ADSP-2184x/ADSP-SC84x SHARC-FX Processor

Preliminary Hardware Reference

Revision Revision 0.2, January 2026

Part Number
82-100163-01

,

ANALOG DEVICES

Notices

Copyright Information

© 2026 Analog Devices, Inc., ALL RIGHTS RESERVED. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

Printed in the USA.

Disclaimer

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

Trademark and Service Mark Notice

The Analog Devices logo, Blackfin, Blackfin+, CrossCore, EngineerZone, EZ-Board, EZ-KIT Lite, EZ-KIT Mini, EZ-Extender, SHARC, SHARC+, Blackfin+, A^2B, E^2B, EdgeStudio, SigmaStudio, SigmaStudio+, and VisualDSP++ are registered trademarks of Analog Devices, Inc.

All other brand and product names are trademarks or service marks of their respective owners.

Contents

Arm Cortex-A55 Subsystem

Cortex-A55 Features 1–2
Functional Description 1–2
Cortex-A55 Block Diagram 1–2
Instruction Fetch Unit (IFU) 1–3
Data Processing Unit (DPU) 1–3
Performance Monitor Unit (PMU) 1–3
Advanced SIMD and Floating-point support 1–4
Cryptographic Extension 1–4
Memory Management Unit (MMU) 1–4
L1 Memory System 1–6
L2 Memory System 1–7
Reliability, Availability, and Serviceability (RAS) 1–7
Generic Interrupt Controller (GIC) 1–7
A55 Core in DynamIQ Shared Unit (DSU) 1–8
System Control 1–8
Generic Timer 1–8
A55 Configurations 1–9
DynamIQ Shared Unit Interfaces 1–10
Performance Monitor Unit (PMU) 1–12

Clock Generation Unit (CGU)

CGU Features 2–1
CGU Functional Description 2–1
ADSP-2184x CGU Register List 2–2
ADSP-2184x CGU Interrupt List 2–2
ADSP-2184x CGU Trigger List 2–3

CGU Definitions 2–3
CGU PLL Block Diagram 2–4
CLKOUT Selections 2–5
CGU Operating Modes 2–6
CGU Power-up Sequence 2–7
CGU Event Control 2–7
CGU Programming Model 2–7
Configuring CGU Modes 2–7
Changing the Clock Frequencies 2–8
Changing the PLL Clock Frequency 2–8
Changing the CCLKn or SYSCLK Frequency Without Modifying the PLLCLK Frequency 2–9
Changing the OCLK Frequency 2–10
Selecting SCLK1_0 Source 2–10
Changing SCLK1_0 Frequencies 2–10
Aligning All Clocks 2–11
Valid Clock Multiplier Settings 2–11
PLL Bypass and PLL Disable 2–11
Processor Specific Information 2–12
ADSP-2184x CGU Register Descriptions 2–12
Core Clock Buffer Disable Register 2–14
Core Clock Buffer Status Register 2–15
CLKOUT Select Register 2–16
Control Register 2–18
Clocks Divisor Register 2–20
DIV Register Extension 2–23
PLL Control Register 2–24
Revision ID Register 2–26
System Clock Buffer Disable Register 2–27
System Clock Buffer Status Register 2–29
Status Register 2–31

Time Stamp Counter 32 LSB Register 2–34
Time Stamp Counter 32 MSB Register 2–35
Time Stamp Control Register 2–36
Time Stamp Counter Initial 32 LSB Value Register 2–37
Time Stamp Counter Initial MSB Value Register 2–38

Clock Distribution Unit (CDU)

CDU Features 3–1
CDU Functional Description 3–1
CDU Block Diagram 3–1
CDU Definitions 3–2
ADSP-2184x CDU Register List 3–3
CDU Clock Configuration Options 3–3
CDU Programming Model 3–4
Changing the PLL and Clock Frequency 3–4
Changing the Clock Frequency 3–5
ADSP-2184x CDU Register Descriptions 3–5
CDU Configuration 3–7
CDU Revision ID 3–8
CDU Status 3–9

Fractional PLL (Frac-N PLL)

Frac-N PLL Features 4–1
Frac-N PLL Definitions 4–2
Frac-N PLL Functional Description 4–4
Frac-N PLL Block Diagram 4–4
FPLL Detailed Block Diagram 4–4
DPLL and FPLL Block Diagram 4–6
ADSP-2184x FRACNPLL Register List 4–7
ADSP-2184x FRACNPLL Interrupt List 4–7
ADSP-2184x FRACNPLL Trigger List 4–8

Frac-N PLL Architectural Concepts 4–8
Interrupts and Triggers 4–8
Lock Interrupt-Trigger 4–8
DPLL Unlock Interrupt 4–9
TWAIT Trigger Receiver 4–9
DAI Clock Selection 4–10
Frac-N PLL CLKOUT Clock Selection 4–10
Usage of Frac-N PLL Output Clock 4–11
Frac-N PLL Operating Modes 4–14
FPLL Only Mode 4–14
DPLL Plus FPLL Mode 4–14
FIN (External Reference Clock) Sources 4–15
DPLL Plus FPLL Clock Multiplexer Configuration 4–17
Frac-N PLL Programming Model 4–17
Programming FPLL Only Mode 4–17
FPLL Only Mode TWAIT 4–18
Programming Model for DPLL Plus FPLL Mode 4–18
DPLL Plus FPLL Mode TWAIT 4–19
Programming the FIN Unlock and Re-lock DPLL 4–20
ADSP-2184x FRACNPLL Register Descriptions 4–20
Integration Control Register 4–22
DPLL Control Register 0 4–25
DPLL Control Register 1 4–26
DPLL Control Register 2 4–27
DPLL Control Register 3 4–28
DPLL Control Register 4 4–29
DPLL Control Register 5 4–30
DPLL Control Register 6 4–31
DPLL Control Register 7 4–32
DPLL Control Register 8 4–34

DPLL Control Register 9 4–35
DPLL Status Register 0 4–36
DPLL Status Register 1 4–37
DPLL Status Register 2 4–38
Fractional PLL Control Register 0 4–39
Fractional PLL Control Register 1 4–41
Integration Status Register 4–42

Dynamic Power Management (DPM)

DPM Features 5–1
DPM Functional Description 5–1
ADSP-2184x DPM Register List 5–1
DPM Definitions 5–2
DPM Operating Modes 5–2
Reset State 5–2
Full-on Mode 5–2
DPM Event Control 5–3
ADSP-2184x DPM Register Descriptions 5–3
Control Register 5–4
Peripherals Disable Register 0 5–5
Peripherals Disable Register 1 5–7
Revision ID 5–9
Status Register 5–10

Reset Control Unit (RCU)

RCU Features 6–1
RCU Functional Description 6–2
ADSP-2184x RCU Register List 6–2
ADSP-2184x RCU Trigger List 6–3
RCU Definitions 6–3
RCU Architectural Concepts 6–3

RCU Status and Error Signals 6–4
Core n Reset In 6–4
Core n Reset Out 6–5
ADSP-2184x RCU Register Descriptions 6–5
Boot Code Register 6–6
Core Reset Outputs Control Register 6–9
Core Reset Outputs Status Register 6–10
Control Register 6–11
Message Register 6–13
Message Clear Bits Register 6–16
Message Set Bits Register 6–17
System Interface Disable Register 6–18
System Interface Status Register 6–19
System Reset Request Status Register 6–20
Status Register 6–21
Software Vector Register 0 6–23
Software Vector Register 1 6–24
Software Vector Register 2 6–25
SVECT Lock Register 6–26

System Event Controller (SEC) and Generic Interrupt Controller (GIC)

SEC Features 7–1
SEC Functional Description 7–2
ADSP-2184x SEC Register List 7–2
ADSP-2184x SEC Interrupt List 7–3
ADSP-2184x SEC Trigger List 7–3
ADSP-2184x Interrupt List 7–3
SEC Definitions 7–16
SEC System Event Aggregator (SEA) 7–17
SEC Block Diagram 7–18

SEC Fault Interface (SFI) 7–18
SEC Core Interface (SCI) 7–19
SEC Source Interface (SSI) 7–20
SEC Architectural Concepts 7–20
System Interrupt Groups 7–20
System Interrupt Flow 7–20
System Interrupt Priorities 7–22
SEC Error 7–22
SEC Programming Model 7–22
Programming Concepts 7–23
Programming Examples 7–23
Fault Management Interface Programming Model 7–23
Configuring a System Source to Interrupt a Core 7–24
Core/SEC Handshake Requirements to Ensure Proper Interrupt Handling 7–25
Configuring a System Source as a Fault 7–25
Configuring the WDOG Expiry Event to Issue a System Reset 7–26
SEC Programming Restrictions 7–26
ADSP-2184x SEC Register Descriptions 7–27
SCI Active Register n 7–28
SCI Control Register n 7–29
SCI Group Mask Register n 7–31
SCI Priority Level Register n 7–33
SCI Priority Mask Register n 7–35
Core Pending Register n 7–36
SCI Source ID Register n 7–37
SCI Status Register n 7–38
Global End Register 7–40
Fault COP Period Register 7–41
Fault COP Period Current Register 7–42
Fault Control Register 7–43
Fault Delay Register 7–46

Fault Delay Current Register 7–47
Fault End Register 7–48
Fault Source ID Register 7–49
Fault System Reset Delay Register 7–50
Fault System Reset Delay Current Register 7–51
Fault Status Register 7–52
Global Control Register 7–54
Global Status Register 7–55
Global Raise Register 7–57
Source Control Register n 7–58
Source Status Register n 7–61
GIC Overview 7–62
GIC Functional Description 7–63
GIC Block Diagram 7–64
GIC Performance Monitoring Unit (GIC PMU) 7–64

Trigger Routing Unit (TRU)

TRU Features 8–1
TRU Functional Description 8–1
ADSP-2184x TRU Register List 8–2
ADSP-2184x TRU Interrupt List 8–2
ADSP-2184x Trigger List 8–3
TRU Definitions 8–16
TRU Block Diagram 8–17
TRU Architectural Concepts 8–17
TRU Programming Model 8–17
Programming Concepts 8–18
Programming Examples 8–18
Configuring a Simple Trigger Sequence 8–18
TRU Event Control 8–18

TRU Status and Error Signals 8–18
ADSP-2184x TRU Register Descriptions 8–18
Error Address Register 8–20
Global Control Register 8–21
Generator Trigger Register 8–23
Receiver Select Register 8–24
Status Information Register 8–25

L2 System Memory

L2 System Memory Features 9–1
L2 System Memory Functional Description 9–1
ADSP-2184x L2CTL Register List 9–2
ADSP-2184x L2CTL Interrupt List 9–3
ADSP-2184x L2CTL Trigger List 9–3
L2 System Memory Block Diagram 9–3
L2 System Memory Architectural Concepts 9–4
Access Characteristics 9–4
Read/Write Latency and Throughput 9–4
L2 Memory Controller Block Diagram (Instance) 9–5
Arbitration and Priority 9–5
Data Integrity 9–7
ECC Algorithm 9–7
ECC Hardware Control 9–9
ECC Error Management 9–9
Memory Refresh 9–10
Power Modes 9–10
L2 System Memory Event Control 9–11
ECC Error Interrupt 9–11
ADSP-2184x L2CTL Register Descriptions 9–11
Control Register 9–13
Error Type 0 Address Register 9–16

Error Type 1 Address Register 9–17
Error Type 2 Address Register 9–18
Error Type 3 Address Register 9–19
Error Type 4 Address Register 9–20
ECC Error Address 0 Register 9–21
ECC Error Address 1 Register 9–22
ECC Error Address 2 Register 9–23
ECC Error Address 3 Register 9–24
ECC Error Address 4 Register 9–25
ECC Error Address 5 Register 9–26
ECC Error Address 6 Register 9–27
ECC Error Address 7 Register 9–28
ECC Error Address 8 Register 9–29
Error Type 0 Register 9–30
Error Type 1 Register 9–32
Error Type 2 Register 9–33
Error Type 3 Register 9–34
Error Type 4 Register 9–35
Initialization Register 9–36
Initialization Status Register 9–38
Power Control Register 9–40
Revision ID Register 9–42
Read Priority Count Register 9–43
Read Priority Count Register 9–44
Scrub Start Address Register 9–45
Scrub Count Register 9–46
Scrub Control Register 9–47
Status Register 9–49
L2 Port Error Status Register 9–52
Write Priority Count Register 9–53

Write Priority Count Register 9–54

DDR Pre-fetch Buffer (DDRPFB)

DDRPFB Features 10–2
Smart Pre-fetch 10–2
Data Invalidation 10–2
Direct Access 10–2
Return Zero 10–2
Controller ID Based Direct Access 10–2
Range Based Pre-fetch 10–4
ADSP-2184x PFB Register List 10–5
DDRPFB Programming Model 10–6
Enabling the Pre-fetch Buffer and PFB Idle State 10–6
General Programming Guidelines 10–6
Guidelines For Achieving Maximum Performance 10–6
ADSP-2184x PFB Register Descriptions 10–7
Prefetch buffer fourth control register 10–9
Prefetch buffer third control register 10–10
Control Register 0 10–11
Control Register 1 10–13
Range Select Register 10–15
Requester ID 0 Register 10–16
Requester ID 0 Mask Register 10–17
Requester ID 1 Register 10–18
Requester ID 1 Mask Register 10–19
Requester ID 2 Register 10–20
Requester ID 2 Mask Register 10–21
Requester ID 3 Register 10–22
Requester ID 3 Mask Register 10–23
Requester ID Enable Register 10–24

Prefetch Buffer First Status Register 10–25
Return Zero End Address Register 10–26
Return Zero Start Address Register 10–27

One-Time Programmable Memory Controller (OTPC)

OTPC Features 11–1
OTPC Functional Description 11–1
ADSP-2184x OTPC Register List 11–1
ADSP-2184x OTPC Interrupt List 11–2
Error Correction 11–2
OTP Layout 11–2
OTPC Event Control 11–8
OTPC Interrupt Signals 11–8
OTPC Status and Error Signals 11–8
OTP API Overview 11–8
OTP Programming 11–9
OTP Reading 11–9
OTP Get Field 11–9
ADSP-2184x OTPC Register Descriptions 11–9
OTP Security State Register 11–11
OTP Status Register 11–12

System Memory Protection Unit (SMPU)

SMPU Features 12–1
SMPU Functional Description 12–2
ADSP-2184x SMPU Register List 12–3
SMPU Interrupts 12–4
Memory Writes 12–4
Memory Reads 12–4
ID Comparison 12–5
Memory Region 12–8

SMPU Definitions 12–9
SMPU Block Diagram 12–9
SMPU Architectural Concepts 12–10
SMPU Operating Modes 12–10
SMPU Interrupt Signals 12–10
SMPU Status and Error Signals 12–11
ADSP-2184x SMPU Register Descriptions 12–11
Bus Error Address Register 12–13
Bus Error Details Register 12–14
SMPU Control Register 12–15
Exclusive Access IDn Address 12–17
Exclusive Access Status 12–18
Interrupt Address Register 12–19
Interrupt Details Register 12–20
Region n Address Register 12–21
Region n Control Register 12–22
SMPU Revision ID Register 12–25
Region n ID A Register 12–26
Region n ID B Register 12–27
Region n ID Mask A Register 12–28
Region n ID Mask B Register 12–29
SMPU Control Secure Accesses Register 12–30
Region n Control Secure Accesses Register 12–32
SMPU Status Register 12–33

General-Purpose Ports (PORT)

PORT Features 13–2
PORT Functional Description 13–2
ADSP-2184x PORT Register List 13–3
ADSP-2184x PORT Trigger List 13–3

ADSP-2184x PINT Register List 13–4
ADSP-2184x PINT Interrupt List 13–4
ADSP-2184x PINT Trigger List 13–5
ADSP-2184x PADS Register List 13–5
PORT Architectural Concepts 13–7
Internal Interfaces 13–8
External Interfaces 13–8
GPIO Pin Function 13–8
Input Mode 13–8
Output Mode 13–8
Trigger Toggle Mode 13–8
Open-Drain Mode 13–9
Port Multiplexing Control 13–9
PORT Event Control 13–9
PORT Interrupt Signals 13–9
PORT Programming Model 13–11
Programmable Pull Enable and Pull Select Resistors for PORT and DAI 13–15
ADSP-2184x PORT Register Descriptions 13–16
Port x GPIO Data Register 13–17
Port x GPIO Data Clear Register 13–21
Port x GPIO Data Set Register 13–25
Port x GPIO Output Toggle Register 13–28
Port x GPIO Direction Register 13–31
Port x GPIO Direction Clear Register 13–35
Port x GPIO Direction Set Register 13–39
Port x Function Enable Register 13–42
Port x Function Enable Clear Register 13–45
Port x Function Enable Set Register 13–48
Port x GPIO Input Enable Register 13–51
Port x GPIO Input Enable Clear Register 13–54

Port x GPIO Input Enable Set Register 13–57
Port x GPIO Lock Register 13–60
Port x Multiplexer Control Register 13–62
Port x GPIO Polarity Invert Register 13–64
Port x GPIO Polarity Invert Clear Register 13–68
Port x GPIO Polarity Invert Set Register 13–71
Port x GPIO Trigger Toggle Register 13–74
ADSP-2184x PINT Register Descriptions 13–75
PINT Assign Register 13–77
PINT Edge Clear Register 13–79
PINT Edge Set Register 13–82
PINT Invert Clear Register 13–85
PINT Invert Set Register 13–88
PINT Latch Register 13–91
PINT Mask Clear Register 13–95
PINT Mask Set Register 13–98
PINT Pin State Register 13–101
PINT Request Register 13–105
ADSP-2184x PADS Register Descriptions 13–109
DAI0 Pins 1 to 8 Drive Strength Control Register 13–111
DAI0 Pins 9 to 16 Drive Strength Control Register 13–112
DAI0 Pins 17 to 20 Drive Strength Control Register 13–113
DAI0 Port Input Enable Control Register 13–114
DAI0 Pull Enable Register 13–115
DAI0 Pull Selector Register 13–117
DAI0 Slew Rate Control Register 13–119
DAI0 Strong Pull Up Control Register 13–121
DAI1 Pins 1 to 8 Drive Strength Control Register 13–123
DAI1 Pins 9 to 16 Drive Strength Control Register 13–124
DAI1 Pins 17 to 20 Drive Strength Control Register 13–125

DAI1 Port Input Enable Control Register 13–126
DAI1 Pull Enable Register 13–127
DAI1 Pull Selector Register 13–129
DAI1 Slew Rate Control Register 13–131
DAI1 Strong Pull up Control Register 13–133
Non-GPIO Control Register 13–135
Non-GPIO Drive Strength Register 13–137
EMAC Non-GPIO Drive Strength Register 13–138
OTPC Boot Status Register 13–139
Peripheral PAD Configuration 0 Register 13–140
Peripheral Configuration 1 Register 13–142
PORTA Pins 0 to 7 Drive Strength Control Register 13–143
PORTA Pins 8 to 15 Drive Strength Control Register 13–144
PORTB Pins 0 to 7 Drive Strength Control Register 13–145
PORTB Pins 8 to 15 Drive Strength Control Register 13–146
PORTC Pins 0 to 7 Drive Strength Control Register 13–147
PORTC Pins 8 to 15 Drive Strength Control Register 13–148
PORTD Pins 0 to 7 Drive Strength Control Register 13–149
PORTD Pins 8 to 15 Drive Strength Control Register 13–150
PORTE Pins 0 to 7 Drive Strength Control Register 13–151
PORTE Pins 8 to 15 Drive Strength Control Register 13–152
PORTF Pins 0 to 7 Drive Strength Control Register 13–153
PORTF Pins 8 to 15 Drive Strength Control Register 13–154
PORTG Pins 0 to 7 Drive Strength Control Register 13–155
PORTG Pins 8 to 15 Drive Strength Control Register 13–156
PORTH Pins 0 to 7 Drive Strength Control Register 13–157
PORTH Pins 8 and 9 Drive Strength Control Register 13–158
PORTA and PORTB Pull Enable Register 13–159
PORTA and PORTB Pull Selector Register 13–162
PORTA and PORTB Slew Rate Control Register 13–165

PORTA and PORTB Strong Pull Up Control Register 13–168
PORTC and PORTD Pull Enable Register 13–171
PORTC and PORTD Pull Selector Register 13–174
PORTC and PORTD Slew Rate Control Register 13–177
PORTC and PORTD Strong Pull Up Control Register 13–180
PORTE and PORTF Pull Enable Register 13–183
PORTE and PORTF Pull Selector Register 13–186
PORTE and PORTF Slew Rate Control Register 13–189
PORTE and PORTF Strong Pull Up Control Register 13–192
PORTG and PORTH Pull Enable Register 13–195
PORTG and PORTH Pull Selector Register 13–198
PORTG and PORTH Slew Rate Control Register 13–201
PORTG and PORTH Strong Pull Up Control Register 13–204
Status Register 13–207

Thermal Monitoring Unit (TMU)

TMU Features 14–1
TMU Functional Description 14–1
ADSP-2184x TMU Register List 14–1
ADSP-2184x TMU Interrupt List 14–2
ADSP-2184x TMU Trigger List 14–2
TMU Definitions 14–2
TMU Block Diagram 14–3
TMU Architectural Concepts 14–3
TMU and HADC 14–4
TMU Event Control 14–5
Status and Error Signals 14–5
TMU Programming Guidelines 14–5
Reading Temperature Data 14–6
TMU Calibration 14–6

ADSP-2184x TMU Register Descriptions 14–7
Alert High Limit Register 14–8
Alert Low Limit Register 14–9
Averaging Register 14–10
Temperature Conversion Blank Register 14–11
TMU Control Register 14–12
Fault High Limit Register 14–13
Fault Low Limit Register 14–14
Gain Value Register 14–15
Interrupt Mask Register 14–16
Offset Register 14–17
Temperature Refresh Counter 14–18
Status Register 14–19
Temperature Value Register 14–21

Housekeeping ADC (HADC)

HADC Features 15–1
HADC Functional Description 15–1
ADSP-2184x HADC Register List 15–2
ADSP-2184x HADC Interrupt List 15–2
ADSP-2184x HADC Trigger List 15–3
HADC Definitions 15–3
HADC Block Diagram 15–4
HADC Signal Descriptions 15–5
HADC Architectural Concepts 15–5
Converter Operation 15–6
Auto-Scan 15–6
Channel Sequence Programming 15–7
ADC Transfer Function 15–7
Results 15–7
HADC Operating Modes 15–7

HADC Event Control 15–8
HADC Programming Model 15–8
ADSP-2184x HADC Register Descriptions 15–9
Channel Mask Register 15–10
Control Register 15–11
Channel Data Registers 15–13
Interrupt Mask Register 15–14
Status Register 15–15

Controller Area Network Flexible Data Rate (CANFD)

CANFD Features 16–1
CANFD Functional Description 16–2
ADSP-2184x CANFD Register List 16–3
ADSP-2184x CANFD Interrupt List 16–5
ADSP-2184x CANFD Trigger List 16–5
CANFD Architectural Concepts 16–6
Block Diagram 16–6
Message Buffer (MB) 16–7
Message Buffer Memory Map 16–13
Memory Partition 16–18
Rx FIFO Structure 16–19
CANFD Processes 16–22
Transmit Process 16–22
Arbitration Process 16–23
Lowest Number Mailbox First 16–23
Highest-Priority Mailbox First 16–23
Arbitration Completion 16–24
Arbitration Start and Stop Conditions 16–25
Receive Process 16–26
Matching Process 16–27
Receive Process Pretending Network Mode 16–32

Move Process 16–35
Move-In 16–35
Move-Out 16–37
Data Coherence 16–37
Transmission Abort Mechanism 16–37
Mailbox Inactivation 16–38
Mailbox Lock Mechanism 16–39
Rx FIFO 16–40
Rx FIFO Under DMA Operation 16–41
Clear FIFO Operation 16–41
CAN Protocol Features 16–42
CAN FD Frames 16–42
Transceiver Delay Compensation 16–46
Remote Frames 16–47
Overload Frames 16–48
Time Stamp 16–48
Protocol Timing 16–49
Arbitration and Matching Timing 16–52
Tx Arbitration Start Delay 16–53
CANFD Clock Domains and Restrictions 16–56
CANFD Operating Modes 16–59
Normal Mode 16–59
Freeze Mode 16–60
Loop-Back Mode 16–60
Listen-Only Mode 16–61
CAN FD Active Mode 16–61
Module Disable Mode 16–61
Doze Mode 16–62
Stop Mode 16–63
Pretended Network Mode 16–64
CANFD Event Control 16–65

Interrupts 16–65
Bus Interface 16–66
Detection and Correction of Memory Errors 16–67
Sources of Memory Access 16–69
Error Indication 16–69
Error Reporting 16–70
Response to Errors 16–70
Error Injection 16–71
CANFD Programming Model 16–72
Initialization 16–73
Pretended Network - Doze Mode 16–75
Pretended Network - Stop Mode 16–76
ADSP-2184x CANFD Register Descriptions 16–76
Can Bit Timing Register 16–79
CRC Register 16–82
Control 1 Register 16–83
Pretended Networking Control1 Register 16–89
Control 2 Register 16–92
Pretended Networking Control2 Register 16–97
Error Counter Register 16–98
Error Injection Address Register 16–101
Error Injection Data Pattern Register 16–102
Error Injection Parity Pattern Register 16–103
Error Status Register 16–104
Error and Status 1 Register 16–107
Error and Status 2 Register 16–118
CANFD Bit Timing Register 16–120
CANFD CRC Register 16–122
CANFD Control Register 16–124
Pretended Networking DLC Filter Register 16–128

Pretended Networking ID Filter1 Register 16–129
Pretended Networking ID Filter2 / IDMask Register 16–130
Mailbox Interrupt Flag 1 Register 16–132
Mailbox Interrupt Flag 2 Register 16–137
Mailbox Interrupt Mask 1 Register 16–138
Mailbox Interrupt Mask 2 Register 16–139
Module Configuration Register 16–140
Memory Error Control Register 16–149
Pretended Networking Payload Low Filter2 Register 16–152
Pretended Networking Payload Low Filter1 Register 16–153
Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register 16–154
Pretended Networking Payload Low Filter2 / Payload Low Mask Register 16–155
Error Report Address Register 16–156
Error Report Data Register 16–158
Error Report Syndrome Register 16–159
Receive Mailbox14 Mask Register 16–161
Receive Mailbox15 Mask Register 16–162
Receive FIFO Global Mask Register 16–163
Receive FIFO Information Register 16–164
Receive Individual Mask Register 16–165
Receive Mailbox Global Mask Register 16–166
Free Running Timer Register 16–167
Pretended Networking Wake Up Message Buffer 0 Control Status Register 16–168
Pretended Networking Wake Up Message Buffer 0 Data 0-3 Register 16–169
Wakeup Message Buffer Data 4-7 Register 16–170
Pretended Networking Wake Up Message Buffer ID 0 Register 16–171
Pretended Networking Wake Up Message Buffer 1 Control Status Register 16–172
Pretended Networking Wake Up Message Buffer 1 Data 0-3 Register 16–173
Pretended Networking Wake Up Message Buffer 1 Data 4-7 Register 16–174
Pretended Networking Wake Up Message Buffer ID 1 Register 16–175

Pretended Networking Wake Up Message Buffer 2 Control Status Register 16–176
Pretended Networking Wake Up Message Buffer 2 Data 0-3 Register 16–177
Pretended Networking Wake Up Message Buffer 2 Data 4-7 Register 16–178
Pretended Networking Wake Up Message Buffer ID 2 Register 16–179
Pretended Networking Wake Up Message Buffer 3 Control Status Register 16–180
Pretended Networking Wake Up Message Buffer 3 Data 0-3 Register 16–181
Pretended Networking Wake Up Message Buffer 3 Data 4-7 Register 16–182
Pretended Networking Wake Up Message Buffer ID 3 Register 16–183
Pretended Networking Wakeup Match Register 16–184

Watchdog Timer (WDOG)

WDOG Features 17–1
WDOG Functional Description 17–2
ADSP-2184x WDOG Register List 17–2
ADSP-2184x WDOG Interrupt List 17–3
ADSP-2184x WDOG Trigger List 17–3
WDOG Block Diagram 17–3
Internal Interface 17–4
External Interface 17–4
ADSP-2184x WDOG Register Descriptions 17–4
Count Register 17–5
Control Register 17–6
Watchdog Timer Status Register 17–7
Watchdog Timer Window Register 17–9

Link Port (LP)

LP Features 18–1
LP Functional Description 18–1
ADSP-2184x LP Register List 18–2
ADSP-2184x LP Interrupt List 18–2
ADSP-2184x LP Trigger List 18–2

ADSP-2184x LP DMA Channel List 18–3
Block Diagram 18–3
External Connections 18–4
Internal Blocks 18–4
Architectural Concepts 18–4
FIFO Buffers 18–4
Handshake for Link Port Enable Process 18–6
Multi-Processor Connectivity 18–7
LP Operating Modes 18–8
LP Data Transfer Modes 18–8
Core Data Transfers 18–8
DMA Data Transfers 18–8
LP Event Control 18–9
Interrupt Signals 18–9
Enabling Link Port Interrupts 18–10
Status and Error Signals 18–10
LP Programming Model 18–11
Setting Up a DMA Transmit Operation 18–11
Setting Up a DMA Receive Operation 18–11
Setting Up a Core Transmit Operation 18–12
Setting Up a Core Receive Operation 18–13
ADSP-2184x LP Register Descriptions 18–13
Control Register 18–14
Clock Divider Value Register 18–16
Receive Buffer Register 18–17
Status Register 18–18
Transmit Buffer Register 18–20
Shadow Input Transmit Buffer Register 18–21
Shadow Output Transmit Buffer Register 18–22

Pulse Density Modulation (PDM) Microphone Interface

PDM Features 19–1
PDM Functional Description 19–1
ADSP-2184x PDM Register List 19–2
PDM Block Diagram 19–2
PDM Architectural Concepts 19–3
PDM Initialization and Clocking 19–3
SPORT Interface 19–4
SPORT Timing 19–5
High Pass Filter 19–7
PDM Programming Model 19–8
Configuring the PDM Interface 19–8
Software Reset 19–9
ADSP-2184x PDM Register Descriptions 19–9
PDM Control Register 19–10
High Pass Filter Control Register 19–11
Software Reset Register 19–12
Serial Port Control0 Register 19–13
Serial Port Control1 Register 19–15

Serial Peripheral Interface (SPI)

SPI Features 20–1
SPI Functional Description 20–2
ADSP-2184x SPI Register List 20–2
ADSP-2184x SPI Interrupt List 20–3
ADSP-2184x SPI Trigger List 20–4
ADSP-2184x SPI DMA Channel List 20–5
SPI Block Diagram 20–5
Transfer Protocol 20–6
Clock Considerations 20–7
Controlling Delay Between Frames 20–8
Flow Control 20–10

Target Select Operation 20–11
Beginning and Ending a Non-DMA SPI Transfer 20–12
Transmit Operation in Non-DMA Mode 20–13
Receive Operation in Non-DMA Mode 20–13
Dual I/O Mode 20–13
Quad I/O Mode (SPI1/SPI2) 20–14
Fast Mode 20–15
Memory-Mapped Mode (SPI1/SPI2) 20–16
Memory-Mapped Description of Operation 20–17
Memory-Mapped Architectural Concepts 20–18
Memory-Mapped Read Accesses 20–20
Memory-Mapped High-Performance Features 20–23
Merged Read Accesses 20–23
Wrap Around Accesses 20–23
Execute-In-Place (XIP, SPI2 only) 20–24
Memory-Mapped Mode Error Status Bits 20–25
Memory-Mapped Programming Guidelines 20–26
SPI Interrupt Signals 20–28
Data Interrupts 20–28
Status Interrupts 20–29
Error Conditions 20–29
SPI Programming Concepts 20–30
Controller Operation in Non-DMA Modes 20–31
Target Operation in Non-DMA Modes 20–31
Configuring DMA Controller Mode 20–32
Configuring DMA Target Mode Operation 20–33
ADSP-2184x SPI Register Descriptions 20–34
Clock Rate Register 20–36
Control Register 20–37
Delay Register 20–43

Masked Interrupt Condition Register 20–44
Masked Interrupt Clear Register 20–46
Interrupt Mask Register 20–49
Interrupt Mask Clear Register 20–51
Interrupt Mask Set Register 20–54
Memory Mapped Read Header 20–57
SPI Memory Top Address 20–61
Receive FIFO Data Register 20–62
Received Word Count Register 20–63
Received Word Count Reload Register 20–64
Receive Control Register 20–65
Target Select Register 20–68
Status Register 20–71
Transmit FIFO Data Register 20–76
Transmitted Word Count Register 20–77
Transmitted Word Count Reload Register 20–78
Transmit Control Register 20–79

Extended Serial Peripheral Interface (xSPI)

xSPI Features 21–1
Supported Memory Devices 21–2
xSPI Functional Description 21–2
ADSP-2184x XSPI Register List 21–2
ADSP-2184x xSPI Interrupt List 21–5
xSPI Block Diagram 21–6
Architectural Concepts 21–6
Operating Modes 21–7
Direct Mode 21–7
STIG Mode 21–8
ACMD Mode 21–9
Integrated Soft PHY 21–10

XIP Mode 21–10
Configuring the Controller 21–11
Operating in Direct Mode 21–12
Write Limitations 21–13
AXI Wrapping Bursts 21–13
Entering and Exiting XIP Mode 21–14
Operating in ACMD Mode 21–17
Using the Auto Command in CDMA Mode 21–18
Using the Auto Command in PIO Mode 21–24
Status Checking 21–27
Selecting the DMA Interface 21–29
Rules and Limitations 21–30
Operating in STIG Mode 21–31
Instruction Variants 21–31
Glued Data Instruction 21–40
Instruction Type Variants 21–44
Switching Work Modes 21–47
Programming STIG Mode 21–47
STIG Mode Status 21–48
STIG Limitations 21–49
Sequencing STIG Operation 21–50
Event Handling 21–51
Device Discovery 21–53
Discovery Modes 21–53
Discovery Flow 21–54
SFDP Command Structure 21–55
SPI NAND Workflow 21–62
Discovery Status 21–63
Discovery Restrictions 21–65
Discovery Limitations 21–66
Legacy Devices 21–66

Device Specific Configuration 21–66
Controller Reconfiguration 21–67
Common Sequence Register Updates 21–68
Reset Sequence Parameters for Profile 1 Devices 21–70
Erase Sequence Register Updates 21–71
Write Enable Sequence Register Updates 21–71
Program Sequence Register Updates 21–71
Read Sequence Register Updates 21–73
Status Sequence Register Updates 21–77
Boot Engine 21–82
Configuring the Boot Engine 21–83
xSPI PHY Controller 21–85
PHY Initialization 21–85
Initial PHY Registers to Configure 21–86
XSPI_PHY_DLL_CTL Register 21–86
XSPI_PHY_DQ_TR Register 21–87
XSPI_PHY_DQS_TR Register 21–87
XSPI_PHY_GATE_LPBK_CTL Register 21–87
XSPI_PHY_DLL_REQ_CTL Register 21–88
XSPI_PHY_DLL_COMP_CTL Register 21–88
PHY Re-initialization 21–91
Resynchronization of Target Delay Lines 21–91
PHY Read Sampling Method Selection 21–92
DQS Mode for PHY Read Sampling 21–92
Non-DQS Mode (Loopback) for PHY Read Sampling 21–92
Memory Device Data Integrity 21–93
ECC Data Integrity 21–93
CRC Parity Check 21–93
CRC Using Direct or ACMD Mode 21–93
CRC Using STIG Mode 21–94
Miscellaneous Low-level Configurations 21–94
Write Protection 21–94

Device Reset 21–94
SPI Clock CPOL and CPHA Configuration 21–94
JEDEC Reset Delay Settings 21–94
Device Delay Settings 21–94
Reset Recovery Delay Settings 21–95
t_{CMS} Timing Pause Refresh Settings 21–95
Legacy HyperFlash or Profile 2 Address Settings 21–95
Multiple Device Connections to Controller 21–95
SPI NAND Device Operation Settings 21–95
ADSP-2184x XSPI Register Descriptions 21–96
Boot Status Register 21–99
Command Register 0 21–100
Command Register 1 21–101
Command Register 2 21–102
Command Register 3 21–103
Command Register 4 21–104
Command Register 5 21–105
Command Status Register 21–106
Command Status Pointer Register 21–107
DAC Configuration Register 21–108
Address Remapping Register 0 21–110
Address Remapping Register 1 21–111
Device Discovery Control Register 21–112
DMA Interface Control Register 21–116
DMA Error Address High Register 21–118
DMA Error Address Low Register 21–119
Erase Sequence Configuration Register 0 21–120
Erase Sequence Configuration Register 1 21–122
Erase Sequence Configuration Register 2 21–123
Controller Features Register 21–124

General Controller Status Register 21–126
Interrupt Enable Register 21–128
Interrupt Status Register 21–131
Long Polling Count Register 21–134
Device Active Maximum Clock Cycle Register 21–135
Clock Mode Control Register 21–136
Device Delay Register 21–137
HyperFlash Offset Register 21–138
JEDEC Reset Delay Register 21–139
Hardware Reset Control Register 21–140
Reset Recovery Delay Register 21–143
Write Protect Register 21–144
PHY DLL Observable Points 0 Register 21–145
PHY DLL Observable Points 1 Register 21–147
PHY DLL Control Register 21–148
Bus Requester DLL Control Register 21–150
Bus Completer DLL Control Register 21–152
PHY DLL Resynchronization Register 21–153
PHY DQS Timing Register 21–154
PHY DQ Timing Register 21–156
PHY Features Register 21–157
PHY Gate Loopback Control Register 21–160
PHY Global Control Register 21–162
PHY GPIO Control Register 0 21–163
PHY GPIO Control Register 1 21–164
PHY GPIO Status Register 0 21–165
PHY GPIO Status Register 1 21–166
PHY Global Termination Control Register 21–167
PHY DQS Input Enable Timing Register 21–168
PHY Observable Points Register 21–169

PHY Revision ID Register 21–171
PHY Static Aging Register 21–172
PHY Deskew Write Register 21–173
Program Sequence Configuration Register 0 21–175
Program Sequence Configuration Register 1 21–177
Program Sequence Configuration Register 2 21–178
Read Sequence Configuration Register 0 21–180
Read Sequence Configuration Register 1 21–182
Read Sequence Configuration Register 2 21–184
Revision ID Register 21–186
Reset Sequence Configuration Register 0 21–187
Reset Sequence Configuration Register 1 21–189
Host DMA Buffer Address Register 0 21–190
Host DMA Buffer Address Register 1 21–191
Host DMA Block Size Register 21–192
Host DMA Thread Status Register 21–193
Sequence Configuration Register 0 21–194
Sequence Configuration Register 1 21–198
Short Polling Count Register 21–200
Status Checking Sequence Configuration Register 0 21–201
Status Checking Sequence Configuration Register 1 21–203
Status Checking Sequence Configuration Register 10 21–205
Status Checking Sequence Configuration Register 2 21–207
Status Checking Sequence Configuration Register 3 21–208
Status Checking Sequence Configuration Register 4 21–209
Status Checking Sequence Configuration Register 5 21–210
Status Checking Sequence Configuration Register 7 21–212
Status Checking Sequence Configuration Register 8 21–213
Status Checking Sequence Configuration Register 9 21–214
Auto Command Engine Interrupt Status Thread Register 21–215

Thread Error Interrupt Enable Register 21–216
Thread Error Interrupt Status Register 21–217
Auto Command Engine Thread Status Register 21–219
WEL Sequence Configuration Register 21–220
Device Control Register 21–222
XIP Configuration Register 21–223

Universal Asynchronous Receiver/Transmitter (UART)

UART Features 22–1
UART Functional Description 22–2
ADSP-2184x UART Register List 22–3
ADSP-2184x UART Interrupt List 22–3
ADSP-2184x UART Trigger List 22–4
ADSP-2184x UART DMA Channel List 22–5
UART Block Diagram 22–5
UART Architectural Concepts 22–5
Internal Interface 22–5
External Interface 22–6
Hardware Flow Control 22–6
Bit Rate Generation 22–7
Autobaud Detection 22–7
UART Debug Features 22–9
UART Operating Modes 22–9
UART Mode 22–10
IrDA SIR Mode 22–10
Multi-Drop Bus Mode 22–10
UART Data Transfer Modes 22–12
UART Mode Transmit Operation (Core) 22–12
UART Mode LIN Break Command 22–12
UART Mode Receive Operation (Core) 22–13
IrDA Transmit Operation 22–14

IrDA Receive Operation 22–14
MDB Transmit Operation 22–15
MDB Receive Operation 22–16
DMA Mode 22–16
Mixing DMA and Core Modes 22–17
Setting Up Hardware Flow Control 22–17
UART Event Control 22–18
Interrupt Masks 22–18
Interrupt Servicing 22–18
Transmit Interrupts 22–19
Receive Interrupts 22–20
Status Interrupts 22–21
Multi-Drop Bus Events 22–22
UART Programming Model 22–22
Detecting Autobaud 22–22
Using Common Initialization Steps 22–23
Using Core Transfers 22–23
Using DMA Transfers 22–23
Using Interrupts 22–23
Setting Up Hardware Flow Control 22–23
ADSP-2184x UART Register Descriptions 22–24
Clock Rate Register 22–25
Control Register 22–26
Interrupt Mask Register 22–32
Interrupt Mask Clear Register 22–36
Interrupt Mask Set Register 22–38
Receive Buffer Register 22–40
Receive Shift Register 22–41
Receive Counter Register 22–42
Scratch Register 22–43

Status Register 22–44
Transmit Address/Insert Pulse Register 22–49
Transmit Hold Register 22–50
Transmit Shift Register 22–51
Transmit Counter Register 22–52

General-Purpose Timer (TIMER)

GP Timer Features 23–1
ADSP-2184x TIMER Register List 23–2
ADSP-2184x TIMER Interrupt List 23–2
ADSP-2184x TIMER Trigger List 23–3
Timer Block Diagram 23–5
Internal Interface 23–5
Internal Timer Connections 23–6
External Interface 23–6
GP Timer Operating Modes 23–6
General Operation 23–7
Period, Width and Delay Register Interaction 23–7
Single-Pulse PWMOUT Mode 23–8
Continuous PWMOUT Mode 23–9
Width Capture (WIDCAP) Mode 23–10
Width Capture Mode Overflow 23–13
Windowed Watchdog (WATCHDOG) Modes 23–15
Windowed Watchdog Width Mode 23–15
Windowed Watchdog Period Mode 23–17
Pin Interrupt (PININT) Mode 23–19
External Clock (EXTCLK) Mode 23–19
GP Timer Programming Concepts 23–20
Setting Up Constantly Changing Timer Conditions 23–20
Configuring, Enabling, and Disabling One or More Timers 23–21
Configuring Timer Data and Status Interrupts 23–21

Configuring the Timer as a Trigger Receiver 23–21
Ordered Trigger Toggle Mode 23–22
Using the Timer Broadcast Feature 23–22
Timer Illegal States 23–23
Continuous PWMOUT Mode 23–23
Single Pulse PWMOUT Mode 23–25
WIDCAP Mode 23–25
EXTCLK Mode 23–25
WATCHDOG Events 23–26
ADSP-2184x TIMER Register Descriptions 23–27
Broadcast Delay Register 23–29
Broadcast Period Register 23–30
Broadcast Width Register 23–31
Data Interrupt Latch Register 23–32
Data Interrupt Mask Register 23–33
Error Type Status Register 23–34
Run Register 23–38
Run Clear Register 23–39
Run Set Register 23–40
Status Interrupt Latch Register 23–41
Status Interrupt Mask Register 23–42
Stop Configuration Register 23–43
Stop Configuration Clear Register 23–44
Stop Configuration Set Register 23–45
Timer n Configuration Register 23–46
Timer n Counter Register 23–51
Timer n Delay Register 23–52
Timer n Period Register 23–53
Timer n Width Register 23–54
Trigger Receiver Enable Register 23–55

Trigger Generator Mask Register 23–56

Pulse-Width Modulator (PWM)

PWM Features 24–1
Functional Description 24–2
ADSP-2184x PWM Register List 24–2
ADSP-2184x PWM Interrupt List 24–4
ADSP-2184x PWM Trigger List 24–4
PWM Definitions 24–4
Architectural Concepts 24–5
Block Diagram 24–5
Enhanced PWM Clocking 24–6
Timer Units 24–8
PWM Timer Period (PWM_TM) Registers 24–8
Timer Unit Operation 24–9
Phase Offset Control 24–10
Channel Timing Control Unit 24–13
Channel Control 24–13
Pulse Positioning and Duty Cycle Registers 24–14
Duty Cycle and Pulse Positioning Control 24–14
Channel Low Side Output Dependent Operation Mode and Dead Time 24–15
Channel High Side and Low Side Outputs, Independent Operation Mode 24–18
Switched Reluctance Motors Application 24–19
Switching Dead Time (PWM_DT) Register 24–20
Duty Cycle with Dead Time Control: Calculations for PULSEMODE 00 24–20
Special Consideration for PWM Operation in Over-Modulation 24–22
Gate Drive Unit 24–24
Output Control Feature Precedence 24–25
Operating Modes 24–25
Sync Operation Modes 24–25
External (Triggered) PWM Sync Generation 24–25
Output Disable and Cross-Over Modes 24–26

Brushless DC Motor (Electronically Commutated Motor) Control 24–27
Emulation Mode 24–28
Event Control 24–28
Trip Control Unit 24–29
Programming Model 24–30
Programming Model for Three-Phase AC Motor Control 24–30
PWM Initialization for Motor Control 24–32
PWM Enable for Motor Control 24–34
PWM Response to Sync Interrupt for Motor Control 24–34
PWM Disable (and Stop the Motor) for Motor Control 24–35
ADSP-2184x PWM Register Descriptions 24–35
Channel A Control Register 24–38
Channel A-High Duty-0 Register 24–40
Channel A-High Duty-1 Register 24–41
Channel A-High Full Duty0 Register 24–42
Channel A-High Full Duty1 Register 24–43
Channel A-Low Duty-0 Register 24–44
Channel A-Low Duty-1 Register 24–45
Channel A-Low Full Duty0 Register 24–46
Channel A-Low Full Duty1 Register 24–47
Channel B Control Register 24–48
Channel B-High Duty-0 Register 24–50
Channel B-High Duty-1 Register 24–51
Channel B-High Full Duty0 Register 24–52
Channel B-High Full Duty1 Register 24–53
Channel B-Low Duty-0 Register 24–54
Channel B-Low Duty-1 Register 24–55
Channel B-Low Full Duty0 Register 24–56
Channel B-Low Full Duty1 Register 24–57
Channel C Control Register 24–58

Channel C-High Pulse Duty Register 0 ..... **24–60**
Channel C-High Pulse Duty Register 1 ..... **24–61**
Channel Configuration Register **24–62**
Channel A Dead-time Register **24–68**
Channel B Dead-time Register **24–69**
Channel C Dead-time Register **24–70**
Channel D Dead-time Register **24–71**
Chop Configuration Register **24–72**
Channel C-High Full Duty0 Register **24–73**
Channel C-High Full Duty1 Register **24–74**
Channel C-Low Pulse Duty Register 0 ..... **24–75**
Channel C-Low Duty-1 Register **24–76**
Channel C-Low Full Duty0 Register **24–77**
Channel C-Low Full Duty1 Register **24–78**
Control Register **24–79**
Channel D Control Register **24–83**
Channel D-High Duty-0 Register **24–85**
Channel D-High Pulse Duty Register 1 ..... **24–86**
Channel D-High Full Duty0 Register **24–87**
Channel D-High Full Duty1 Register **24–88**
Channel D-Low Pulse Duty Register 0 ..... **24–89**
Channel D-Low Pulse Duty Register 1 ..... **24–90**
Channel A Delay Register **24–91**
Channel B Delay Register **24–92**
Channel C Delay Register **24–93**
Channel D Delay Register **24–94**
Channel D-Low Full Duty0 Register **24–95**
Channel D-Low Full Duty1 Register **24–96**
Interrupt Latch Register **24–97**
Interrupt Mask Register **24–99**

Status Register 24–101
Software Trip Register 24–106
Sync Pulse Width Register 24–108
Timer 0 Period Register 24–109
Timer 1 Period Register 24–110
Timer 2 Period Register 24–111
Timer 3 Period Register 24–112
Timer 4 Period Register 24–113
Trip Configuration Register 24–114
Trip Polarity Register 24–122

General-Purpose Counter (CNT)

GP Counter Features 25–1
GP Counter Functional Description 25–1
ADSP-2184x CNT Register List 25–2
ADSP-2184x CNT Interrupt List 25–2
ADSP-2184x CNT Trigger List 25–3
GP Counter Operating Modes 25–3
Quadrature Encoder Mode 25–3
Binary Encoder Mode 25–4
Up/Down Counter Mode 25–4
Direction Counter Mode 25–5
Timed Direction Mode 25–5
GP Counter Programming Model 25–5
GP Counter General Programming Flow 25–5
GP Counter Mode Configuration 25–5
Configuring GP Counter Push-Button Operation 25–6
Configuring Zero-Marker-Zeros-Counter Mode 25–6
Configuring Zero-Marker-Error Mode 25–6
Configuring Zero-Once Mode 25–6
Configuring Boundary Auto-Extend Mode 25–7

Configuring Boundary Capture Mode 25–7
Configuring Boundary Compare and Boundary Zero Modes 25–7
Configuring GP Counter Push-Button Operation 25–8
GP Counter Programming Concepts 25–8
CNT Input Noise Filtering 25–8
Capturing Counter Interval and CNT_CNTR Read Timing 25–9
Capturing Time Interval Between Successive Counter Events 25–10
GP Counter Event Control 25–11
Illegal Gray and Binary Code Events 25–11
Up/Down Count Events 25–11
Zero-Count Events 25–12
Overflow Events 25–12
Boundary Match Events 25–12
Zero Marker Events 25–12
ADSP-2184x CNT Register Descriptions 25–12
Configuration Register 25–14
Command Register 25–17
Counter Register 25–20
Debounce Register 25–21
Interrupt Mask Register 25–23
Maximum Count Register 25–26
Minimum Count Register 25–27
Status Register 25–28

Media Local Bus (MLB)

Features 26–1
MLB Definitions 26–2
Clocking 26–3
Functional Description 26–3
ADSP-2184x MLB Register List 26–3
ADSP-2184x MLB Interrupt List 26–4

MediaLB Protocol 26–5
MLB Architectural Concepts 26–6
MediaLB Block Diagram 26–6
MediaLB Interface 26–6
Routing Fabric 26–7
Data Buffer RAM 26–7
Channel Table RAM 26–7
Address Mapping 26–7
Channel Allocation Table 26–8
Channel Set Up 26–10
Channel Descriptor Tables 26–10
AHB Descriptor Table (ADT) 26–15
Interrupt Interface Block 26–19
Operating Modes 26–19
Isochronous Data Exchange 26–19
Asynchronous and Control Data Exchange 26–21
Synchronous Data Exchange 26–21
Data Transfer 26–22
DMA 26–22
Programming Model 26–22
Channel Initialization 26–22
Configure the Hardware 26–22
Program the CAT and the CDT 26–22
Program the ADT 26–24
Service 26–25
Servicing the DMA Channel Interrupts 26–25
Servicing the MediaLB Status Interrupts 26–25
Polling for MediaLB System Commands 26–26
ADSP-2184x MLB Register Descriptions 26–26
Peripheral Channel Mask 0 Register 26–28
Peripheral Channel Mask 1 Register 26–29

Peripheral Channel Status 0 Register 26–30
Peripheral Channel Status 1 Register 26–31
Bus Control Register 26–32
MediaLB Control 0 Register 26–34
Control 1 Register 26–36
MLB Global Control Register 26–37
HBI Channel Busy 0 Register 26–38
HBI Channel Busy 1 Register 26–39
HBI Channel Error 0 Register 26–40
HBI Channel Error 1 Register 26–41
HBI Channel Mask 0 Register 26–42
HBI Channel Mask 1 Register 26–43
HBI Control Register 26–44
Memory Interface Address Register 26–45
Memory Interface Control Register 26–46
Memory Interface Control Data 0 Register 26–47
Memory Interface Control Data 1 Register 26–48
Memory Interface Control Data 2 Register 26–49
Memory Interface Control Data 3 Register 26–50
Memory Interface Control Data Write Enable 0 Register 26–51
Memory Interface Control Data Write Enable 1 Register 26–52
Memory Interface Control Data Write Enable 2 Register 26–53
Memory Interface Control Data Write Enable 3 Register 26–54
Interrupt Enable Register 26–55
Channel Status 0 Register 26–58
Channel Status 1 Register 26–59
System Data Register 26–60
System Status Register 26–61

Two-Wire Interface (TWI)

TWI Features 27–1
TWI Functional Description 27–2
ADSP-2184x TWI Register List 27–2
ADSP-2184x TWI Interrupt List 27–3
TWI Block Diagram 27–3
External Interface 27–3
Serial Clock Signal (SCL) 27–4
Serial Data Signal (SDA) 27–4
Internal Interface 27–5
TWI Architectural Concepts 27–5
TWI Protocol 27–5
Clock Generation and Synchronization 27–6
Bus Arbitration 27–6
Start and Stop Conditions 27–7
General Call Support 27–7
Fast Mode 27–8
TWI Operating Modes 27–8
Repeated Start 27–8
Transmit Receive Repeated Start 27–8
Receive Transmit Repeated Start 27–9
Clock Stretching 27–10
Clock Stretching During FIFO Underflow 27–10
Clock Stretching During FIFO Overflow 27–11
Clock Stretching During Repeated Start 27–11
TWI Programming Model 27–12
General Setup 27–12
Target Mode 27–13
Controller Mode Program Flow 27–14
Controller Mode Clock Setup 27–15
Controller Mode Transmit 27–15
Controller Mode Receive 27–16

ADSP-2184x TWI Register Descriptions 27–17
SCL Clock Divider Register 27–18
Control Register 27–19
FIFO Control Register 27–21
FIFO Status Register 27–23
Interrupt Mask Register 27–24
Interrupt Status Register 27–26
Controller Mode Address Register 27–29
Controller Mode Control Registers 27–30
Controller Mode Status Register 27–33
Rx Data Double-Byte Register 27–36
Rx Data Single-Byte Register 27–37
Target Mode Address Register 27–38
Target Mode Control Register 27–39
Target Mode Status Register 27–41
Tx Data Double-Byte Register 27–42
Tx Data Single-Byte Register 27–43

Ethernet Media Access Controller (EMAC)

EMAC Features 28–1
EMAC Functional Description 28–6
ADSP-2184x EMAC Register List 28–8
ADSP-2184x EMAC Interrupt List 28–16
ADSP-2184x EMAC Trigger List 28–17
EMAC Definitions 28–19
EMAC Block Diagram and Interfaces 28–22
EMAC CORE Sub Blocks 28–25
EMAC PHY Interface 28–27
RGMII Board Design Recommendations 28–27
Clock Sources 28–29
EMAC Architectural Concepts 28–30

EMAC System Crossbar Interface (EMAC SCB) 28–30
Priority of SCB Requests 28–31
SCB Interface Programming Options 28–31
Burst Reordering and Data Interleaving 28–33
Posted Writes 28–33
SCB Bus Transaction Status 28–33
DMA Controller (EMAC DMA) 28–33
DMA Bursts Using the SCB Interface 28–35
Application Data Buffer Alignment 28–36
Buffer Size Calculations 28–36
DMA Transmit Operation 28–37
DMA Receive Operation 28–41
DMA Error Response 28–43
DMA Descriptors 28–43
MAC Transaction Layer (MTL) 28–71
Transmit Path and Operation 28–71
Receive Path and Operation 28–73
EMAC CORE 28–74
EMAC CORE Transmission Engine 28–75
EMAC Core Reception 28–78
Source Address, VLAN, CRC Insertion/Replacement 28–81
Double VLAN Processing 28–81
Source Address and VLAN Insert, Replace, or Delete 28–83
Queue/Channel Based VLAN Tag Insertion on Transmit 28–84
CRC Insert and Replace 28–84
Packet Filtering 28–84
Source Address or Destination Address Filtering 28–85
VLAN Filtering 28–88
Layer 3 and Layer 4 Frame Filtering 28–94
EMAC Station Management Interface (SMI) 28–96
MDC Clock Frequency 28–97
MDIO Frame Structure 28–98

SMI Write Operation 28–99
SMI Read Operation 28–100
Preamble Suppression 28–100
Trailing Clocks and Back-to-Back Transactions 28–100
Interrupt for MDIO Transaction Completion 28–101
EMAC Management Counters (MMC) 28–101
MMC Receive Interrupt Register 28–103
MMC Transmit Interrupt Register 28–103
MMC Receive Checksum Offload Interrupt Register 28–103
EMAC Precision Time Protocol (PTP) Engine 28–103
IEEE1588 and the PTP Engine 28–103
Block Diagram 28–107
PTP Module Clock 28–108
Time Stamp Module 28–109
System Time 28–118
Target Time Trigger (Alarm) 28–121
Pulse-Per-Second (PPS) 28–121
PTP Interrupts 28–124
Media Clock Generation and Recovery (MCGR) 28–124
PTP Time Stamp Offload Function 28–128
One Step Time Stamp 28–133
Multiple Channels and Queues Support 28–140
Multiple Queues Support in the Transmit Path 28–140
Fixed Priority Scheme 28–140
Multiple Queues in the Receive Path 28–141
Assigning Tag Priorities to Transmit and Receive Queues 28–144
Distribution of Receive Packets from MAC to Receive Queues 28–145
Traffic Shaper 28–147
TCP/IP Offloading Features 28–157
Transmit Checksum Offload Engine 28–158
Receive Checksum Offload Engine 28–161
TCP/IP Segmentation Offload (TSO) Engine 28–165

UDP/IPv4 Fragment Offload (UFO) Engine 28–171
IPv4 ARP Offload Engine 28–174
Energy Efficient Ethernet (EEE) Mode 28–175
Transmit Path Functions 28–176
Receive Path Functions 28–177
LPI Timers 28–179
Low Power Interrupt (LPI) 28–179
Time-Sensitive Networking 28–180
Enhancements to Scheduled Traffic (EST) 28–180
Gate Control List (GCL) Implementation 28–182
Frame Preemption (FPE) 28–191
Time-Based Scheduling 28–203
Flow Control 28–209
Transmit Flow Control 28–209
Receive Flow Control 28–212
Loopback Mode 28–214
Guidelines for Using Loopback Mode 28–214
Enabling Loopback Mode 28–214
Endian Support 28–215
EMAC Event Control 28–215
EMAC Interrupt Signals 28–215
Interrupts from the MAC 28–216
Interrupts from the MTL 28–216
Interrupts from DMA 28–217
Periodic Scheduling of Transmit and Receive Interrupt 28–217
Per Channel Transfer Complete Interrupt 28–218
EMAC Triggers 28–219
EMAC Programming Model 28–219
Initializing DMA 28–219
Initializing the MTL Registers 28–220
Initializing the MAC 28–221

Performing Normal Transmit and Receive Operations 28–222
Stopping and Starting Transmission 28–223
Switching to New Descriptor List in Receive DMA 28–223
Multi-Channel Multi-Queuing 28–223
Recovering from DMA Channel Failure 28–225
Disabling the Receive Queue 28–225
IEEE 1588 Time Stamping (PTP) 28–226
Initialization Guidelines for System Time Generation 28–226
System Time Correction 28–226
Coarse Correction Method 28–227
Fine Correction Method 28–227
PTP Offloading 28–227
Automatic Periodic Generation of PTP Sync Messages 28–227
Periodic Generation of PTP Pdelay_Req Messages 28–228
Response Generation - Ordinary or Boundary Requester Mode 28–228
Response Generation - Ordinary or Boundary Completer Mode 28–229
Response Generation - Transparent Completer Mode 28–229
Response Generation - Transparent Requester Mode 28–230
Response Generation - Peer-to-Peer Transparent Mode 28–230
Using the Audio Video Feature 28–231
Initializing the DMA in Audio Video Feature 28–231
Enabling Slot Number Checking 28–232
Disabling Flow Control for AV Enabled Queues 28–232
Programming Energy Efficient Ethernet 28–233
Entering and Exiting the Transmit LPI Mode 28–233
Programming Flexible Pulse-Per-Second Output 28–234
Generating Single Pulse on PPS 28–234
Generating Next Pulse on PPS 28–234
Generating a Pulse Train on PPS 28–234
Programming TSO 28–235
Programming UFO 28–236

Programming Header Payload Split Receive 28–236
Programming VLAN Filtering on Receive 28–237
Programming Extended VLAN Filtering and Routing on Receive 28–238
Programming Queue/Channel Based VLAN Inclusion Registers 28–238
Writing to the Indirect Registers 28–238
Reading the Indirect Registers 28–239
Programming EST 28–239
GCL and GCL Linked Registers 28–239
EST Registers 28–240
Programming Launch Time in Time-Based Scheduling 28–241
Programming Media Clock Generation and Recovery 28–241
Media Clock Generation 28–241
Media Clock Recovery 28–241
EMAC Programming Concepts 28–242
IEEE 802.3 Ethernet Packet Structure 28–242
Frame Size Statistics for Application Software 28–243
Software Visualization of Programmable Packet Size 28–243
Ethernet Packet Structure in C 28–243
PTP Header Structure in C 28–243
ADSP-2184x EMAC Register Descriptions 28–244
DMA Channel n Control Register 28–254
DMA Channel n Current Application Receive Buffer Address Register 28–256
DMA Channel n Current Application Receive Descriptor Register 28–257
DMA Channel n Current Application Transmit Buffer Address Register 28–258
DMA Channel n Current Application Transmit Descriptor Register 28–259
DMA Channel n Interrupt Enable Register 28–260
DMA Channel n Missed Frame Counter Register 28–264
DMA Channel n Receive Descriptor List Address Register 28–265
DMA Channel n Receive Descriptor Tail Pointer Register 28–266
DMA Channel n Receive Control Register 28–267

DMA Channel n Extended Receive Control Register 28–270
DMA Channel n Receive Interrupt Watchdog Timer Register 28–271
DMA Channel n Slot Function Control and Status register 28–272
DMA Channel n Status Register 28–274
DMA Channel n Transmit Descriptor List Address Register 28–281
DMA Channel n Transmit Descriptor Ring Length Register 28–282
DMA Channel n Transmit Descriptor Tail Pointer Register 28–283
DMA Channel n Transmit Control Register 28–284
DMA Debug Status 0 Register 28–288
DMA Debug Status 1 Register 28–292
DMA Debug Status 2 Register 28–296
DMA Interrupt Status Register 28–298
DMA Mode Register 28–301
DMA System Bus Mode Register 28–304
DMA TBS Control 0 Register 28–307
DMA TBS Control 1 Register 28–308
DMA TBS Control 2 Register 28–309
DMA TBS Control 3 Register 28–310
LPI One Microsecond Tic Counter Register 28–311
Address Resolution Protocol (ARP) Register 28–312
MAC Address 0 High Register 28–313
MAC Address 0 Low Register 28–314
MAC Address n High Register 28–315
MAC Address n Low Register 28–317
Auxiliary Time Stamp Control Register 28–318
Auxiliary Time Stamp Nanoseconds Register 28–320
Auxiliary Time Stamp Seconds Register 28–321
CSR Software Control Register 28–322
MAC Configuration Register 28–323
Debug Register 28–332

MAC Split Mode and Offset Configuration Register 28–334
MAC Extended Configuration Register 28–336
Frame Preemption Control and Status Register 28–340
Hardware Feature Presence Register 0 28–343
Hardware Feature Presence Register 1 28–347
Hardware Feature Presence Register 2 28–352
Hardware Feature Presence Register 3 28–355
MAC Hardware Feature Presence Register 4 28–359
Hash Table Register 0 28–360
Hash Table Register 1 28–361
Hash Table Register 2 28–362
Hash Table Register 3 28–363
Hash Table Register 4 28–364
Hash Table Register 5 28–365
Hash Table Register 6 28–366
Hash Table Register 7 28–367
Indirect Access Control Register 28–368
Indirect Access Data Register 28–370
Inner VLAN Tag Inclusion or Replacement Register 28–371
MAC Interrupt Enable Register 28–373
MAC Interrupt Status Register 28–375
Layer 3 and Layer 4 Control Register 28–380
Layer 3 and Layer 4 Control Register 28–385
LPI Control and Status Register 28–390
Transmit LPI Entry Timer Register 28–394
LPI Timers Control Register 28–395
Layer 3 Address 0 Register 7 28–396
Layer 3 Address 0 Register [i] 28–397
Layer 3 Address 1 Register 7 28–398
Layer 3 Address 1 Register [i] 28–399

Layer 3 Address 2 Register 7 28–400
Layer 3 Address 2 Register [i] 28–401
Layer 3 Address 3 Register 7 28–402
Layer 3 Address 3 Register [i] 28–403
Layer 4 Address Register [i] 28–404
Layer 4 Address Register [i] 28–405
PTP Log Message Interval Register 28–406
MDIO Address Register 28–408
MDIO Data Register 28–413
PHY Interface Control and Status Register 28–414
PPS n Interval Register 28–416
PPS n Target Time Nanoseconds Register 28–417
PPS n Target Time Seconds Register 28–418
PPS n Width Register 28–419
PPS Control Register 28–420
PTP Time Stamp Offload Control Register 28–426
MAC Packet Filter Register 28–429
Presentation Time Update Register 28–434
Presentation Time Nanosecond Register 28–435
Queue 0 Transmit Flow Control Register 28–436
Queue [i] Transmit Flow Control Register 28–439
Receive Queue Control 0 Register 28–442
Receive Queue Control 1 Register 28–445
Receive Queue Control 2 Register 28–449
Receive Queue Control 3 Register 28–451
Receive Queue Control 4 Register 28–453
MAC Receive Time Domain Increment Register 28–456
Receive Flow Control Register 28–457
Receive Transmit Status Register 28–459
Source Port Identity 0 Register 28–462

Source Port Identity 1 Register **28–463**
Source Port Identity 2 Register **28–464**
Subsecond Increment Register **28–465**
System Time Higher Word Seconds Register **28–466**
System Time Nanoseconds Register **28–467**
System Time Nanoseconds Update Register **28–468**
System Time Seconds Register **28–470**
System Time Seconds Update Register **28–471**
Time Stamp Addend Register **28–472**
Time Stamp Control Register **28–473**
Time Stamp Egress Asymmetry Correction Register **28–480**
Time Stamp Egress Correction Nanosecond Register **28–481**
Time Stamp Egress Correction Sub Nanosecond Register **28–482**
Time Stamp Egress Latency Register **28–483**
Time Stamp Ingress Asymmetry Correction Register **28–484**
Time Stamp Ingress Correction Nanosecond Register **28–485**
Time Stamp Ingress Correction Sub Nanosecond Register **28–486**
Time Stamp Ingress Latency Register **28–487**
Time Stamp Status Register **28–488**
Transmit Queue Priority Mapping 0 Register **28–494**
Transmit Queue Priority Mapping 1 Register **28–495**
MAC Transmit Time Domain Increment Register **28–496**
Transmit Time Stamp Status Nanoseconds Register **28–497**
Transmit Time Stamp Status Seconds Register **28–498**
VLAN Hash Table Register **28–499**
VLAN Tag Inclusion or Replacement Register **28–500**
VLAN Tag Control Register **28–503**
VLAN Tag Indirect Access Data Register **28–508**
Watchdog and Jabber Timeout Register **28–511**
MMC Control Register **28–514**

MMC Receive FPE Interrupt Status Register 28–517
MMC Receive FPE Interrupt Mask Register 28–519
MMC Transmit FPE Interrupt Status Register 28–521
MMC Transmit FPE Interrupt Mask Register 28–523
MMC Receive IPC Interrupt Status Register 28–524
MMC Receive IPC Interrupt Mask Register 28–533
MMC Receive FPE Fragment Counter Register 28–542
MMC Receive Interrupt Status Register 28–543
MMC Receive Interrupt Mask Register 28–552
MMC Receive FPE Packet Assembly Error Counter Register 28–561
MMC Receive FPE Packet Assembly Okay Counter Register 28–562
MMC Receive FPE Packet SMD Error Counter Register 28–563
MMC Transmit FPE Packet Fragment Counter Register 28–564
MMC Transmit FPE Hold Request Counter Register 28–565
MMC Transmit Interrupt Status Register 28–566
MMC Transmit Interrupt Mask Register 28–575
EST Control Register 28–584
EST Extended Control Register 28–587
EST Frame Size Capture Register 28–588
EST Frame Size Error Register 28–589
EST Gate Control List Control Register 28–590
EST Gate Control List Data Register 28–593
EST Interrupt Enable Register 28–594
EST Scheduling Error Register 28–596
EST Status Register 28–597
Frame Preemption Advance Register 28–600
MTL Frame Preemption Control and Status Register 28–601
MTL Interrupt Status Register 28–603
MTL Operation Mode Register 28–606
Queue 0 Interrupt Control and Status Register 28–608

Queue n Interrupt Control and Status Register 28–610
Receive Queue Control Register 28–612
Receive Queue 0 Debug Register 28–614
Receive Queue 0 Missed Packet and Overflow Counter Register 28–616
Receive Queue 0 Operation Mode Register 28–618
Receive Queue n Control Register 28–622
Receive Queue n Debug Register 28–624
Receive Queue n Missed Packet and Overflow Counter Register 28–626
Receive Queue n Operation Mode Register 28–628
Receive Queue to DMA Channel Mapping Register 0 28–632
Receive Queue to DMA Channel Mapping Register 1 28–636
MTL Time Based Scheduling Control Register 28–640
Transmit Queue 0 Debug Register 28–642
Transmit Queue 0 ETS Status Register 28–644
Transmit Queue 0 Operation Mode Register 28–645
Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register 28–647
Transmit Queue 0 Underflow Packet Counter Register 28–648
Transmit Queue n Debug Register 28–649
Transmit Queue n ETS Control Register 28–651
Transmit Queue n ETS Status Register 28–653
Transmit Queue n CBS HiCredit Register 28–654
Transmit Queue n CBS LoCredit Register 28–655
Transmit Queue n Operation Mode Register 28–656
Transmit Queue n IdleSlopeCredit, Quantum or Weights Register 28–658
Transmit Queue n SendSlopeCredit Register 28–660
Transmit Queue n Underflow Packet Counter Register 28–661
Receive ICMP Error Octets Counter Register 28–662
Receive ICMP Error Packets Counter Register 28–663
Receive ICMP Good Octets Counter Register 28–664
Receive ICMP Good Packets Counter Register 28–665

Receive IPv4 Fragmented Octets Counter Register **28–666**
Receive IPv4 Fragmented Packets Counter Register **28–667**
Receive IPv4 Good Octets Counter Register **28–668**
Receive IPv4 Good Packets Counter Register **28–669**
Receive IPv4 Header Error Octets Counter Register **28–670**
Receive IPv4 Header Error Packets Counter Register **28–671**
Receive IPv4 Payload Octets Counter Register **28–672**
Receive IPv4 Payload Packets Counter Register **28–673**
Receive IPv4 UDP Checksum Disable Octets Counter Register **28–674**
Receive IPv4 UDP Checksum Disable Packets Counter Register **28–675**
Receive IPv6 Good Octets Counter Register **28–676**
Receive IPv6 Good Packets Counter Register **28–677**
Receive IPv6 Header Error Octets Counter Register **28–678**
Receive IPv6 Header Error Packets Counter Register **28–679**
Receive IPv6 Payload Octets Counter Register **28–680**
Receive IPv6 Payload Packets Counter Register **28–681**
Receive TCP Error Octets Counter Register **28–682**
Receive TCP Error Packets Counter Register **28–683**
Receive TCP Good Octets Counter Register **28–684**
Receive TCP Good Packets Counter Register **28–685**
Receive UDP Error Octets Counter Register **28–686**
Receive UDP Error Packets Counter Register **28–687**
Receive UDP Good Octets Counter Register **28–688**
Receive UDP Good Packets Counter Register **28–689**
Receive 1024 - Maximum Octets Packets Good Bad Counter Register **28–690**
Receive 128 - 255 Octets Packets Good Bad Counter Register **28–691**
Receive 256 - 511 Octets Packets Good Bad Counter Register **28–692**
Receive 512 - 1023 Octets Packets Good Bad Counter Register **28–693**
Receive 64 Octets Packets Good Bad Counter Register **28–694**
Receive 65 - 127 Octets Packets Good Bad Counter Register **28–695**

Receive Alignment Error Packets Counter Register **28–696**
Receive Broadcast Packets Good Counter Register **28–697**
Receive CRC Error Packets Counter Register **28–698**
Receive Control Packets Good Counter Register **28–699**
Receive FIFO Overflow Packets Counter Register **28–700**
Receive Jabber Error Packets Counter Register **28–701**
Receive LPI Transition Counter Register **28–702**
Receive LPI Microseconds Counter Register **28–703**
Receive Length Error Packets Counter Register **28–704**
Receive Multicast Packets Good Counter Register **28–705**
Receive Octet Good Counter Register **28–706**
Receive Octet Good Bad Counter Register **28–707**
Receive Out of Range Type Packets Counter Register **28–708**
Receive Oversize Packets Good Counter Register **28–709**
Receive Packets Good Bad Counter Register **28–710**
Receive Pause Packets Counter Register **28–711**
Receive Error Packets Counter Register **28–712**
Receive Runt Error Packets Counter Register **28–713**
Receive Undersize Packets Good Counter Register **28–714**
Receive Unicast Packets Good Counter Register **28–715**
Receive VLAN Packets Good Bad Counter Register **28–716**
Receive Watchdog Error Packets Counter Register **28–717**
Transmit 1024-Max Octets Packets Good Bad Counter Register **28–718**
Transmit 128-255 Octets Packets Good Bad Counter Register **28–719**
Transmit 256-511 Octets Packets Good Bad Counter Register **28–720**
Transmit 512-1023 Octets Packets Good Bad Counter Register **28–721**
Transmit 64 Octets Packets Good Bad Counter Register **28–722**
Transmit 65-127 Octets Packets Good Bad Counter Register **28–723**
Transmit Broadcast Packets Good Counter Register **28–724**
Transmit Broadcast Packets Good Bad Counter Register **28–725**

Transmit Carrier Error Packets Counter Register 28–726
Transmit Deferred Packets Counter Register 28–727
Transmit Excessive Collision Packets Counter Register 28–728
Transmit Excessive Deferral Error Counter Register 28–729
Transmit LPI Transition Counter Register 28–730
Transmit LPI Microseconds Counter Register 28–731
Transmit Late Collision Packets Counter Register 28–732
Transmit Multicast Packets Good Counter Register 28–733
Transmit Multicast Packets Good Bad Counter Register 28–734
Transmit Multiple Collision Packets Good Counter Register 28–735
Transmit Osize Packets Good Counter Register 28–736
Transmit Octet Good Counter Register 28–737
Transmit Octet Good Bad Counter Register 28–738
Transmit Packet Good Counter Register 28–739
Transmit Packet Good Bad Counter Register 28–740
Transmit Pause Packets Counter Register 28–741
Transmit Single Collision Packets Good Counter Register 28–742
Transmit Underflow Error Packets Counter Register 28–743
Transmit Unicast Packets Good Bad Counter Register 28–744
Transmit VLAN Packets Good Counter Register 28–745

Digital Audio Interface (DAI)

SRU Features 29–1
Functional Description 29–2
ADSP-2184x DAI Register List 29–2
ADSP-2184x DAI Interrupt List 29–5
ADSP-2184x DAI Trigger List 29–6
DAI Block Diagram 29–6
DAI Signal Naming Conventions 29–7
I/O Pin Buffers 29–8

Pin Buffer Signals 29–8
Pin Buffer Input Signal 29–8
Pin Buffer Enable Signal 29–8
Pin Buffer Functions 29–9
Pin Buffers as Signal Input 29–9
Pin Buffers As Signal Output 29–9
DAI Pin Buffer Status 29–10
DAIn Peripherals 29–10
Output Signals With Pin Buffer Enable Control 29–10
Output Signals Without Pin Buffer Enable Control 29–10
Signal Routing Units (SRUs) 29–10
Signal Routing Matrix by Groups 29–11
DAI Group Routing 29–11
Rules for SRU Connections 29–12
Miscellaneous Buffers and Functions 29–13
DAI Routing Unit (DRU) 29–13
DAI Routing Capabilities 29–14
DAI Default Routing 29–17
Unused DAI Connections 29–19
DAI Operating Modes 29–19
DAI Pin Buffer Polarity 29–20
DAI Miscellaneous Buffer Polarity 29–20
DAI System Interrupt Controller (SIC) 29–20
Signal Routing Unit Effect Latency 29–23
DAI Programming Model 29–23
Debug Features 29–23
DAI Sources Overview 29–23
Group A – Clock Routing Signals 29–24
Group B – Serial Data Source Signals 29–29
Group C – Frame Sync Source Signals 29–33
Group D – Pin Signal Assignment Source Signals 29–36

Group E – Miscellaneous Source Signals 29–44
Group F – Pin Output Enable Source Signals 29–47
DAI Destination Registers Overview 29–52
SPORT Grouping with DAI 29–60
Global PCG Enable 29–60
ADSP-2184x DAI Register Descriptions 29–60
Clock Routing Control Register 0 29–64
Clock Routing Control Register 1 29–66
Clock Routing Control Register 2 29–68
Clock Routing Control Register 3 29–69
Clock Routing Control Register 4 29–70
Clock Routing Control Register 5 29–71
Clock Routing Control Register 6 29–73
Clock Routing Control Register 7 29–75
DAI MISC Control Register 29–76
Serial Data Routing Control Register 0 29–77
Serial Data Routing Control Register 1 29–78
Serial Data Routing Control Register 2 29–79
Serial Data Routing Control Register 3 29–80
Serial Data Routing Control Register 4 29–81
Serial Data Routing Control Register 5 29–82
Serial Data Routing Control Register 6 29–83
Serial Data Routing Control Register 7 29–84
Serial Data Routing Control Register 8 29–85
Extended Clock Routing Control Register 0 29–86
Extended Clock Routing Control Register 1 29–88
Extended Clock Routing Control Register 2 29–90
Extended Clock Routing Control Register 3 29–91
Extended Clock Routing Control Register 4 29–92
Extended Clock Routing Control Register 5 29–93

Extended Clock Routing Control Register 6 ..... **29–95**
Extended Clock Routing Control Register 7 ..... **29–97**
Extended Serial Data Routing Control Register 0 ..... **29–98**
Extended Serial Data Routing Control Register 1 ..... **29–99**
Extended Serial Data Routing Control Register 2 ..... **29–100**
Extended Serial Data Routing Control Register 3 ..... **29–101**
Extended Serial Data Routing Control Register 4 ..... **29–102**
Extended Serial Data Routing Control Register 5 ..... **29–103**
Extended Serial Data Routing Control Register 6 ..... **29–104**
Extended Serial Data Routing Control Register 7 ..... **29–105**
Extended Serial Data Routing Control Register 8 ..... **29–106**
Extended Frame Sync Routing Control Register 0 ..... **29–107**
Extended Frame Sync Routing Control Register 1 ..... **29–109**
Extended Frame Sync Routing Control Register 2 ..... **29–111**
Extended Frame Sync Routing Control Register 4 ..... **29–112**
Extended Frame Sync Routing Control Register 5 ..... **29–113**
Extended Frame Sync Routing Control Register 6 ..... **29–115**
Extended Miscellaneous Control Register 0 ..... **29–116**
Extended Miscellaneous Control Register 1 ..... **29–118**
Extended Miscellaneous Control Register 2 ..... **29–119**
Extended Pin Buffer Enable Register 0 ..... **29–120**
Extended Pin Buffer Enable Register 1 ..... **29–121**
Extended Pin Buffer Enable Register 2 ..... **29–122**
Extended Pin Buffer Enable Register 3 ..... **29–123**
Extended Pin Buffer Assignment Register 0 ..... **29–124**
Extended Pin Buffer Assignment Register 1 ..... **29–125**
Extended Pin Buffer Assignment Register 2 ..... **29–126**
Extended Pin Buffer Assignment Register 3 ..... **29–127**
Extended Pin Buffer Assignment Register 4 ..... **29–128**
Frame Sync Routing Control Register 0 ..... **29–130**

Frame Sync Routing Control Register 1 ..... **29–132**
Frame Sync Routing Control Register 2 ..... **29–134**
Frame Sync Routing Control Register 4 ..... **29–135**
Frame Sync Routing Control Register 5 ..... **29–136**
Frame Sync Routing Control Register 6 ..... **29–138**
Global SPORT Interrupt Grouping Register **29–139**
Global PCG Enable Control Register **29–142**
Global SPORT Enable Register **29–145**
Falling-Edge Interrupt Mask Register **29–148**
Core Interrupt Priority Assignment Register **29–152**
Rising-Edge Interrupt Mask Register **29–156**
High Priority Interrupt Latch Register **29–160**
Shadow High Priority Interrupt Latch Register **29–164**
Low Priority Interrupt Latch Register **29–169**
Shadow Low Priority Interrupt Latch Register **29–173**
Miscellaneous Control Register 0 ..... **29–178**
Miscellaneous Control Register 1 ..... **29–180**
Miscellaneous Control Register 1 ..... **29–182**
Pin Buffer Enable Register 0 ..... **29–183**
Pin Buffer Enable Register 1 ..... **29–184**
Pin Buffer Enable Register 2 ..... **29–185**
Pin Buffer Enable Register 3 ..... **29–186**
Pin Buffer Assignment Register 0 ..... **29–187**
Pin Buffer Assignment Register 1 ..... **29–188**
Pin Buffer Assignment Register 2 ..... **29–189**
Pin Buffer Assignment Register 3 ..... **29–190**
Pin Buffer Assignment Register 4 ..... **29–191**
Dummy Pin Buffer Assignment Register **29–193**
Pin Status Register **29–194**

Serial Port (SPORT)

Features 30–1
Signal Descriptions 30–3
SRU Programming 30–6
Functional Description 30–6
ADSP-2184x SPORT Register List 30–6
ADSP-2184x SPORT Interrupt List 30–7
ADSP-2184x SPORT Trigger List 30–9
ADSP-2184x SPORT DMA Channel List 30–10
Block Diagram 30–11
Architectural Concepts 30–12
Multiplexer Logic 30–13
Data Types and Companding 30–16
Companding as a Function 30–17
Transmit Path 30–17
Receive Path 30–18
Operating Modes and Options 30–19
Serial Word Length 30–21
Clock Sample and Drive Edges 30–21
Frame Sync Options 30–22
Data-Dependent versus Data-Independent Frame Syncs 30–22
Support for Edge-Detected and Level-Sensitive Frame Syncs 30–23
Early versus Late Frame Syncs 30–24
Framed versus Unframed Frame Syncs 30–25
Frame Sync Polarity 30–25
Premature Frame Sync Error Detection 30–26
Mode Selection 30–26
Standard DSP Serial Mode 30–27
Stereo Modes 30–28
I^2S Mode 30–28

Left-Justified Mode 30–29
Right-Justified Mode 30–30
Multichannel (TDM) Mode 30–32
Packed I^2S Mode 30–36
Gated Clock Mode 30–37
Data Transfers and Interrupts 30–38
Data Buffers 30–38
Data Buffer Status 30–40
Single-Word (Core) Transfers 30–40
DMA Transfers 30–41
Data Transfer Interrupt 30–42
Error Detection (Status) Interrupt 30–43
Grouping of SPORTs 30–44
SPORT Programming Model 30–46
Initializing Core-Driven (Non-MCM) Transfers 30–46
Initializing Multichannel Transfers 30–47
Using DMA for SPORT Transfers 30–49
Using Companding as a Function 30–49
Programming Global SPORT Groups 30–50
Disabling Global SPORT Groups 30–51
ADSP-2184x SPORT Register Descriptions 30–51
Half SPORT 'A' Multichannel 0-31 Select Register 30–53
Half SPORT 'B' Multichannel 0-31 Select Register 30–54
Half SPORT 'A' Multichannel 32-63 Select Register 30–55
Half SPORT 'B' Multichannel 32-63 Select Register 30–56
Half SPORT 'A' Multichannel 64-95 Select Register 30–57
Half SPORT 'B' Multichannel 64-95 Select Register 30–58
Half SPORT 'A' Multichannel 96-127 Select Register 30–59
Half SPORT 'B' Multichannel 96-127 Select Register 30–60
Half SPORT 'A' Control 2 Register 30–61

Half SPORT 'B' Control 2 Register 30–62
Half SPORT 'A' Control Register 30–63
Half SPORT 'B' Control Register 30–71
Half SPORT 'A' Divisor Register 30–80
Half SPORT 'B' Divisor Register 30–82
Half SPORT 'A' Error Register 30–84
Half SPORT 'B' Error Register 30–86
Half SPORT 'A' Multichannel Control Register 30–88
Half SPORT 'B' Multichannel Control Register 30–90
Half SPORT 'A' Multichannel Status Register 30–92
Half SPORT 'B' Multichannel Status Register 30–93
Half SPORT 'A' Rx Buffer (Primary) Register 30–94
Half SPORT 'B' Rx Buffer (Primary) Register 30–95
Half SPORT 'A' Rx Buffer (Secondary) Register 30–96
Half SPORT 'B' Rx Buffer (Secondary) Register 30–97
Half SPORT 'A' Tx Buffer (Primary) Register 30–98
Half SPORT 'B' Tx Buffer (Primary) Register 30–99
Half SPORT 'A' Tx Buffer (Secondary) Register 30–100
Half SPORT 'B' Tx Buffer (Secondary) Register 30–101

Precision Clock Generators (PCG)

Features 31–1
Functional Description 31–2
ADSP-2184x PCG Register List 31–2
ADSP-2184x PCG Trigger List 31–3
Internal Interface 31–4
Serial Clock 31–4
Frame Sync 31–4
Frame Sync Output 31–4
Divider Mode Selection 31–5
Phase Shift 31–5

Pulse Width 31–6
Default Pulse Width 31–6
Input Clock Source Considerations 31–7
Timing Example for I²S Mode 31–7
Cross Mode Connections 31–7
Operating Modes 31–7
Normal Mode 31–7
Bypass Mode 31–8
One-Shot Mode 31–8
Audio System Example 31–9
Hardware Trigger Control 31–11
Clock Configuration Examples 31–11
Global PCG Enable 31–12
PCG Event Control 31–12
External Event Trigger 31–12
External Event Trigger Delay 31–13
Programming Model 31–13
Frame Sync Phase Setting 31–14
External Event Trigger 31–14
Debug Features 31–14
ADSP-2184x PCG Register Descriptions 31–14
Precision Clock A Control 0 Register 31–16
Precision Clock A Control 1 Register 31–17
Precision Clock B Control 0 Register 31–18
Precision Clock B Control 1 Register 31–19
Precision Clock C Control 0 Register 31–20
Precision Clock C Control 1 Register 31–21
Precision Clock D Control 0 Register 31–22
Precision Clock D Control 1 Register 31–23
Precision Clock E Control 0 Register 31–24

Precision Clock E Control 1 Register 31–25
Precision Clock F Control 0 Register 31–26
Precision Clock F Control 1 Register 31–27
Precision Clock G Control 0 Register 31–28
Precision Clock G Control 1 Register 31–29
Precision Clock H Control 0 Register 31–30
Precision Clock H Control 1 Register 31–31
Precision Clock Pulse Width Control 1 Register 31–32
Precision Clock Pulse Width Control 2 Register 31–34
Precision Clock Pulse Width Control 3 Register 31–36
Precision Clock Pulse Width Control 4 Register 31–38
Precision Clock Frame Sync Synchronization 1 Register 31–40
Precision Clock Frame Sync Synchronization 2 Register 31–45
Precision Clock Frame Sync Synchronization 3 Register 31–48
Precision Clock Frame Sync Synchronization 4 Register 31–53

Asynchronous Sample Rate Converter (ASRC)

Features 32–1
Functional Description 32–2
ADSP-2184x ASRC Register List 32–2
ASRC Interrupt List 32–2
ASRC Block Diagram 32–2
SRU Programming 32–3
Clocking 32–3
I/O Ports 32–4
De-Emphasis Filter 32–4
Mute Control 32–4
SRC Core 32–4
RAM FIFO 32–4
Digital Servo Loop 32–5
FIR Filter 32–5

Sample Rate Sensing 32–5
Digital Filter Group Delay 32–5
Data Format 32–6
Operating Modes 32–7
TDM Input Mode 32–7
TDM Output Mode 32–7
Matched-Phase Mode 32–8
Bypass Mode 32–9
De-Emphasis Mode 32–9
Dithering Mode 32–9
Muting Modes 32–10
Soft Mute 32–10
Hard Mute 32–10
Auto Mute 32–10
Interrupts 32–10
Sources 32–10
SRC Mute Out 32–11
Masking 32–11
Service 32–11
Programming Model 32–11
Debug Features 32–11
ADSP-2184x ASRC Register Descriptions 32–11
Control Register for ASRC 0 and 1 32–13
Control Register for ASRC 2 and 3 32–18
Control Register For ASRC 4 And 5 32–23
Control Register for ASRC 6 and 7 32–28
Mute Register 32–33
Ratio Register for ASRC 0 and 1 32–35
Ratio Register for ASRC 2 and 3 32–37
Ratio Register For ASRC 4 and 5 32–39

Ratio Register for ASRC 6 and 7 32–41

Sony/Philips Digital Interface (S/PDIF)

Features 33–1
ADSP-2184x SPDIF Register List 33–1
SRU Programming 33–3
S/PDIF Interrupt List 33–3
Clocking 33–3
S/PDIF Transmitter 33–3
Functional Description 33–3
Input Data Formats 33–5
Operating Modes 33–6
Full Serial Mode 33–6
Standalone Mode 33–6
Data Output Mode 33–7
S/PDIF Receiver 33–7
Functional Description 33–8
Clock Recovery 33–8
Output Data Format 33–9
Channel Status 33–9
Operating Modes 33–9
Compressed or Non-linear Audio Data 33–9
Emphasized Audio Data 33–10
Single-Channel Double-Frequency Mode 33–10
Clock Recovery Modes 33–10
Interrupts 33–11
Sources 33–11
Transmit Block Start 33–11
Receiver Status 33–11
Receiver Error 33–11
Masking 33–11

Service 33–11
Programming Model 33–11
Programming the Transmitter 33–11
Programming the Receiver 33–12
Interrupted Data Streams on the Receiver 33–12
Debug Features 33–13
Loopback Routing 33–13
ADSP-2184x SPDIF Register Descriptions 33–13
Receive Control 33–15
Receive Status Register 33–17
Receive Status A0 Register 33–20
Receive Status B0 Register 33–21
Receive Status A1 Register 33–22
Receive Status B1 Register 33–23
Transmit Control Register 33–24
Transmit Status A0 Register 33–27
Transmit Status A1 Register 33–28
Transmit Status A2 Register 33–29
Transmit Status A3 Register 33–30
Transmit Status A4 Register 33–31
Transmit Status A5 Register 33–32
Transmit Status B0 Register 33–33
Transmit Status B1 Register 33–34
Transmit Status B2 Register 33–35
Transmit Status B3 Register 33–36
Transmit Status B4 Register 33–37
Transmit Status B5 Register 33–38
Transmit User Buffer A0 Register 33–39
Transmit User Buffer A1 Register 33–40
Transmit User Buffer A2 Register 33–41

Transmit User Buffer A3 Register 33–42
Transmit User Buffer A4 Register 33–43
Transmit User Buffer A5 Register 33–44
Transmit User Buffer B0 Register 33–45
Transmit User Buffer B1 Register 33–46
Transmit User Buffer B2 Register 33–47
Transmit User Buffer B3 Register 33–48
Transmit User Buffer B4 Register 33–49
Transmit User Buffer B5 Register 33–50
User Bit Update Register 33–51

Direct Memory Access (DMA)

DMA Channel Features 34–1
DMA Channel Functional Description 34–3
ADSP-2184x DMA Register List 34–3
ADSP-2184x DMA Channel List 34–4
DMA Definitions 34–5
Block Diagram 34–7
Architectural Concepts 34–9
DMA Channel SCB Interface 34–9
SCB Interface Signals 34–9
SCB Burst Transfers 34–9
Data Address Alignment 34–10
Descriptor Set Address Alignment 34–11
Peripheral Control Commands 34–11
Idle Command 34–11
Request-Data Command 34–12
Request-Data Urgent Command 34–12
Peripheral-Control Command Restrictions 34–12
DMA Channel Peripheral DMA Bus 34–12
Memory DMA and Triggering 34–12

Medium Band Width DMA Channel MMR Access Bus 34–15
DMA Channel Operation Flow 34–15
Startup Flow 34–15
Refresh Flow 34–17
Work Unit Transition Flow 34–18
Transfer Termination and Shutdown Flow 34–20
DMA Channel Errors 34–22
Status and Debug Errors 34–22
DMA Configuration Register Errors 34–22
Illegal Register Write During Run 34–23
Address Alignment Error 34–23
Memory Access Error 34–23
Trigger Overrun Error 34–24
Bandwidth-Monitor Error 34–24
Control Interface Error 34–24
DMA Operating Modes 34–24
Register-Based Flow Modes 34–24
Stop Mode 34–25
Autobuffer Mode 34–25
Descriptor-Based Flow Modes 34–25
Descriptor-Array Mode 34–26
Descriptor-List Mode 34–26
Descriptor-On-Demand Modes 34–27
Data Transfer Modes 34–28
Two-Dimensional DMA 34–28
DMA Channel Event Control 34–29
Event Signals 34–30
Work Unit State Events 34–30
Peripheral Interrupt Request Events 34–31
Peripheral Data Request Events 34–31
DMA Channel Triggers 34–32

Issuing Triggers 34–32
Waiting For Triggers 34–32
DMA Channel Programming Model 34–33
Mode Configuration 34–34
Register-Based Linear-Buffer Stop Flow Mode 34–34
Register-Based Autobuffer Flow Mode 34–35
Descriptor-Array Flow Mode 34–36
Descriptor-List Flow Mode 34–37
Register-Based Memory-to-Memory Transfer in Stop Flow Mode 34–38
Programming Concepts 34–39
Synchronization of Software and DMA 34–39
Interrupt and Trigger Event-Based Synchronization 34–40
Register Polling Based Synchronization 34–40
Descriptor Queues 34–41
Queues Using Event Generation for Every Descriptor Set 34–41
Queues Using Minimal Events 34–42
ADSP-2184x DMA Register Descriptions 34–43
Start Address of Current Buffer Register 34–44
Current Address Register 34–45
Bandwidth Limit Count Register 34–46
Bandwidth Limit Count Current Register 34–47
Bandwidth Monitor Count Register 34–48
Bandwidth Monitor Count Current Register 34–49
Configuration Register 34–50
Current Descriptor Pointer Register 34–58
Pointer to Next Initial Descriptor Register 34–59
Previous Initial Descriptor Pointer Register 34–60
Status Register 34–61
Inner Loop Count Start Value Register 34–64
Current Count (1D) or Intra-row XCNT (2D) Register 34–65
Inner Loop Address Increment Register 34–66

Outer Loop Count Start Value (2D only) Register 34–67
Current Row Count (2D only) Register 34–68
Outer Loop Address Increment (2D only) Register 34–69

Extended Memory DMA (EMDMA)

EMDMA Features 35–1
EMDMA Functional Description 35–1
ADSP-2184x EMDMA Register List 35–2
ADSP-2184x EMDMA Interrupt List 35–2
ADSP-2184x EMDMA Trigger List 35–3
DMA Addressing 35–3
DMA Burst Transfers 35–3
Transfer Control Block (TCB) Memory Storage 35–3
Chain Assignment 35–4
Starting Chain Loading 35–4
Buffered Chain Loading Register 35–4
TCB Storage 35–4
EMDMA Operating Modes 35–7
Standard DMA 35–8
Circular Buffered DMA 35–8
Chained DMA Mode 35–9
Data Direction On-the-Fly 35–10
Write Back Circular Index Pointer 35–10
Scatter/Gather DMA 35–10
Pre Modified Read/Write Index 35–13
Delay Line DMA 35–13
ADSP-2184x EMDMA Register Descriptions 35–16
External Base Address Register 35–17
Circular Buffer Length Register 35–18
Chain Pointer Register 35–19

Internal Count Register 35–21
External Count Register 35–22
External Memory DMA Control Register 35–23
Internal Index Register 35–27
External Index Register 35–28
Internal Modifier Register 35–29
External Modifier Register 35–30
Delay Line Tap Count Register 35–31
Tap List Pointer Register 35–32

Cyclic Redundancy Check (CRC)

CRC Features 36–1
CRC Functional Description 36–2
ADSP-2184x CRC Register List 36–2
ADSP-2184x CRC Interrupt List 36–3
CRC Definitions 36–3
CRC Block Diagram 36–4
Peripheral DMA Bus 36–5
MMR Access Bus 36–5
Mirror Block 36–6
Data FIFO 36–6
DMA Request Generator 36–6
CRC Engine 36–6
Compare Logic 36–6
CRC Architectural Concepts 36–6
Look-up Table 36–7
Data Mirroring 36–7
FIFO Status and Data Requests 36–8
CRC Operating Modes 36–9
Data Transfer Modes 36–9

Memory Scan Compute-and-Compare Mode 36–10
Memory Scan Data Verify 36–10
Memory Transfer Compute-and-Compare Mode 36–10
Memory Transfer Data Fill Mode 36–11
CRC Event Control 36–11
Interrupt Signals 36–11
CRC Programming Model 36–12
CRC Mode Configuration 36–12
Look-up Table Generation 36–12
Core Driven Memory Scan Compute-and-Compare Mode 36–13
DMA Driven Memory Scan Compute-and-Compare Mode 36–14
Core Driven Memory Scan Data Verify Mode 36–16
DMA Driven Memory Scan Data Verify Mode 36–17
Core Driven Memory Transfer Compute-and-Compare Mode 36–19
DMA Driven Memory Transfer Compute-and-Compare Mode 36–20
DMA Driven Memory Transfer Data Fill Mode 36–22
ADSP-2184x CRC Register Descriptions 36–23
Data Compare Register 36–25
Control Register 36–26
Data Word Count Register 36–29
Data Count Capture Register 36–30
Data Word Count Reload Register 36–31
Data FIFO Register 36–32
Fill Value Register 36–33
Interrupt Enable Register 36–34
Interrupt Enable Clear Register 36–35
Interrupt Enable Set Register 36–36
Polynomial Register 36–37
CRC Current Result Register 36–38
CRC Final Result Register 36–39
Status Register 36–40

System Security

Security Features 37–1
Security Functional Description 37–2
Hardware Security Module (HSM) 37–4
Security Mode Configuration 37–4
HSM Assisted Secure Boot Flow 37–4
Status and Error Signals 37–5

System Protection Unit (SPU)

SPU Features 38–1
SPU Functional Description 38–1
ADSP-2184x SPU Register List 38–1
ADSP-2184x SPU Interrupt List 38–2
Peripheral Register Write Protection 38–2
SPU Block Diagram 38–5
SPU Architectural Concepts 38–5
SPU Event Control 38–5
SPU Programming Model 38–6
SPU Mode Configuration 38–7
Locking Write-Protect Registers 38–7
Protecting a Peripheral 38–7
Configuring Security Privileges of a Peripheral 38–8
ADSP-2184x SPU Register Descriptions 38–8
Control Register 38–9
Secure Check Register 38–11
Secure Control Register 38–12
Secure Core Registers 38–14
Secure Peripheral Register 38–15
Status Register 38–16
Write Protect Register n 38–17

Write-Protect, Secure Peripheral, and Secure Core Registers 38–17
Processor-Specific Information 38–24

Security Packet Engine (PKTE)

PKTE Features 39–1
PKTE Functional Description 39–1
ADSP-2184x PKTE Register List 39–1
ADSP-2184x PKTE Interrupt List 39–3
PKTE Definitions 39–4
Cipher Module 39–5
Hash Module 39–6
Pseudo-Random Number Generator 39–6
Packet Engine Processing Details 39–8
Crypto Padding 39–8
Pad Generation and Insertion 39–9
Pad Types 39–9
Pad Length 39–10
Pad Verification and Consumption 39–12
Crypto and Hash Algorithms 39–14
IV Processing 39–17
ARC4 Processing 39–18
Hash State Loading 39–19
Sequence Number Processing 39–19
Sequence Number Processing in Extended SSL/TLS 39–19
Sequence Number Processing in DTLS 39–20
PKTE Block Diagram 39–23
PKTE Architectural Concepts 39–24
Packet Engine 39–24
Input/Output FIFO Buffers 39–24
Parallel Operations 39–24
DMA Controller 39–24

Interrupt Controller 39–25
Clock Controller 39–25
PKTE Operating Modes 39–25
Autonomous Ring Mode (ARM) 39–26
Target Command Mode (TCM) 39–27
Direct Host Mode (DHM) 39–27
PKTE Event Control 39–27
PKTE Interrupt Signals 39–27
PKTE Programming Model 39–30
PKTE Mode Configuration 39–33
PKTE Programming Concepts 39–33
Packet Engine Descriptor 39–33
Descriptor Processing 39–35
Descriptor Ownership 39–36
SA Record and State Record Structure 39–37
SA Record Structure 39–37
SA State Structure 39–40
ARC4 State Structure 39–41
Configuring Operations in the PKTE 39–41
Basic Operations and Decoding 39–41
Error Code Description 39–43
Extended Error Codes 39–43
Number Format 39–46
PKTE Programming Examples 39–46
Calculating SHA in Direct Host Mode 39–46
Performing AES Decryption in Direct Host Mode 39–47
ADSP-2184x PKTE Register Descriptions 39–48
Packet Engine ARC4 State Record Address 39–51
Starting Entry of 256-byte ARC4 State Buffer 39–52
Packet Engine Buffer Pointer Register 39–53
Packet Engine Buffer Threshold Register 39–54

Packet Engine Command Descriptor Ring Base Address 39–56
Packet Engine Command Descriptor Count Register 39–57
Packet Engine Command Descriptor Count Increment Register 39–58
Packet Engine Configuration Register 39–59
PE Clock Control Register 39–62
PKTE Continue Register 39–64
Packet Engine Control Register 39–65
Starting Entry of 256-byte Data Input/Output Buffer 39–70
Packet Engine Destination Address 39–71
Packet Engine DMA Configuration Register 39–72
Packet Engine Endian Configuration Register 39–74
Packet Engine Halt Control Register 39–76
Packet Engine Halt Status Register 39–78
Interrupt Mask Disable Register 39–81
Interrupt Mask Enable Register 39–83
Interrupt Masked Status Register 39–85
Packet Engine Input Buffer Count Register 39–87
Packet Engine Input Buffer Count Increment Register 39–88
Interrupt Configuration Register 39–89
Interrupt Clear Register 39–90
Interrupt Enable Register 39–92
Interrupt Unmasked Status Register 39–94
Packet Engine Length Register 39–96
Packet Engine Output Buffer Count Register 39–98
Packet Engine Output Buffer Count Decrement Register 39–99
PE Alternative Key stat register 39–100
Packet Engine Cache Control 0 39–101
Packet Engine Cache Control 1 39–102
Packet Engine Result Descriptor Ring Base Address 39–103
Packet Engine Result Descriptor Count Registers 39–104

Packet Engine Result Descriptor Count Decrement Registers 39–105
Packet Engine Ring Configuration 39–106
Packet Engine Ring Pointer Status 39–107
Packet Engine Ring Status 39–108
Packet Engine Ring Threshold Registers 39–109
Packet Engine SA Address 39–111
ARC4 i and j Pointer Register 39–112
SA Command 0 39–113
SA Command 1 39–118
SA Inner Hash Digest Registers 39–122
SA Key Registers 39–123
SA Initialization Vector Register 39–124
SA Outer Hash Digest Registers 39–125
SA Ready Indicator 39–127
SA Sequence Number Register 39–128
SA Sequence Number Mask Registers 39–129
SA SPI Register 39–130
Packet Engine Source Address 39–131
Packet Engine Status Register 39–132
Packet Engine State Record Address 39–136
State Hash Byte Count Registers 39–137
State Inner Digest Registers 39–138
State Initialization Vector Registers 39–139
Packet Engine User ID 39–140

Public Key Accelerator (PKA)

PKA Features 40–1
PKA Functional Description 40–1
ADSP-2184x PKA Register List 40–2
PKA Definitions 40–2

PKA Architectural Concepts 40–3
PKA Block Diagram 40–3
PKCP Vector Operations 40–4
Modular Exponentiation Operations 40–6
Modular Inversion 40–13
Modular Inversion with an Even Modulus 40–14
Modular Inversion with a Prime Modulus 40–14
ECC Operations 40–14
ADSP-2184x PKA Register Descriptions 40–20
PKA Vector_A Length 40–21
PKA Vector_A Address 40–22
PKA Vector_B Length 40–23
PKA Vector_B Address 40–24
PKA Compare Result 40–25
PKA Vector_C Address 40–26
PKA Most-Significant-Word of Divide Remainder 40–27
PKA Vector_D Address 40–28
PKA Function 40–29
Start of PKA RAM space 40–32
PKA Most-Significant-Word of Result Vector 40–33
PKA Bit Shift Value 40–34

Public Key Interrupt Controller (PKIC)

PKIC Functional Description 41–1
ADSP-2184x PKIC Register List 41–1
ADSP-2184x PKIC Interrupt List 41–2
PKIC Programming Model 41–2
PKIC Programming Concepts 41–2
ADSP-2184x PKIC Register Descriptions 41–3
Acknowledge Register 41–4

Enable Clear Register 41–5
Enable Control Register 41–6
Enable Set Register 41–7
Enabled Status Register 41–8
Polarity Control Register 41–9
Raw Status Register 41–10
Type Control Register 41–11

True Random Number Generator (TRNG)

TRNG Features 42–1
TRNG Functional Description 42–1
ADSP-2184x TRNG Register List 42–1
Random Number Generation 42–2
Locking Detection and Prevention 42–3
Run Testing 42–4
Monobit Testing 42–4
Poker Testing 42–5
Data for Tests 42–6
X9.31 Postprocessing 42–6
TRNG Block Diagram 42–6
TRNG Architectural Concepts 42–7
TRNG Operating Modes 42–8
TRNG Data Transfer Modes 42–9
TRNG Event Control 42–9
TRNG Interrupt Signals 42–9
ADSP-2184x TRNG Register Descriptions 42–10
TRNG Alarm Counter Register 42–12
TRNG Alarm Mask Register 42–14
TRNG Alarm Stop Register 42–15
TRNG Block Count Register 42–16

TRNG Configuration Register 42–17
Counter Register 42–19
TRNG Control Register 42–20
TRNG FRO De-tune Register 42–23
TRNG FRO Enable Register 42–24
TRNG Input Registers 42–25
TRNG Interrupt Acknowledge Register 42–26
Post-Process Key Registers 42–28
TRNG LFSR Access Register 42–29
TRNG LFSR Access Register 42–30
TRNG LFSR Access Register 42–31
TRNG Monobit Test Result Register 42–32
TRNG Output Registers 42–33
TRNG Poker Test Result Registers 42–34
TRNG Run Count Registers 42–35
TRNG Run Test State and Result Registers 42–36
TRNG Status Register 42–38
TRNG Test Register 42–40
TRNG Post-Process "V" Value Registers 42–43

FIR Accelerator (FIR)

Features 43–1
Clocking 43–1
Functional Description 43–2
ADSP-2184x FIR Register List 43–2
ADSP-2184x FIR Interrupt List 43–3
ADSP-2184x FIR Trigger List 43–4
Compute Block 43–4
Partial Sum Register 43–5
Delay Line Memory 43–5

Coefficient Memory 43–5
Prefetch Data Buffer 43–5
Processing Output 43–6
System Memory Storage 43–7
Coefficients and Input Buffer Storage 43–7
Single Rate Input Filtering 43–7
Decimation 43–7
Interpolation 43–8
Operating Modes 43–8
Single Rate Processing 43–8
Single Iteration 43–8
Floating-point Multi-Iteration 43–8
Window Processing 43–9
Multi-Rate Processing 43–9
Decimation 43–9
Interpolation 43–9
Floating-Point Data Format 43–10
Fixed-Point Data Format 43–10
Auto Configuration Mode (ACM) 43–10
Data Transfer 43–11
Chain Assignment 43–11
DMA Access 43–13
Burst Access Support 43–13
Data and Coefficient Load in Parallel 43–13
Prefetch Buffer for DMA Channels 43–13
Accelerator TCB 43–13
Chain Pointer DMA 43–14
Programming Model 43–15
Legacy Mode 43–15
Auto Configuration Mode (ACM) 43–16

Debug Mode 43–17
Write to Local Memory 43–17
Read from Local Memory 43–18
Single-Step Mode 43–18
Computing FIR Output, Tap Length Greater than 4096 43–18
Debug Features 43–19
Local Memory Access 43–19
Single-Step Mode 43–20
Emulation Considerations 43–20
Interrupts 43–20
Sources 43–20
Window Complete 43–21
All Channels Complete 43–21
MAC Status 43–21
Service 43–21
ADSP-2184x FIR Register Descriptions 43–22
FIR Chain Pointer Register 43–23
FIR Coefficient Count Register 43–24
FIR Coefficient Index Register 43–25
FIR Coefficient Modifier Register 43–26
FIR Global Control Register 43–27
FIR Channel Control Register 43–31
Debug Address Register 43–33
FIR Debug Control Register 43–34
FIR Debug Data Read Register 43–35
FIR Debug Data Write Register 43–36
FIR DMA Status Register 43–37
FIR Input Data Base Register 43–39
FIR Input Data Count Register 43–40
FIR Input Data Index Register 43–41

FIR Input Data Modifier Register 43–42
FIR MAC Status Register 43–43
FIR Output Data Base Register 43–46
FIR Output Data Count Register 43–47
FIR Output Data Index Register 43–48
FIR Output Data Modifier Register 43–49
Software Control Register 1 43–50
Software Control Register 2 43–51
Secondary Global Control Register 43–52

IIR Accelerator (IIR)

Features 44–1
IIR Definitions 44–1
Clocking 44–3
Functional Description 44–3
ADSP-2184x IIR Register List 44–5
ADSP-2184x IIR Interrupt List 44–6
ADSP-2184x IIR Trigger List 44–6
Multiply and Accumulate (MAC) Unit 44–7
Input Data and Biquad State 44–7
Coefficient Memory 44–7
Internal Memory Storage 44–7
Coefficient Memory Storage 44–8
Target Coefficient Memory Storage 44–8
Operating Modes 44–8
Window Processing Mode 44–8
40-Bit Floating-Point Mode 44–9
Save Biquad State Mode 44–9
Auto Configuration Mode (ACM) 44–10
Coefficient Slewing 44–11
Coefficient Slewing Features 44–11

Coefficient Slewing Parameters 44–12
Coefficient Real-time Slewing 44–13
Data Transfers 44–13
IIR Accelerator TCB 44–13
DMA Access 44–16
Burst Mode Access 44–16
Chain Pointer DMA 44–16
Interrupts 44–17
Sources 44–18
Window Complete 44–18
All Channels Complete 44–18
MAC Status 44–18
Service 44–19
Programming Model 44–19
Programming Legacy Mode and ACM 44–19
Legacy Mode Programming 44–20
Auto Configuration Mode Programming 44–21
Coefficient Slewing Programming 44–22
Coefficient Slewing Program Flow 44–23
Programming Constant Alpha Slewing Mode 44–23
Programming Linear Changing Alpha Slewing Mode 44–24
Programming Intermediate Target Coefficient Update Mode 44–24
ADSP-2184x IIR Register Descriptions 44–25
Channel n Alpha 44–26
Chain Pointer Register 44–27
Coefficient Buffer Index Register 44–28
Coefficient Buffer Length Register 44–29
Coefficient Index Modifier Register 44–30
Global Control Register 44–31
Channel Control Register 44–35
IIR Debug Address Register 44–37

IIR Debug Control Register 44–38
IIR Debug Read Data High Register 44–39
IIR Debug Read Data Low Register 44–40
IIR Debug Write Data High Register 44–41
IIR Debug Write Data Low Register 44–42
DMA Status Register 44–43
Input Buffer Base Register 44–45
Input Data Index Register 44–46
Input Data Buffer Length Register 44–47
Input Data Index Modifier Register 44–48
MAC Status Register 44–49
Output Buffer Base Register 44–51
Output Data Buffer Index Register 44–52
IIR Output Data Buffer Length Register 44–53
IIR Output Data Index Modifier Register 44–54
Software Control Register1 44–55
Software Control Register2 44–56
Secondary Global Control Register 44–57
Channel 0 Slewing Duration and Frequency 44–58

System Crossbars (SCB)

SCB Features 45–1
SCB Functional Description 45–1
SCB Architectural Concepts 45–2
SCB Block Diagram 45–2
SCB Block Diagram 45–3
Address Remapping for Backwards Compatibility 45–12
Port Security 45–14
System Crossbars 45–14
SCB Bus Requester IDs 45–15
SCB Programming Model 45–18

SCB Programming Concepts 45–20
QoS Programming 45–24
ADSP-2184x SCB0 Register Descriptions 45–25
..... 45–34
..... 45–35
..... 45–36
..... 45–37
..... 45–38
..... 45–39
..... 45–40
..... 45–41
..... 45–42
..... 45–43
..... 45–44
..... 45–45
..... 45–46
..... 45–47
..... 45–48
..... 45–49
..... 45–50
..... 45–51
..... 45–52
..... 45–53
..... 45–54
..... 45–55
..... 45–56
..... 45–57
..... 45–58
..... 45–59
..... 45–60

45–61
45–62
45–63
45–64
45–65
45–66
45–67
45–68
45–69
45–70
45–71
45–72
45–73
45–74
45–75
45–76
45–77
45–78
45–79
45–80
45–81
45–82
45–83
45–84
45–85
45–86
45–87
45–88
45–89
45–90

45–91
45–92
45–93
45–94
45–95
45–96
45–97
45–98
45–99
45–100
45–101
45–102
45–103
45–104
45–105
45–106
45–107
45–108
45–109
45–110
45–111
45–112
45–113
45–114
45–115
45–116
45–117
45–118
45–119
45–120

45–121
45–122
45–123
45–124
45–125
45–126
45–127
45–128
45–129
45–130
45–131
45–132
45–133
45–134
45–135
45–136
45–137
45–138
45–139
45–140
45–141
45–142
45–143
45–144
45–145
45–146
45–147
45–148
45–149
45–150

45–151
45–152
45–153
45–154
45–155
45–156
45–157
45–158
45–159
45–160
45–161
45–162
45–163
45–164
45–165
45–166
45–167
45–168
45–169
45–170
45–171
45–172
45–173
45–174
45–175
45–176
45–177
45–178
45–179
45–180

45–181
45–182
45–183
45–184
45–185
45–186
45–187
45–188
45–189
45–190
45–191
45–192
45–193
45–194
45–195
45–196
45–197
45–198
45–199
45–200
45–201
45–202
45–203
45–204
45–205
45–206
45–207
45–208
45–209
45–210

45–211
45–212
45–213
45–214
45–215
45–216
45–217
45–218
45–219
45–220
45–221
45–222
45–223
45–224
45–225
45–226
45–227
45–228
45–229
45–230
45–231
45–232
45–233
45–234
45–235
45–236
45–237
45–238
45–239
45–240

45–241
45–242
45–243
45–244
45–245
45–246
45–247
45–248
45–249
45–250
45–251
45–252
45–253
45–254
45–255
45–256
45–257
45–258
45–259
45–260
45–261
45–262
45–263
45–264
45–265
45–266
45–267
45–268
45–269
45–270

45–271
45–272
45–273
45–274
45–275
45–276
45–277
45–278
45–279
45–280
45–281
45–282
45–283
45–284
45–285
45–286
45–287
45–288
45–289
45–290
45–291
45–292
45–293
45–294
45–295
45–296
45–297
45–298
45–299
45–300

45–301
45–302
45–303
45–304
45–305
45–306
45–307
45–308
45–309
45–310
45–311
ADSP-2184x SCB1 Register Descriptions 45–311
45–312
ADSP-2184x SCB3 Register Descriptions 45–312
45–314
45–315
45–316
45–317
45–318
45–319
45–320
45–321
45–322
45–323
45–324
45–325
45–326
45–327
45–328
45–329

45–330
45–331
45–332
45–333
45–334
45–335
45–336
45–337
45–338
45–339
45–340
45–341
45–342
45–343
45–344
45–345
45–346
45–347
45–348
ADSP-2184x SCB4 Register Descriptions 45–348
45–352
45–353
45–354
45–355
45–356
45–357
45–358
45–359
45–360
45–361

45–362
45–363
45–364
45–365
45–366
45–367
45–368
45–369
45–370
45–371
45–372
45–373
45–374
45–375
45–376
45–377
45–378
45–379
45–380
45–381
45–382
45–383
45–384
45–385
45–386
45–387
45–388
45–389
45–390
45–391

45–392
45–393
45–394
45–395
45–396
45–397
45–398
45–399
45–400
45–401
45–402
45–403
45–404
45–405
45–406
45–407
45–408
45–409
45–410
45–411
45–412
45–413
45–414
45–415
45–416
45–417
45–418
45–419
45–420
45–421

45–422
45–423
45–424
45–425
45–426
45–427
45–428
45–429
45–430
45–431
45–432
45–433
45–434
45–435
45–436
45–437
45–438
45–439
45–440
45–441
45–442
45–443
45–444
45–445
ADSP-2184x INST1 Register Descriptions 45–445
45–447
45–448
45–449
45–450
45–451

45–452
45–453
45–454
45–455
45–456
45–457
45–458
45–459
45–460
45–461
45–462
45–463
45–464
45–465
45–466
45–467

System Watchpoint Unit (SWU)

SWU Features 46–1
SWU Functional Description 46–1
ADSP-2184x SWU Register List 46–1
ADSP-2184x SWU Interrupt List 46–2
ADSP-2184x SWU Trigger List 46–3
SWU Definitions 46–4
SWU Architectural Concepts 46–4
SWU-to-SCB Interface 46–4
SWU Block Diagram 46–4
SCB Interface Block 46–5
MMR Interface Block 46–5
SWU Operating Modes 46–5
Bandwidth Mode 46–5

Watchpoint Mode 46–5
Match Block 46–5
Scaling 46–6
SWU Event Control 46–6
SWU Interrupts 46–6
SWU Status and Errors 46–6
Triggers 46–7
SWU Programming Model 46–7
SWU Mode Configuration 46–8
Configuring the SWU for Bandwidth Mode 46–8
Configuring the SWU for Watchpoint Mode 46–9
ADSP-2184x SWU Register Descriptions 46–10
Count Register n 46–11
Control Register n 46–12
Current Register n 46–17
Global Control Register 46–18
Global Status Register 46–19
Bandwidth History Register n 46–23
ID Register n 46–24
Lower Address Register n 46–25
Target Register n 46–26
Upper Address Register n 46–27

Memory Error Protection Unit (MEPU)

Memory Error Controller (MEC) 47–1
MEC Features 47–1
Parity Controller (PCTL) 47–2
PCTL Features 47–2
MEC Functional Description 47–2
ADSP-2184x MEC Register List 47–2

ADSP-2184x MEC Trigger List 47–4
ADSP-2184x MEC Interrupt List 47–5
MEPU Block Diagram 47–6
MEC Block Diagram 47–7
PCTL Block Diagram 47–7
MECx Input and Output Block Diagram 47–7
MEC Architectural Concepts 47–8
MEC Configuration 47–8
Error Status and Interrupts 47–11
Requester Port Bus Interrupts 47–13
PCTL Integration 47–13
ADSP-2184x MEC Register Descriptions 47–13
A55 Core Memory Errors Interrupt Request Global Control Register 47–16
A55 Error Interrupt Request Global Status Register 47–17
A55 Error Control Register 47–18
A55 Error Interrupt Mask Register 47–19
A55 Error Status Register 47–20
A55 Core Memory Errors Interrupt Request Global Control Register 47–21
A55 Error Interrupt Request Global Status Register 47–22
A55 Error Control Register 47–23
A55 Error Interrupt Mask Register 47–24
A55 Error Status Register 47–25
Component ID0 Register 47–26
Component ID1 Register 47–27
Component ID2 Register 47–28
Component ID3 Register 47–29
Clear Register 47–30
Core Mem ECC Error Interrupt Request Global Control Register 47–31
Core Memory ECC Error Interrupt Request Global Status Register 47–32
Core Memory ECC Error Control Register 47–33

Core Memory ECC Error Interrupt Mask Register 47–34
Core Memory ECC Error Status Register 47–35
ECC Error Control Register 47–36
ECC Error Interrupt Mask Register 47–37
ECC Error Status Register 47–38
ECC Error Interrupt Request Global Control Register 47–39
ECC Error Interrupt Request Global Status Register 47–40
Parity Error Interrupt Request Global Control Register 47–41
Parity Error Interrupt Request Global Status Register 47–42
Parity Error Control Register 47–43
Parity Error Control Register 47–45
Parity Error Interrupt Mask Register 47–47
Parity Error Interrupt Mask Register 47–49
Parity Error Status Register 47–51
Parity Error Status Register 47–53
Peripheral ID0 Register 47–55
Peripheral ID1 Register 47–56
Peripheral ID2 Register 47–57
Peripheral ID3 Register 47–58
Peripheral ID4 Register 47–59

System Debug and Trace Unit (DBG)

DBG Features 48–1
Debug Functional Description 48–2
DBG Block Diagram 48–2
DBG Definitions 48–2
Test Access Port Controller (TAPC) 48–3
Embedded Trace Macrocell (ETM) 48–4
Debug Access Ports 48–4
Trace Unit 48–4

Programmable Flow Trace (CSPFT) 48–4
System Trace Module (STM) 48–5
Embedded Cross Trigger (ECT) 48–5
ADSP-2184x A55_CLUSTER Register Descriptions 48–7
CTI External Multiplexer Control Register 48–19
CTI Application Trigger Clear Register 48–20
CTI Application Pulse Register 48–21
CTI Application Trigger Set Register 48–22
CTI Authentication Status Register 48–23
CTI Channel In Status Register 48–24
CTI Channel Out Status Register 48–25
CTI Component Identification 0 Register 48–26
CTI Component Identification 1 Register 48–27
CTI Component Identification 2 Register 48–28
CTI Component Identification 3 Register 48–29
CTI Claim Tag Clear Register 48–30
CTI Claim Tag Set Register 48–31
CTI Control Register 48–32
CTI Device Affinity 0 Register 48–33
CTI Device Affinity 1 Register 48–34
CTI Device Architecture Register 48–35
CTI Device ID 0 Register 48–36
CTI Device ID 1 Register 48–38
CTI Device ID 2 Register 48–39
CTI Device Type Register 48–40
CTI Channel Gate Enable Register 48–41
CTI Input Trigger to Output Channel Enable 0 Register 48–42
CTI Input Trigger to Output Channel Enable 1 Register 48–43
CTI Input Trigger to Output Channel Enable 2 Register 48–44
CTI Input Trigger to Output Channel Enable 3 Register 48–45

CTI Input Trigger to Output Channel Enable 4 Register 48–46
CTI Input Trigger to Output Channel Enable 5 Register 48–47
CTI Input Trigger to Output Channel Enable 6 Register 48–48
CTI Input Trigger to Output Channel Enable 7 Register 48–49
CTI Output Trigger Acknowledge Register 48–50
CTI Integration mode Control Register 48–51
CTI Lock Access Register 48–52
CTI Lock Status Register 48–53
CTI Input Channel to Output Trigger Enable 0 Register 48–55
CTI Input Channel to Output Trigger Enable 1 Register 48–56
CTI Input Channel to Output Trigger Enable 2 Register 48–57
CTI Input Channel to Output Trigger Enable 3 Register 48–58
CTI Input Channel to Output Trigger Enable 4 Register 48–59
CTI Input Channel to Output Trigger Enable 5 Register 48–60
CTI Input Channel to Output Trigger Enable 6 Register 48–61
CTI Input Channel to Output Trigger Enable 7 Register 48–62
CTI Peripheral Identification 0 Register 48–63
CTI Peripheral Identification 1 Register 48–64
CTI Peripheral Identification 2 Register 48–65
CTI Peripheral Identification 3 Register 48–66
CTI Peripheral Identification 4 Register 48–67
CTI Peripheral Identification 5 Register 48–68
CTI Peripheral Identification 6 Register 48–69
CTI Peripheral Identification 7 Register 48–70
CTI Trigger In Status Register 48–71
CTI Trigger Out Status Register 48–72
Debug Authentication Status Register 48–73
External Debug Component Identification 0 Register 48–75
External Debug Component Identification 1 Register 48–76
External Debug Component Identification 2 Register 48–77

External Debug Component Identification 3 Register 48–78
External Debug Device Affinity 0 Register 48–79
External Debug Device Affinity 1 Register 48–80
External Debug Device Architecture Register 48–81
External Debug Device ID 0 Register 48–82
External Debug Device ID 1 Register 48–84
External Debug Device ID 2 Register 48–85
External Debug Device Type Register 48–86
External Debug Feature Register 48–87
External Debug Feature Register 48–89
External Debug Execution Control Register 48–90
External Debug Lock Access Register 48–91
External Debug Lock Status Register 48–92
External Debug Processor Feature Register 48–94
External Debug Processor Feature Register 48–97
External Debug Peripheral Identification 0 Register 48–98
External Debug Peripheral Identification 1 Register 48–99
External Debug Peripheral Identification 2 Register 48–100
External Debug Peripheral Identification 3 Register 48–101
External Debug Peripheral Identification 4 Register 48–102
External Debug Peripheral Identification 5 Register 48–103
External Debug Peripheral Identification 6 Register 48–104
External Debug Peripheral Identification 7 Register 48–105
Main ID Register 48–106
Trace Authentication Status Register 48–108
Trace Component ID 0 Register 48–110
Trace Component ID 1 Register 48–111
Trace Component ID 2 Register 48–112
Trace Component ID 3 Register 48–113
Trace Device Affinity 0 Register 48–114

Trace Device Affinity 1 Register 48–115
Trace Device Architecture Register 48–116
Trace Device ID Register 48–117
Trace Device Type Register 48–118
Trace Software Lock Access Register 48–119
Trace Software Lock Status Register 48–120
Trace PowerDown Control Register 48–121
Trace Peripheral ID 0 Register 48–122
Trace Peripheral ID 1 Register 48–123
Trace Peripheral ID 2 Register 48–124
Trace Peripheral ID 3 Register 48–125
Trace Peripheral ID 4 Register 48–126
Performance Monitors Authentication Status Register 48–127
Performance Monitors Component Identification 0 Register 48–128
Performance Monitors Component Identification 1 Register 48–129
Performance Monitors Component Identification 2 Register 48–130
Performance Monitors Component Identification 3 Register 48–131
Performance Monitors Device Affinity 0 Register 48–132
Performance Monitors Device Affinity 1 Register 48–133
Performance Monitors Device Architecture Register 48–134
Performance Monitors Device Type Register 48–135
Performance Monitors Lock Access Register 48–136
Performance Monitors Lock Status Register 48–137
Performance Monitors Peripheral Identification 0 Register 48–139
Performance Monitors Peripheral Identification 1 Register 48–140
Performance Monitors Peripheral Identification 2 Register 48–141
Performance Monitors Peripheral Identification 3 Register 48–142
Performance Monitors Peripheral Identification 4 Register 48–143
Performance Monitors Peripheral Identification 5 Register 48–144
Performance Monitors Peripheral Identification 6 Register 48–145

Performance Monitors Peripheral Identification 7 Register 48–146
CTI External Multiplexer Control Register 48–147
CTI Application Trigger Clear Register 48–148
CTI Application Pulse Register 48–149
CTI Application Trigger Set Register 48–150
CTI Authentication Status Register 48–151
CTI Channel In Status Register 48–152
CTI Channel Out Status Register 48–153
CTI Component Identification 0 Register 48–154
CTI Component Identification 1 Register 48–155
CTI Component Identification 2 Register 48–156
CTI Component Identification 3 Register 48–157
CTI Claim Tag Clear Register 48–158
CTI Claim Tag Set Register 48–159
CTI Control Register 48–160
CTI Device Affinity 0 Register 48–161
CTI Device Affinity 1 Register 48–162
CTI Device Architecture Register 48–163
CTI Device ID 0 Register 48–164
CTI Device ID 1 Register 48–166
CTI Device ID 2 Register 48–167
CTI Device Type Register 48–168
CTI Channel Gate Enable Register 48–169
CTI Input Trigger to Output Channel Enable 0 Register 48–170
CTI Input Trigger to Output Channel Enable 1 Register 48–171
CTI Input Trigger to Output Channel Enable 2 Register 48–172
CTI Input Trigger to Output Channel Enable 3 Register 48–173
CTI Input Trigger to Output Channel Enable 4 Register 48–174
CTI Input Trigger to Output Channel Enable 5 Register 48–175
CTI Input Trigger to Output Channel Enable 6 Register 48–176

CTI Input Trigger to Output Channel Enable 7 Register **48–177**
CTI Output Trigger Acknowledge Register **48–178**
CTI Integration mode Control Register **48–179**
CTI Lock Access Register **48–180**
CTI Lock Status Register **48–181**
CTI Input Channel to Output Trigger Enable 0 Register **48–183**
CTI Input Channel to Output Trigger Enable 1 Register **48–184**
CTI Input Channel to Output Trigger Enable 2 Register **48–185**
CTI Input Channel to Output Trigger Enable 3 Register **48–186**
CTI Input Channel to Output Trigger Enable 4 Register **48–187**
CTI Input Channel to Output Trigger Enable 5 Register **48–188**
CTI Input Channel to Output Trigger Enable 6 Register **48–189**
CTI Input Channel to Output Trigger Enable 7 Register **48–190**
CTI Peripheral Identification 0 Register **48–191**
CTI Peripheral Identification 1 Register **48–192**
CTI Peripheral Identification 2 Register **48–193**
CTI Peripheral Identification 3 Register **48–194**
CTI Peripheral Identification 4 Register **48–195**
CTI Peripheral Identification 5 Register **48–196**
CTI Peripheral Identification 6 Register **48–197**
CTI Peripheral Identification 7 Register **48–198**
CTI Trigger In Status Register **48–199**
CTI Trigger Out Status Register **48–200**
Debug Authentication Status Register **48–201**
External Debug Component Identification 0 Register **48–203**
External Debug Component Identification 1 Register **48–204**
External Debug Component Identification 2 Register **48–205**
External Debug Component Identification 3 Register **48–206**
External Debug Device Affinity 0 Register **48–207**
External Debug Device Affinity 1 Register **48–208**

External Debug Device Architecture Register 48–209
External Debug Device ID 0 Register 48–210
External Debug Device ID 1 Register 48–212
External Debug Device ID 2 Register 48–213
External Debug Device Type Register 48–214
External Debug Feature Register 48–215
External Debug Feature Register 48–217
External Debug Execution Control Register 48–218
External Debug Lock Access Register 48–219
External Debug Lock Status Register 48–220
External Debug Processor Feature Register 48–222
External Debug Processor Feature Register 48–225
External Debug Peripheral Identification 0 Register 48–226
External Debug Peripheral Identification 1 Register 48–227
External Debug Peripheral Identification 2 Register 48–228
External Debug Peripheral Identification 3 Register 48–229
External Debug Peripheral Identification 4 Register 48–230
External Debug Peripheral Identification 5 Register 48–231
External Debug Peripheral Identification 6 Register 48–232
External Debug Peripheral Identification 7 Register 48–233
Main ID Register 48–234
Trace Authentication Status Register 48–236
Trace Component ID 0 Register 48–238
Trace Component ID 1 Register 48–239
Trace Component ID 2 Register 48–240
Trace Component ID 3 Register 48–241
Trace Device Affinity 0 Register 48–242
Trace Device Affinity 1 Register 48–243
Trace Device Architecture Register 48–244
Trace Device ID Register 48–245

Trace Device Type Register **48–246**
Trace Software Lock Access Register **48–247**
Trace Software Lock Status Register **48–248**
Trace PowerDown Control Register **48–249**
Trace Peripheral ID 0 Register **48–250**
Trace Peripheral ID 1 Register **48–251**
Trace Peripheral ID 2 Register **48–252**
Trace Peripheral ID 3 Register **48–253**
Trace Peripheral ID 4 Register **48–254**
Performance Monitors Authentication Status Register **48–255**
Performance Monitors Component Identification 0 Register **48–256**
Performance Monitors Component Identification 1 Register **48–257**
Performance Monitors Component Identification 2 Register **48–258**
Performance Monitors Component Identification 3 Register **48–259**
Performance Monitors Device Affinity 0 Register **48–260**
Performance Monitors Device Affinity 1 Register **48–261**
Performance Monitors Device Architecture Register **48–262**
Performance Monitors Device Type Register **48–263**
Performance Monitors Lock Access Register **48–264**
Performance Monitors Lock Status Register **48–265**
Performance Monitors Peripheral Identification 0 Register **48–267**
Performance Monitors Peripheral Identification 1 Register **48–268**
Performance Monitors Peripheral Identification 2 Register **48–269**
Performance Monitors Peripheral Identification 3 Register **48–270**
Performance Monitors Peripheral Identification 4 Register **48–271**
Performance Monitors Peripheral Identification 5 Register **48–272**
Performance Monitors Peripheral Identification 6 Register **48–273**
Performance Monitors Peripheral Identification 7 Register **48–274**
Authentication Status Register **48–275**
Component Identification 0 Register **48–278**

Component Identification 1 Register **48–279**
Component Identification 2 Register **48–280**
Component Identification 3 Register **48–281**
Debug Power Control 0 Register **48–282**
Debug Power Control 1 Register **48–283**
Debug Power Control 2 Register **48–284**
Debug Power Control 3 Register **48–285**
Debug Power Control 4 Register **48–286**
Debug Power Control 5 Register **48–287**
Debug Power Control 6 Register **48–288**
Debug Power Control 7 Register **48–289**
Debug Power Control 8 Register **48–290**
Debug Power Status 0 Register **48–291**
Debug Power Status 1 Register **48–292**
Debug Power Status 2 Register **48–293**
Debug Power Status 3 Register **48–294**
Debug Power Status 4 Register **48–295**
Debug Power Status 5 Register **48–296**
Debug Power Status 6 Register **48–297**
Debug Power Status 7 Register **48–298**
Debug Power Status 8 Register **48–299**
Device Architecture Register **48–300**
Device ID Register **48–301**
Device Type Register **48–303**
Peripheral Identification 0 Register **48–304**
Peripheral Identification 1 Register **48–305**
Peripheral Identification 2 Register **48–306**
Peripheral Identification 3 Register **48–307**
Peripheral Identification 4 Register **48–308**
Peripheral Identification 5 Register **48–309**

Peripheral Identification 6 Register 48–310
Peripheral Identification 7 Register 48–311
Power Reset Identification 0 Register 48–312
ROM Table Entry 0 Register 48–313
ROM Table Entry 1 Register 48–315
ROM Table Entry 10 Register 48–317
ROM Table Entry 11 Register 48–319
ROM Table Entry 12 Register 48–321
ROM Table Entry 13 Register 48–323
ROM Table Entry 14 Register 48–325
ROM Table Entry 15 Register 48–327
ROM Table Entry 16 Register 48–329
ROM Table Entry 17 Register 48–331
ROM Table Entry 18 Register 48–333
ROM Table Entry 19 Register 48–335
ROM Table Entry 2 Register 48–337
ROM Table Entry 20 Register 48–339
ROM Table Entry 21 Register 48–341
ROM Table Entry 22 Register 48–343
ROM Table Entry 23 Register 48–345
ROM Table Entry 24 Register 48–347
ROM Table Entry 25 Register 48–349
ROM Table Entry 26 Register 48–351
ROM Table Entry 27 Register 48–353
ROM Table Entry 28 Register 48–355
ROM Table Entry 29 Register 48–357
ROM Table Entry 3 Register 48–359
ROM Table Entry 30 Register 48–361
ROM Table Entry 31 Register 48–363
ROM Table Entry 32 Register 48–365

ROM Table Entry 33 Register 48–367
ROM Table Entry 34 Register 48–369
ROM Table Entry 35 Register 48–371
ROM Table Entry 36 Register 48–373
ROM Table Entry 37 Register 48–375
ROM Table Entry 38 Register 48–377
ROM Table Entry 39 Register 48–379
ROM Table Entry 4 Register 48–381
ROM Table Entry 40 Register 48–383
ROM Table Entry 41 Register 48–385
ROM Table Entry 42 Register 48–387
ROM Table Entry 43 Register 48–389
ROM Table Entry 44 Register 48–391
ROM Table Entry 45 Register 48–393
ROM Table Entry 46 Register 48–395
ROM Table Entry 47 Register 48–397
ROM Table Entry 48 Register 48–399
ROM Table Entry 49 Register 48–401
ROM Table Entry 5 Register 48–403
ROM Table Entry 50 Register 48–405
ROM Table Entry 51 Register 48–407
ROM Table Entry 52 Register 48–409
ROM Table Entry 53 Register 48–411
ROM Table Entry 54 Register 48–413
ROM Table Entry 55 Register 48–415
ROM Table Entry 56 Register 48–417
ROM Table Entry 57 Register 48–419
ROM Table Entry 58 Register 48–421
ROM Table Entry 59 Register 48–423
ROM Table Entry 6 Register 48–425

ROM Table Entry 60 Register 48–427
ROM Table Entry 61 Register 48–429
ROM Table Entry 62 Register 48–431
ROM Table Entry 63 Register 48–433
ROM Table Entry 64 Register 48–435
ROM Table Entry 65 Register 48–437
ROM Table Entry 66 Register 48–439
ROM Table Entry 67 Register 48–441
ROM Table Entry 68 Register 48–443
ROM Table Entry 69 Register 48–445
ROM Table Entry 7 Register 48–447
ROM Table Entry 70 Register 48–449
ROM Table Entry 71 Register 48–451
ROM Table Entry 72 Register 48–453
ROM Table Entry 73 Register 48–455
ROM Table Entry 74 Register 48–457
ROM Table Entry 75 Register 48–459
ROM Table Entry 76 Register 48–461
ROM Table Entry 77 Register 48–463
ROM Table Entry 78 Register 48–465
ROM Table Entry 79 Register 48–467
ROM Table Entry 8 Register 48–469
ROM Table Entry 80 Register 48–471
ROM Table Entry 81 Register 48–473
ROM Table Entry 9 Register 48–475
ADSP-2184x SOC600CTI_TRACE Register Descriptions 48–476
External Multiplexer Control register. 48–479
Authentication Status Register. 48–480
Component Identification Register 0. 48–484
Component Identification Register 1. 48–485

Component Identification Register 2. 48–486
Component Identification Register 3. 48–487
Claim Tag Clear Register. 48–488
Claim Tag Set Register. 48–489
CTI Application Channel Clear register. 48–490
CTI Application Channel Pulse register. 48–491
CTI Application Channel Set register. 48–492
CTI Channel Input Status register. 48–493
CTI Channel Output Status register. 48–494
CTI Control register. 48–495
Enable CTI Channel Gate register. 48–496
CTI Trigger 0 to Channel Enable register. 48–497
CTI Trigger 1 to Channel Enable register. 48–498
CTI Trigger 2 to Channel Enable register. 48–499
CTI Trigger 3 to Channel Enable register. 48–500
CTI Trigger 4 to Channel Enable register. 48–501
CTI Trigger 5 to Channel Enable register. 48–502
CTI Trigger 6 to Channel Enable register. 48–503
CTI Interrupt Acknowledge register. 48–504
CTI Channel to Trigger 0 Enable register. 48–505
CTI Channel to Trigger 1 Enable register. 48–506
CTI Channel to Trigger 2 Enable register. 48–507
CTI Channel to Trigger 3 Enable register. 48–508
CTI Channel to Trigger 4 Enable register. 48–509
CTI Channel to Trigger 5 Enable register. 48–510
CTI Trigger Input Status register. 48–511
CTI Trigger Output Status register. 48–512
Device Affinity register 0. 48–513
Device Affinity register 1. 48–514
Device Architecture Register. 48–515

Device Configuration Register. 48–516
Device Type Identifier Register. 48–517
Integration Test Channel Input register. 48–518
Integration Test Channel Output register. 48–519
Integration Mode Control Register. 48–520
Integration Test Trigger Input register. 48–521
Integration Test Trigger Output register. 48–522
Peripheral Identification Register 0. 48–523
Peripheral Identification Register 1. 48–524
Peripheral Identification Register 2. 48–525
Peripheral Identification Register 3. 48–526
Peripheral Identification Register 4. 48–527
Peripheral Identification Register 5. 48–528
Peripheral Identification Register 6. 48–529
Peripheral Identification Register 7. 48–530
ADSP-2184x SYSCTI Register Descriptions 48–530
External Multiplexer Control register. 48–533
Authentication Status Register. 48–534
Component Identification Register 0. 48–538
Component Identification Register 1. 48–539
Component Identification Register 2. 48–540
Component Identification Register 3. 48–541
Claim Tag Clear Register. 48–542
Claim Tag Set Register. 48–543
CTI Application Channel Clear register. 48–544
CTI Application Channel Pulse register. 48–545
CTI Application Channel Set register. 48–546
CTI Channel Input Status register. 48–547
CTI Channel Output Status register. 48–548
CTI Control register. 48–549

Enable CTI Channel Gate register. 48–550
CTI Trigger 0 to Channel Enable register. 48–551
CTI Trigger 1 to Channel Enable register. 48–552
CTI Trigger 2 to Channel Enable register. 48–553
CTI Trigger 3 to Channel Enable register. 48–554
CTI Trigger 4 to Channel Enable register. 48–555
CTI Trigger 5 to Channel Enable register. 48–556
CTI Trigger 6 to Channel Enable register. 48–557
CTI Trigger 7 to Channel Enable register. 48–558
CTI Interrupt Acknowledge register. 48–559
CTI Channel to Trigger 0 Enable register. 48–560
CTI Channel to Trigger 1 Enable register. 48–561
CTI Channel to Trigger 10 Enable register. 48–562
CTI Channel to Trigger 11 Enable register. 48–563
CTI Channel to Trigger 12 Enable register. 48–564
CTI Channel to Trigger 13 Enable register. 48–565
CTI Channel to Trigger 2 Enable register. 48–566
CTI Channel to Trigger 3 Enable register. 48–567
CTI Channel to Trigger 4 Enable register. 48–568
CTI Channel to Trigger 5 Enable register. 48–569
CTI Channel to Trigger 6 Enable register. 48–570
CTI Channel to Trigger 7 Enable register. 48–571
CTI Channel to Trigger 8 Enable register. 48–572
CTI Channel to Trigger 9 Enable register. 48–573
CTI Trigger Input Status register. 48–574
CTI Trigger Output Status register. 48–575
Device Affinity register 0. 48–576
Device Affinity register 1. 48–577
Device Architecture Register. 48–578
Device Configuration Register. 48–579

Device Type Identifier Register. 48–580
Integration Test Channel Input register. 48–581
Integration Test Channel Output register. 48–582
Integration Mode Control Register. 48–583
Integration Test Trigger Input register. 48–584
Integration Test Trigger Output register. 48–585
Peripheral Identification Register 0. 48–586
Peripheral Identification Register 1. 48–587
Peripheral Identification Register 2. 48–588
Peripheral Identification Register 3. 48–589
Peripheral Identification Register 4. 48–590
Peripheral Identification Register 5. 48–591
Peripheral Identification Register 6. 48–592
Peripheral Identification Register 7. 48–593
ADSP-2184x TAPC Register Descriptions 48–593
Debug Control Register 48–595
IDCODE Register 48–597
Secure Debug Key 0 Register 48–598
Secure Debug Key 1 Register 48–599
Secure Debug Key 2 Register 48–600
Secure Debug Key 3 Register 48–601
Secure Debug Key 4 Register 48–602
Secure Debug Key 5 Register 48–603
Secure Debug Key 6 Register 48–604
Secure Debug Key 7 Register 48–605
Secure Debug Key Control Register 48–606
Secure Debug Key Status Register 48–607
USERCODE Register 48–608
ADSP-2184x TRACE Register Descriptions 48–608
Authentication Status Register. 48–610

Component Identification Register 0. 48–614
Component Identification Register 1. 48–615
Component Identification Register 2. 48–616
Component Identification Register 3. 48–617
Claim Tag Clear Register. 48–618
Claim Tag Set Register. 48–619
Current Port Size Register. 48–620
Current Test Patterns/Modes Register. 48–621
Device Architecture Register. 48–623
Device Configuration Register. 48–624
Device Type Identifier Register. 48–626
External Control Port In Register. 48–627
External Control Port Out Register. 48–628
Formatter and Flush Control Register. 48–629
Formatter and Flush Status Register. 48–632
Formatter Synchronization Count Register. 48–633
Integration Test ATB Control Register 0. 48–634
Integration Test ATB Control Register 1. 48–635
Integration Test ATB Control Register 2. 48–636
Integration Test ATB Data Register 0. 48–637
Integration Mode Control Register. 48–638
Integration Test Output Control Register. 48–639
Integration Test Trigger In and Flush In Register. 48–640
Peripheral Identification Register 0. 48–641
Peripheral Identification Register 1. 48–642
Peripheral Identification Register 2. 48–643
Peripheral Identification Register 3. 48–644
Peripheral Identification Register 4. 48–645
Peripheral Identification Register 5. 48–646
Peripheral Identification Register 6. 48–647

Peripheral Identification Register 7. 48–648
Supported Port Size Register. 48–649
Supported Trigger Modes Register. 48–650
Supported Test Patterns/Modes Register. 48–652
Trigger Counter Multiplier Register. 48–654
Trigger Counter Value Register. 48–656
Test Pattern Repeat Counter Register. 48–657
ADSP-2184x CSETF Register Descriptions 48–657
Authentication Status Register. 48–660
Buffer Level Water Mark. 48–664
Current Buffer Fill Level. 48–665
Component Identification Register 0. 48–666
Component Identification Register 1. 48–667
Component Identification Register 2. 48–668
Component Identification Register 3. 48–669
Claim Tag Clear Register. 48–670
Claim Tag Set Register. 48–671
Control Register. 48–672
Device Configuration Register. 48–673
Device Configuration Register 1. 48–675
Device Type Identifier Register. 48–676
Formatter and Flush Control Register. 48–677
Formatter and Flush Status Register. 48–681
Integration Test ATB Control 0 Register. 48–682
Integration Test ATB Control 1 Register. 48–683
Integration Test ATB Control 2 Register. 48–684
Integration Test ATB Data 0 Register. 48–685
Integration Test ATB Transmitter Control 0 register. 48–686
Integration Test ATB Transmitter Control 1 register. 48–687
Integration Test ATB Transmitter Control 2 register. 48–688

Integration Test ATB Transmitter Data 0 register. 48–689
Integration Mode Control Register. 48–690
Integration Test Event and Interrupt Control Register. 48–691
Integration Test Trigger In and Flush In register. 48–692
Latched Buffer Fill Level. 48–693
Mode register. 48–694
Peripheral Identification Register 0. 48–695
Peripheral Identification Register 1. 48–696
Peripheral Identification Register 2. 48–697
Peripheral Identification Register 3. 48–698
Peripheral Identification Register 4. 48–699
Peripheral Identification Register 5. 48–700
Peripheral Identification Register 6. 48–701
Peripheral Identification Register 7. 48–702
Periodic Synchronization Counter Register. 48–703
RAM Read Data register. 48–704
RAM Read Pointer register. 48–705
RAM Size register. 48–706
RAM Write Data register. 48–707
RAM Write Pointer register. 48–708
Status register. 48–709
Trigger Counter register. 48–711
ADSP-2184x CSETR Register Descriptions 48–711
Authentication Status Register. 48–714
AXI Control Register. 48–718
AXI Control Register 1. 48–720
Buffer Level Water Mark. 48–721
Current Buffer Fill Level. 48–722
Component Identification Register 0. 48–723
Component Identification Register 1. 48–724

Component Identification Register 2. ..... **48–725**
Component Identification Register 3. ..... **48–726**
Claim Tag Clear Register. **48–727**
Claim Tag Set Register. **48–728**
Control Register. **48–729**
Data Buffer Address HIGH register. **48–730**
Data Buffer Address Low register. **48–731**
Device Configuration Register. **48–732**
Device Configuration Register 1. ..... **48–735**
Device Type Identifier Register. **48–736**
Formatter and Flush Control Register. **48–737**
Formatter and Flush Status Register. **48–740**
Integration Test ATB Control 0 Register. **48–741**
Integration Test ATB Control 1 Register. **48–742**
Integration Test ATB Control 2 Register. **48–743**
Integration Test ATB Data 0 Register. **48–744**
Integration Mode Control Register. **48–745**
Integration Test Event and Interrupt Control Register. **48–746**
Integration Test Trigger In and Flush In register. **48–747**
Latched Buffer Fill Level. **48–748**
Mode register. **48–749**
Peripheral Identification Register 0. ..... **48–750**
Peripheral Identification Register 1. ..... **48–751**
Peripheral Identification Register 2. ..... **48–752**
Peripheral Identification Register 3. ..... **48–753**
Peripheral Identification Register 4. ..... **48–754**
Peripheral Identification Register 5. ..... **48–755**
Peripheral Identification Register 6. ..... **48–756**
Peripheral Identification Register 7. ..... **48–757**
Periodic Synchronization Counter Register. **48–758**

RAM Read Data register. 48–759
RAM Read Pointer register. 48–760
RAM Read Pointer High register. 48–761
RAM Size register. 48–762
RAM Update Read Pointer register. 48–763
RAM Write Data register. 48–764
RAM Write Pointer register. 48–765
RAM Write Pointer High register. 48–766
Status register. 48–767
Trigger Counter register. 48–769
ADSP-2184x CSTF Register Descriptions 48–769
Authentication Status Register. 48–771
Component Identification Register 0. 48–775
Component Identification Register 1. 48–776
Component Identification Register 2. 48–777
Component Identification Register 3. 48–778
Claim Tag Clear Register. 48–779
Claim Tag Set Register. 48–780
Device Affinity register 0. 48–781
Device Affinity register 1. 48–782
Device Architecture Register. 48–783
Device Configuration Register. 48–784
Device Configuration Register 1. 48–785
Device Configuration Register 2. 48–786
Device Type Identifier Register. 48–787
Funnel Control register. 48–788
Integration test control register 0. 48–790
Integration test control register 1. 48–791
Integration test control register 2. 48–792
Integration test control register 3. 48–793

Integration test data register. 48–794
Integration Mode Control Register. 48–796
Peripheral Identification Register 0. 48–797
Peripheral Identification Register 1. 48–798
Peripheral Identification Register 2. 48–799
Peripheral Identification Register 3. 48–800
Peripheral Identification Register 4. 48–801
Peripheral Identification Register 5. 48–802
Peripheral Identification Register 6. 48–803
Peripheral Identification Register 7. 48–804
Priority Control register. 48–805
ADSP-2184x A55_SYSTMRRD Register Descriptions 48–805
Component Identification Register 0. 48–807
Component Identification Register 1. 48–808
Component Identification Register 2. 48–809
Component Identification Register 3. 48–810
Current value of Counter[31:0]. 48–811
Current value of Counter[63:32]. 48–812
Peripheral Identification Register 0. 48–813
Peripheral Identification Register 1. 48–814
Peripheral Identification Register 2. 48–815
Peripheral Identification Register 3. 48–816
Peripheral Identification Register 4. 48–817
Peripheral Identification Register 5. 48–818
Peripheral Identification Register 6. 48–819
Peripheral Identification Register 7. 48–820
ADSP-2184x A55_C0 Register Descriptions 48–820
Debug Breakpoint Control Registers. 48–826
Debug Breakpoint Control Registers. 48–830
Debug Breakpoint Control Registers. 48–834

Debug Breakpoint Control Registers. 48–838
Debug Breakpoint Control Registers. 48–842
Debug Breakpoint Control Registers. 48–846
Debug Claim Tag Clear register. 48–850
Debug Claim Tag Set register. 48–851
Debug Data Transfer Register, Receive. 48–852
Debug Data Transfer Register, Transmit. 48–853
Debug Watchpoint Control Registers. 48–854
Debug Watchpoint Control Registers. 48–857
Debug Watchpoint Control Registers. 48–860
Debug Watchpoint Control Registers. 48–863
Debug Watchpoint Value Registers. 48–866
Debug Watchpoint Value Registers. 48–867
Debug Watchpoint Value Registers. 48–868
Debug Watchpoint Value Registers. 48–869
Debug Watchpoint Value Registers. 48–870
Debug Watchpoint Value Registers. 48–871
Debug Watchpoint Value Registers. 48–872
Debug Watchpoint Value Registers. 48–873
External Debug Context ID Sample Register. 48–874
External Debug Exception Catch Control Register. 48–875
External Debug Event Status Register. 48–877
External Debug Integration mode Control register. 48–879
External Debug Instruction Transfer Register. 48–880
External Debug Power/Reset Control Register. 48–881
External Debug Processor Status Register. 48–883
External Debug Reserve Control Register. 48–889
External Debug Status and Control Register. 48–891
External Debug Watchpoint Address Register. 48–897
External Debug Watchpoint Address Register. 48–898

OS Lock Access Register. **48–899**
Trace Address Comparator Access Type Registers. **48–900**
Trace Address Comparator Access Type Registers. **48–905**
Trace Address Comparator Access Type Registers. **48–906**
Trace Address Comparator Access Type Registers. **48–911**
Trace Address Comparator Access Type Registers. **48–912**
Trace Address Comparator Access Type Registers. **48–917**
Trace Address Comparator Access Type Registers. **48–918**
Trace Address Comparator Access Type Registers. **48–923**
Trace Address Comparator Access Type Registers. **48–924**
Trace Address Comparator Access Type Registers. **48–929**
Trace Address Comparator Access Type Registers. **48–930**
Trace Address Comparator Access Type Registers. **48–935**
Trace Address Comparator Access Type Registers. **48–936**
Trace Address Comparator Access Type Registers. **48–941**
Trace Address Comparator Access Type Registers. **48–942**
Trace Address Comparator Access Type Registers. **48–947**
Trace Address Comparator Value Registers. **48–948**
Trace Address Comparator Value Registers. **48–949**
Trace Address Comparator Value Registers. **48–950**
Trace Address Comparator Value Registers. **48–951**
Trace Address Comparator Value Registers. **48–952**
Trace Address Comparator Value Registers. **48–953**
Trace Address Comparator Value Registers. **48–954**
Trace Address Comparator Value Registers. **48–955**
Trace Address Comparator Value Registers. **48–956**
Trace Address Comparator Value Registers. **48–957**
Trace Address Comparator Value Registers. **48–958**
Trace Address Comparator Value Registers. **48–959**
Trace Address Comparator Value Registers. **48–960**

Trace Address Comparator Value Registers. 48–961
Trace Address Comparator Value Registers. 48–962
Trace Address Comparator Value Registers. 48–963
Trace Auxiliary Control Register. 48–964
Trace Branch Broadcast Control Register. 48–967
Trace Cycle Count Control Register. 48–969
Trace Context ID Comparator Control Register 0. 48–970
Trace Context ID Comparator Value Registers. 48–972
Trace Context ID Comparator Value Registers. 48–973
Trace Claim Tag Clear Register. 48–974
Trace Claim Tag Set Register. 48–975
Trace Counter Control Registers. 48–976
Trace Counter Control Registers. 48–978
Trace Counter Reload Value Registers. 48–980
Trace Counter Reload Value Registers. 48–981
Trace Counter Value Registers. 48–982
Trace Counter Value Registers. 48–983
Trace Configuration Register. 48–984
Trace Event Control 0 Register. 48–988
Trace Event Control 1 Register. 48–991
Trace External Input Select Register. 48–993
Trace ID Register 0. 48–995
Trace ID Register 1. 48–1000
Trace ID Register 2. 48–1002
Trace ID Register 3. 48–1005
Trace ID Register 4. 48–1008
Trace ID Register 5. 48–1011
Trace IMPLEMENTATION DEFINED Register 0. 48–1014
Integration ATB Identification Register. 48–1016
Trace Integration Mode Control Register. 48–1017

Integration Instruction ATB In Register. 48–1018
Integration Instruction ATB Out Register. 48–1019
Integration Instruction ATB Data Register. 48–1020
Trace OS Lock Access Register. 48–1021
Trace OS Lock Status Register. 48–1022
Trace PowerDown Status Register. 48–1023
Trace Programming Control Register. 48–1025
Trace Resource Selection Control Registers. 48–1026
Trace Resource Selection Control Registers. 48–1028
Trace Resource Selection Control Registers. 48–1030
Trace Resource Selection Control Registers. 48–1032
Trace Resource Selection Control Registers. 48–1034
Trace Resource Selection Control Registers. 48–1036
Trace Resource Selection Control Registers. 48–1038
Trace Resource Selection Control Registers. 48–1040
Trace Resource Selection Control Registers. 48–1042
Trace Resource Selection Control Registers. 48–1044
Trace Resource Selection Control Registers. 48–1046
Trace Resource Selection Control Registers. 48–1048
Trace Resource Selection Control Registers. 48–1050
Trace Resource Selection Control Registers. 48–1052
Trace Sequencer State Transition Control Registers. 48–1054
Trace Sequencer State Transition Control Registers. 48–1055
Trace Sequencer State Transition Control Registers. 48–1056
Trace Sequencer Reset Control Register. 48–1057
Trace Sequencer State Register. 48–1058
Trace Single-Shot Comparator Control Registers. 48–1059
Trace Single-Shot Comparator Status Registers. 48–1061
Trace Stall Control Register. 48–1063
Trace Status Register. 48–1065

Trace Synchronization Period Register. **48–1066**
Trace Trace ID Register. **48–1067**
Trace Global Timestamp Control Register. **48–1068**
Trace ViewInst Main Control Register. **48–1069**
Trace ViewInst Include/Exclude Control Register. **48–1072**
Trace ViewInst Start/Stop Control Register. **48–1074**
Trace VMID Comparator Value Registers. **48–1076**
Trace VMID Comparator Value Registers. **48–1077**
Performance Monitors Cycle Counter Filter Register. **48–1078**
Performance Monitors Cycle Counter. **48–1080**
Performance Monitors Cycle Counter. **48–1081**
Performance Monitors Common Event Identification register 0. ..... **48–1082**
Performance Monitors Common Event Identification register 1. ..... **48–1083**
Performance Monitors Common Event Identification register 2. ..... **48–1084**
Performance Monitors Common Event Identification register 3. ..... **48–1085**
Performance Monitors Configuration Register. **48–1086**
Performance Monitors Count Enable Clear register. **48–1088**
Performance Monitors Count Enable Set register. **48–1097**
Performance Monitors Control Register. **48–1106**
Performance Monitors Event Count Registers. **48–1109**
Performance Monitors Event Count Registers. **48–1110**
Performance Monitors Event Count Registers. **48–1111**
Performance Monitors Event Count Registers. **48–1112**
Performance Monitors Event Count Registers. **48–1113**
Performance Monitors Event Count Registers. **48–1114**
Performance Monitors Event Type Registers. **48–1115**
Performance Monitors Event Type Registers. **48–1118**
Performance Monitors Event Type Registers. **48–1121**
Performance Monitors Event Type Registers. **48–1124**
Performance Monitors Event Type Registers. **48–1127**

Performance Monitors Event Type Registers. 48–1130
Performance Monitors Interrupt Enable Clear register. 48–1133
Performance Monitors Interrupt Enable Set register. 48–1150
Performance Monitors Integration mode Control register. 48–1167
Performance Monitors Overflow Flag Status Clear register. 48–1168
Performance Monitors Overflow Flag Status Set register. 48–1180
Performance Monitors Software Increment register. 48–1192
ADSP-2184x A55_C2 Register Descriptions 48–1192
Debug Breakpoint Control Registers. 48–1198
Debug Breakpoint Control Registers. 48–1202
Debug Breakpoint Control Registers. 48–1206
Debug Breakpoint Control Registers. 48–1210
Debug Breakpoint Control Registers. 48–1214
Debug Breakpoint Control Registers. 48–1218
Debug Claim Tag Clear register. 48–1222
Debug Claim Tag Set register. 48–1223
Debug Data Transfer Register, Receive. 48–1224
Debug Data Transfer Register, Transmit. 48–1225
Debug Watchpoint Control Registers. 48–1226
Debug Watchpoint Control Registers. 48–1229
Debug Watchpoint Control Registers. 48–1232
Debug Watchpoint Control Registers. 48–1235
Debug Watchpoint Value Registers. 48–1238
Debug Watchpoint Value Registers. 48–1239
Debug Watchpoint Value Registers. 48–1240
Debug Watchpoint Value Registers. 48–1241
Debug Watchpoint Value Registers. 48–1242
Debug Watchpoint Value Registers. 48–1243
Debug Watchpoint Value Registers. 48–1244
Debug Watchpoint Value Registers. 48–1245

External Debug Context ID Sample Register. 48–1246
External Debug Exception Catch Control Register. 48–1247
External Debug Event Status Register. 48–1249
External Debug Integration mode Control register. 48–1251
External Debug Instruction Transfer Register. 48–1252
External Debug Power/Reset Control Register. 48–1253
External Debug Processor Status Register. 48–1255
External Debug Reserve Control Register. 48–1261
External Debug Status and Control Register. 48–1263
External Debug Watchpoint Address Register. 48–1269
External Debug Watchpoint Address Register. 48–1270
OS Lock Access Register. 48–1271
Trace Address Comparator Access Type Registers. 48–1272
Trace Address Comparator Access Type Registers. 48–1277
Trace Address Comparator Access Type Registers. 48–1278
Trace Address Comparator Access Type Registers. 48–1283
Trace Address Comparator Access Type Registers. 48–1284
Trace Address Comparator Access Type Registers. 48–1289
Trace Address Comparator Access Type Registers. 48–1290
Trace Address Comparator Access Type Registers. 48–1295
Trace Address Comparator Access Type Registers. 48–1296
Trace Address Comparator Access Type Registers. 48–1301
Trace Address Comparator Access Type Registers. 48–1302
Trace Address Comparator Access Type Registers. 48–1307
Trace Address Comparator Access Type Registers. 48–1308
Trace Address Comparator Access Type Registers. 48–1313
Trace Address Comparator Access Type Registers. 48–1314
Trace Address Comparator Access Type Registers. 48–1319
Trace Address Comparator Value Registers. 48–1320
Trace Address Comparator Value Registers. 48–1321

Trace Address Comparator Value Registers. 48–1322
Trace Address Comparator Value Registers. 48–1323
Trace Address Comparator Value Registers. 48–1324
Trace Address Comparator Value Registers. 48–1325
Trace Address Comparator Value Registers. 48–1326
Trace Address Comparator Value Registers. 48–1327
Trace Address Comparator Value Registers. 48–1328
Trace Address Comparator Value Registers. 48–1329
Trace Address Comparator Value Registers. 48–1330
Trace Address Comparator Value Registers. 48–1331
Trace Address Comparator Value Registers. 48–1332
Trace Address Comparator Value Registers. 48–1333
Trace Address Comparator Value Registers. 48–1334
Trace Address Comparator Value Registers. 48–1335
Trace Auxiliary Control Register. 48–1336
Trace Branch Broadcast Control Register. 48–1339
Trace Cycle Count Control Register. 48–1341
Trace Context ID Comparator Control Register 0. 48–1342
Trace Context ID Comparator Value Registers. 48–1344
Trace Context ID Comparator Value Registers. 48–1345
Trace Claim Tag Clear Register. 48–1346
Trace Claim Tag Set Register. 48–1347
Trace Counter Control Registers. 48–1348
Trace Counter Control Registers. 48–1350
Trace Counter Reload Value Registers. 48–1352
Trace Counter Reload Value Registers. 48–1353
Trace Counter Value Registers. 48–1354
Trace Counter Value Registers. 48–1355
Trace Configuration Register. 48–1356
Trace Event Control 0 Register. 48–1360

Trace Event Control 1 Register. 48–1363
Trace External Input Select Register. 48–1365
Trace ID Register 0. 48–1367
Trace ID Register 1. 48–1372
Trace ID Register 2. 48–1374
Trace ID Register 3. 48–1377
Trace ID Register 4. 48–1380
Trace ID Register 5. 48–1383
Trace IMPLEMENTATION DEFINED Register 0. 48–1386
Integration ATB Identification Register. 48–1388
Trace Integration Mode Control Register. 48–1389
Integration Instruction ATB In Register. 48–1390
Integration Instruction ATB Out Register. 48–1391
Integration Instruction ATB Data Register. 48–1392
Trace OS Lock Access Register. 48–1393
Trace OS Lock Status Register. 48–1394
Trace PowerDown Status Register. 48–1395
Trace Programming Control Register. 48–1397
Trace Resource Selection Control Registers. 48–1398
Trace Resource Selection Control Registers. 48–1400
Trace Resource Selection Control Registers. 48–1402
Trace Resource Selection Control Registers. 48–1404
Trace Resource Selection Control Registers. 48–1406
Trace Resource Selection Control Registers. 48–1408
Trace Resource Selection Control Registers. 48–1410
Trace Resource Selection Control Registers. 48–1412
Trace Resource Selection Control Registers. 48–1414
Trace Resource Selection Control Registers. 48–1416
Trace Resource Selection Control Registers. 48–1418
Trace Resource Selection Control Registers. 48–1420

Trace Resource Selection Control Registers. 48–1422
Trace Resource Selection Control Registers. 48–1424
Trace Sequencer State Transition Control Registers. 48–1426
Trace Sequencer State Transition Control Registers. 48–1427
Trace Sequencer State Transition Control Registers. 48–1428
Trace Sequencer Reset Control Register. 48–1429
Trace Sequencer State Register. 48–1430
Trace Single-Shot Comparator Control Registers. 48–1431
Trace Single-Shot Comparator Status Registers. 48–1433
Trace Stall Control Register. 48–1435
Trace Status Register. 48–1437
Trace Synchronization Period Register. 48–1438
Trace Trace ID Register. 48–1439
Trace Global Timestamp Control Register. 48–1440
Trace ViewInst Main Control Register. 48–1441
Trace ViewInst Include/Exclude Control Register. 48–1444
Trace ViewInst Start/Stop Control Register. 48–1446
Trace VMID Comparator Value Registers. 48–1448
Trace VMID Comparator Value Registers. 48–1449
Performance Monitors Cycle Counter Filter Register. 48–1450
Performance Monitors Cycle Counter. 48–1452
Performance Monitors Cycle Counter. 48–1453
Performance Monitors Common Event Identification register 0. 48–1454
Performance Monitors Common Event Identification register 1. 48–1455
Performance Monitors Common Event Identification register 2. 48–1456
Performance Monitors Common Event Identification register 3. 48–1457
Performance Monitors Configuration Register. 48–1458
Performance Monitors Count Enable Clear register. 48–1460
Performance Monitors Count Enable Set register. 48–1469
Performance Monitors Control Register. 48–1478

Performance Monitors Event Count Registers. 48–1481
Performance Monitors Event Count Registers. 48–1482
Performance Monitors Event Count Registers. 48–1483
Performance Monitors Event Count Registers. 48–1484
Performance Monitors Event Count Registers. 48–1485
Performance Monitors Event Count Registers. 48–1486
Performance Monitors Event Type Registers. 48–1487
Performance Monitors Event Type Registers. 48–1490
Performance Monitors Event Type Registers. 48–1493
Performance Monitors Event Type Registers. 48–1496
Performance Monitors Event Type Registers. 48–1499
Performance Monitors Event Type Registers. 48–1502
Performance Monitors Interrupt Enable Clear register. 48–1505
Performance Monitors Interrupt Enable Set register. 48–1522
Performance Monitors Integration mode Control register. 48–1539
Performance Monitors Overflow Flag Status Clear register. 48–1540
Performance Monitors Overflow Flag Status Set register. 48–1552
Performance Monitors Software Increment register. 48–1564

ADSP-2184x MISCREG Register Descriptions

CANFD Low Power Selection Mode 49–3
Miscellaneous Clock Multiplexer Selection Register 49–5
Miscellaneous Clock Multiplexer Selection 1 Register 49–9
LPDDR Miscellaneous Register 49–10
EMAC Error Register 49–11
EMAC Miscellaneous Register 49–13
EMAC Miscellaneous Register 2 49–14
EMMC MISC Register 49–15
MINI CGU0 MISC Register 49–16
MINI CGU0 MISC Register 49–18

SH0 Prefetch Range Selection Register 49–20
SH1 Prefetch Range Selection Register 49–21
SPORT Direct Interrupt Enable for SH0 and SH1 49–22
XSPI1 Boot/Init Control Register 0 49–23
XSPI1 Boot/Init Control Register 1 49–25
XSPI1 Boot/Init Control Register 2 49–26
XSPI1 Boot/Init Control Register 3 49–27
XSPI1 Boot/Init Control Register 4 49–28
XSPI1 Boot/Init Control Register 5 49–29
XSPI1 Boot/Init Control Register 6 49–30
XSPI1 Boot/Init Control Register 7 49–31
XSPI1 Boot/Init Control Register 8 49–32
XSPI1 Boot/Init Reset Control Register 49–33
XSPI1 Boot/Init Status Register 49–34
XSPI Boot/Init Control Register 0 49–35
XSPI Boot/Init Control Register 1 49–37
XSPI Boot/Init Control Register 2 49–38
XSPI Boot/Init Control Register 3 49–39
XSPI Boot/Init Control Register 4 49–40
XSPI Boot/Init Control Register 5 49–41
XSPI Boot/Init Control Register 6 49–42
XSPI Boot/Init Control Register 7 49–43
XSPI Boot/Init Control Register 8 49–44
XSPI Boot/Init Reset Control Register 49–45
XSPI Boot/Init Status Register 49–46

ADSP-2184x Register List

Preface

Purpose of This Manual

The ***ADSP-2184x/ADSP-SC84x SHARC-FX Processor Hardware Reference*** provides architectural information about the ADSP-2184x/SC84x processors. This hardware reference provides the main architectural information about these processors. The architectural descriptions cover functional blocks, buses, and ports, including all features and processes that they support.

For timing, electrical, and package specifications, refer to the ***ADSP-21843/844/845/846/ADSP-SC843/844/845/836 High Performance SHARC-FX DSP Core With Connectivity/Safety/Security Data Sheet***.

Intended Audience

The primary audience for this manual is a programmer who is familiar with Analog Devices products. The manual assumes the audience has a working knowledge of the appropriate product architecture and instruction set. Programmers who are unfamiliar with Analog Devices processors can use this manual, but should supplement it with other texts, such as programming reference books and data sheets, that describe their target architecture.

What's New in This Manual

This manual is the a preliminary revision (0.2) of the ADSP-2184x/SC84x SHARC-FX Processor Hardware Reference.

NOTE: Analog Devices is in the process of updating documentation to provide terminology and language that is culturally appropriate. This is a process with a wide scope and will be phased in as quickly as possible. Thank you for your patience.

Technical or Customer Support

You can reach customer and technical support for processors and peripherals from Analog Devices using the following resources:

- Post your questions in the processors and DSP support community at ***EngineerZone***®:

  http://ez.analog.com/community/dsp

- Submit your questions to technical support at ***Connect with ADI Specialists***:

  http://www.analog.com/support

- Contact your Analog Devices sales office or authorized distributor. Locate one at:

http://www.analog.com/adi-sales

Product Information

Product information can be obtained from the Analog Devices internet site.

Analog Devices Web Site

The Analog Devices Web site, http://www.analog.com, provides information about a broad range of products—analog integrated circuits, amplifiers, converters, and digital signal processors.

To access technical documentation, search the product number on the website to find the appropriate product page. The documentation on the product page includes manuals as well as data sheets and application notes. If there is an errata list, it is linked below the document listing on the product page.

Also note, MyAnalog.com is a free feature of the Analog Devices Web site that allows customization of a Web page to display only the latest information about products you are interested in. You can choose to receive weekly e-mail notifications containing updates to the Web pages that meet your interests, including documentation errata against all manuals. MyAnalog.com provides access to books, application notes, data sheets, code examples, and more.

Visit MyAnalog.com to sign up. If you are a registered user, just log on. Your user name is your e-mail address.

EngineerZone

EngineerZone is a technical support forum from Analog Devices. It allows you direct access to ADI technical support engineers. You can search FAQs and technical information to get quick answers to your design questions.

Use EngineerZone to connect with other developers who face similar design challenges. You can also use this open forum to share knowledge and collaborate with the ADI support team and your peers. Visit http://ez.analog.com to sign up.

Notation Conventions

Text conventions used in this manual are identified and described as follows. Additional conventions, which apply only to specific chapters, may appear throughout this document.

Example	Description
File > ***Close***	Titles in reference sections indicate the location of an item within the CrossCore Embedded Studio IDE's menu system (for example, the ***Close*** command appears on the ***File*** menu).
`{this | that}`	Alternative required items in syntax descriptions appear within curly brackets and separated by vertical bars; read the example as `this` or `that`. One or the other is required.
`[this | that]`	Optional items in syntax descriptions appear within brackets and separated by vertical bars; read the example as an optional `this` or `that`.

Example	Description
`[this, …]`	Optional item lists in syntax descriptions appear within brackets delimited by commas and terminated with an ellipse; read the example as an optional comma-separated list of `this`.
`.SECTION`	Commands, directives, keywords, and feature names are in text with `Letter Gothic` font.
`filename`	Non-keyword placeholders appear in text with italic style format.
NOTE:	***NOTE:*** For correct operation, ... A note provides supplementary information on a related topic. In the online version of this book, the word ***NOTE:*** appears instead of this symbol.
CAUTION:	***CAUTION:*** Incorrect device operation may result if ... ***CAUTION:*** Device damage may result if ... A caution identifies conditions or inappropriate usage of the product that could lead to undesirable results or product damage. In the online version of this book, the word ***CAUTION:*** appears instead of this symbol.
ATTENTION:	***ATTENTION:*** Injury to device users may result if ... A warning identifies conditions or inappropriate usage of the product that could lead to conditions that are potentially hazardous for devices users. In the online version of this book, the word ***ATTENTION:*** appears instead of this symbol.
Registers/Bits	All registers and bits in this manual are linked (clickable) to their respective descriptions in the "Register Descriptions" of each chapter.
Miscellaneous Conventions	Interrupt and internal signals are shown in all caps with no other formatting. For example the SPDIFn_RX or SCLK signal or the PKTE0_IRQ interrupt. An overbar denotes an active-low signal as in $\overline{\text{SYS_FAULT}}$.

Register Documentation Conventions

Register diagrams use the following conventions:

- The descriptive name of the register appears at the top with the short form of the name.
- If a bit has a short name, the short name appears first in the bit description, followed by the long name.
- The reset value appears in binary in the individual bits and in hexadecimal to the left of the register.
- Bits marked ***X*** have an unknown reset value. Consequently, the reset value of registers that contain such bits is undefined or dependent on pin values at reset.
- Shaded bits are reserved

NOTE: To ensure upward compatibility with future implementations, write back the value that is read for reserved bits in a register, unless otherwise specified.

Register description tables use the following conventions:

- Each bit's or bit field's access type appears beneath the bit number in the table in the form (read-access/write-access). The access types include:
  - R = read, RC = read clear, RS = read set, R0 = read zero, R1 = read one, Rx = read undefined
  - W = write, NW = no write, W1C = write one to clear, W1S = write one to set, W0C = write zero to clear, W0S = write zero to set, WS = write to set, WC = write to clear, W1A = write one action
- Many bit and bit field descriptions include enumerations, identifying bit values and related functionality. Unless otherwise indicated (with a prefix), these enumerations are decimal values.

1 Arm Cortex-A55 Subsystem

The ADSP-SC844/SC846 processor includes a dual Arm® Cortex-A55® core. The Cortex-A55 core is a mid-range, low-power core that implements the Armv8-A architecture with support for the Armv8.1-A extension, the Armv8.2-A extension and the reliability, availability, and serviceability (RAS) extension. Each core has a Level 1 (L1) memory system, and private Level 2 (L2) cache. The cores are implemented inside the DynamIQ Shared Unit (DSU) as a little core.

The dual Cortex-A55 sub-system in the ADSP-SC844/SC846 processor includes a data engine unit that implements the advanced SIMD and floating-point architecture support, cryptographic extension, generic interrupt controller, performance monitoring unit, and a generic timer. The Cortex-A55 also includes support for a L1-cache sub-system and a full-fledged memory management unit. The dual Cortex-A55 implements the Armv8 architecture and runs 64-bit Arm instructions in AArch64 execution state and 32-bit Arm and 32-bit Thumb instructions in AArch32 execution state.

This document describes the Arm Cortex-A55 core and memory architecture used on the ADSP-SC844/SC846 processor but does not provide detailed programming information for the Arm processor. For more information about programming the Arm processor, visit the Arm Information Center:

- ARM Documentation

The applicable documentation for programming the Arm Cortex-A55 processor include:

- Arm Cortex-A55 Core Technical Reference Manual, Revision: r2p0
- Arm Cortex-A Series Programmer's Guide for ARMv8-A, Version: 1.0
- ARM CoreLink NIC-400 Network Interconnect Technical Reference Manual
- Arm Architecture Reference Manual Armv8, for Armv8-A architecture profile
- Arm DynamIQ™ Shared Unit Technical Reference Manual, Revision: r4p1
- ARM NEON™ Programmer’s Guide
- Arm Cortex®-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual
- Arm Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and 4.0

Cortex-A55 Features

The Arm Cortex-A55 subsystem has the following features.

Core Features

- Full implementation of the Arm8.2-A A64, A32, and T32 instruction sets
- Both the AArch32 and AArch64 execution states at all exception levels (EL0 to EL3)
- In-order pipeline with direct and indirect branch prediction
- Separate L1 data and instruction side memory systems with a memory management unit (MMU)
- Support for Arm TrustZone® technology
- Data engine unit that implements the advanced SIMD and floating-point architecture support
- Cryptographic extension: This architectural extension is only available if the data engine is present
- Generic interrupt controller (GIC) CPU interface to connect to an external distributor
- Generic timers interface that support a 64-bit count input from an external system counter

Cache Features

- Unified private L2 cache
- L1 and L2 cache protection in the form of error correction code (ECC) or parity on all RAM instances
- Shared L3 cache

Debug Features

- Reliability, availability, and serviceability (RAS) extension
- Arm8.2-A debug logic
- Performance monitoring unit (PMU)
- Embedded trace macrocell (ETM) that supports instruction trace only

Functional Description

The following sections provide information on the function of the subsystem.

Cortex-A55 Block Diagram

The ***A55 Dual Core Subsystem Block Diagram*** shows the primary blocks of the dual Cortex-A55 subsystem. The cluster consists of the cores and the DynamIQ Shared Unit (DSU), which connects the cores to an external memory system.

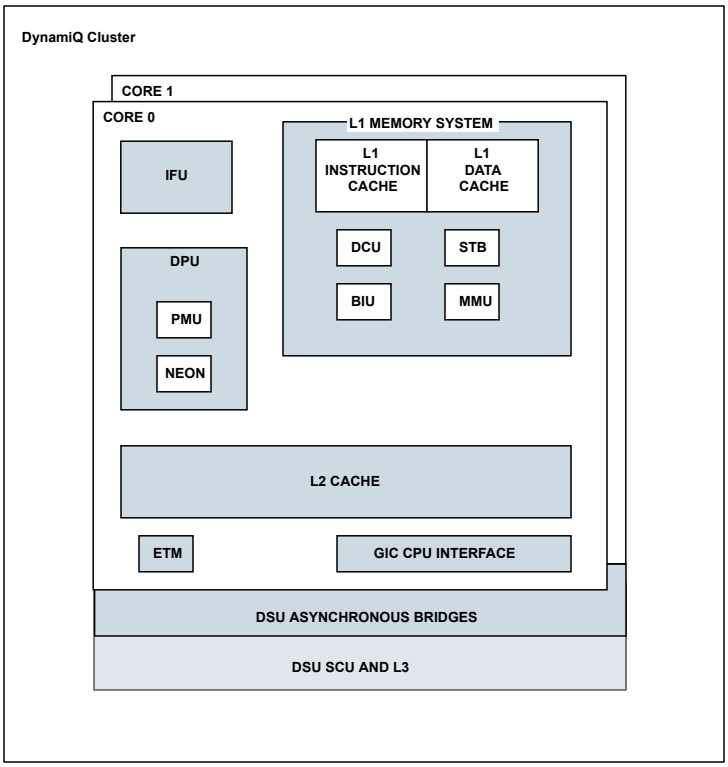


Figure 1-1: A55 Dual Core Subsystem Block Diagram

Instruction Fetch Unit (IFU)

The IFU fetches instructions from the instruction cache or from external memory and predicts the outcome of branches in the instruction stream. It passes the instructions to the Data Processing Unit (DPU) for processing.

Data Processing Unit (DPU)

The DPU decodes and executes instructions. It executes instructions that require data transfer to or from the memory system by interfacing to the Data Cache Unit (DCU). The DPU includes the PMU, the Advanced SIMD and floating-point support, and the Cryptographic Extension.

Performance Monitor Unit (PMU)

The Cortex-A55 core includes performance monitors that can be configured to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the code that can be used for debug and code profiling. The PMU provides six counters. Each counter can count any of the events available in the core.

See the Performance Monitor Unit (PMU) section below for details of the PMU register set and events.

Advanced SIMD and Floating-point support

Advanced SIMD is a media and signal processing architecture that adds instructions primarily for audio, video, 3D graphics, image, and speech processing. The floating-point architecture provides support for single-precision and double-precision floating-point operations.

The Cortex-A55 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set and the Advanced SIMD and floating-point instructions in the A32 and T32 instruction sets. The A64 instruction set offers additional Advanced SIMD instructions, including double-precision floating-point vector operations.

The Cortex-A55 floating-point implementation:

- Does not generate floating-point exceptions
- Implements all scalar operations in hardware with support for all combinations of:
    - Rounding modes
    - Flush-to-zero
    - Default Not a Number (NaN) modes

The Advanced SIMD architecture, its associated implementation, and supporting software, are also referred to as NEON™ technology.

See the *Arm® Cortex®-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual* for details of the instruction and register set.

Cryptographic Extension

The Cortex-A55 core Cryptographic Extension supports the Armv8-A cryptographic extension. The cryptographic extension adds new A64, A32, and T32 instructions to advanced SIMD that accelerate:

- Advanced Encryption Standard (AES) encryption and decryption
- The Secure Hash Algorithm (SHA) functions SHA-1, SHA-224, and SHA-256
- Finite field arithmetic used in algorithms such as Galois/Counter mode and elliptic curve cryptography

See the *Arm® Architecture Reference Manual Armv8*, for Armv8-A architecture profile for details of the instruction and register set.

Memory Management Unit (MMU)

The MMU is responsible for translating addresses of code and data from the virtual view of memory to the physical addresses in the real system. The translation is carried out by the MMU hardware and is transparent to the application. In addition, the MMU controls such things as memory access permissions, memory ordering and cache policies for each region of memory.

The MMU enables tasks or applications to be written in a way that requires them to have no knowledge of the physical memory-map of the system, or about other programs that might be running simultaneously. This enables

you to use the same virtual memory address space for each program. It also lets you work with a contiguous virtual memory map, even if the physical memory is fragmented. This virtual address space is separate from the actual physical map of memory in the system. Applications are written, compiled, and linked to run in the virtual memory space. Virtual addresses are those used by you, and the compiler and linker, when placing code in memory. Physical addresses are those used by the actual hardware system.

The three main functions of the MMU are:

Page table entries support:

- Control the translation table walk hardware that accesses translation tables in main memory
- Translate virtual addresses (VAs) to physical addresses (PAs)
- Provide fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables

Each stage of address translation uses a set of address translations and associated memory properties that are held in memory-mapped tables called translation tables. Translation table entries can be cached into a Translation Lookaside Buffer (TLB). The following table describes the components included in the MMU.

Table 1-1: TLBs and TLB Caches in the MMU

Component	Description
Instruction L1 TLB	15 entries, fully associative
Data L1 TLB	16 entries, fully associative
L2 TLB	1024 entries, 4-way set associative
Walk cache RAM	64 entries, 4-way set associative
IPA cache RAM	64 entries, 4-way set associative

The Cortex-A55 core supports a 40-bit physical address range, which allows 1TB of physical memory to be addressed.

The Cortex-A55 core implements a two-level TLB structure.

L1 TLB

The first level of caching for the translation table information is an L1 TLB. It is implemented on both the instruction and data sides. The Cortex-A55 L1 instruction TLB supports 4 KB, 16 KB, 64 KB, and 2 MB pages. The Cortex-A55 L1 data TLB supports 4 KB pages only. Any other page sizes are fractured after the L2 TLB and the appropriate page size are sent to the L1 TLB. All TLB-related maintenance operations result in flushing both the instruction and data L1 TLBs.

L2 TLB

A unified L2 TLB handles the misses from the L1 instruction and data TLBs using a 4-way, set-associative, 1024-entry cache. In implementations with core cache protection, parity bits protect the TLB RAMs by detecting any single-bit error. If an error is detected, the entry is invalidated and fetched again.

IPA Cache RAM

The Intermediate Physical Address (IPA) cache RAM holds mappings between the IPAs and Physical Addresses (PAs). Only non-secure EL1 and EL0 stage 2 translations use the IPA cache. When a stage 2 translation completes, the cache is updated. The IPA cache is checked whenever a stage 2 translation is required. Like the L2 TLB, the IPA cache RAM can hold entries for different sizes.

Walk Cache RAM

The walk cache RAM holds the result of a stage 1 translation up to, but not including the last level.

NOTE: Virtual memory translation tables are typically created by operating systems and are often dynamically managed by the memory management layer. However, even a bare metal system can enable the MMU. For this, a flat mapping technique is used, where all virtual memory addresses are programmed exactly as the physical memory addresses in the system.

L1 Memory System

The dual Cortex-A55 processor L1 memory system consists of separate instruction and data caches that run at Arm core clock speed. The L1 instruction cache size is 32 KB and data cache size is 32 KB.

L1 Instruction-side Memory System

The L1 instruction-side memory system provides an instruction stream to the DPU. Its key features are:

- 64-byte instruction side cache line length
- 4-way set associative L1 instruction cache
- 128-bit read interface to the L2 memory system

The Cortex-A55 core uses extensive branch prediction to improve Instructions Per Clock (IPC) and power efficiency.

L1 Data-side Memory System

The L1 data-side memory system responds to load and store requests from the DPU. It also responds to SCU snoop requests from other cores, or external controllers. Its key features are:

- 64-byte data side cache line length
- 4-way set associative L1 data cache
- A read buffer that services both the Data Cache Unit (DCU), and the Instruction Fetch Unit (IFU)
- A 64-bit read path from the data L1 memory system to the data path
- A 128-bit write path from the data path to the L1 memory system
- Merging store buffer capability which writes to all types of memory (device, normal cacheable and normal non-cacheable)

- A data side prefetch engine that detects patterns of strides with multiple streams are allowed in parallel, capable of detecting both constant and patterns of strides

L2 Memory System

The dual Cortex-A55 L2 memory system is required to interface the Cortex-A55 cores to the L3 memory system. The main features of the L2 memory system are:

- Strictly exclusive with L1 data cache
- Pseudo-inclusive with L1 instruction cache
- Private per-core unified L2 cache
- 40-bit physical address space
- Physically indexed, physically tagged

The L2 memory subsystem consists of:

- A 4-way, set-associative L2 cache with a size of 256 KB. Cache lines have a fixed length of 64 bytes.
- Optional ECC protection for tag, data, and L2 data buffer RAM structures

The shared L3 memory system consists of:

- Unified 16-way set-associative L3 cache of 512 KB
- Partial L3 cache power-down support
- Optional cache protection in the form of error correcting code (ECC) on L3 cache
- L3 memory system operating at Arm core clock

Reliability, Availability, and Serviceability (RAS)

The Cortex-A55 core implements the RAS extension to the Armv8-A architecture which provides mechanisms for the standardized reporting of errors generated by cache protection mechanisms. When configured with core cache protection, the Cortex-A55 core can detect and correct a 1-bit error in any RAM and detect 2-bit errors in some RAMs.

See the *Arm Cortex-A55 Core Technical Reference Manual* for more information on RAS.

Generic Interrupt Controller (GIC)

The GIC-600 is an Arm architecture compliant System-on-Chip (SoC) peripheral. It is a high-performance, area-optimized interrupt controller. The GIC implements the Arm generic interrupt controller architecture. The GIC takes interrupts asserted at the system level and signals them to each connected processor as appropriate. The GIC has the following features.

- Registers for managing interrupt sources, interrupt behavior, and interrupt routing to one or more processors

- Support for the Arm architecture security extensions
- The ability to enable, disable, and generate processor interrupts from hardware (peripheral) interrupt sources
- The ability to generate software interrupts
- Support for interrupt masking and prioritization

Refer to the GIC Overview section for more information on the processor-specific configuration of the GIC.

The GIC-600 contains a performance monitoring unit (PMU) for counting key GIC events from the distributor.

Refer to the GIC Performance Monitoring Unit (GIC PMU) section for more information.

A55 Core in DynamIQ Shared Unit (DSU)

The ADSP-SC844/SC846 processor implements a dual A55 cores (LITTLE core) with L1 and L2 cache inside the DynamIQ cluster, which has a shared-L3 cache and snoop control unit (SCU) with snoop filter. Both A55 cores are configured to run synchronously with the DSU.

See the *Arm DynamIQ™ Shared Unit Technical Reference Manual* for further details on DSU.

System Control

The system registers control and provide status information for the functions that the core implements. The main functions of the system registers are:

- Overall system control and configuration
- MMU configuration and management
- Cache configuration and management
- System performance monitoring
- GIC configuration and management

The system registers are accessible in the AArch64 and AArch32 execution states.

Generic Timer

The generic timer can schedule events and trigger interrupts that are based on an incrementing counter value. It generates timer events as active-low interrupt outputs and event streams. The timer contains:

- An EL1 non-secure physical timer
- An EL2 hypervisor physical timer
- An EL3 secure physical timer
- A virtual timer
- A hypervisor virtual timer

The Cortex-A55 core does not include the system counter. This resides in the SoC. The system counter value is distributed to the core with a 64-bit bus.

See the *Arm DynamIQ Shared Unit Technical Reference Manual* and the *Arm Architecture Reference Manual Armv8*, for Armv8-A architecture profile for more information on the generic timer.

Generic Timer Interrupts

Timers can be configured to generate an interrupt. The active-low, level sensitive timer interrupt signals are routed to the GIC as a private peripheral interrupt (PPI).

A55 Configurations

The ***Core Configuration*** and ***DSU Configuration*** tables describe the Cortex A55 and processor configurations.

Table 1-2: Core Configuration

Core Feature	Comment
L1 instruction cache size	32 KB
L1 data cache size	32 KB
L2 cache	Included
L2 cache size	256 KB
ECC or parity core cache protection	Included
Advanced SIMD and floating-point support (including Dot Product instruction support)	Included
Cryptographic Extension	Included
CoreSight Embedded Logic Analyzer (ELA)	False
CoreSight ELA RAM address size	N/A
Page Based Hardware Attributes (PBHA) support	Not included

Table 1-3: DSU Configuration

Core Feature	Comment
NUM_BIG_CORES	0
NUM_LITTLE_CORES	2
NUM_OTHER_CORES	0
BIG_CORE_TYPE	N/A
LITTLE_CORE_TYPE	Ananke
OTHER_CORE_TYPE	N/A
MODULE	N/A
MODULE_DEBUG_BLOCK	N/A

Table 1-3: DSU Configuration (Continued)

Core Feature	Comment
ACE	True
REQUESTER_DATA_WIDTH	128
ACP	False
PERIPH_PORT	True
SCU_CACHE_PROTECTION	True
L3_CACHE	True

DynamIQ Shared Unit Interfaces

This section describes the DSU interface signals.

Clock and Clock Enable Signals

The ***Clock Signals*** and ***Clock Enable Signals*** tables describe the clock and clock enable signals.

Table 1-4: Clock Signals

Signal Name	Description	Connection
SCLK	Clock for the SCU/L3 and the AMBA interface	CLKO1
PCLK	Clock for the debug peripheral interface and the timers, power management, and other miscellaneous logic	SYSCLK
ATCLK	Clock for the ATB trace interface	SYSCLK
GICCLK	Clock for the GIC interface	SYSCLK

The *Default Configuration* table describe the default configuration of the dual Cortex-A55.

Table 1-5: Default Configuration

Configuration	Status
Default Execution Mode in EL3	AArch64
Endianness configuration, controls the reset value of the SCTLR_EL3/SCTLR EE bit	Little Endian
Enable Thumb exceptions, controls the reset value of the SCTLR.TE bit	Enabled
Value read in ClusterID Affinity Level-2 field, MPIDR bits [23:16]	8'h0
Value read in ClusterID Affinity Level-3 field, MPIDR bits [39:32]	8'h0
Disables the Cryptographic Extensions	False
Globally disables the CPU interface logic and routes the external interrupt signals directly to the cores	False

GIC Signals

This section describes the GIC default configuration.

Table 1-6: GIC Default Configuration (Interrupts)

Signal Name	Description	Connection
$\overline{\text{FIQ}}$	Active-low, level-sensitive fast interrupt request	Disabled
$\overline{\text{IRQ}}$	Active-low, level-sensitive interrupt request	Disabled
$\overline{\text{VFIQ}}$	Active-low, level-sensitive virtual fast interrupt request	Disabled
$\overline{\text{VIRQ}}$	Active-low, level-sensitive virtual interrupt request	Disabled
IRIT I/F	Distributor to GIC CPU Interface messages	From GIC
ICC I/F	GIC CPU Interface to distributor messages	To GIC

WFE Wakeup Event Signals

This section describes the wakeup event signals

Table 1-7: WFE Wakeup Event Signals

Signal Name	Description	TRU Requester/Completer ID
EVENTIREQ	Event input request for core wake-up from WFE state. It must remain asserted until EVENTIACK is asserted and must not be re-asserted until EVENTIACK is low.	TRU Completer 116
EVENTIACK	Event input request acknowledge. It is not asserted until EVENTIREQ is high, and then remains asserted until after EVENTIREQ goes low.	TRU Requester 69
EVENTOREQ	Event output request for core wake-up, triggered by SEV instruction. It is only asserted when EVENTOACK is low, and then remains high until after EVENTOACK goes high.	TRU Requester 70
EVENTOACK	Event output request acknowledge. It must not be asserted until EVENTOREQ is high, and then must remain asserted until after EVENTOREQ goes low.	TRU Completer 117

Error Signals

This section describes the error signals

Table 1-8: Error Signals

Signal Name	Description
$\overline{\text{ERRIRQ[0]}}$ A55_SCU_ERR (A55 SCU Error)	Active-low, level-sensitive error indicator for an ECC error that causes potential data corruption or loss of coherency in L3 and snoop filter RAMs or ACE write transactions with a write response condition. It is output from the SCLK domain.

Table 1-8: Error Signals (Continued)

Signal Name	Description
$\overline{\text{ERRIRQ}}$[2:1] A55_L1L2_ERR (A55 L1,L2 Error)	Active-low, level-sensitive error indicator for an ECC error that causes potential data corruption or loss of coherency for the L1 and L2 caches in the respective core. The bits are output from the PERIPHCLK domain.

NOTE: The bit[1] is for core 0 and bit[2] for core 1

Debug Block APB Interface Signals (Debug)

The interface between the debug block and the DSU consists of a pair of APB interfaces, one in each direction. The following table shows the signals between the cluster APB controller and the debug block APB target.

Table 1-9: Debug Block APB Interface Signals (Debug)

Signal Name	Description
$\overline{\text{COMMIRQ}}$	Active-low, level-sensitive comms channel receive or transmit interrupt request
DBGEN	Invasive debug enable
NIDEN	Non-invasive debug enable
SPIDEN	Secure privilege invasive debug enable
SPNIDEN	Secure privilege non-invasive debug enable

Performance Monitor Unit (PMU)

The Cortex-A55 core includes performance monitors that can be configured to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the code that can used for debug and code profiling. The PMU provides six counters. Each counter can count any of the events available in the core.

For complete register information refer to the *Arm Cortex-A55 Core Technical Reference Manual* and the *Arm Cortex-A Series Programmer's Guide for ARMv8-A*.

Functional Description

This section describes the functionality of the PMU. The PMU includes the following interfaces and counters:

- Events from all other units from across the design are provided to the PMU
- The PMU registers can be programmed using the system registers or the external peripheral interface
- The PMU has 32-bit counters that increment when they are enabled, based on events, and a 64-bit cycle counter
- The Cortex-A55 core supports access to the performance monitor registers from the internal system register interface and a memory-mapped interface

PMU Events

The PMU event counters can gather various statistics on the operation of the core and its memory system, like cycle count, cache miss/hit, memory read/write, load and prefetch instructions. For the all the PMU events, please refer to *Arm Cortex-A55 Core Technical Reference Manual*.

PMU Interrupts

The Cortex-A55 core asserts the nPMUIRQ signal when the PMU generates an interrupt. The active-low, level sensitive PMU interrupt signal is routed to the GIC as a private peripheral interrupt.

Interrupt ID	Name	Description	Sensitivity
16	PMU_IRQ	PMU interrupt request	Level

2 Clock Generation Unit (CGU)

The Clock Generation Unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a requester clock that runs at a frequency that is a multiple of the CLKIN input clock frequency. The PCU divides down the requester clock to generate various system clocks and synchronization signals.

CGU Features

The CGU module supports the following features:

- Provides smooth transitions from the current clock condition to a new condition with PLL logic and executes the changes to clocks due to register programming.
- Provides PLL and clock domain status reporting for event management.
- Supports the capability to bypass the PLL for power savings.
- Software controlled dynamic power management allows control of the core clock frequency (f_{CCLK}).
- Controls clock gating of the core and system clocks.

NOTE: For more information about processor-specific CGU features, see the processor data sheet.

CGU Functional Description

The CGU generates all on-chip clocks and synchronization signals based on the programmed PLL multiplication factor and dividers. The CGU provides the following functionality.

Change the PLL Clock Frequency

The CGU allows programs to change the PLL clock frequency by writing new values to bits in the control register. Any time the PLL relocks, the CGU aligns all core and system clocks.

Change Other Clock Frequencies

The CGU allows programs to change the CCLKn, SYSCLK, SCLKn, DCLK, and OCLK frequencies by writing values to the CGU_DIV register. Any time the clock frequency is changed, the OCLK, CCLKn, SYSCLK, DCLK, and SCLKn clocks exit the frequency change sequence aligned.

Perform Clock Alignment

The CGU can align all clocks by writing to the CGU_DIV register. This function aligns all PLL-based clocks.

For more information on these functions, see the CGU Programming Model section.

ADSP-2184x CGU Register List

The clock generation unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a clock, running at a frequency that is a multiple of the CLKIN input clock's frequency. The CGU also generates all on-chip clocks and synchronization signals. The PCU permits application software control of the PLL's operation. A set of registers govern CGU operations. For more information on CGU functionality, see the CGU register descriptions.

Table 2-1: ADSP-2184x CGU Register List

Name	Description
CGU_CCBF_DIS	Core Clock Buffer Disable Register
CGU_CCBF_STAT	Core Clock Buffer Status Register
CGU_CLKOUTSEL	CLKOUT Select Register
CGU_CTL	Control Register
CGU_DIV	Clocks Divisor Register
CGU_DIVEX	DIV Register Extension
CGU_PLLCTL	PLL Control Register
CGU_REVID	Revision ID Register
CGU_SCBF_DIS	System Clock Buffer Disable Register
CGU_SCBF_STAT	System Clock Buffer Status Register
CGU_STAT	Status Register
CGU_TSCOUNT0	Time Stamp Counter 32 LSB Register
CGU_TSCOUNT1	Time Stamp Counter 32 MSB Register
CGU_TSCTL	Time Stamp Control Register
CGU_TSVALUE0	Time Stamp Counter Initial 32 LSB Value Register
CGU_TSVALUE1	Time Stamp Counter Initial MSB Value Register

ADSP-2184x CGU Interrupt List

Table 2-2: ADSP-2184x CGU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
7	CGU0_EVT	CGU0 Event	Edge	
8	CGU1_EVT	CGU1 Event	Edge	

ADSP-2184x CGU Trigger List

Table 2-3: ADSP-2184x CGU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
3	`CGU0_EVT`	CGU0 Event	Edge
4	`CGU1_EVT`	CGU1 Event	Edge

Table 2-4: ADSP-2184x CGU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

CGU Definitions

DPM

The Dynamic Power Management module (DPM) works with the CGU to provide flexible power dissipation modes for the processor.

PCU

The PLL control unit (PCU) in the CGU controls PLL operations. The MMR registers of the CGU are implemented in the PCU.

PLL

The phase-locked loop (PLL) operates within the CGU.

RCU

The reset control unit (RCU) provides input to the CGU to manage clocks during processor reset.

CDU

The clock distribution unit distributes the clocks from the CGU to different clock domains

CGU

The clock generation unit (CGU) is comprised of the PLL and PCU. The CGU generates the clocks listed in the ***Clock Descriptions*** table.

Table 2-5: Clock Descriptions

Clock	Description
CCLK0_0	CCLK0 derived from CGU0
SYSCLK_0	SYSCLK derived from CGU0
SCLK0_0	SCLK0 derived from CGU0
SCLK1_0	SCLK1 derived from CGU0
DCLK0_0	DCLK derived from CGU0
OCLK_0	OCLK derived from CGU0
CCLK2_0	CCLK (VCO/3) derived from VCO clock of CGU0
DCLK1_0	DCLK1(VCO/3) derived from VCO clock of CGU0
CCLK0_1	CCLK0 derived from CGU1
SYSCLK_1	SYSCLK derived from CGU1
SCLK0_1	SCLK0 derived from CGU1
SCLK1_1	SCLK1 derived from CGU1
DCLK_1	DCLK derived from CGU1
OCLK_1	OCLK derived from CGU1
CCLK2_1	CCLK (VCO/3) derived from VCO clock of CGU1
DCLK1_1	DCLK1(VCO/3) derived from VCO clock of CGU1

CGU PLL Block Diagram

The ***CGU PLL Block Diagram*** provides a top-level block diagram of the phase locked loop (PLL). The main blocks of the PLL are the phase/frequency detector (PFD), the charge pump, the loop filter and the voltage controlled oscillator (VCO). The VCO multiplies the `SYS_CLKIN0` input to a higher frequency.

Additional configuration options are configured using the Clock Distribution Unit.

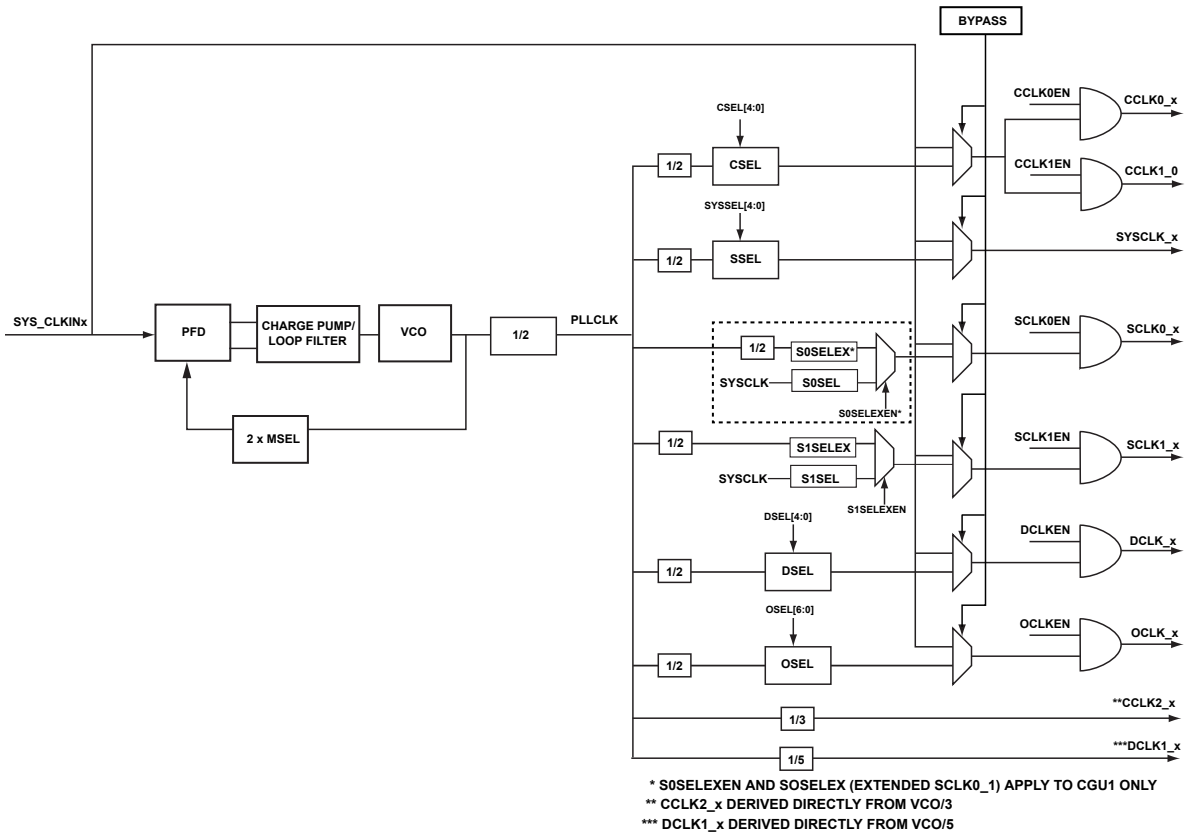


Figure 2-1: CGU PLL Block Diagram

CLKOUT Selections

The ***SYS_CLKOUT Generation*** figure is a conceptual representation of the CLKOUT module. Different clocks that originate from the CGU blocks are available on the SYS_CLKOUT output pin. The selection of the clock output on the SYS_CLKOUT pin is controlled by the `CGU_CLKOUTSEL.CLKOUTSEL` bit field. Most of the clock frequencies are divided with a fixed divider value, for example CGU0_SYSCLK0 has a fixed divider of 8 for the CLKOUT pin.

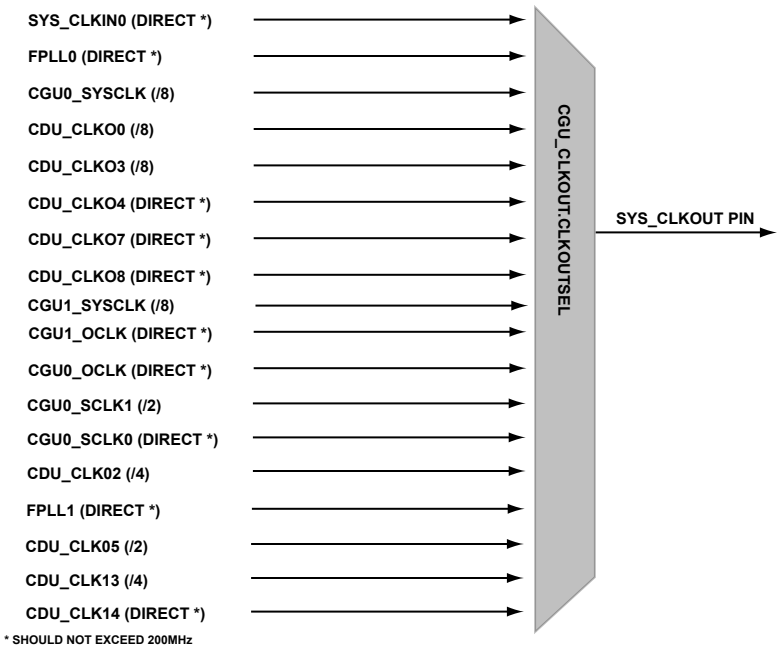


Figure 2-2: SYS_CLKOUT Generation

The BMODE pin affects the CLKOUT pins after a hardware reset is deasserted.

BMODE= 0 — When a hardware reset is deasserted, CGU0_OCLK is selected to come out of the CLKOUT pin. While programming all values of the `CGU_CLKOUTSEL.CLKOUTSEL` bit field is possible, changing it to 0 (SYS_CLKIN) is not possible until and unless a non-zero value is first programmed to the the `CGU_CLKOUTSEL.CLKOUTSEL` bit field.

BMODE = (non zero) — When a hardware reset is deasserted, SYS_CLKIN is selected by default.

CGU Operating Modes

The CGU does not have configurable operating modes, but CGU operations affect the operating modes of other modules. Some CGU operation issues that affect operation of other modules include the following:

- The PLL of the CGU operates in either normal mode (CGU clock divisors applied) or bypass mode (CGU PLL is bypassed and clock divisors ignored).
- The SCB uses the CGU for clock synchronization across clock domains. For more information, see System Crossbars (SCB).
- The DPM uses the CGU for clock management as power state transitions occur. For more information, see the Dynamic Power Management (DPM) chapter.
- The CGU uses clock gating control to obtain flexible low-power modes.

CGU Power-up Sequence

See the product data sheet for exact power-up requirements. The processor is configured to come up in clock bypass mode. The programs is required to configure full speed clocks and safety monitors. SYS_CLKIN0 and all supplies should be stable before the SYS_HWRST signal is deasserted.

CGU Event Control

The CGU generates an event or error for several different reasons. Events and errors are described in the following sections.

Events

After a frequency change, a CGU event indicates that the PLL has locked, and clocks are synchronized. If a core was idled while changing frequencies, the CGU can use an event interrupt to break the core idle. While in active mode, a CGU event indicates that the PLL has locked.

CGU Errors

A CGU error occurs under following conditions:

- A write access to the CGU_DIV/CGU_DIVEX registers trigger an alignment sequence while the PLL is locked and is still aligning the clocks.

  The CGU_STAT.WDIVERR bit state indicates this error. If this error occurs, clear the CGU_STAT.WDIVERR bit and rewrite the desired values to the CGU_DIV/CGU_DIVEX registers.

- A write access to the CGU_CTL.MSEL bit field occurs while the PLL is locking.

  The CGU_STAT.WDFMSERR bit state indicates this error. If this error occurs, wait until the PLL has finished locking, cleared the error, and rewritten the desired value change.

CGU Generated Bus Errors

The CGU generates a bus error when a read or write transaction to an unused address within the CGU address range is attempted. It also generates a bus error when a misaligned access is made to a CGU register. In addition to the bus error, the CGU_STAT.ADDRERR bit is set. If a write to a write-protected CGU register is attempted, the CGU generates a bus error. In addition, the CGU_STAT.LWERR bit is set.

CGU Programming Model

The programming model for the CGU involves the various mode configuration techniques.

Configuring CGU Modes

Use the following procedures to configure the clocks and PLL.

NOTE: The program needs to clear the CGU_STAT.CLKSALGN bit before changing clocks. The following sequence is executed once, inside the application, after coming out of reset.

```
*pREG_CGU0_PLLCTL |= BITM_CGU_PLLCTL_PLLBPCL; // come out of bypass and enter
Full ON
while( (pADI_CGU0 ->STAT & 0xF) != 0x5 ) { } // poll
// now clocks are running with hardware default divisors.
// now program can change frequencies If desired the program can put the PLL
again into bypass.
```

Changing the Clock Frequencies

Applications change clock frequencies in two ways. The first way is modifying the PLL multiplication value by writing to the CGU_CTL register and the second is modifying the clock dividers by writing to the CGU_DIV register. Both actions have different implications even if the frequencies of the final clock are the same. Write accesses to change the CGU_CTL.MSEL bit field while the PLL is locking set the CGU_STAT.WDFMSERR error bit. The CGU_STAT.WDIVERR error bit is set when one of following accesses is attempted while the PLL is locked, but still aligning the clocks:

- A write access to the CGU_DIV/CGU_DIVEX to trigger an alignment sequence.
- A write access to the CGU_DIV/CGU_DIVEX to change the CGU_DIV.CSEL/, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, or CGU_DIV.DSEL/CGU_DIVEX.S1SELEX bits.

Read-after-write accesses to these registers return the new value, even if the frequency of the clock change is still in-progress.

Modifying the PLL multiplier requires the PLL to relock. Once the PLL locks, the CGU synchronizes the clocks. Changes to the CGU_CTL.MSEL bit field result in bypassing the PLL. By setting the CGU_CTL.WFI bit, programs force the PLL to wait for the core to return to the idle or reset state before the frequency changes. If necessary, clear the CGU_DIV.UPDT bit to avoid multiple clock alignment sequences. If the CGU_DIV register is not updated, the CGU uses the current values to determine the frequencies of the clock. It is the programs responsibility to guarantee that the new CGU_CTL.MSEL and CGU_DIV/CGU_DIVEX combinations are legal.

Changing the PLL Clock Frequency

To change the phase-locked loop clock (*PLLCLK*) frequency, write new values to the CGU_CTL.MSEL field. Any time the PLL relocks, all core and system clocks are aligned.

1. Read CGU_STAT register and verify that:
    a. The CGU_STAT.PLLEN bit =1 (PLL enabled)
    b. The CGU_STAT.PLOCK bit =1 (PLL is not locking)
    c. The CGU_STAT.CLKSALGN bit =0 (clocks aligned)

2. Write the desired values to the clock divisor select fields of the CGU_DIV/CGU_DIVEX register with the CGU_DIV.UPDT bit =0.
3. Write the desired values to the CGU_CTL.MSEL field.
    a. To change the PLL frequency while the core is idle, write to the CGU_CTL register with the CGU_CTL.WFI bit =1.
    b. To change the PLL frequency while the core is active, write to the CGU_CTL register with the CGU_CTL.WFI bit =0.

This sequence performs these actions:

1. Updates the corresponding CGU registers.
2. Bypasses the PLL.
3. Makes the PLL lock to the new values in the CGU_CTL.MSEL field.
4. Changes the clock frequencies.
5. Exits the PLL bypass with all clocks aligned.

When exiting the PLL bypass state, a CGU event occurs.

The CGU_STAT register exits this sequence with the CGU_STAT.PLLEN bit =1, the CGU_STAT.PLOCK bit =1, the CGU_STAT.PLLBP bit =0, and the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.PLOCK bit, CGU_STAT.PLLBP bit, and CGU_STAT.CLKSALGN bit to discover when the PLL is locked and the clocks are aligned.

Changing the frequency of the PLL is allowed while the PLL is bypassed. In this case the new PLLCLK frequency is not used until the PLL is no longer bypassed.

Changing the CCLKn or SYSCLK Frequency Without Modifying the PLLCLK Frequency

To change the clock frequencies, write new values to CGU_DIV.CSEL or CGU_DIV.SYSSEL bits. The frequency change occurs only when the PLL is not bypassed. Any time the CCLKn or SYSCLK clock frequencies are changed, they exit the frequency change sequence aligned.

1. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write the desired CGU_DIV.CSEL, CGU_DIV.SYSSEL, and CGU_DIV.OSEL bitfield values with the CGU_DIV.UPDT bit = 1 and/or, write to the CGU_DIVEX.S1SELEX field. This write updates the CGU_DIV register, changes the SCLKn and SYSCLK frequencies, and aligns the clocks. When the clocks are aligned, a CGU event occurs.

The CGU_STAT register exits this sequence with the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.CLKSALGN bit to discover when the clocks are aligned. Any write attempt to change the CGU_DIV.S0SEL or CGU_DIV.S1SEL bit fields while CGU_STAT.CLKSALGN bit =1 (clocks alignment in progress) triggers an MMR access bus error and the CGU_DIV register is not modified.

Also, any write attempt to change the CGU_DIVEX.S1SELEX field while the CGU_STAT.CLKSALGN bit =1 (clocks alignment in progress) triggers an MMR access bus error and the CGU_DIVEX register is not modified.

Programming the SYSCLK frequency to a higher value than CCLKn also triggers an MMR access bus error and the CGU_DIV/CGU_DIVEX register is not modified.

Writing to the CGU_DIV/CGU_DIVEX register is allowed while the processor is in active (PLL bypassed) mode. In this case the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

Accessing the DDR memory while changing the SYSCLK frequency is not supported and can have unpredictable results.

Changing the OCLK Frequency

To change the OCLK clock frequency, write a new CGU_DIV.OSEL bit value. Any time the OCLK clock frequency is changed, the OCLK, CCLKn, SYSCLK, and SCLKn clocks exit the frequency change sequence aligned.

1. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write the desired CGU_DIV.OSEL value with the CGU_DIV.UPDT bit =1. This write updates the CGU_DIV register, changes the OCLK frequency, and aligns all clocks except OCLK.

The CGU_STAT register exits this sequence with the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.CLKSALGN bit to discover when the clocks are aligned. Any write attempt to change the CGU_DIV.DSEL field while the CGU_STAT.CLKSALGN bit =1 (clock alignment in progress) triggers an MMR access bus error and the CGU_DIV/CGU_DIVEX is not modified. When the clocks are aligned, a CGU event occurs.

Writing to the CGU_DIV.OSEL bit field is allowed while the processor is in active (PLL bypassed) mode. In this case the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

Selecting SCLK1_0 Source

SCLK1_0 sources the SPDIF receiver clock. This can be sourced from PLL clock or SYSCLK0_0. Programming the CGU_DIVEX enables PLL clock as source (bypassing SYSCLK0_0 that continues to source other peripherals). When CGU_DIVEX is enabled, the PLL clock is divided from CGU_DIVEX register value.

Changing SCLK1_0 Frequencies

Complete the following steps to change the SCLK0 or SCLK1 frequency.

1. Read the CGU_STAT register. Verify that CGU_STAT.CLKSALGN = 0 (Clocks aligned).
2. Read the CGU_CTL register. Verify that the CGU_CTL.S1SELEXEN bit is set as needed. (These bits can only be modified when the PLL is bypassed).

3. Read the CGU_STAT register. Verify that the PLL is not bypassed. (If it is bypassed, see PLL Bypass and PLL Disable topic.)
4. Read the CGU_DIV register. Verify that the CGU_DIV.UPDT bit is set.
5. Write the desired CGU_DIVEX.S1SELEX value to the CGU_DIVEX registers .

Aligning All Clocks

To align the clocks, write 1 to the CGU_DIV.ALGN bit. The frequency can also be changed, if necessary. The clocks aligned include:

- CCLKn
- SYSCLK
- SCLKn
- DCLK
- OCLK

1. Read the CGU_STAT register to verify that CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write 1 to the CGU_DIV.ALGN bit. All other fields can change.

   ADDITIONAL INFORMATION: This write does not alter the CGU_DIV/CGU_DIVEX register unless one of the clock-select fields is modified. When the clocks are aligned, a CGU event occurs.

The CGU_STAT register exits this sequence with the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.CLKSALGN bit to discover when the clocks are aligned. Any write to the CGU_DIV/CGU_DIVEX register intended to align clocks or to change a clock select field while the CGU_STAT.CLKSALGN bit =1 (clocks alignment in progress) triggers an MMR access bus error. In this case, the CGU_DIV/CGU_DIVEX register is not modified.

Writing 1 to the CGU_DIV.ALGN bit has no effect while the processor is in active (PLL bypassed) mode.

Valid Clock Multiplier Settings

Processor operations depend on valid settings in the CGU_CTL and CGU_DIV registers. These registers control the clock multiplier and divisor values. Set these registers such that the minimum and maximum clocks specified in the data sheet are not violated. All other clock specifications in the data sheet must also be adhered to for correct operation of the processor.

PLL Bypass and PLL Disable

Writing 1 to the CGU_PLLCTL.PLLBPST bit tells the PLL to apply OSC_CLKIN clock to CCLK, SYSCLK, SCLK0, SCLK1, DCLK (PLL Bypass), and OCLK outputs. Writing 1 to the CGU_PLLCTL.PLLBPCL bit

tells the PLL to exit its PLL Bypass state and make all output clocks align and transition to their programmed frequencies.

The PLL can be disabled by clearing the CGU_PLLCTL.PLLEN bit while in the bypass state. If necessary, clock buffers can be disabled.

CCLK0 clock can be disabled or enabled by writing 1 or 0 to the corresponding bit in the CGU_CCBF_DIS register.

To determine which core clock buffers were disabled since the last read, software reads the CGU_CCBF_STAT register. The SCLK0, SCLK1, DCLK and OCLK clocks can be disabled or enabled by writing 1 or 0 to the corresponding bit in the CGU_SCBF_DIS register. Software cannot disable SYSCLK.

Processor Specific Information

The processor has one system crystal oscillator and two system CGU units to provide the clocks to the system. Both of the CGUs come up in bypass mode out of reset.

CGU0 is the main CGU which provides most of the clocks including CCLK0, SYSCLK (SYSCLK_0), SCLK0 (SCLK0_0), and SCLK1 (SCKL1_0) to the system buses, infrastructure, and most of the peripherals. The DCLK source can be selected between CGU0 and CGU1, through CDU0.

For details, refer to the Clock Distribution Unit (CDU).

NOTE: The frequency ratio of the core clock to SYSCLK is 2:1.

The frequency ratios of SYSCLK to SCLK0 are 6:1, 4:1 or 2:1.

The processor does not support any other frequency ratio.

ADSP-2184x CGU Register Descriptions

Clock Generation Unit (CGU) contains the following registers.

Table 2-6: ADSP-2184x CGU Register List

Name	Description
CGU_CCBF_DIS	Core Clock Buffer Disable Register
CGU_CCBF_STAT	Core Clock Buffer Status Register
CGU_CLKOUTSEL	CLKOUT Select Register
CGU_CTL	Control Register
CGU_DIV	Clocks Divisor Register
CGU_DIVEX	DIV Register Extension
CGU_PLLCTL	PLL Control Register
CGU_REVID	Revision ID Register

Table 2-6: ADSP-2184x CGU Register List (Continued)

Name	Description
CGU_SCBF_DIS	System Clock Buffer Disable Register
CGU_SCBF_STAT	System Clock Buffer Status Register
CGU_STAT	Status Register
CGU_TSCOUNT0	Time Stamp Counter 32 LSB Register
CGU_TSCOUNT1	Time Stamp Counter 32 MSB Register
CGU_TSCTL	Time Stamp Control Register
CGU_TSVALUE0	Time Stamp Counter Initial 32 LSB Value Register
CGU_TSVALUE1	Time Stamp Counter Initial MSB Value Register

Core Clock Buffer Disable Register

The CGU_CCBF_DIS register controls each core's clock buffer to determine if the CCLK is enabled.

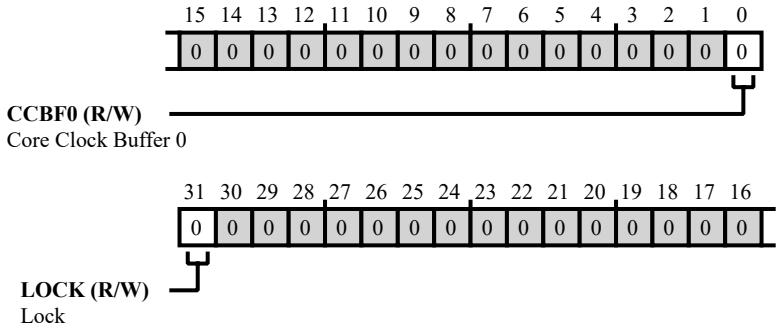


Figure 2-3: CGU_CCBF_DIS Register Diagram

Table 2-7: CGU_CCBF_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If set (=1) the CGU_CCBF_DIS.LOCK bit locks the CGU_CCBF_DIS register.	
		0	Unlock register
		1	Lock register
0 (R/W)	CCBF0	Core Clock Buffer 0. The CGU_CCBF_DIS.CCBF0 bit enables (=0) or disables (=1) CCLK0s buffer.	
		0	Enable buffer
		1	Disable buffer

Core Clock Buffer Status Register

The CGU_CCBF_STAT register shows which core clock buffer(s) are disabled. For example clearing the CGU_CCBF_DIS.CCBF0 bit clears the CGU_CCBF_STAT.CCBF0 bit after a number of cycles. To guarantee that the correct value is read, this register should be read twice and the second result used.

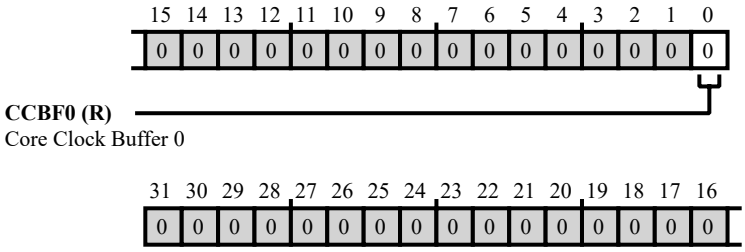


Figure 2-4: CGU_CCBF_STAT Register Diagram

Table 2-8: CGU_CCBF_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	CCBF0	Core Clock Buffer 0. The CGU_CCBF_STAT.CCBF0 bit reports the status of the CGU_CCBF_DIS.CCBF0 bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled

CLKOUT Select Register

The CGU_CLKOUTSEL selects the signal that the CGU drives through the CLKOUT multiplexer.

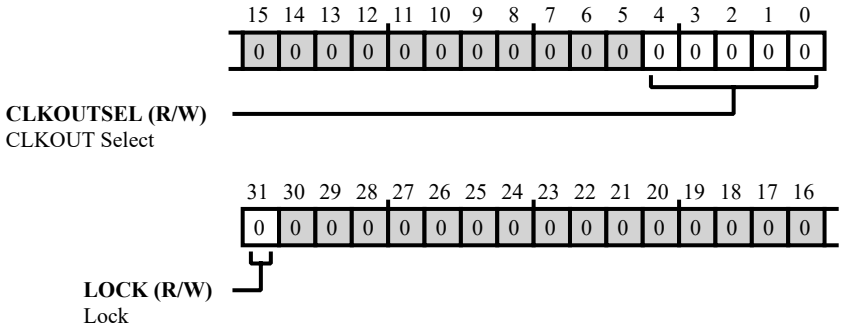


Figure 2-5: CGU_CLKOUTSEL Register Diagram

Table 2-9: CGU_CLKOUTSEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the CGU_CLKOUTSEL.LOCK bit is set, the CGU_CLKOUTSEL register is read only (locked).	
		0	Unlock
		1	Lock
4:0 (R/W)	CLKOUTSEL	CLKOUT Select. The CGU_CLKOUTSEL.CLKOUTSEL selects the signal that the CGU drives through the CLKOUT pin multiplexer.	
		0	CLKIN0
		1	FPLL0 CLKOUT
		2	PLLTOP0.SYSCLKO
		3	PLLTOP0.CCLK0
		4	CDU.CLKO3
		5	CDU.CLKO4
		6	CDU.CLKO7
		7	CDU.CLKO8
		8	PLLTOP1.SYSCLKO
		9	PLLTOP1.OCLK
		10	PLLTOP0.OCLK

Table 2-9: CGU_CLKOUTSEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		11	Reserved
		13	Reserved
		14	PLLTOP0.SCLK1
		15	Reserved
		16	PLLTOP0.SCLK0
		18	Reserved
		19	CDU.CLKO2
		20	FPLL1 CLKOUT
		21	CDU.CLKO5
		22	CDU.CLKO13
		24	CDU.CLKO14
		25	Reserved
		26	Reserved
		27	Reserved
		28	Reserved
		29	Reserved
		30	GND
		31	GND

Control Register

The CGU_CTL controls the clock generation divisors for SYS_CLKIN and the PLL. Read after write accesses to the CGU_CTL register returns the new value even if the clock's frequency change is still in progress.

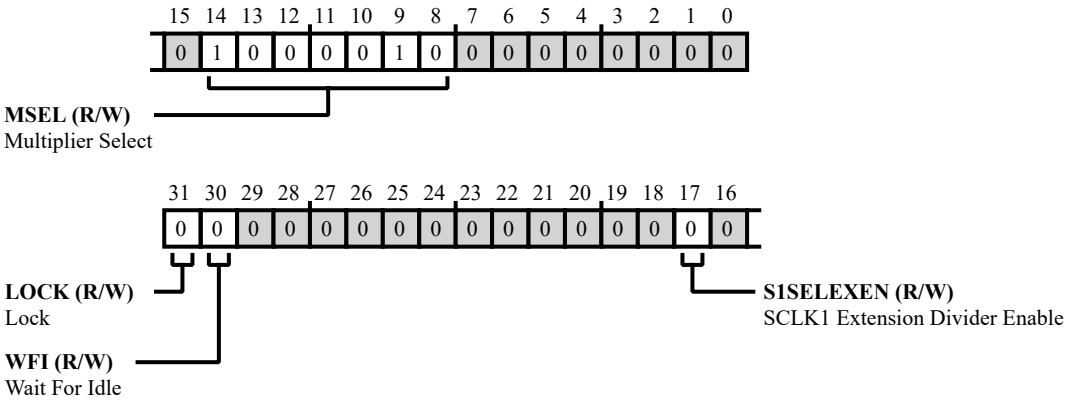


Figure 2-6: CGU_CTL Register Diagram

Table 2-10: CGU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the CGU_CTL.LOCK bit is set, the CGU_CTL register is read only (locked).	
		0	Unlock
		1	Lock
30 (R/W)	WFI	Wait For Idle. Modifying the PLL multiplier requires the PLL to re-lock and once the PLL locks, clocks have to be synchronized. Changes to the CGU_CTL.MSEL bit values results in bypassing the PLL. The CGU_CTL.WFI bit forces the PLL to wait for all processor cores to be in an idle or reset state before changing frequencies as a result of changes to the CGU_CTL.MSEL bits. Write accesses to the CGU_CTL to change the CGU_CTL.MSEL bit values while the PLL is locking sets the CGU_STAT.WDFMSERR bit.	
		0	Update Immediately
		1	Wait for Idle

Table 2-10: CGU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	S1SELEXEN	SCLK1 Extension Divider Enable. The `CGU_CTL.S1SELEXEN` directs the CGU to select S1SELEX instead of S1SEL. This bit can only be updated when the PLL is bypassed.	
		0	Selects SYSCLK to SCLK1 Divider
		1	Selects PLLCLK to SCLK1 Divider
16 (R/W)	S0SELEXEN	SCLK0 Extension Divider Enable. The `CGU_CTL.S0SELEXEN` directs the CGU to select S0SELEX instead of S0SEL. This bit can only be updated when the PLL is bypassed.	
		0	Selects SYSCLK to SCLK0 Divider
		1	Selects PLLCLK to SCLK0 Divider
14:8 (R/W)	MSEL	Multiplier Select. The `CGU_CTL.MSEL` bit field selects the multiplier in the PLLCLK equation: PLLCLK frequency = (SYS_CLKIN frequency * MSEL) Where the value of MSEL is between 1 and 127. When the processor comes out of reset, the default multiplier value is 40.	
		0	MSEL = 128
		1-127	MSEL = 1 to 127

Clocks Divisor Register

The CGU_DIV register controls clock divisors for core clocks, system clocks, external (off core) memory clocks, and output clock. Read after write accesses to the CGU_DIV register returns the new value even if the clock's frequency change is still in progress.

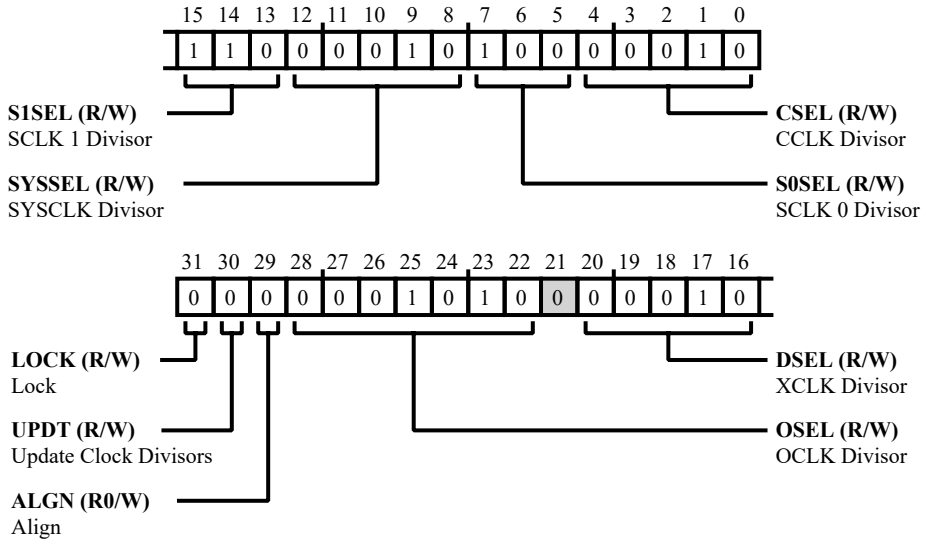


Figure 2-7: CGU_DIV Register Diagram

Table 2-11: CGU_DIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the CGU_DIV.LOCK bit is set, the CGU0_DIV and CGU0_DIVEX registers are read only (locked).	
		0	Unlock
		1	Lock
30 (R/W)	UPDT	Update Clock Divisors. The CGU_DIV.UPDT controls whether the CGU drives new CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, and CGU_DIV.OSEL values to PLL after CGU_DIV register update.	
		0	No PLL Update
		1	Drive Updated SEL Values to PLL

Table 2-11: CGU_DIV Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R0/W)	ALGN	Align. The CGU_DIV.ALGN directs the CGU to align the PLL-based clocks. The divisor selections (CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, and/or CGU_DIV.OSEL) do not have to change.	
		0	No Action
		1	Align PLL Clocks
28:22 (R/W)	OSEL	OCLK Divisor. The CGU_DIV.OSEL selects the divisor in the OCLK equation: OCLK frequency = (SYS_CLKIN frequency * MSEL) / (2 * CGU_DIV.OSEL) Where the value of CGU_DIV.OSEL is between 1 and 127.	
		0	OSEL = 128
		1-127	OSEL = 1 to 127
20:16 (R/W)	DSEL	XCLK Divisor. The CGU_DIV.DSEL selects the divisor in the XCLK equation: XCLK frequency = (SYS_CLKIN frequency * MSEL) / (2 * CGU_DIV.DSEL) Where the value of CGU_DIV.DSEL is between 1 and 31.	
		0	DSEL = 32
		1-31	DSEL = 1 to 31
15:13 (R/W)	S1SEL	SCLK 1 Divisor. The CGU_DIV.S1SEL selects the divisor in the SCLK1 equation: SCLK1 frequency = (SYSCLK frequency) / CGU_DIV.S1SEL Where the value of CGU_DIV.S1SEL is between 1 and 7.	
		0	S1SEL = 8
		1-7	S1SEL = 1 to 7
12:8 (R/W)	SYSSEL	SYSCLK Divisor. The CGU_DIV.SYSSEL selects the divisor in the SYSCLK equation: SYSCLK frequency = (SYS_CLKIN frequency * MSEL) / (2 * CGU_DIV.SYSSEL) Where the value of CGU_DIV.SYSSEL is between 1 and 31.	
		0	SYSSEL = 32
		1-31	SYSSEL = 1 to 31

Table 2-11: CGU_DIV Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:5 (R/W)	S0SEL	SCLK 0 Divisor. The CGU_DIV.S0SEL selects the divisor in the SCLK0 equation: SCLK0 frequency = (SYSCLK frequency) / CGU_DIV.S0SEL Where the value of CGU_DIV.S0SEL is between 1 and 7.	
		0	S0SEL = 8
		1-7	S0SEL = 1 to 7
4:0 (R/W)	CSEL	CCLK Divisor. The CGU_DIV.CSEL bit field selects the divisor in the CCLK equation: CCLK frequency = (SYS_CLKIN frequency * MSEL) / (2 * CGU_DIV.CSEL) Where the value of CGU_DIV.CSEL is between 1 and 31. When the processor comes out of reset, the default CGU_DIV.CSEL bit field value is 2.	
		0	CSEL = 32
		1-31	CSEL= 1 to 31

DIV Register Extension

The CGU_DIVEX register contains bitfields for programming the divisor values of the system clock [SCLK0 and SCLK1] in extended divisor mode.

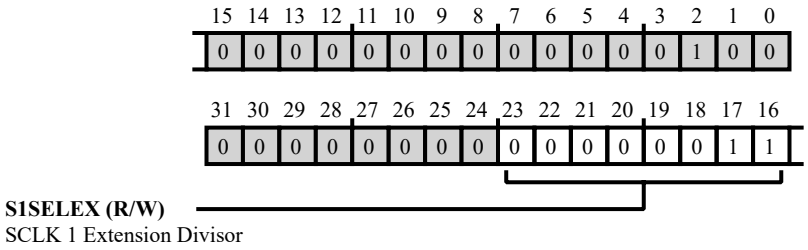


Figure 2-8: CGU_DIVEX Register Diagram

Table 2-12: CGU_DIVEX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	S1SELEX	SCLK 1 Extension Divisor. The CGU_DIVEX.S1SELEX selects the divisor in the SCLK1EX equation: SCLK1EX frequency = (SYS_CLKIN frequency * MSEL) / (2 * CGU_DIVEX.S1SELEX) Where the value of CGU_DIVEX.S1SELEX is between 1 and 255.	
		0	S1SELEX = 256
		1-255	S1SELEX = 1 to 255
7:0 (R/W)	S0SELEX	SCLK 0 Extension Divisor. The CGU_DIVEX.S0SELEX selects the divisor in the SCLK0EX equation: SCLK0EX frequency = (SYS_CLKIN frequency * MSEL) / (2 * CGU_DIVEX.S0SELEX) Where the value of CGU_DIVEX.S0SELEX is between 1 and 255.	
		0	S1SELEX = 256
		1-255	S1SELEX = 1 to 255

PLL Control Register

The CGU_PLLCTL register contains bits that enable and disable the PLL as well as control its function.

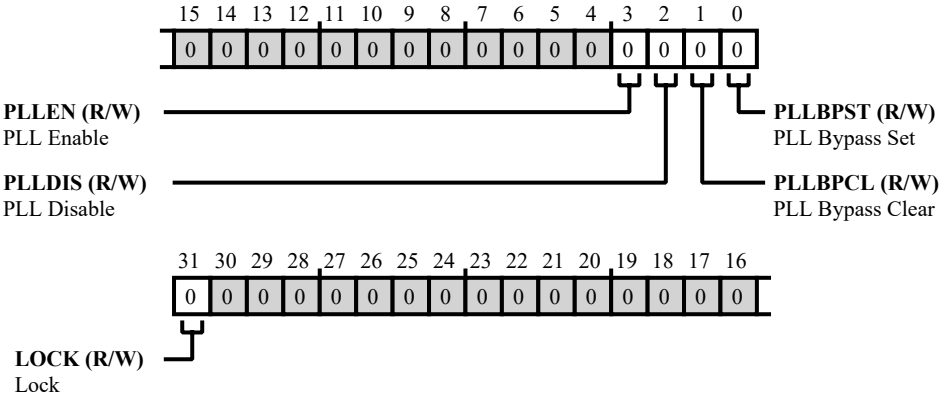


Figure 2-9: CGU_PLLCTL Register Diagram

Table 2-13: CGU_PLLCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. Setting (=1) the CGU_PLLCTL.LOCK bit locks access to the CGU_PLLCTL register.	
		0	Unlock register
		1	Lock register
3 (R/W)	PLLEN	PLL Enable. Setting (=1) the CGU_PLLCTL.PLLEN bit enables the PLL. Check the CGU_STAT.PLLEN bit to verify that the action is complete.	
		0	No action
		1	Enable PLL
2 (R/W)	PLLDIS	PLL Disable. Setting (=1) the CGU_PLLCTL.PLLDIS bit disables the PLL. Check the CGU_STAT.PLLEN bit to verify that the action is complete.	
		0	No action
		1	Disable PLL
1 (R/W)	PLLBPCL	PLL Bypass Clear. Setting (=1) the CGU_PLLCTL.PLLBPCL bit takes the PLL out of bypass mode. Check the CGU_STAT.PLLBP bit to verify that the action is complete.	
		0	No action
		1	Exit bypass mode

Table 2-13: CGU_PLLCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PLLBPST	PLL Bypass Set. Setting (=1) the `CGU_PLLCTL.PLLBPST` bit bypasses the PLL and all the clocks run on CLKIN. Check the `CGU_STAT.PLLBP` bit to verify that the action is complete.	
		0	Use PLL
		1	Bypass PLL

Revision ID Register

The CGU_REVID register reports the version of the CGU.

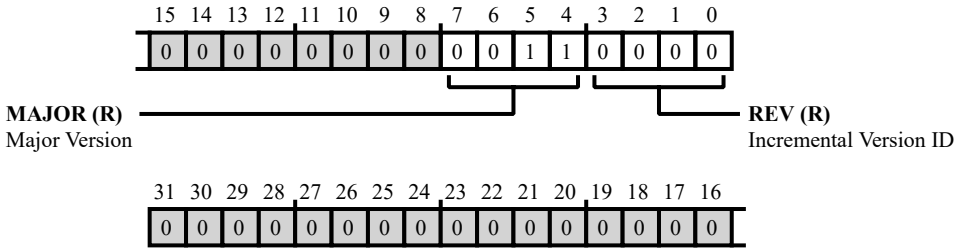


Figure 2-10: CGU_REVID Register Diagram

Table 2-14: CGU_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version.
3:0 (R/NW)	REV	Incremental Version ID.

System Clock Buffer Disable Register

The CGU_SCBF_DIS register controls each system's clock buffer to determine if the SCLKn buffer is enabled.

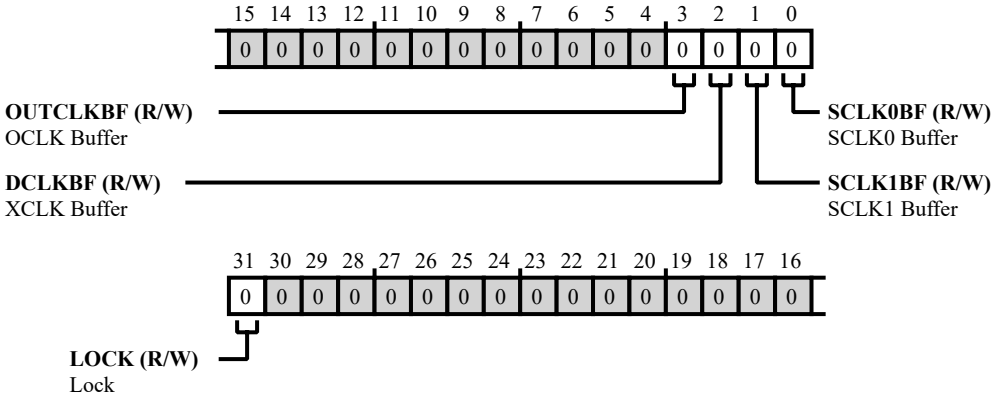


Figure 2-11: CGU_SCBF_DIS Register Diagram

Table 2-15: CGU_SCBF_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. The CGU_SCBF_DIS.LOCK bit allows writes to the CGU_SCBF_DIS register when cleared (=0) or blocks writes if set (=1) and the SPU_CTL.GLCK bit is set.	
		0	Unlock register
		1	Lock register
3 (R/W)	OUTCLKBF	OCLK Buffer. The CGU_SCBF_DIS.OUTCLKBF bit enables (=0, default) or disables (=1) OCLKs buffer.	
		0	Enable buffer
		1	Disable buffer
2 (R/W)	DCLKBF	XCLK Buffer. The CGU_SCBF_DIS.DCLKBF bit enables (=0, default) or disables (=1) XCLKs buffer.	
		0	Enable buffer
		1	Disable buffer

Table 2-15: CGU_SCBF_DIS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SCLK1BF	SCLK1 Buffer. The `CGU_SCBF_DIS.SCLK1BF` bit enables (=0, default) or disables (=1) SCLK1s buffer.	
		0	Enable buffer
		1	Disable buffer
0 (R/W)	SCLK0BF	SCLK0 Buffer. The `CGU_SCBF_DIS.SCLK0BF` bit enables (=0, default) or disables (=1) SCLK0s buffer.	
		0	Enable buffer
		1	Disable buffer

System Clock Buffer Status Register

The CGU_SCBF_STAT register shows which system clock buffer(s) are disabled. For example clearing the CGU_CCBF_DIS.CCBF0 bit clears the CGU_SCBF_STAT.SCLK0BF bit after a number of cycles. To guarantee that the correct value is read, this register should be read twice and the second result used.

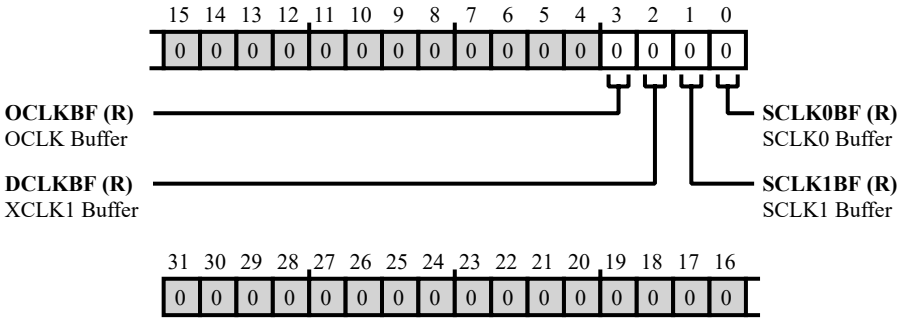


Figure 2-12: CGU_SCBF_STAT Register Diagram

Table 2-16: CGU_SCBF_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	OCLKBF	OCLK Buffer. The CGU_SCBF_STAT.OCLKBF bit reports the status of the CGU_SCBF_DIS.OUTCLKBF bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled
2 (R/NW)	DCLKBF	XCLK1 Buffer. The CGU_SCBF_STAT.DCLKBF bit reports the status of the CGU_SCBF_DIS.DCLKBF bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled
1 (R/NW)	SCLK1BF	SCLK1 Buffer. The CGU_SCBF_STAT.SCLK1BF bit reports the status of the CGU_SCBF_DIS.SCLK1BF bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled

Table 2-16: CGU_SCBF_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SCLK0BF	SCLK0 Buffer. The `CGU_SCBF_STAT.SCLK0BF` bit reports the status of the `CGU_SCBF_DIS.SCLK0BF` bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled

Status Register

The CGU_STAT register reflects the PLL status and errors detected during the PLL configuration. This register may be cleared asynchronously by a reset signal from the RCU module. All bits---except those defined as W1C (write-1-to-clear)---are read only.

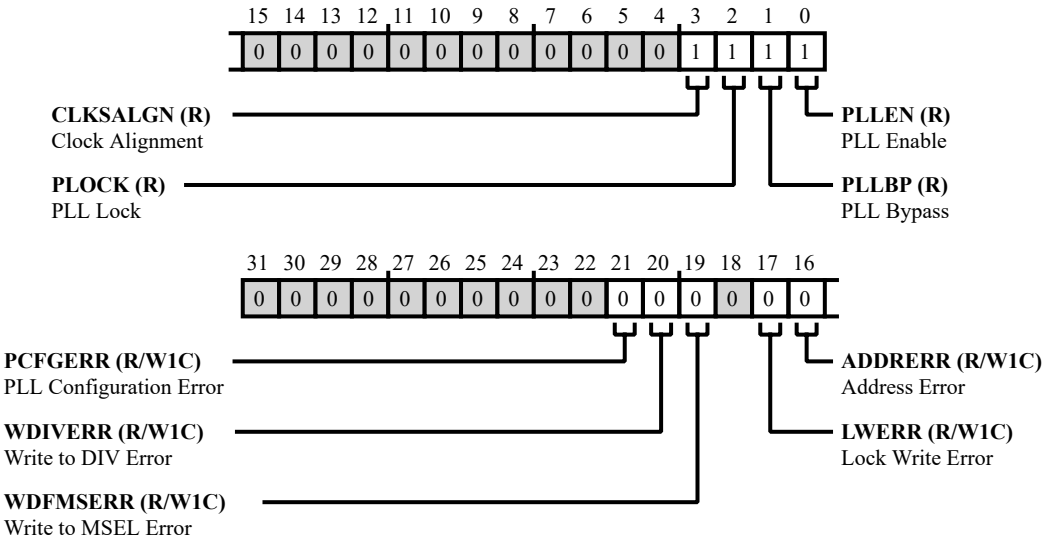


Figure 2-13: CGU_STAT Register Diagram

Table 2-17: CGU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W1C)	PCFGERR	PLL Configuration Error. If the CGU_PLLCTL.PLLBPST and the CGU_PLLCTL.PLLBPCL bits are set (=1) simultaneously or the CGU_PLLCTL.PLLDIS bit was set (=1) in full-on mode or while trying to enter full-on mode (CGU_PLLCTL.PLLBPCL =1), the CGU_STAT.PCFGERR bit triggers the bus error.	
		0	No Error
		1	Configuration Error
20 (R/W1C)	WDIVERR	Write to DIV Error. The CGU_STAT.WDIVERR bit indicates a write access to the CGU_DIV register (to trigger an alignment sequence or to change the CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, or CGU_DIV.DSEL bit values) while the PLL is locked, but still aligning the clocks. Read after write accesses to the CGU_STAT and CGU_DIV registers return the new value even if the clock frequency change is still in progress.	
		0	No Error
		1	Write DIV Error

Table 2-17: CGU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W1C)	WDFMSERR	Write to MSEL Error. The CGU_STAT.WDFMSERR bit indicates a write access to the CGU_CTL register to change the CGU_CTL.MSEL bit values while the PLL is locking.	
		0	No Error
		1	Write MSEL Error
17 (R/W1C)	LWERR	Lock Write Error. The CGU_STAT.LWERR bit indicates an attempt to write to write-protected (locked) CGU registers. The CGU issues a bus error for this condition.	
		0	No Error
		1	Lock Write Error
16 (R/W1C)	ADDRERR	Address Error. The CGU_STAT.ADDRERR bit indicates an attempt to make a read or write access to unimplemented addresses or accesses are non-aligned. The CGU issues a bus error for this condition.	
		0	No Error
		1	Address Error
3 (R/NW)	CLKSALGN	Clock Alignment. The CGU_STAT.CLKSALGN bit indicates whether a clock alignment sequence is in progress. This bit is set when clocks alignment is required by changes to CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, or CGU_DIV.OSEL. The CGU_STAT.CLKSALGN bit is cleared when clocks are aligned. Note that (after a PLL frequency change in active state) the CGU_STAT.CLKSALGN bit may indicate that clocks are not aligned even though the clocks are aligned (all clocks are aligned and running at CLKIN frequency).	
		0	Clocks are Aligned
		1	Clocks not Aligned (alignment in progress)
2 (R/NW)	PLOCK	PLL Lock. The CGU_STAT.PLOCK bit indicates whether the PLL is locked. This bit is set when the PLL locks (PLL lock counter end-of-count). The CGU_STAT.PLOCK bit is cleared when requested PLL frequency change (for PLL reset, PLL disable-to-enable transition, or a change to the CGU_CTL.MSEL values) is in progress.	
		0	PLL not Locked (PLL frequency change in progress)
		1	PLL Locked

Table 2-17: CGU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	PLLBP	PLL Bypass. The `CGU_STAT.PLLBP` bit indicates whether the PLL is bypassed. The default value for the `CGU_STAT.PLLBP` bit is determined by the bypass strap pin.	
		0	PLL not Bypassed
		1	PLL Bypassed
0 (R/NW)	PLLEN	PLL Enable. The `CGU_STAT.PLLEN` bit indicates whether the PLL is enabled.	
		0	Disabled
		1	Enabled

Time Stamp Counter 32 LSB Register

The CGU_TSCOUNT0 register address is used to read the CoreSight time stamp counter LSB 32-bit (bits [31:0]) value.

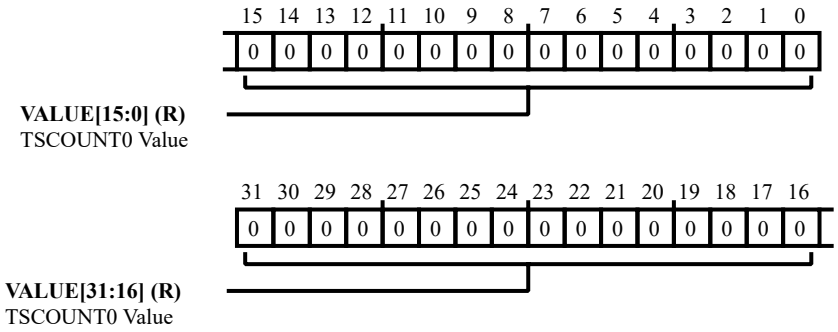


Figure 2-14: CGU_TSCOUNT0 Register Diagram

Table 2-18: CGU_TSCOUNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	TSCOUNT0 Value. The CGU_TSCOUNT0.VALUE bit field holds the time stamp counter 32 LSBs.

Time Stamp Counter 32 MSB Register

The CGU_TSCOUNT1 register address is used to read the CoreSight time stamp counter MSB 32-bit (bits [63:32]) value.

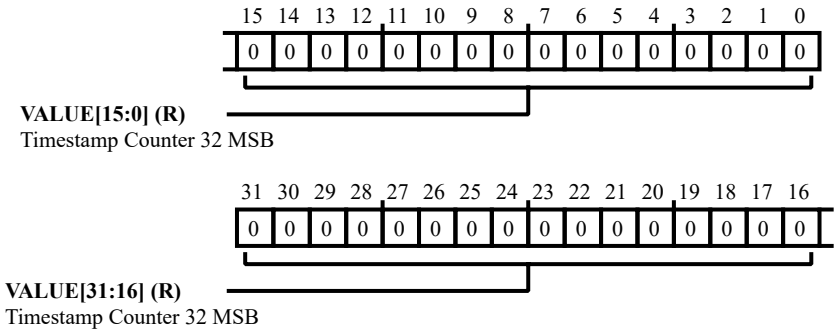


Figure 2-15: CGU_TSCOUNT1 Register Diagram

Table 2-19: CGU_TSCOUNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Timestamp Counter 32 MSB. The CGU_TSCOUNT1.VALUE bit field holds the time stamp counter 32 MSBs.

Time Stamp Control Register

The CGU_TSCTL register controls the operation of the CoreSight time stamp counter.

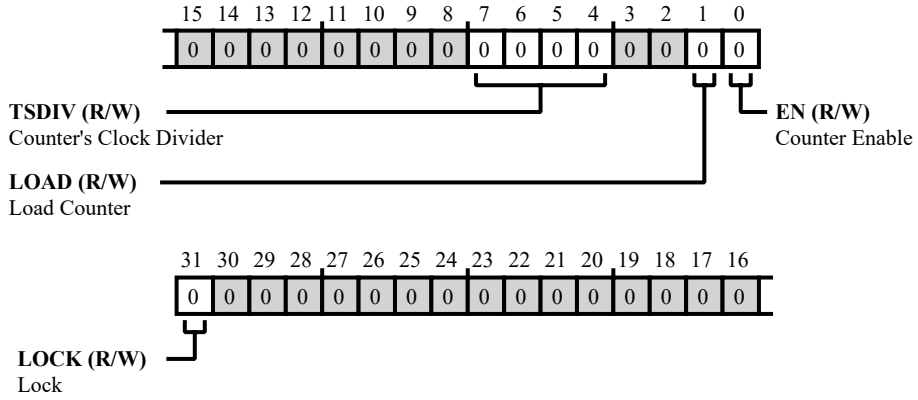


Figure 2-16: CGU_TSCTL Register Diagram

Table 2-20: CGU_TSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. Setting the CGU_TSCTL.LOCK bit locks this register.	
		0	Unlock
		1	Lock
7:4 (R/W)	TSDIV	Counter's Clock Divider. The CGU_TSCTL.TSDIV bit field divides SYSCLK by 2^{TSDIV}.	
		0-15	Divides SYSCLK by 2^{TSDIV}
1 (R/W)	LOAD	Load Counter. Writing one to the CGU_TSCTL.LOAD bit causes CoreSight time stamp counter to be loaded from the CGU_TSVALUE0 and CGU_TSVALUE1 registers.	
		0	Always read as "0"
0 (R/W)	EN	Counter Enable. The CGU_TSCTL.EN bit enables or disables the CoreSight time stamp counter.	
		0	Counter Disabled
		1	Counter Enabled

Time Stamp Counter Initial 32 LSB Value Register

The CGU_TSVALUE0 register holds the least significant bits (bits [31:0]) value that is initially loaded to the CoreSight time stamp counter.

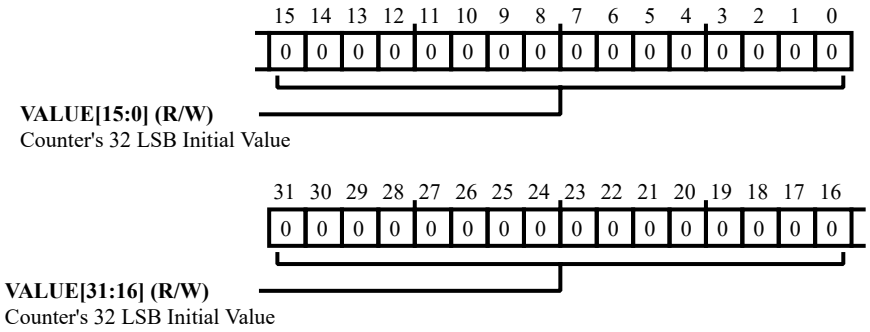


Figure 2-17: CGU_TSVALUE0 Register Diagram

Table 2-21: CGU_TSVALUE0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Counter's 32 LSB Initial Value. The CGU_TSVALUE0.VALUE bit field holds the LSBs value that is initially loaded to the CoreSight time stamp counter.

Time Stamp Counter Initial MSB Value Register

The CGU_TSVALUE1 register holds the most significant bits (bits [63:32]) value that is initially loaded to the CoreSight time stamp counter.

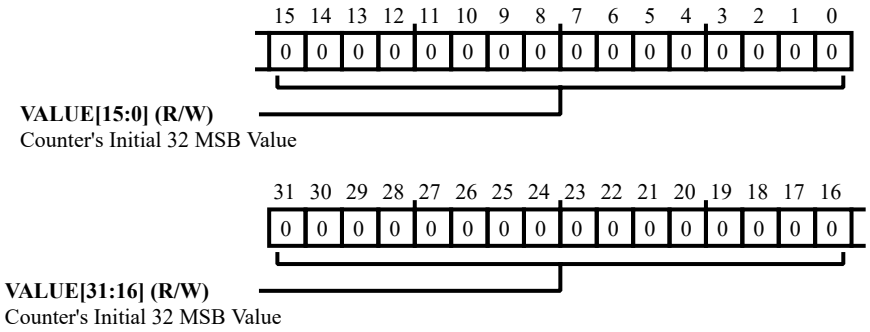


Figure 2-18: CGU_TSVALUE1 Register Diagram

Table 2-22: CGU_TSVALUE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Counter's Initial 32 MSB Value. The CGU_TSVALUE1.VALUE bit field holds the MSBs value that is initially loaded to the CoreSight time stamp counter.

3 Clock Distribution Unit (CDU)

The Clock Distribution Unit (CDU) consists of an array of multiplexors that select clocks originated from up to four different clock sources. These sources are different clocks that are generated from the Clock Generation Units (CGUs). The multiplexors are configured by software. All clocks multiplexors have up to four different sources. Unused input clocks are grounded internally and never selected. The output clock signal for each multiplexor is assigned to one or more destinations within the processor.

CDU Features

The CDU modules supports the following features:

- Generation of up to 14 output clocks
- Output clock buffers that can be disabled by software
- Multiplexors that are configured by writing to configuration registers (`CDU_CFG[n]`)
- Clocks originated in CGU0 that are selected by default
- A CDU status register (`CDU_STAT`) that indicates a configuration change is in-progress

CDU Functional Description

The CDU functions as a set of software-configurable multiplexors that select clocks from different sources.

CDU Block Diagram

The ***CGU/CDU Block Diagram*** figure shows the functional blocks within the CDU. As shown in the figure, the CDU takes different clocks generated by the CGU blocks and provides flexibility to route any clock from possible options to the output CDU clocks. The output clocks from the CDU are connected to specific targets such as the DDR module. All of the targets are clocked through the CDU output clocks rather than being directly clocked by CGU clocks.

This configuration provides the flexibility to meet the specific clock requirements of the different modules in the system (such as the core, DDR, or SPI module) without compromising the clocking of other modules. Such a flexibility is not possible with a single CGU in the system. With two CGUs:

- Each CGU has its own set of clock multipliers and dividers, providing a greater number of orthogonal clocks than possible with a single CGU.

Figure 3-1: CGU/CDU Block Diagram

The DDR is clocked from the DCLK (DCLK_0 and DCLK_1) from CGU0 or CGU1. This configuration provides the flexibility to program the DDR with frequencies orthogonal to the core clock frequencies. (Orthogonal frequencies have independent signal states. For example, frequency and amplitude modulation signals on the same carrier can be detected without interfering with each other).

CDU Definitions

The ***Clock Descriptions*** table provides a brief description of the clocks supported by the processor.

Table 3-1: Clock Descriptions

Clock	Description
CCLK0_0	CCLK0 derived from CGU0
CCLK1_0	CCLK1 derived from CGU0
SYSCLK_0	SYSCLK derived from CGU0
SCLK0_0	SCLK0 derived from CGU0
SCLK1_0	SCLK1 derived from CGU0
DCLK_0	DCLK derived from CGU0
OCLK_0	OCLK derived from CGU0
CCLK0_1	CCLK0 derived from CGU1
SYSCLK_1	SYSCLK derived from CGU1
SCLK0_1	SCLK0 derived from CGU1
SCLK1_1	SCLK1 derived from CGU1
DCLK_1	DCLK derived from CGU1
OCLK_1	OCLK derived from CGU1
CDU_CLKOn	Clocks that come out of the CDU module and go to different blocks

ADSP-2184x CDU Register List

Clock Distribution Unit

Table 3-2: ADSP-2184x CDU Register List

Name	Description
CDU_CFG[n]	CDU Configuration
CDU_REVID	CDU Revision ID
CDU_STAT	CDU Status

CDU Clock Configuration Options

The ***CDU Targets*** table provides information about clock source and destination options.

Table 3-3: CDU Targets

CDU0 Input				CDU0 Output	Target
IN0_CLKOn	IN1_CLKOn	IN2_CLKOn	IN3_CLKOn		
CCLK0_0	CCLK0_1	CCLK2_0	N/A	CLKO0	SHARC-FX and its accelerators
CCLK0_0	CCLK0_1	CCLK2_0	N/A	CLKO1	A55 Subsystem

Table 3-3: CDU Targets (Continued)

CDU0 Input				CDU0 Output	Target
IN0_CLKOn	IN1_CLKOn	IN2_CLKOn	IN3_CLKOn		
CCLK0_0	CCLK0_1	CCLK2_0	N/A	CLKO2	Reserved
DCLK_0	DCLK_1	DCLK1_0	N/A	CLKO3	LPDDR4
OCLK_0	OCLK_1	N/A	N/A	CLKO4	CAN
SCLK1_0	N/A	N/A	N/A	CLKO5	SPDIF
SCLK0_0	OCLK_0	N/A	N/A	CLKO6	SPI
SCLK0_0	SCLK0_1	N/A	N/A	CLKO7	GIGE
SCLK0_0	OCLK_0	SCLK1_1	N/A	CLKO8	xSPI1
DCLK_0	CCLK2_0	DCLK_1	CCLK2_1	CLKO9	LINKPORT
SCLK0_0	OCLK_0	SCLK1_1	N/A	CLKO10	xSPI0
SCLK0_0	N/A	N/A	N/A	CLKO12	TRACE
SYSCLK_0	CLKPWM*1	N/A	N/A	CLKO13	PWM
SCLK0_0	SYSCLK_1	N/A	N/A	CLKO14	MSHC

*1 See the Enhanced PWM Clocking section in the PWM chapter.

CDU Programming Model

The CDU_CFG[n] registers are a system configuration resource. These registers are accessed by a system configuration routine that also handles the configuration of other system modules. Writes to a CDU_CFG[n] register must occur when there is not a CDU_CLKOn configuration change in progress. If a CGUn inputs are selected, that CGUn configuration must be completed first.

Changing the PLL and Clock Frequency

1. Read the CGU_STAT register. Verify that:
   - CGU_STAT.PLLEN =1 (enabled)
   - CGU_STAT.PLOCK =1 (PLL is not locking)
   - CGU_STAT.CLKSALGN =0 (clocks aligned)
2. Write the desired values into the CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.SYSSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL and CGU_DIV.OSEL bits with the CGU_DIV.UPDT bit cleared (= 0).
3. Read the CGU_STAT register. Verify that:
   - CGU_STAT.PLLEN = 1 (enabled)

- CGU_STAT.PLOCK = 1 (not locking)
- CGU_STAT.CLKSALGN = 0 (clocks aligned)

4. Read the CDU_STAT register. Verify that the CDU_STAT.CLKO0 = 0 (no CLKOn configuration change in progress).
5. Write to the CDU_CFG[n].SEL bit to select the clock source (the CDU_CFG[n].EN bit should =1.)
6. Read the CDU_CFG[n] register. Verify that the CDU_CFG[n].SEL has the programmed value.

CLKOn is reconfigured.

Changing the Clock Frequency

Use the following procedure to change a clock frequency.

1. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write the desired values into the CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.SYSSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL and CGU_DIV.OSEL bits with the CGU_DIV.UPDT bit = 1.
3. Read the CGU_DIV registers to verify that the CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.SYSSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL and CGU_DIV.OSEL bit values are correct.
4. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN = 0 (clocks aligned).
5. If clocks switch from CGUm clocks to CGUn input clocks, read the CGU_SCBF_STAT and CGU_CCBF_STAT registers. The CGU_CCBF_STAT[1:0] bit field corresponds to the CCLK1 and CCLK0 clock buffers, respectively. The CGU_SCBF_STAT[3:0] bit field corresponds to the OUTCLK, DCLK, SCLK1, and SCLK0 clock buffers, respectively.
6. Read the CDU_STAT register.
7. Write to the CDU_CFG[n].SEL bit to select the clock source (CDU_CFG[n].EN = 1).
8. Read the CDU_CFG[n] register. Verify that the CDU_CFG[n].SEL bit field has the programmed value.
9. Verify that the CDU_STAT.CLKO0 bit field = 0.

ADSP-2184x CDU Register Descriptions

Clock Distribution Unit (CDU) contains the following registers.

Table 3-4: ADSP-2184x CDU Register List

Name	Description
CDU_CFG[n]	CDU Configuration
CDU_REVID	CDU Revision ID

Table 3-4: ADSP-2184x CDU Register List (Continued)

Name	Description
CDU_STAT	CDU Status

CDU Configuration

The CDU_CFG[n] registers control the configuration of the clock multiplexors. CDU0_CFG[n] corresponds to output clock CDU_CLKO[n].

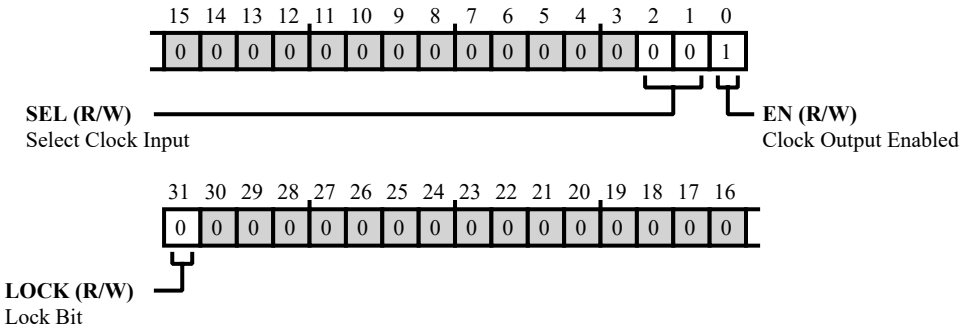


Figure 3-2: CDU_CFG[n] Register Diagram

Table 3-5: CDU_CFG[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
2:1 (R/W)	SEL	Select Clock Input.	
		0	IN0_CLKOn Selected
		1	IN1_CLKOn Selected
		2	IN2_CLKOn Selected
		3	IN3_CLKOn Selected
0 (R/W)	EN	Clock Output Enabled. The CDU_CFG[n].EN bit enables clock output.	

CDU Revision ID

The CDU_REVID register reports the version of the CDU.

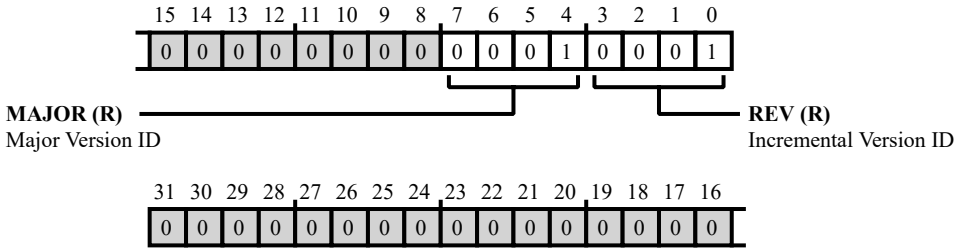


Figure 3-3: CDU_REVID Register Diagram

Table 3-6: CDU_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version ID.
3:0 (R/NW)	REV	Incremental Version ID.

CDU Status

The CDU_STAT register reflects the status a change in the configuration of the clock muxes.

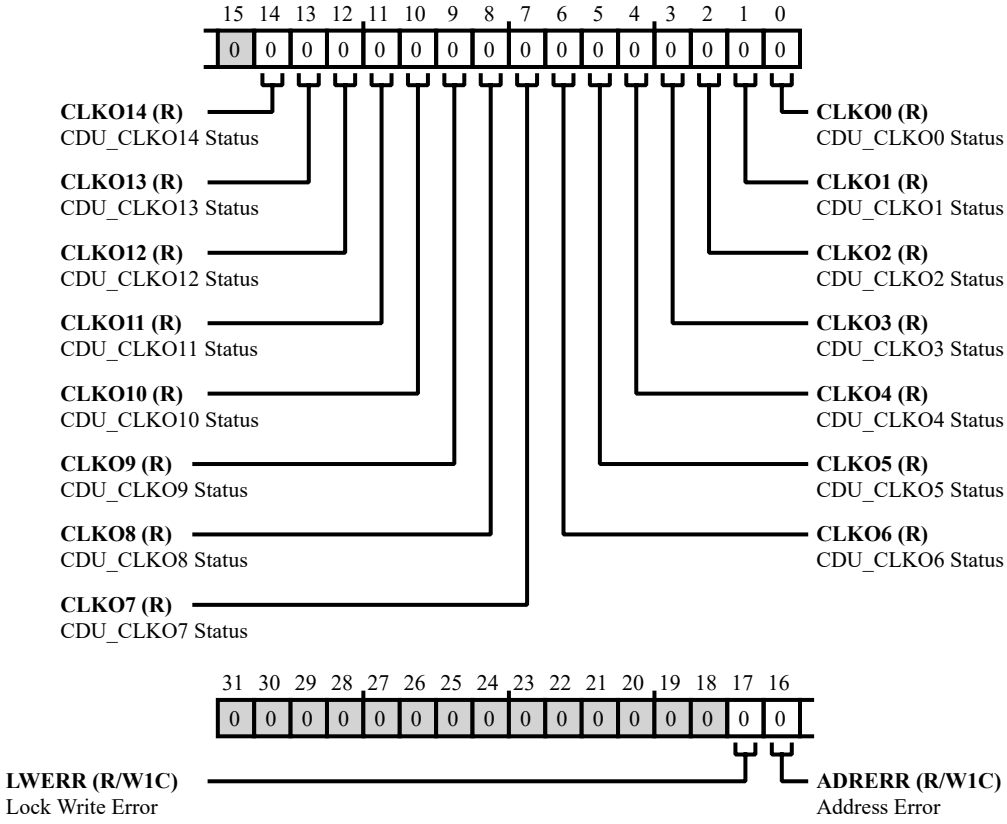


Figure 3-4: CDU_STAT Register Diagram

Table 3-7: CDU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	LWERR	Lock Write Error. The CDU_STAT.LWERR bit indicates a lock error where write transactions try to access write protected registers.	
		0	No Lock Write Error
		1	Lock Write Error
16 (R/W1C)	ADRERR	Address Error. The CDU_STAT.ADRERR bit indicates an address error where read or write transactions try to access unimplemented addresses or accesses are non-aligned.	
		0	No Address Error
		1	Address Error

Table 3-7: CDU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/NW)	CLKO14	CDU_CLKO14 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
13 (R/NW)	CLKO13	CDU_CLKO13 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
12 (R/NW)	CLKO12	CDU_CLKO12 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
11 (R/NW)	CLKO11	CDU_CLKO11 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
10 (R/NW)	CLKO10	CDU_CLKO10 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
9 (R/NW)	CLKO9	CDU_CLKO9 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
8 (R/NW)	CLKO8	CDU_CLKO8 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
7 (R/NW)	CLKO7	CDU_CLKO7 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
6 (R/NW)	CLKO6	CDU_CLKO6 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
5 (R/NW)	CLKO5	CDU_CLKO5 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress

Table 3-7: CDU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	CLKO4	CDU_CLKO4 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
3 (R/NW)	CLKO3	CDU_CLKO3 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
2 (R/NW)	CLKO2	CDU_CLKO2 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
1 (R/NW)	CLKO1	CDU_CLKO1 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
0 (R/NW)	CLKO0	CDU_CLKO0 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress

4 Fractional PLL (Frac-N PLL)

This processor uses the Fractional Phased-locked Loop (Frac-N PLL) to generate precise and low-jitter audio clock frequencies.

IMPORTANT: In this chapter, ADI uses the abbreviation Frac-N PLL to refer to the entire Fractional PLL module. The Frac-N PLL subblocks are the Fractional PLL (FPLL) and Digital PLL (DPLL). The FPLL subblock generates a clock frequency that is a fractional multiple of CLKIN. The DPLL subblock performs phase alignment with the output clock to an external reference clock (FIN).

The Fractional Phased-locked Loop (Frac-N PLL) operates in one of two modes:

- FPLL Only Mode–generates a clock with frequency values that are fractional multiples of the CLKIN frequency. As an example, the processor can output an audio clock of frequency 24.576 MHz generated from a 25 MHz CLKIN frequency.
- DPLL Plus FPLL Mode–consists of a digital filter and controller (DPLL), combined with the FPLL to create a dual-loop PLL. The DPLL provides fractional and integer parts of divider values to the FPLL block. The Frac-N PLL output clock feeds back the signal to the DPLL for phase-alignment with the external reference clock (FIN). As an example, you can obtain a 1 KHz PPS output from the EMAC block, to use as a reference for generating a 24.576 MHz audio clock that is phase-aligned to the pulses per second (PPS) output.

Frac-N PLL Features

The Frac-N PLL supports the following features:

- Generation of a clock with frequency values that are fractional multiples of the CLKIN frequency.
- Phase-alignment of the output clock with a selected external reference clock (FIN).
- FIN sources from DAI0/DAI1 PCGx_EXTCLK, GPIO (FRACNPLL_PTP_CLK), or RGMII PTP PPS0-3 signals.
- FPLL frequency, DPLL frequency, and phase lock interrupt and trigger.
- DPLL frequency unlock interrupt.
- FPLL and DPLL trigger wait (TWAIT).
- Routing of the Frac-N PLL signals to the SYS_CLKOUT.

- Routing of the Frac-N PLL output clock to the Timer and PWM blocks.

Frac-N PLL Definitions

To make the best use of the Frac-N PLL, it is helpful to understand the following terms.

Phase-locked Loop (PLL)

A feedback circuit designed to synchronize one circuit board to the phase of its on board clock with an external timing signal. PLL circuits operate by comparing the phase of an external signal to the phase of a clock signal produced by a voltage controlled crystal oscillator (VCXO).

Fractional Phase-locked Loop (Frac-N PLL)

The entire fractional PLL module used to generate precise and low-jitter audio clock frequencies. The Frac-N PLL consists of two subblocks, the FPLL and the DPLL.

Fractional PLL (FPLL)

The FPLL subblock of the Frac-N PLL that generates a clock frequency, which is a fractional multiple of the CLKIN.

Digital PLL (DPLL)

The DPLL subblock of the Frac-N PLL that performs phase alignment with the output clock to an external reference clock (FIN).

Digital Audio Interfaces (DAI)

Consists of a group audio peripherals connected using the Signal Routing Unit (SRU). The SRU connects inputs and outputs of the DAI peripherals with each other and to the external pins. This arrangement permits peripherals to be interconnected and accommodates different audio configurations without making separate external pin connections.

Ethernet Media Access Controller (EMAC)

The EMAC peripheral in the processor enables network connectivity to applications through an Ethernet interface. This function enables applications to support TCP/IP based network communication. At the system end, the module supports direct connection with the system crossbar bus for memory or memory-mapped registers (MMR) transactions.

Pulse-Width Modulator (PWM)

The processor component that provides a way of controlling certain analog quantities, by varying the pulse width of a fixed frequency rectangular waveform.

FIN (external reference clock)

An external reference clock used to compare against the Frac-N PLL output clock by the DPLL for phase-alignment corrections.

Phase/Frequency Detector (PFD)

A linear or binary phase detector that compares the phase difference between the input reference signal and the output signal from a voltage controlled oscillator (VCO).

Precision Clock Generator (PCG)

Used to produce a pair of signals from a clock input signal. The PCG normally generates two signals used as a serial bit clock and frame sync pair.

Precision Time Protocol (PTP)

A protocol used to synchronize clocks in processors and their peripherals. On a processor, it achieves clock accuracy in the sub-microsecond range, making it suitable for measurement and control systems.

Pulses-per-Second (PPS)

A signal that has a width of less than one second and a sharply rising or abruptly falling edge that accurately repeats once per second.

Signal Routine Unit (SRU)

A precision clock generator component that routes all of the precision clock generator (PCG) signals in one digital audio interface (DAI) for external usage.

Trigger Routing Unit (TRU)

The processor component that provides a system level sequence control without core intervention. It maps the trigger controllers (trigger generators) to trigger targets (triggers receivers). The receiver endpoints can be configured to respond to triggers in different ways.

Voltage Controlled Oscillator (VCO)

An electronic oscillator where the voltage input controls the oscillation frequency. The applied input voltage determines the instantaneous oscillation frequency.

Frac-N PLL Functional Description

This section provides a description of the Frac-N PLL, including a list of its registers, interrupts, and functional block diagrams.

Frac-N PLL Block Diagram

The *Frac-N PLL Block Diagram* shows the organization and communication of the Frac-N PLL It contains the following major components.

- FPLL subblock
- DPLL subblock
- Clock multiplexers–that switch between functional modes and select the FIN source
- REG-interface–configure and control the FPLL, DPLL, and clock multipliers. The REG-IF also reads the status parameters.

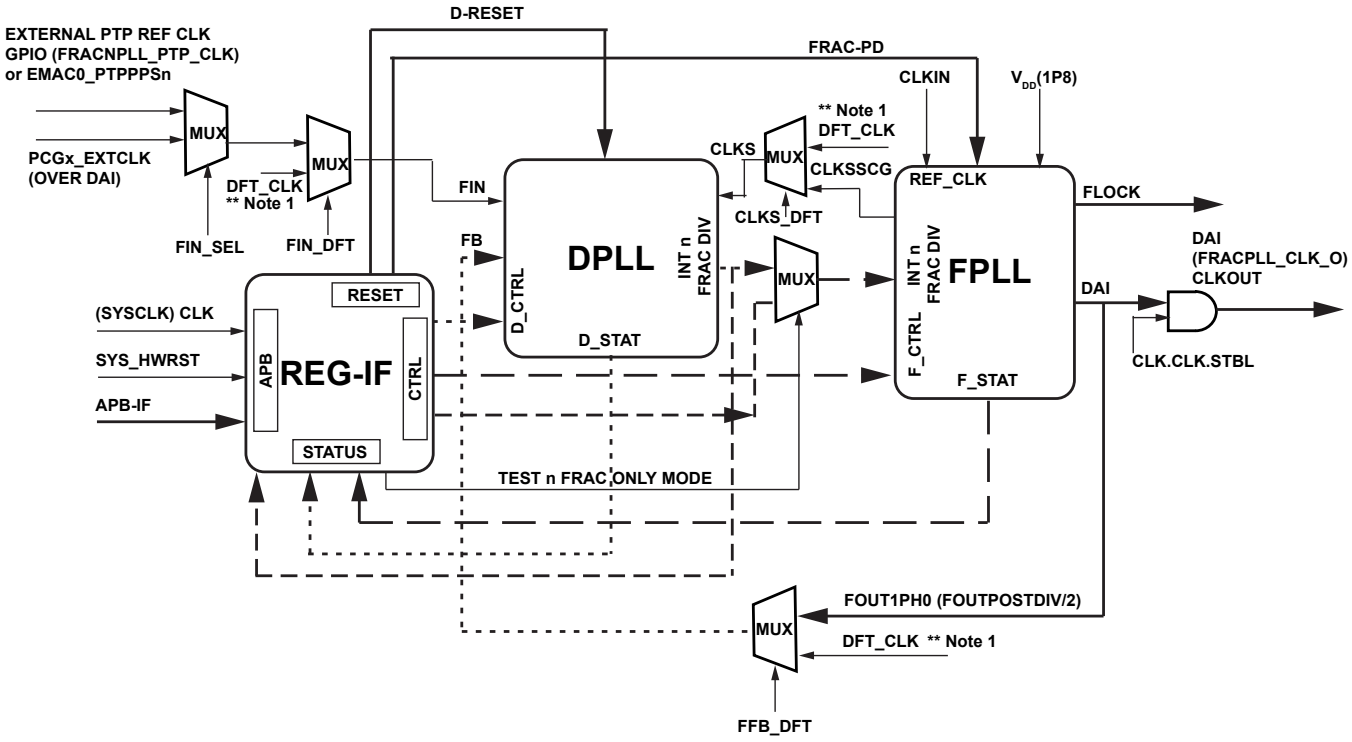


Figure 4-1: Frac-N PLL Block Diagram © Silicon Creations

NOTE: 1. The DFT_CLK is for Analog Devices internal use only.

FPLL Detailed Block Diagram

The FPLL generates a clock with a frequency value that is a fractional multiple of the CLKIN frequency.

The *FPLL Detailed Subblock Diagram* shows how the FPLL generates a clock with a frequency value that is a fractional multiple of the CLKIN frequency.

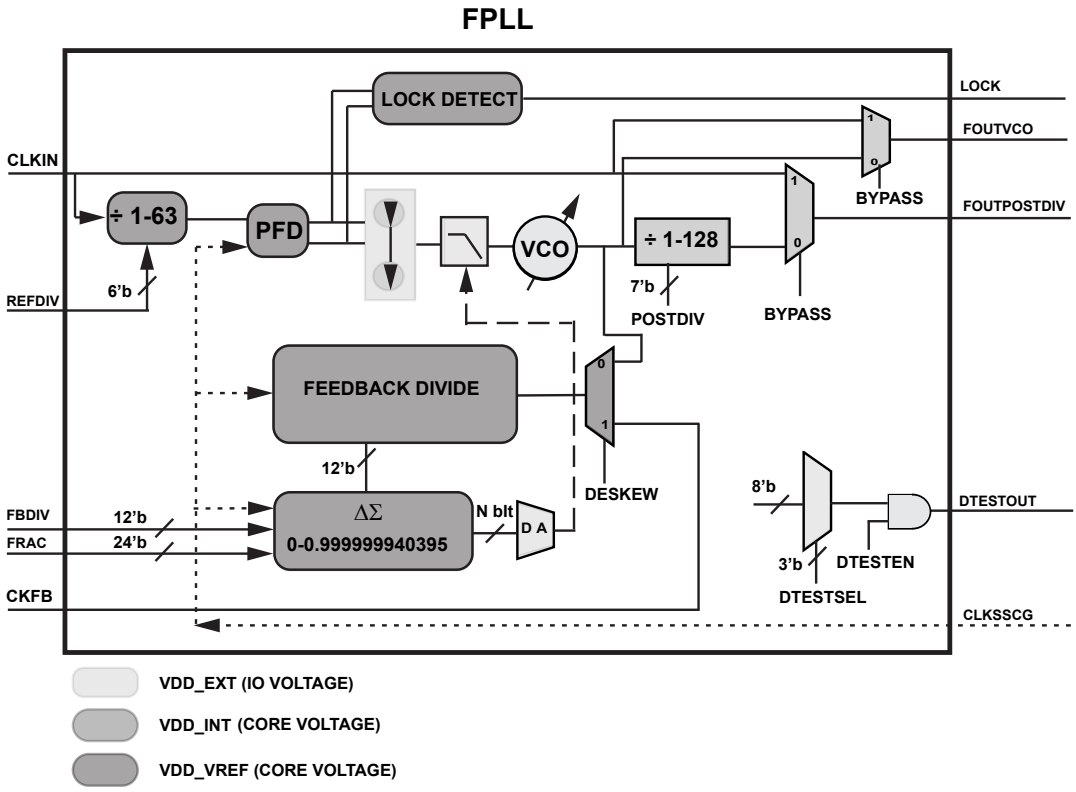


Figure 4-2: FPLL Detailed Subblock Diagram © Silicon Creations

The CLKIN is the input clock source for the FPLL. (See the *High Performance SHARC-FX DSP Core With Arm-Based Connectivity/Safety-Preliminary Data Sheet* for the specific CLKIN frequency range.) The CLKIN frequency is divided with the `FRACNPLL_FPLL_CTL1.REFDIV` field and then goes to the voltage controlled oscillator (VCO). The VCO multiplies the divided CLKIN with the fractional value using the configured `FRACNPLL_FPLL_CTL1.INT` and `FRACNPLL_FPLL_CTL0.FRAC` fields. Engineers can further divide the VCO output (FOUTVCO) with the post divider (POSTDIV), using the configured `FRACNPLL_FPLL_CTL1.POSTDIV` field. FOUTPOSTDIV is the resulting output clock.

Set (=1) the `FRACNPLL_FPLL_CTL0.BYPASS` bit and bypass the CLKIN directly to the FOUTPOSTDIV output. They can further divide the FOUTPOSTDIV frequency 2, 4, 8, and 16 to generate the FOUTPOSTDIV/2, FOUTPOSTDIV/4, FOUTPOSTDIV/8 or FOUTPOSTDIV/16 signals. The CLKSSCG signal works as a synchronization signal, and feeds back to the DPLL when in the DPLL plus FPLL Mode. This clock typically operates at a low frequency (same frequency as the PLL input clock), but also has a low duty cycle (typically about 10-20%).

DPLL and FPLL Block Diagram

The DPLL synchronizes the FPLL clock output with the on-chip (RGMII PTP PPS) or the off-chip (DAI0/DAI1/GPIO) reference clock (FIN). The *DPLL Detailed Subblock Diagram with FPLL Interface* shows a detailed diagram of the DPLL subblock and its interface to the FPLL subblock.

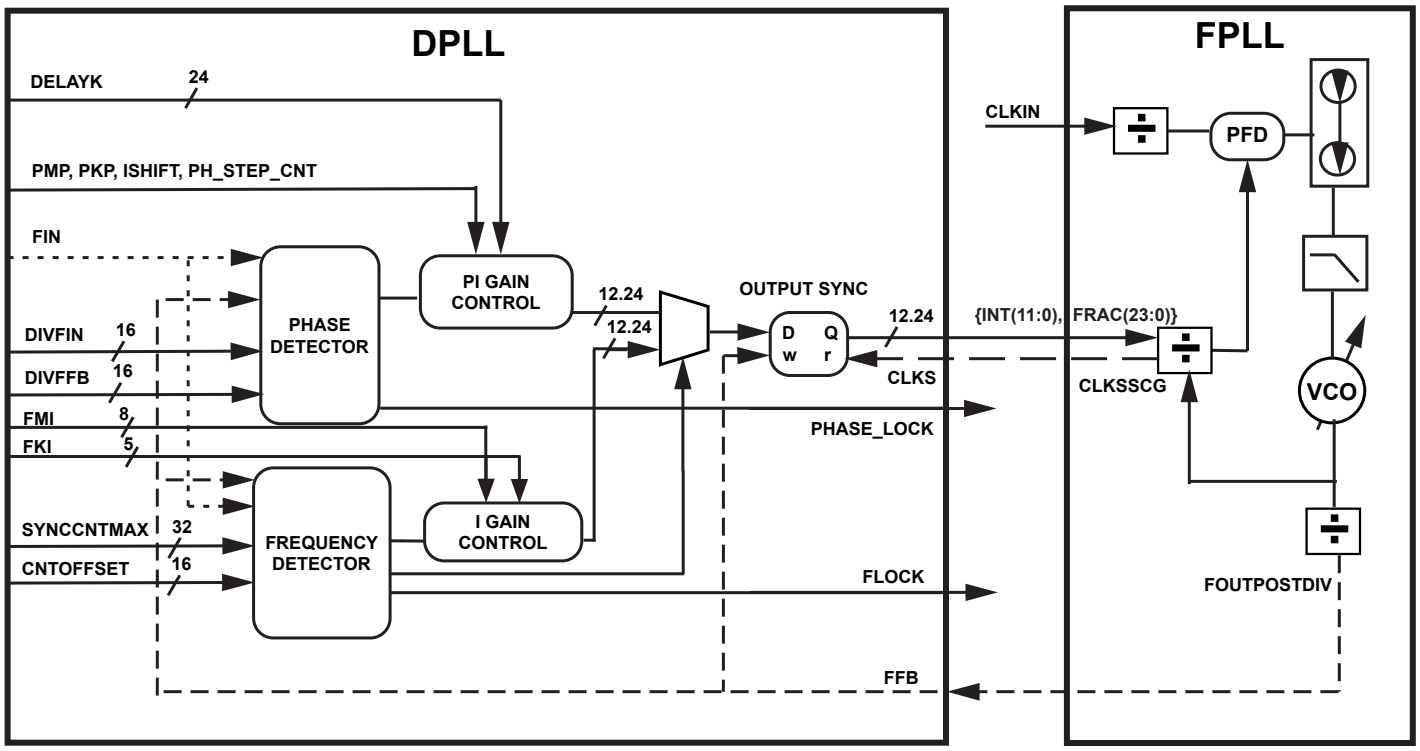


Figure 4-3: DPLL Detailed Subblock Diagram with FPLL Interface © Silicon Creations

The DPLL interfaces to the FPLL by controlling its integer and fractional feedback division bits. The DPLL has a broadly tuneable bandwidth to filter noisy references. Because the FPLL has a separate clean reference, its own noise characteristics and not those of the reference dominates the noise at the FPLL output.

The DPLL uses three phases of acquisition: frequency tracking, phase settling control, and phase tracking. Each phase has separate programmable gains to tune the DPLL performance for the specific application.

- ***Frequency gain control and frequency tolerance control (tracking)***–modify the response time of the PLL when in frequency tracking mode and permit tracking or filtering of a reference, tuned to the expected noise of the reference.
- ***Phase settling control***–manages the transition from frequency tracking mode to phase tracking mode with a high gain to ensure the DPLL can lock under *all* conditions.
- ***Phase tracking***–enables the DPLL to have a very low bandwidth, effectively averaging out higher frequency noise that can happen because of spread spectrum or other noisy input references.

ADSP-2184x FRACNPLL Register List

The FRACnPLL module combines DPLL and Frac PLL to generate an on-chip, low jitter, precision audio clock.

Table 4-1: ADSP-2184x FRACNPLL Register List

Name	Description
FRACNPLL_CTL	Integration Control Register
FRACNPLL_DPLL_CTL0	DPLL Control Register 0
FRACNPLL_DPLL_CTL1	DPLL Control Register 1
FRACNPLL_DPLL_CTL2	DPLL Control Register 2
FRACNPLL_DPLL_CTL3	DPLL Control Register 3
FRACNPLL_DPLL_CTL4	DPLL Control Register 4
FRACNPLL_DPLL_CTL5	DPLL Control Register 5
FRACNPLL_DPLL_CTL6	DPLL Control Register 6
FRACNPLL_DPLL_CTL7	DPLL Control Register 7
FRACNPLL_DPLL_CTL8	DPLL Control Register 8
FRACNPLL_DPLL_CTL9	DPLL Control Register 9
FRACNPLL_DPLL_STAT0	DPLL Status Register 0
FRACNPLL_DPLL_STAT1	DPLL Status Register 1
FRACNPLL_DPLL_STAT2	DPLL Status Register 2
FRACNPLL_FPLL_CTL0	Fractional PLL Control Register 0
FRACNPLL_FPLL_CTL1	Fractional PLL Control Register 1
FRACNPLL_STAT	Integration Status Register

ADSP-2184x FRACNPLL Interrupt List

Table 4-2: ADSP-2184x FRACNPLL Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
114	FRACNPLL0_FRAC_LOCK	FRACNPLL0 Frac Pll Lock Interrupt	Edge	
115	FRACNPLL1_FRAC_LOCK	FRACNPLL1 Frac Pll Lock Interrupt	Edge	
116	FRACNPLL0_FRAC_UN-LOCK	FRACNPLL0 Frac Pll Unlock Interrupt	Edge	
117	FRACNPLL1_FRAC_UN-LOCK	FRACNPLL1 Frac Pll Unlock Interrupt	Edge	

ADSP-2184x FRACNPLL Trigger List

Table 4-3: ADSP-2184x FRACNPLL Trigger List Generators

Trigger ID	Name	Description	Sensitivity
55	FRACNPLL0_FRACPLL_TRIGM	FRACNPLL0 Frac Pll Lock Trigger	Edge
56	FRACNPLL1_FRACPLL_TRIGM	FRACNPLL1 Frac Pll Lock Trigger	Edge

Table 4-4: ADSP-2184x FRACNPLL Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
100	FRACNPLL0_FRACPLL_TRIGS	FRACNPLL0 Frac Pll Start Trigger	Pulse
101	FRACNPLL1_FRACPLL_TRIGS	FRACNPLL1 Frac Pll Start Trigger	Pulse

Frac-N PLL Architectural Concepts

This section covers the following major concepts:

- Interrupts and Triggers
- DAI Clock Selection
- Frac-N PLL CLKOUT Clock Selection
- Usage of Frac-N PLL Output Clock

Interrupts and Triggers

NOTE: For the SHARC-FX core, configure the interrupt controller for the FRACNPLL_FRAC_LOCK interrupt to use an edge-sensitive interrupt to avoid multiple re-entries inside the ISR.

The Cortex-A55 core does not support the FRACNPLL_FRAC_LOCK interrupt.

The following triggers and interrupts are part of the Frac-N PLL.

- Lock Interrupt-Trigger
- DPLL Unlock Interrupt
- TWAIT Trigger Receiver

Lock Interrupt-Trigger

The Frac-N PLL supports the following lock interrupts.

- FPLL frequency lock (FPLL_FLOCK)
- DPLL frequency lock (DPLL_FLOCK)
- DPLL phase lock (DPLL_PLOCK)

These interrupts can be sent to the SEC/NVIC using the interrupt signal FRACNPLL_FRAC_LOCK. The same signal routes the interrupt to the trigger routing unit as the trigger requester signal FRACPLL_TRIGM and to the GPIO (FRACNPLL_FLOCK).

Configure the FRACNPLL_CTL.LOCK_SEL field to select one of the following values:

- 0–routes none of the three interrupts
- 1–uses the FPLL frequency lock (FPLL_FLOCK)
- 2–uses the DPLL frequency lock (DPLL_FLOCK)
- 3–uses the DPLL phase lock (DPLL_PLOCK)

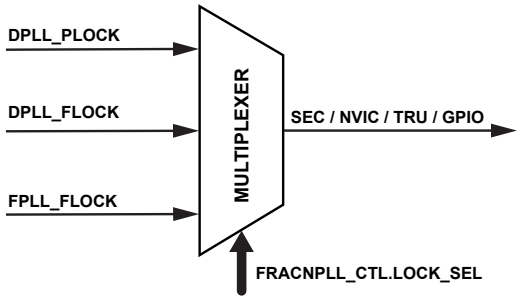


Figure 4-4: Frac-N PLL Lock Interrupt and Trigger Multiplexer

DPLL Unlock Interrupt

Applications can route the DPLL frequency unlock signal to SEC/NVIC FRAC_UNLOCK interrupt. This interrupt can detect the loss of the FIN reference clock signal and take any necessary action at the system level.

TWAIT Trigger Receiver

This processor has an available trigger (receiver) wait or TWAIT to release the power down and reset signal of the FPLL and DPLL. This auto-synchronizes the start of an operation on any available system event. Set (=1) the FRACNPLL_CTL.F_TWAIT (FPLL) and FRACNPLL_CTL.D_TWAIT (DPLL) bits to indicate one of the two operating modes. (See Frac-N PLL Operating Modes for more information.)

Engineers can configure both PLLs and released from reset by clearing (=0) the FRACNPLL_CTL.FCPD (required) and the FRACNPLL_CTL.DCRESET (release from reset) bits. When the respective TWAIT bit is set (=1), an actual reset to FPLL/DPLL does not get unasserted immediately. The reset waits for the trigger to arrive. Once the trigger arrives, the FPLL or DPLL reset is unasserted.

CAUTION: The TWAIT only controls the reset-deassertion. TWAIT bits do not gate the reset assertion to FPLL and DPLL.

DAI Clock Selection

The *Frac-N PLL DAI CLock Multiplexer* figure shows how it provides the FPLL output clock FOUTPOSTDIV in available sub-multiples (2, 4, 6, or 8), which can route different values of clock frequencies to the DAI, without disturbing the feedback clock or the DPLL frequency and phase lock.

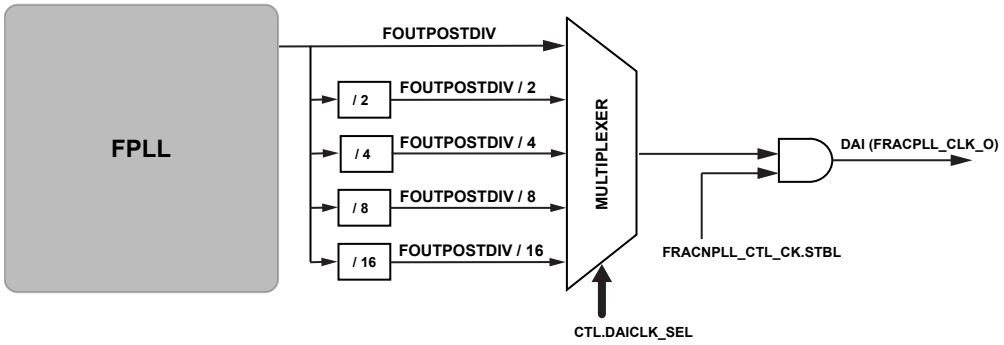


Figure 4-5: Frac-N PLL DAI Clock Multiplexer

For a FOUTPOSTDIV clock of frequency 98.304 MHz, one can program the multiplexer to select output clock values that include: 49.152 MHz (/2), 24.576 MHz (/4), 12.288 MHz (/8), and 6.144 MHz (/16). See the *DAI Output Clock Selection* table for valid configuration values for the `FRACNPLL_CTL.DAICLK_SEL` field. Engineers can set (=1) a clock gating control after the multiplexer using the `FRACNPLL_CTL.CK_STBL` bit to ensure sending out only a stable clock.

Table 4-5: DAI Output Clock Selection

FRACNPLL_CTL. DAICLK_SEL Value	Clock Selected	Example Frequency
0	No Clock	N/A
1	FOUTPOSTDIV/16	6.144 MHz
2	FOUTPOSTDIV/8	12.288 MHz
3	FOUTPOSTDIV/4	24.576 MHz
4	FOUTPOSTDIV/2	45.152 MHz
5	FOUTPOSTDIV (Default)	98.304 MHz

Frac-N PLL CLKOUT Clock Selection

The *FPLL CLKOUT Multiplexer* diagram shows how a multiplexer detects all clocks and synchronization signals (CLKSSCG) for the Frac-N PLL.

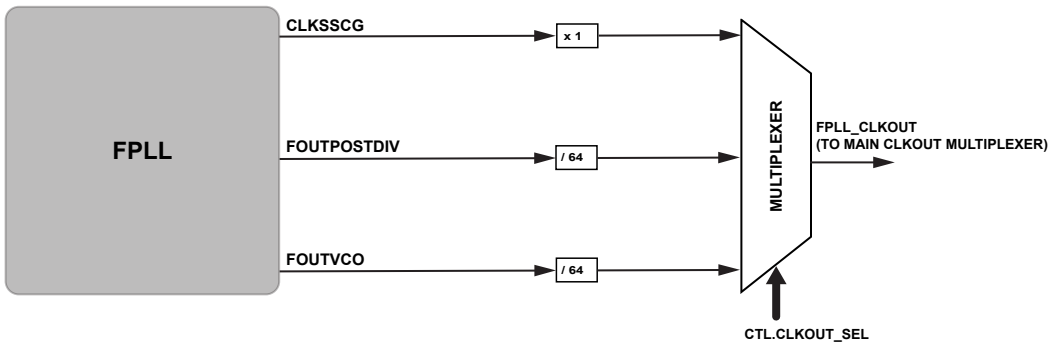


Figure 4-6: FPLL CLKOUT Multiplexer

Configure the `FRACNPLL_CTL.CLKOUT_SEL` field to program the multiplexer by using the values in the *FPLL_CLKOUT Multiplexer Selection* table. To meet the maximum operating pad frequency, adequately-added divider stages exist on each clock path (based on their maximum operating frequency) before connecting to the FPLL CLKOUT multiplexer. The output of the CLKOUT multiplexer merges to the main CLKOUT multiplexer. See the Clock Generation Unit (CGU) section for more information.

Table 4-6: FPLL_CLKOUT Multiplexer Selection

FRACNPLL_CTL. CLKOUT_SEL Value	Signal Routed to FPLL_CLKOUT Multiplexer
1	CLKSSCG
2	FOUTPOSTDIV/64 (Default)
3	FOUTVCO/64

Usage of Frac-N PLL Output Clock

Users can route the FPLL output clock to the following processor modules:

- Digital Audio Interface (DAI)
- Timer
- Pulse-Width Modulator (PWM)

Digital Audio Interface (DAI)

The FPLL output clock is routed to one of the PCGx_EXTCLK signals. The output clock can also be one of the source signals for Group D of the Signal Routing Unit (SRU). This method routes the clock to one or more DAI0/1 pin buffers for external use.

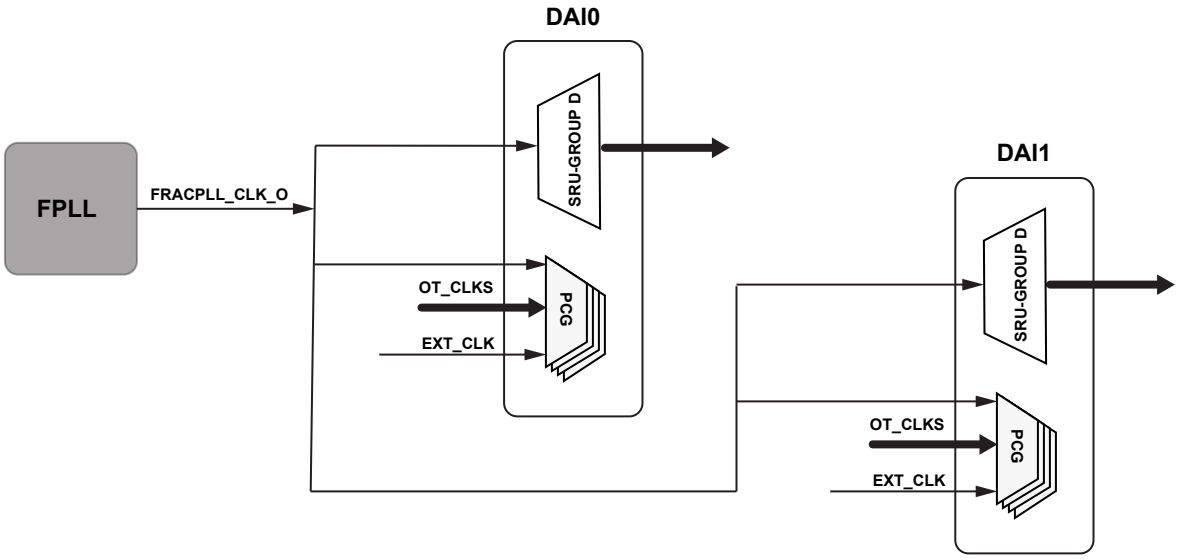


Figure 4-7: FPLL Output to PCG or SRU

Timer

The FPLL output clock connects to the TM0_ACLK port of the all the timers (through multiplexers) and with the other TM0_ACLK sources. Configure the `MISCREG_CLK_MUX_SEL.PCG_CLKSEL` field to select the PCG clock or set (=1) the `MISCREG_CLK_MUX_SEL.PCG_CLKSELF` bit to select the FPLL clock for the timer.

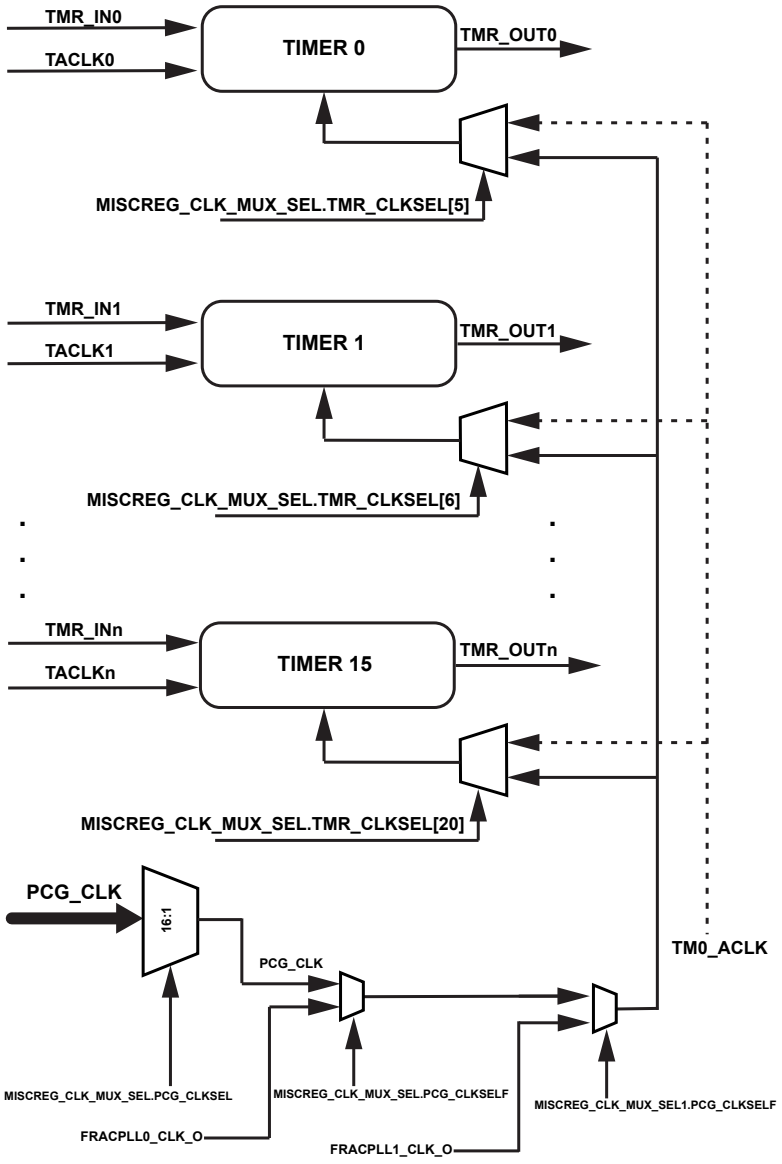


Figure 4-8: FPLL Output Clock Usage for Timer

Pulse-Width Modulator (PWM)

The FPLL output clock is used as an operating clock for the on chip PWM along with the SYSCLK to synchronize the PWM output with the FPLL output clock. Configure the `MISCREG_CLK_MUX_SEL.EPWM_CLKSEL` field to select the PCG clock or set (=1) the `MISCREG_CLK_MUX_SEL.EPWM_CLKSELF` bit to select the FPLL clock for the PWM.

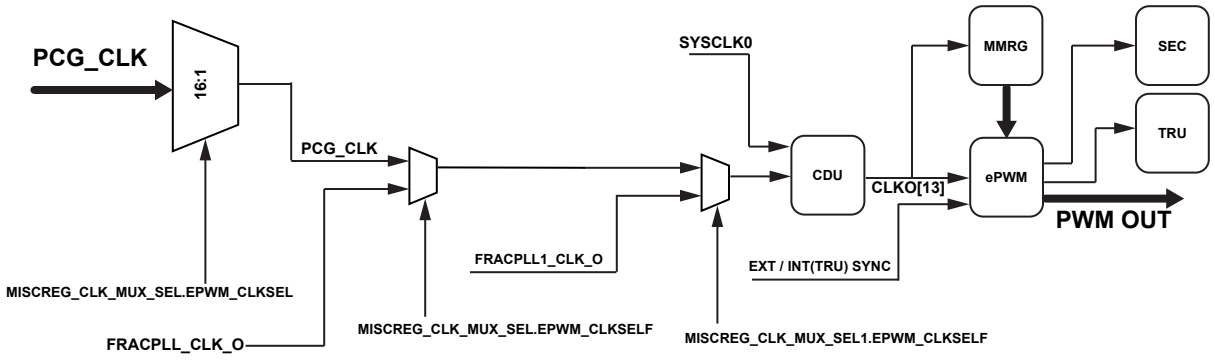


Figure 4-9: FPLL Output Clock Usage for PWM

Frac-N PLL Operating Modes

The Frac-N PLL operates in one of these two modes:

- FPLL Only Mode–clock generation using only the FPLL
- DPLL Plus FPLL Mode–clock generation synchronous with a lower frequency of an external reference clock source (FIN)

FPLL Only Mode

The FPLL generates a clock with frequency values that are fractional multiples of the CLKIN frequency. For example, a CLKIN frequency of 25 MHz can generate an audio clock of frequency 24.576 MHz. The output clock can be used from the FPLL either internally with audio peripherals (using the PCG) or driven out of the chip using DAI pin(s) and connected to external audio components. Configure the `FRACNPLL_FPLL_CTL0` and `FRACNPLL_FPLL_CTL1` registers to control the FPLL clock generation.

The *Clock Multiplexer Configurations for FPLL Only Mode* table shows the values for the different clock multiplexer configurations.

Table 4-7: Clock Multiplexer Configurations for FPLL Only Mode

Multiplexer Name	Register Bit Field	Required Value	Default Value
FIN	FRACNPLL_CTL.FIN_SEL	0	1
FIN-DFT	FRACNPLL_CTL.FIN_DFT	0	1
FFB-DFT	FRACNPLL_CTL.FFB_DFT	0	1
CLKS-DFT	FRACNPLL_CTL.CLKS_DFT	0	1
INT-DFT	FRACNPLL_CTL.DDIV_OVR	1	0
FRAC-DFT	FRACNPLL_CTL.FDIV_OVR	1	0

DPLL Plus FPLL Mode

DPLL Plus FPLL Mode–clock generation synchronized and multiplied with an external clock reference (FIN).

Use the DPLL or multiply and synchronize the output clock of FPLL with a lower-frequency external reference clock source (FIN). The DPLL provides the fractional and integer part of divider values to the FPLL. The processor feeds back the output clock of the FPLL (FOUTPH1 or FOUTPOSTDIV/2) to the DPLL to align with the FIN external clock reference phase.

FIN (External Reference Clock) Sources

Use one of the following options to select the FIN:

- Internal Ethernet PTPPPS signal – RGMII/EMAC0 block, that is, EMAC0_PTPPPS0 through EMAC0_PTPPPS03.
- External Ethernet PTPPPS or any other reference clock source – the GPIO (FRACNPLL_PTP_CLK) of the processor.
- External reference clock source connected – any DAI (0/1) pin of the processor routed using the PCGx_EXTCLK SRU signal.

Use the following steps to configure the FIN source:

1. Configure the value of the `FRACNPLL_CTL.FIN_SEL` field (=1) to select the FIN from DAI (`PCGx_EXTCLK`) or configure the value (=2) to select the FIN from the GPIO (FRACNPLL_PTP_CLK) or one of the EMAC0_PTPPPS0 through EMAC0_PTPPPS3 signals.
2. When the `FRACNPLL_CTL.FIN_SEL` field value (=1), engineers can configure the `MISCREG_CLK_MUX_SEL.PCG_MCLKSEL` bit to select the FIN source from different PCGx_EXTCLK signals. (See the *FIN Source Selection using MISCREG_CLK_MUX_SEL.PCG_MCLKSEL* table.)
3. When the `FRACNPLL_CTL.FIN_SEL` field value (=2), the `MISCREG_CLK_MUX_SEL.PTP_RCLKSEL` field can select the FIN source from either the GPIO (FRACNPLL_PTP_CLK) or one of the EMAC0_PTPPPS0 through EMAC0_PTPPPS3 signals. (See *FIN Source Selection using MISCREG_CLK_MUX_SEL. PTP_RCLKSEL* table).

   NOTE: The `FRACNPLL_CTL.REFCLK_IE` bit must be set (=1) when configuring GPIO (FRACNPLL_PTP_CLK) as the FIN source.

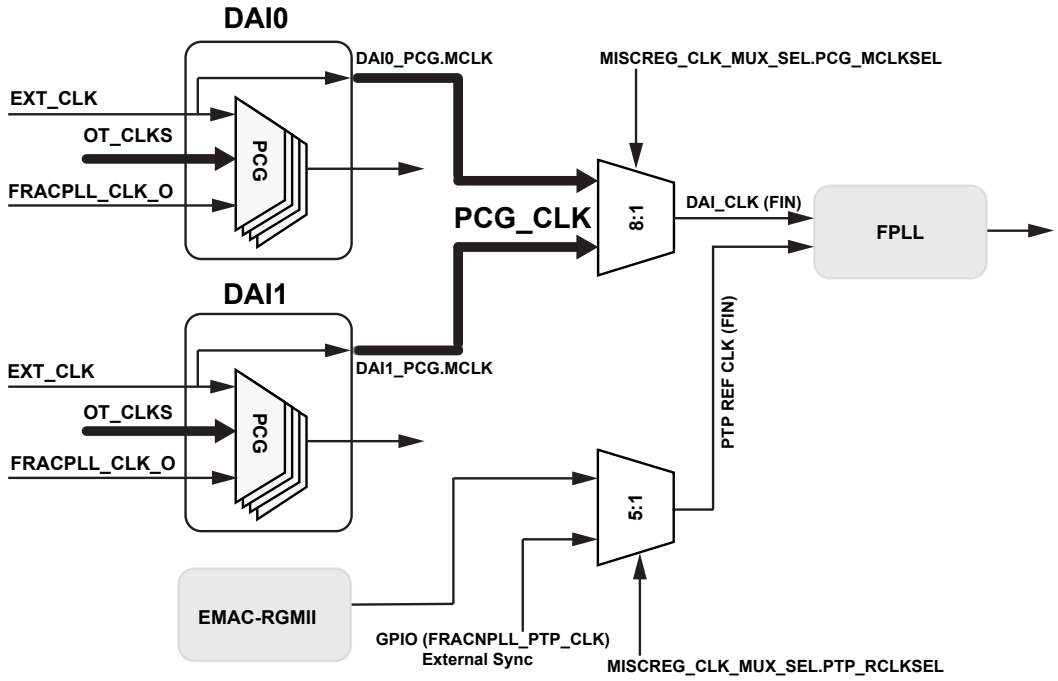


Figure 4-10: Possible FIN Sources

Table 4-8: FIN Source Selection using `MISCREG_CLK_MUX_SEL.PCG_MCLKSEL`

SL Number	**`MISCREG_CLK_MUX_SEL.PCG_MCLKSEL`**	FIN Source
1	0	DAI0_PCGA_EXTCLK
2	1	DAI0_PCGB_EXTCLK
3	2	DAI0_PCGC_EXTCLK
4	3	DAI0_PCGD_EXTCLK
5	4	DAI1_PCGA_EXTCLK
6	5	DAI1_PCGB_EXTCLK
7	6	DAI1_PCGC_EXTCLK
8	7	DAI1_PCGD_EXTCLK

Table 4-9: FIN source selection using `MISCREG_CLK_MUX_SEL.PTP_RCLKSEL`

SL Number	**`MISCREG_CLK_MUX_SEL.PTP_RCLKSEL`**	FIN Source
1	0	GPIO (FRACNPLL_PTP_CLK)
2	1	EMAC0_PTPPPS0

Table 4-9: FIN source selection using `MISCREG_CLK_MUX_SEL.PTP_RCLKSEL` (Continued)

SL Number	`MISCREG_CLK_MUX_SEL.PTP_RCLKSEL`	FIN Source
3	2	EMAC0_PTPPPS1
4	3	EMAC0_PTPPPS2
5	4	EMAC0_PTPPPS3

DPLL Plus FPLL Clock Multiplexer Configuration

The *Clock Multiplexer Configurations for the DPLL Plus FPLL Mode* table describes the multiplexer configurations.

Table 4-10: Clock Multiplexer Configurations for the DPLL Plus FPLL Mode

Multiplexer Name	Register Bit Field	Required Value	Default Value
FIN	`FRACNPLL_CTL.FIN_SEL`	1 or 2	1
FIN-DFT	`FRACNPLL_CTL.FIN_DFT`	1	1
FFB-DFT	`FRACNPLL_CTL.FFB_DFT`	1	1
CLKS-DFT	`FRACNPLL_CTL.CLKS_DFT`	1	1
INT-DFT	`FRACNPLL_CTL.DDIV_OVR`	0	0
FRAC-DFT	`FRACNPLL_CTL.FDIV_OVR`	0	0

Frac-N PLL Programming Model

This section describes the procedures to program the two Frac-N PLL operating modes.

- Programming FPLL Only Mode
- FPLL Only Mode TWAIT
- Programming Model for DPLL Plus FPLL Mode
- DPLL Plus FPLL Mode TWAIT (ADI does not recommend using this procedure.)
- Programming the FIN Unlock and Re-lock DPLL

Programming FPLL Only Mode

Use the following steps to program the FPLL Only Mode multiplexer configuration:

1. Configure the Frac-N PLL clock multiplexer using Table 4-7 Clock Multiplexer Configurations for FPLL Only Mode.
2. Keep DPLL and FPLL in reset by setting (=1) the `FRACNPLL_CTL.DCRESET` and `FRACNPLL_CTL.FCPD`) bits. By default, these bits are set after system reset.

3. Keep the FRACNPLL_CTL.CK_STBL bit cleared (=0). By default, these bits are cleared after system reset.
4. Configure the FRACNPLL_FPLL_CTL0.FRAC, FRACNPLL_FPLL_CTL1.INT, FRACNPLL_FPLL_CTL1.REFDIV, and FRACNPLL_FPLL_CTL1.POSTDIV fields.
5. Release FPLL from reset by clearing (=0) the FRACNPLL_CTL.FCPD bit.
6. Wait for the FPLL frequency lock by polling the FRACNPLL_STAT.FSLOCK bit (blocking mode) *or* configure the FRACNPLL_CTL.LOCK_SEL bit to generate the FPLL frequency lock interrupt (non-blocking mode).
7. Release the clock by setting (=1) the FRACNPLL_CTL.CK_STBL bit.

FPLL Only Mode TWAIT

Use this procedure to invoke the trigger receiver.

1. Configure Frac-N PLL clock multiplexer using Table 4-7 Clock Multiplexer Configurations for FPLL Only Mode.
2. Keep DPLL and FPLL in reset by setting (=1) the FRACNPLL_CTL.DCRESET and FRACNPLL_CTL.FCPD bits. By default, these bits are set after system reset.
3. Keep the FRACNPLL_CTL.CK_STBL bit cleared (=0). By default, these bits are cleared after system reset.
4. Configure the FRACNPLL_FPLL_CTL0.FRAC, FRACNPLL_FPLL_CTL1.INT, FRACNPLL_FPLL_CTL1.REFDIV, and FRACNPLL_FPLL_CTL1.POSTDIVfields.
5. Wait for the FPLL frequency lock by polling the FRACNPLL_STAT.FSLOCK bit (blocking mode) *or* configure the FRACNPLL_CTL.LOCK_SEL bit to generate the FPLL frequency lock interrupt (non-blocking mode).
6. Release the clock by setting (=1) the FRACNPLL_CTL.CK_STBL bit.
7. Set (=1) the FRACNPLL_CTL.F_TWAIT bit.
8. Set (=1) the FRACNPLL_CTL.FCPD bit.

On arrival of the trigger, the actual reset to FPLL is de-asserted and the FPLL starts operating as expected.

Programming Model for DPLL Plus FPLL Mode

Use the following steps to program the DPLL plus FPLL mode:

1. Configure Frac-N PLL clock multiplexer as per Table 4-10 Clock Multiplexer Configurations for the DPLL Plus FPLL Mode.
2. Keep DPLL and FPLL in reset by setting (=1) the FRACNPLL_CTL.DCRESET and FRACNPLL_CTL.FCPD bits. By default, these bits are set after a system reset.

3. Keep the FRACNPLL_CTL.CK_STBL bit cleared (=0). By default, these bits are cleared after system reset.
4. Configure parameters in the DPLL registers FRACNPLL_DPLL_CTL0 through FRACNPLL_DPLL_CTL9, including values for FRACNPLL_DPLL_CTL5.INT_PRESET and FRACNPLL_DPLL_CTL4.FRAC_PRESET. Clear the FRACNPLL_DPLL_CTL4.HOLD and FRACNPLL_DPLL_CTL7.PRESET_EN bits.
5. Release the DPLL reset by clearing (=0) the FRACNPLL_CTL.DCRESET bit.
6. Set (= 1) the FRACNPLL_DPLL_CTL7.PRESET_EN bit.
7. Set (= 1) the FRACNPLL_DPLL_CTL4.HOLD bit.
8. Configure the FRACNPLL_FPLL_CTL0/1 registers. FRACNPLL_FPLL_CTL0.FRAC and FRACNPLL_FPLL_CTL1.INT should be set to zero as these values DPLL provides in this mode. FPLL uses the FRACNPLL_DPLL_CTL5.INT_PRESET and FRACNPLL_DPLL_CTL4.FRAC_PRESET values programmed in step four, plus with the values programmed in FRACNPLL_FPLL_CTL1.REFDIV, and FRACNPLL_FPLL_CTL1.POSTDIV bit fields as a starting point for further frequency and phase lock process.
9. Once the FPLL has asserted a frequency lock, set (=1) the FRACNPLL_STAT.FSLOCK bit, and wait for the DPLL to assert its frequency lock (FRACNPLL_STAT.DSFLOCK=1).
10. Once the DPLL has asserted frequency lock, clear (=0) FRACNPLL_DPLL_CTL7.PRESET_EN and FRACNPLL_DPLL_CTL4.HOLD bits and wait for the DPLL to assert its phase lock (FRACNPLL_STAT.DSPHASE_LOCK=1).

    The DPLL phase lock can take few milliseconds to seconds. ADI recommends you use a non-blocking mode to generate an interrupt after the DPLL phase lock event.
11. Release the clock by setting (=1) the FRACNPLL_CTL.CK_STBL bit.

IMPORTANT: Programming the exact values of the FPLL and DPLL register settings for different FIN and FPLL output clock frequencies is not part of this chapter. The supported values of FIN and FPLL output clock frequencies and the required software routines to program the FPLL and DPLL are available using the FPLL Service along with the CrossCore Embedded Studio® software. ADI recommends you always use this service for programming the Frac-N PLL.

DPLL Plus FPLL Mode TWAIT

CAUTION: ADI does not recommend attempting to use the TWAIT feature in the DPLL Plus FPLL Mode.

Because of the complicated programming model, using the TWAIT feature for the DPLL plus FPLL mode may require the trigger to arrive somewhere the middle of the previously mentioned Programming Model for DPLL Plus FPLL Mode steps. This means a user must add an interrupt to use the TWAIT feature.

Programming the FIN Unlock and Re-lock DPLL

When the FIN is temporarily absent, the DPLL can become unlocked. Once the FIN clock signal enables, the DPLL attempts to lock again. To ensure a smooth DPLL unlock and relock sequence, use the following procedure:

1. Configure the DPLL to generate the unlock interrupt. See DPLL Unlock Interrupt for more information.
2. When the DPLL unlocks because of the FIN absence, it generates a DPLL unlock interrupt. Inside the unlock interrupt service routine, set (=1) the `FRACNPLL_DPLL_CTL4.HOLD` bit. Note: the application can use any other interrupt mechanism as well (for example, SPDIF receiver loss of lock interrupt if SPDIF receiver is the source of FIN) instead of DPLL unlock interrupt to signal loss of FIN.
3. DPLL cannot generate lock interrupt when in hold state. So, the application must have a way to interrupt the core when FIN is reconnected back (for example, SPDIF receiver lock interrupt if SPDIF receiver is the source of FIN). Inside the interrupt service routine, clear (=0) the `FRACNPLL_DPLL_CTL4.HOLD` bit to release DPLL hold.
4. Wait for the DPLL to assert a phase lock (`FRACNPLL_STAT.DSPHASE_LOCK`=1) again. Use the DPLL phase lock interrupt in this case for the non-blocking mode.

ADSP-2184x FRACNPLL Register Descriptions

The Fractional Phase-locked loop (FRACNPLL) contains the following registers.

Table 4-11: ADSP-2184x FRACNPLL Register List

Name	Description
FRACNPLL_CTL	Integration Control Register
FRACNPLL_DPLL_CTL0	DPLL Control Register 0
FRACNPLL_DPLL_CTL1	DPLL Control Register 1
FRACNPLL_DPLL_CTL2	DPLL Control Register 2
FRACNPLL_DPLL_CTL3	DPLL Control Register 3
FRACNPLL_DPLL_CTL4	DPLL Control Register 4
FRACNPLL_DPLL_CTL5	DPLL Control Register 5
FRACNPLL_DPLL_CTL6	DPLL Control Register 6
FRACNPLL_DPLL_CTL7	DPLL Control Register 7
FRACNPLL_DPLL_CTL8	DPLL Control Register 8
FRACNPLL_DPLL_CTL9	DPLL Control Register 9
FRACNPLL_DPLL_STAT0	DPLL Status Register 0
FRACNPLL_DPLL_STAT1	DPLL Status Register 1
FRACNPLL_DPLL_STAT2	DPLL Status Register 2

Table 4-11: ADSP-2184x FRACNPLL Register List (Continued)

Name	Description
FRACNPLL_FPLL_CTL0	Fractional PLL Control Register 0
FRACNPLL_FPLL_CTL1	Fractional PLL Control Register 1
FRACNPLL_STAT	Integration Status Register

Integration Control Register

The FRACNPLL_CTL register controls DPLL reset, DFT modes, clock selections, and trigger wait and power down functions.

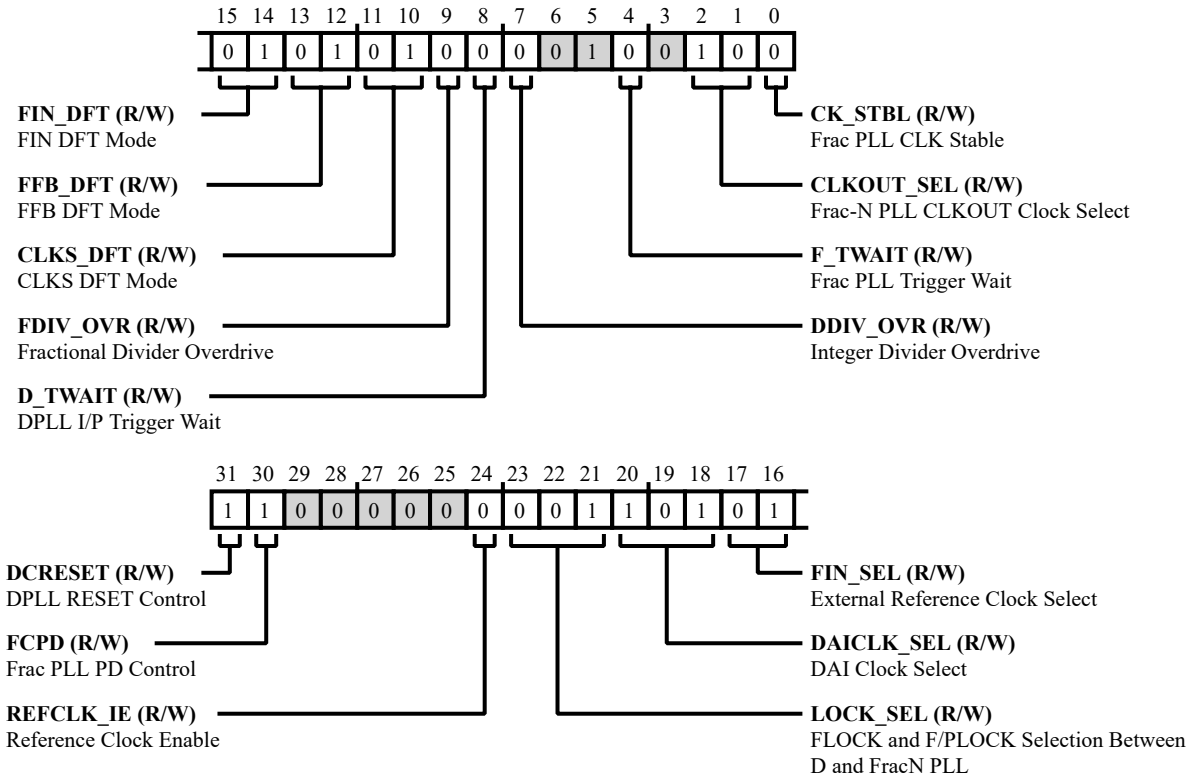


Figure 4-11: FRACNPLL_CTL Register Diagram

Table 4-12: FRACNPLL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	DCRESET	DPLL RESET Control. Resets all internal counters and registers to their default state
30 (R/W)	FCPD	Frac PLL PD Control. Power down for PLL
24 (R/W)	REFCLK_IE	Reference Clock Enable. Input Enable for PLL REFCLK over GPIO
23:21 (R/W)	LOCK_SEL	FLOCK and F/PLOCK Selection Between D and FracN PLL. FPLL FLOCK/PLOCK and DPLL FLOCK/PLOCK selection

Table 4-12: FRACNPLL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20:18 (R/W)	DAICLK_SEL	DAI Clock Select. Selects the output of Frac-N PLL DAI clock multiplexer. 0 - No clock 1 - FOUTPOSTDIV/16 2 - FOUTPOSTDIV/8 3 - FOUTPOSTDIV/4 4 - FOUTPOSTDIV/2 5 - FOUTPOSTDIV
17:16 (R/W)	FIN_SEL	External Reference Clock Select. Multiplex selection for external reference clock. 0 - No clock 1 - Reference clock from DAI 2 - Reference clock from internal or external EMAC-PTPPPS
15:14 (R/W)	FIN_DFT	FIN DFT Mode. DFT mode for external reference clock. 0 - No clock 1 - DAI or EMAC ref clock 2 - DFT ref clock
13:12 (R/W)	FFB_DFT	FFB DFT Mode. DFT mode for feedback clock. 0 - No clock 1 - Feedback clock form FPLL (FOUT1PH0) 2 - DFT feedback clock
11:10 (R/W)	CLKS_DFT	CLKS DFT Mode. DFT mode for CLKS. 0 - No clock 1 - CLKS form FPLL (CLKSSCG) 2 - DFT CLK
9 (R/W)	FDIV_OVR	Fractional Divider Overdrive. Fractional divider override to FPLL in FPLL only mode
8 (R/W)	D_TWAIT	DPLL I/P Trigger Wait.
7 (R/W)	DDIV_OVR	Integer Divider Overdrive. Integer divider override to FPLL in FPLL only mode

Table 4-12: FRACNPLL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	F_TWAIT	Frac PLL Trigger Wait. Frac PLL I/P Trigger Wait
2:1 (R/W)	CLKOUT_SEL	Frac-N PLL CLKOUT Clock Select. Selects the output of FPLL CLKOUT multiplexer which will be given as input to the main CLKOUT multiplexer. 1 - CLKSSCG 2 - FOUTPOSTDIV/64 3 - FOUTVCO/64
0 (R/W)	CK_STBL	Frac PLL CLK Stable. Setting the `FRACNPLL_CTL.CK_STBL` bit ensures sending out a stable clock by enabling clock gating control after the Frac-N PLL DAI clock multiplexer.

DPLL Control Register 0

The FRACNPLL_DPLL_CTL0 register contains the DPLL count offset value used for frequency multiplication in the frequency control loop.

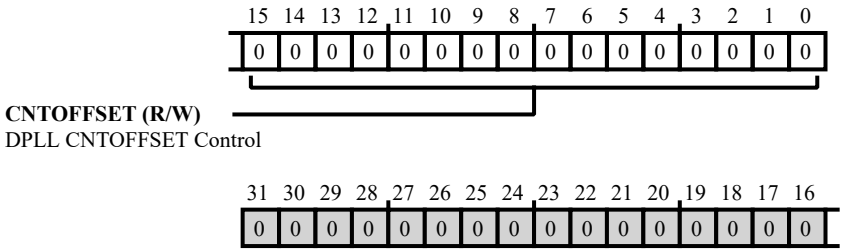


Figure 4-12: FRACNPLL_DPLL_CTL0 Register Diagram

Table 4-13: FRACNPLL_DPLL_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	CNTOFFSET	DPLL CNTOFFSET Control. Allows for frequency multiplication in the frequency control loop. By setting CNTOFFSET to SYNCCNTMAX / K, where K is the multiplication ratio, the frequency detector will output a zero frequency error when CNTOFFSET pulses have been counted from the reference and SYNCCNTMAX pulses have been counted from the feedback.

DPLL Control Register 1

The FRACNPLL_DPLL_CTL1 register controls DPLL delay.

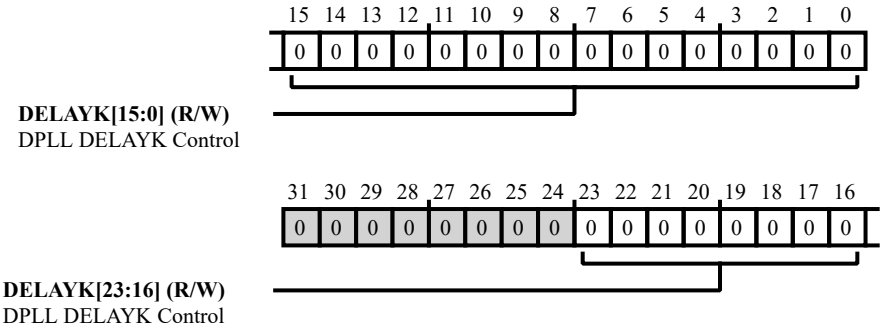


Figure 4-13: FRACNPLL_DPLL_CTL1 Register Diagram

Table 4-14: FRACNPLL_DPLL_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/W)	DELAYK	DPLL DELAYK Control. The FRACNPLL_DPLL_CTL1.DELAYK bit field indicates the phase locking loop counts for DELAYK for each of PH_STEP_CNT gain settings from M * 2^K * $2^{PH_STEP_CNT}$ down to M * 2^K. Total lock count is PH_STEP_CNT*DELAYK

DPLL Control Register 2

The FRACNPLL_DPLL_CTL2 register shows the DPLL integer divider for FIN and FFB.

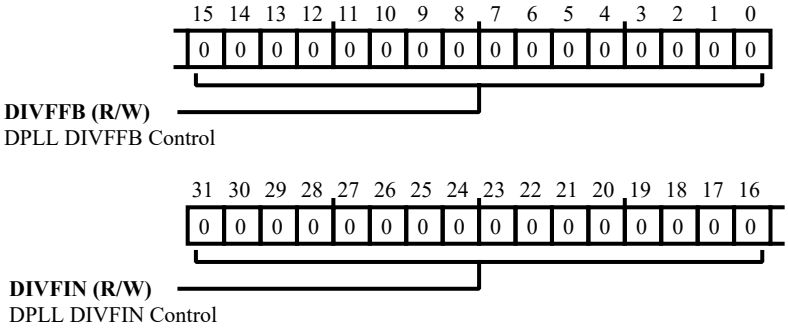


Figure 4-14: FRACNPLL_DPLL_CTL2 Register Diagram

Table 4-15: FRACNPLL_DPLL_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DIVFIN	DPLL DIVFIN Control. The FRACNPLL_DPLL_CTL2.DIVFIN bit field indicates the integer divider for FIN.
15:0 (R/W)	DIVFFB	DPLL DIVFFB Control. The FRACNPLL_DPLL_CTL2.DIVFFB bit field indicates the integer divider for FFB.

DPLL Control Register 3

The FRACNPLL_DPLL_CTL3 register controls the gain for the integral frequency comparison loop and frequency operations.

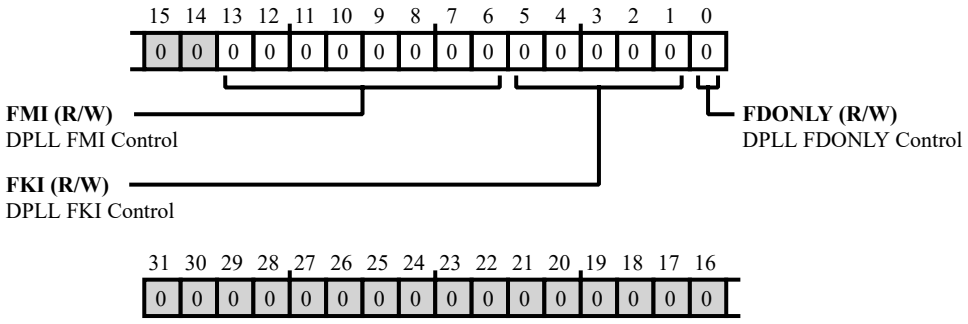


Figure 4-15: FRACNPLL_DPLL_CTL3 Register Diagram

Table 4-16: FRACNPLL_DPLL_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:6 (R/W)	FMI	DPLL FMI Control. The FRACNPLL_DPLL_CTL3.FMI bit field indicates the M gain for the integral frequency comparison loop where Gain=M * 2^K.
5:1 (R/W)	FKI	DPLL FKI Control. The FRACNPLL_DPLL_CTL3.FKI bit field indicates the K gain for the integral frequency comparison loop where Gain=M * 2^K.
0 (R/W)	FDONLY	DPLL FDONLY Control. The FRACNPLL_DPLL_CTL3.FDONLY bit indicates whether test mode or normal operation is used. 0 - Frequency Control Only (Test Mode) 1 - Normal Operation (Frequency and Phase)

DPLL Control Register 4

The FRACNPLL_DPLL_CTL4 register controls the hold output state and FRAC output.

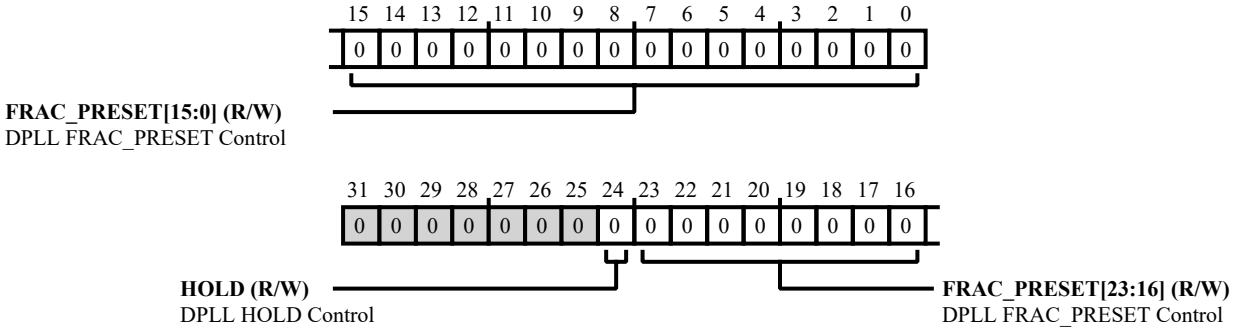


Figure 4-16: FRACNPLL_DPLL_CTL4 Register Diagram

Table 4-17: FRACNPLL_DPLL_CTL4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	HOLD	DPLL HOLD Control. The FRACNPLL_DPLL_CTL4.HOLD bit indicates the hold output state of the DPLL. 0 - Normal operation 1 - Hold. The PLL will not phase lock in this state.
23:0 (R/W)	FRAC_PRESET	DPLL FRAC_PRESET Control. The FRACNPLL_DPLL_CTL4.FRAC_PRESET bit field indicates the preset FRAC output. When PRESET_EN=1, the output is set to this value.

DPLL Control Register 5

The FRACNPLL_DPLL_CTL5 register controls some diagnostic operations and ISHIFT fields.

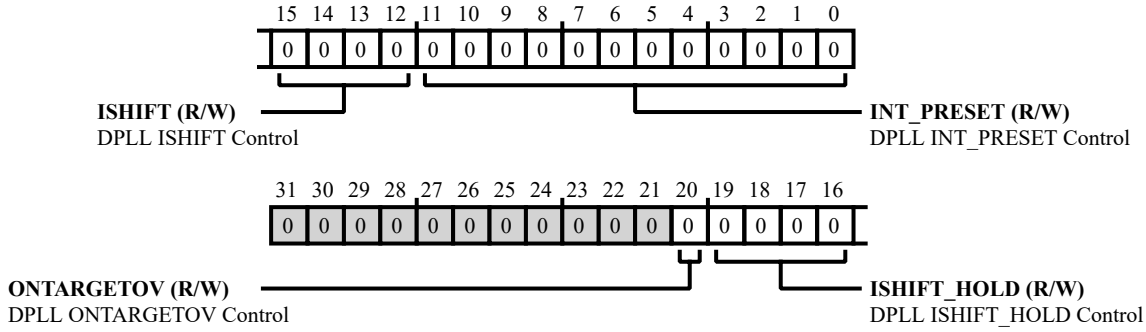


Figure 4-17: FRACNPLL_DPLL_CTL5 Register Diagram

Table 4-18: FRACNPLL_DPLL_CTL5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20 (R/W)	ONTARGETOV	DPLL ONTARGETOV Control. The FRACNPLL_DPLL_CTL5.ONTARGETOV bit indicates diagnostic control (test mode) or normal operation. 0 - Diagnostic Mode 1 - Normal Operation
19:16 (R/W)	ISHIFT_HOLD	DPLL ISHIFT_HOLD Control. The FRACNPLL_DPLL_CTL5.ISHIFT_HOLD bit field indicates the ISHIFT value used after exiting from HOLD Gain of 2^-ISHIFT from P-Path to I-Path integrator in phase lock mode.
15:12 (R/W)	ISHIFT	DPLL ISHIFT Control. The FRACNPLL_DPLL_CTL5.ISHIFT bit field indicates gain of $2^{-ISHIFT}$ from P-Path to I-Path integrator in phase lock mode.
11:0 (R/W)	INT_PRESET	DPLL INT_PRESET Control. The FRACNPLL_DPLL_CTL5.INT_PRESET bit field indicates the preset INT output. When PRESET_EN=1b1, the output is set to this value.

DPLL Control Register 6

The FRACNPLL_DPLL_CTL6 register controls FFB clock delays.

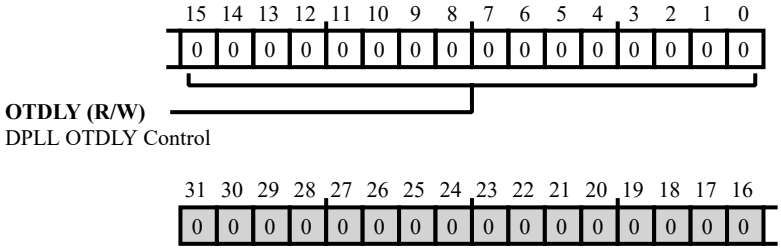


Figure 4-18: FRACNPLL_DPLL_CTL6 Register Diagram

Table 4-19: FRACNPLL_DPLL_CTL6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	OTDLY	DPLL OTDLY Control. The FRACNPLL_DPLL_CTL6.OTDLY bit field indicates the delay in FFB clock periods from when FLOCK=1 to when the DPLL transfers control from the frequency comparison loop to the phase comparison loop. The delay allows additional settling and avoids glitches in FLOCK output. The recommended value is 25.

DPLL Control Register 7

The FRACNPLL_DPLL_CTL7 register controls DPLL bits used for diagnostics and gain functions.

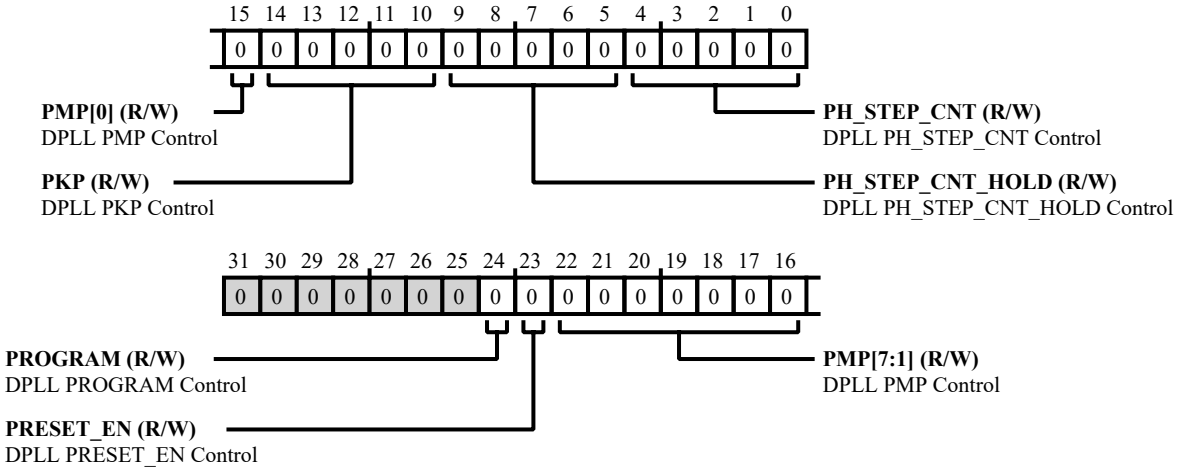


Figure 4-19: FRACNPLL_DPLL_CTL7 Register Diagram

Table 4-20: FRACNPLL_DPLL_CTL7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	PROGRAM	DPLL PROGRAM Control. The FRACNPLL_DPLL_CTL7.PROGRAM bit indicates diagnostic control (test mode) or normal operation. 0 - Diagnostic Mode 1 - Normal Operation
23 (R/W)	PRESET_EN	DPLL PRESET_EN Control. The FRACNPLL_DPLL_CTL7.PRESET_EN bit indicates whether the preset INT and FRAC values are loaded into DPLL. 0 - Normal Operation 1 - Switch to FRAC_PRESET and INT_PRESET and relock
22:15 (R/W)	PMP	DPLL PMP Control. The FRACNPLL_DPLL_CTL7.PMP bit field indicates the M gain for the proportional phase comparison loop where Gain=M * 2^K.
14:10 (R/W)	PKP	DPLL PKP Control. The FRACNPLL_DPLL_CTL7.PKP bit field indicates the K gain for the proportional phase comparison loop where Gain=M * 2^K.

Table 4-20: FRACNPLL_DPLL_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	PH_STEP_CNT_HOLD	DPLL PH_STEP_CNT_HOLD Control. The `FRACNPLL_DPLL_CTL7.PH_STEP_CNT_HOLD` bit field indicates the PH_STEP_CNT value used after exiting from HOLD. Number of steps in phase lock process. Initial gain is Gain_init= M * 2^{K} * $2^{PH_STEP_CNT}$. Total lock count is PH_STEP_CNT*DELAYK.
4:0 (R/W)	PH_STEP_CNT	DPLL PH_STEP_CNT Control. The `FRACNPLL_DPLL_CTL7.PH_STEP_CNT` bit field indicates the number of steps in phase lock process. Initial gain is Gain_init= M * 2^{K} * $2^{PH_STEP_CNT}$. Total lock count is PH_STEP_CNT*DELAYK

DPLL Control Register 8

The FRACNPLL_DPLL_CTL8 register controls inputs for FIN pulses.

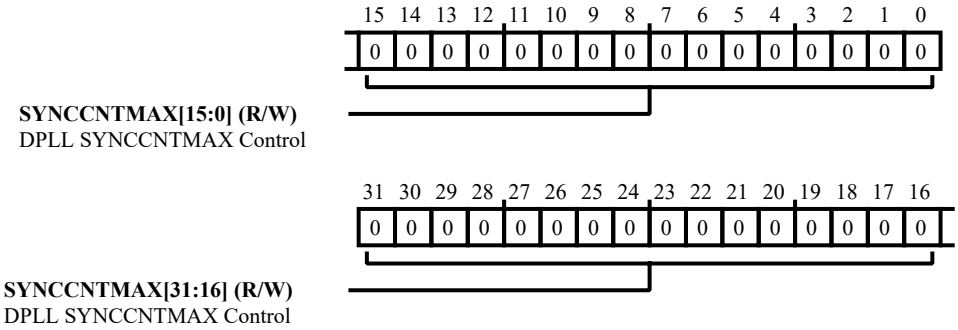


Figure 4-20: FRACNPLL_DPLL_CTL8 Register Diagram

Table 4-21: FRACNPLL_DPLL_CTL8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SYNCCNTMAX	DPLL SYNCCNTMAX Control. FIN pulses are counted for FRACNPLL_DPLL_CTL8.SYNCCNTMAX to determine a frequency match.

DPLL Control Register 9

The FRACNPLL_DPLL_CTL9 register holds the target pulse count used for frequency comparison.

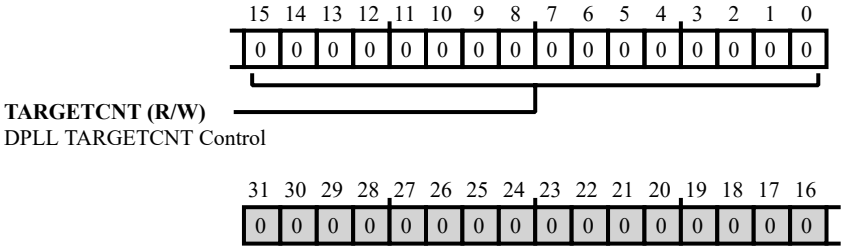


Figure 4-21: FRACNPLL_DPLL_CTL9 Register Diagram

Table 4-22: FRACNPLL_DPLL_CTL9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TARGETCNT	DPLL TARGETCNT Control. The FRACNPLL_DPLL_CTL9.TARGETCNT bit field indicates whether the number of pulse counts between FIN and FFB is less than TARGETCNT. The frequency comparator declares a frequency match (When FLOCK=1).

DPLL Status Register 0

The FRACNPLL_DPLL_STAT0 register contains the fractional portion of the feedback divide value.

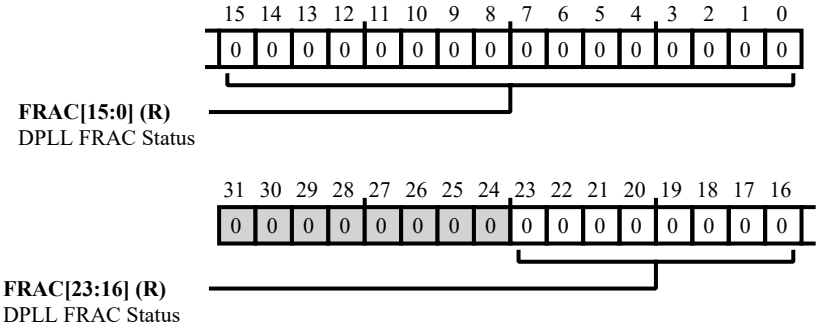


Figure 4-22: FRACNPLL_DPLL_STAT0 Register Diagram

Table 4-23: FRACNPLL_DPLL_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/NW)	FRAC	DPLL FRAC Status. The FRACNPLL_DPLL_STAT0.FRAC bit field indicates the fractional portion of feedback divide value.

DPLL Status Register 1

The FRACNPLL_DPLL_STAT1 register contains the fractional portion of the phase detector integrator output.

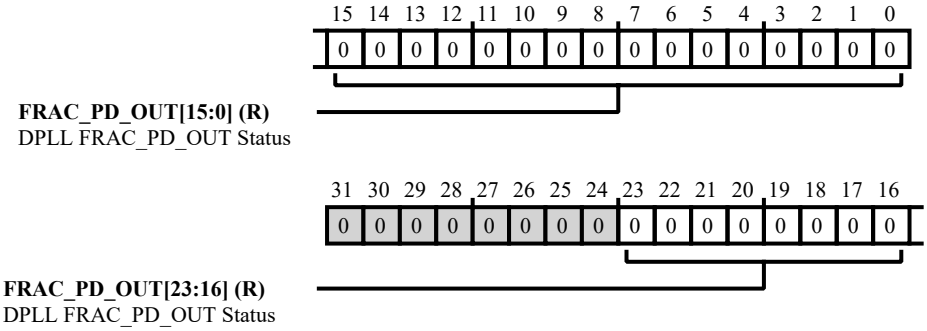


Figure 4-23: FRACNPLL_DPLL_STAT1 Register Diagram

Table 4-24: FRACNPLL_DPLL_STAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/NW)	FRAC_PD_OUT	DPLL FRAC_PD_OUT Status. The FRACNPLL_DPLL_STAT1.FRAC_PD_OUT bit field indicates the fractional bits of the phase detector integrator output.

DPLL Status Register 2

The FRACNPLL_DPLL_STAT2 register contains the integer portion of the phase detector integrator output.

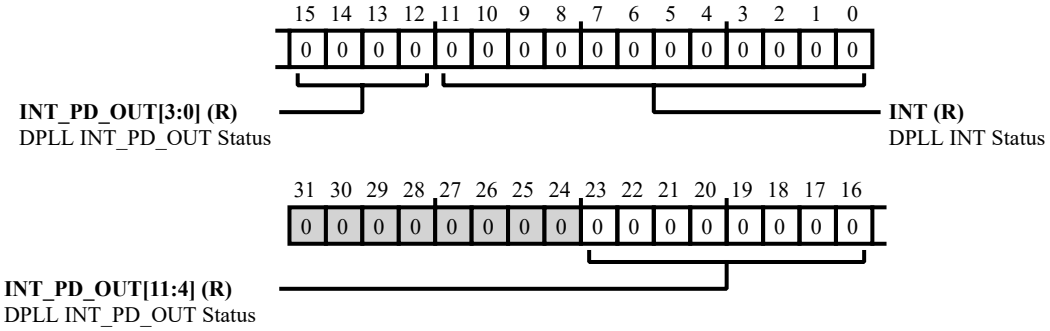


Figure 4-24: FRACNPLL_DPLL_STAT2 Register Diagram

Table 4-25: FRACNPLL_DPLL_STAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:12 (R/NW)	INT_PD_OUT	DPLL INT_PD_OUT Status. The FRACNPLL_DPLL_STAT2.INT_PD_OUT bit field indicates the integer bits of the phase detector integrator output.
11:0 (R/NW)	INT	DPLL INT Status. The FRACNPLL_DPLL_STAT2.INT bit field indicates the integer divider for Frac-N PLL, clocked out on the rising edge of CLKS.

Fractional PLL Control Register 0

The FRACNPLL_FPLL_CTL0 register contains the fractional divider setting for Fractional-N PLL, various power down controls, and bypass control.

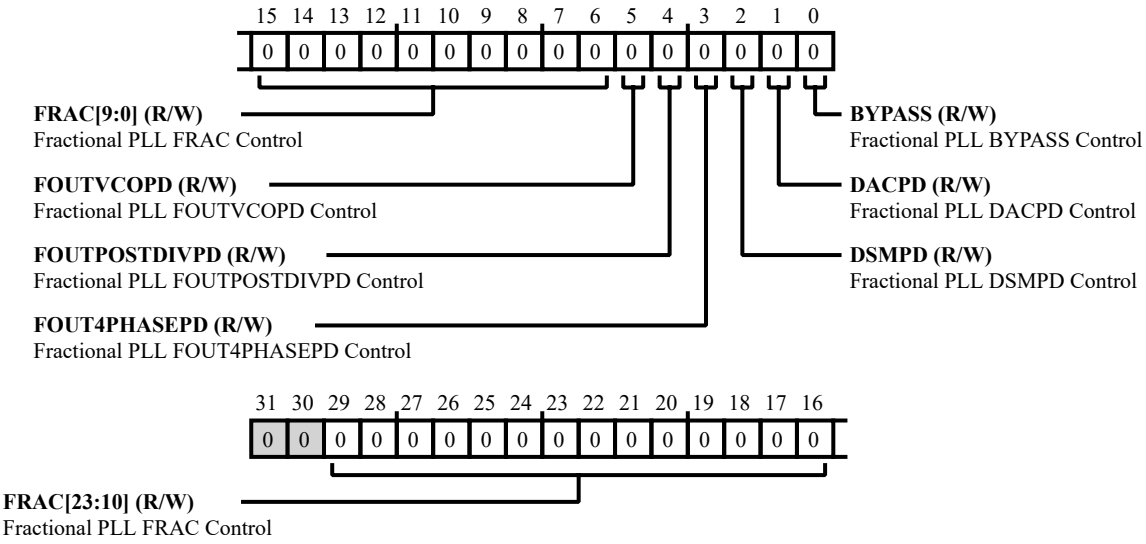


Figure 4-25: FRACNPLL_FPLL_CTL0 Register Diagram

Table 4-26: FRACNPLL_FPLL_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:6 (R/W)	FRAC	Fractional PLL FRAC Control. Fractional divider setting for Fractional-N PLL, clocked out on the rising edge of CLKS
5 (R/W)	FOUTVCOPD	Fractional PLL FOUTVCOPD Control. VCO rate output clock power down
4 (R/W)	FOUTPOSTDIVPD	Fractional PLL FOUTPOSTDIVPD Control. Post divide power down
3 (R/W)	FOUT4PHASEPD	Fractional PLL FOUT4PHASEPD Control. Power down of 4 phase clock generator; 4phase output is also powered down by FOUTPOSTDIVPD
2 (R/W)	DSMPD	Fractional PLL DSMPD Control. Power down Delta-Sigma Modulator; 0-> DSM is active; 1-> DSM is powered down
1 (R/W)	DACPD	Fractional PLL DACPD Control. Power down noise canceling DAC in FRAC mode; 0-> DAC is active (default mode); 1-> DAC is not active (test mode only)

Table 4-26: FRACNPLL_FPLL_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS	Fractional PLL BYPASS Control. FREF is bypassed to FOUTPOSTDIV

Fractional PLL Control Register 1

The FRACNPLL_FPLL_CTL1 register contains the reference divide value, post divide setting, and feedback divide value.

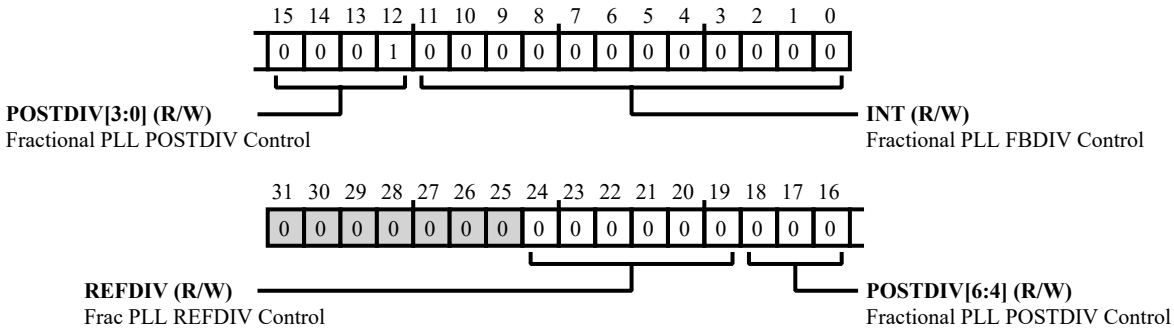


Figure 4-26: FRACNPLL_FPLL_CTL1 Register Diagram

Table 4-27: FRACNPLL_FPLL_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:19 (R/W)	REFDIV	Frac PLL REFDIV Control. Reference divide value (1 to 63)
18:12 (R/W)	POSTDIV	Fractional PLL POSTDIV Control. PLL post divide setting (1 to 127).
11:0 (R/W)	INT	Fractional PLL FBDIV Control. PLL feedback divide value (16 to 4095)

Integration Status Register

The FRACNPLL_STAT status register contains phase, frequency lock status of FPLL and DPLL along with DPLL transition status.

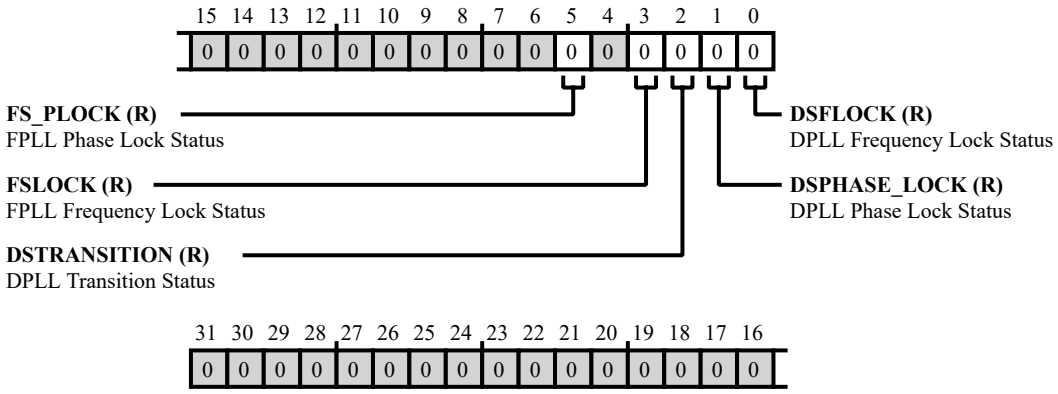


Figure 4-27: FRACNPLL_STAT Register Diagram

Table 4-28: FRACNPLL_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/NW)	FS_PLOCK	FPLL Phase Lock Status. The FRACNPLL_STAT.FS_PLOCK bit indicates whether the FPLL has achieved a phase lock.
3 (R/NW)	FSLOCK	FPLL Frequency Lock Status. The FRACNPLL_STAT.FSLOCK bit indicates whether the FPLL has achieved a frequency lock.
2 (R/NW)	DSTRANSITION	DPLL Transition Status. The FRACNPLL_STAT.DSTRANSITION bit indicates the control transition output. Test Mode Output. When FRACNPLL_STAT.DSTRANSITION = 0, it occurs after transition to fine tune phase control (PHASE LOCK=1b1), When FRACNPLL_STAT.DSTRANSITION = 1, it occurs after phase control engaged (FLOCK=1b1).
1 (R/NW)	DSPHASE_LOCK	DPLL Phase Lock Status. The FRACNPLL_STAT.DSPHASE_LOCK bit indicates whether the DPLL is in fine tune phase tracking mode. When FRACNPLL_STAT.DSPHASE_LOCK = 1, the DPLL is settled and phase locked. When FRACNPLL_STAT.DSPHASE_LOCK = 0, the DPLL is still settling or (if FLOCK=1b0) not frequency locked.

Table 4-28: FRACNPLL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	DSFLOCK	DPLL Frequency Lock Status. The `FRACNPLL_STAT.DSFLOCK` bit indicates whether the DPLL has achieved a frequency lock. When `FRACNPLL_STAT.DSFLOCK` = 1, the frequencies have maintained lock within TARGETCNT for OTDLY FFB clock periods When `FRACNPLL_STAT.DSFLOCK` = 0, the frequency comparator has detected a mismatch in FIN and FFB pulse counts.

5 Dynamic Power Management (DPM)

The dynamic power management (DPM) unit of the processor controls transitions between different power-saving modes.

DPM provides facility for shutting down clocks for various peripherals. Shutting down peripheral's clocks is considered a power saving feature. To support peripheral clock gating functionality, DPM_PER_DIS0 and DPM_PER_DIS1 are available in the DPM module. The bits in these registers need to be programmed to shut off the clocks to peripherals that are not used in an application to save clock switching power.

DPM Features

The DPM allows programs to control the power mode of the processor as follows.

- Permits operation of multiple, external wake-up sources

DPM Functional Description

The DPM can be programmed to transition between power modes.

ADSP-2184x DPM Register List

A set of registers govern DPM operations. For more information on DPM functionality, see the DPM register descriptions.

Table 5-1: ADSP-2184x DPM Register List

Name	Description
DPM_CTL	Control Register
DPM_PER_DIS0	Peripherals Disable Register 0
DPM_PER_DIS1	Peripherals Disable Register 1
DPM_REVID	Revision ID
DPM_STAT	Status Register

DPM Definitions

To make the best use of the DPM, it is useful to understand the following terms.

CGU

Acronym for the clock generation unit (CGU), which is comprised of the PLL and PCU

DPM

Acronym for the dynamic power management (DPM) controller.

Full-on mode

The normal operating mode in which all clock domains are derived from the PLL.

PCU

Acronym for the PLL control unit (PCU).

PLL

Acronym for the phase-locked loop (PLL).

RCU

Acronym for the reset control unit (RCU).

DPM Operating Modes

The DPM includes several operating modes. The modes are:

- Reset
- Full-on

Reset State

Reset is the initial state of the processor and is the result of a hardware or software triggered event. The DPM itself does not trigger entering reset. The external SYS_HWRST pin or the RCU triggers entering reset. The DPM responds to reset by transitioning to its default state.

From Reset, the DPM always transitions to PLL Bypassed state.

Full-on Mode

Full-on mode is the default state of the DPM after Reset.

In full-on mode, the processor can reach its maximum clock rate and power dissipation can be at its highest.

DPM Event Control

The DPM event is triggered when an enabled wake-up is asserted. The DPM generates bus errors when a misaligned access to a register occurs. It also generates errors when an attempt is made to access unused DPM address space or a write-protected register.

DPM Events

The DPM event interrupt is triggered when any bit in the DPM_STAT register is set, indicating that an enabled wake-up was asserted. The DPM event interrupt stays active until the user clears any bits that are set in the DPM_STAT register.

DPM Errors

The DPM generates a bus error when a read or write transaction is attempted to an unused address within the DPM address range. It also generates a bus error when a misaligned access is made to a DPM register. In addition to the bus error, the DPM sets the DPM_STAT.ADDRERR bit.

If a write to a write-protected DPM register is attempted, the DPM generates a bus error. In addition, the DPM sets the DPM_STAT.LWERR bit.

ADSP-2184x DPM Register Descriptions

Dynamic Power Management (DPM) contains the following registers.

Table 5-2: ADSP-2184x DPM Register List

Name	Description
DPM_CTL	Control Register
DPM_PER_DIS0	Peripherals Disable Register 0
DPM_PER_DIS1	Peripherals Disable Register 1
DPM_REVID	Revision ID
DPM_STAT	Status Register

Control Register

The DPM_CTL register controls sleep modes selections and PLL operations of the DPM. A write protect feature permits locking out changes to this register.

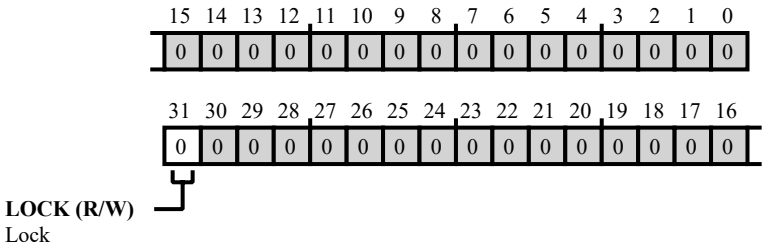


Figure 5-1: DPM_CTL Register Diagram

Table 5-3: DPM_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the DPM_CTL.LOCK bit is set, the DPM_CTL register is read only (locked).	
		0	Unlock
		1	Lock

Peripherals Disable Register 0

The DPM_PER_DIS0 register is used to shut off the clocks to peripherals that are not used in an application in order to save clock switching power.

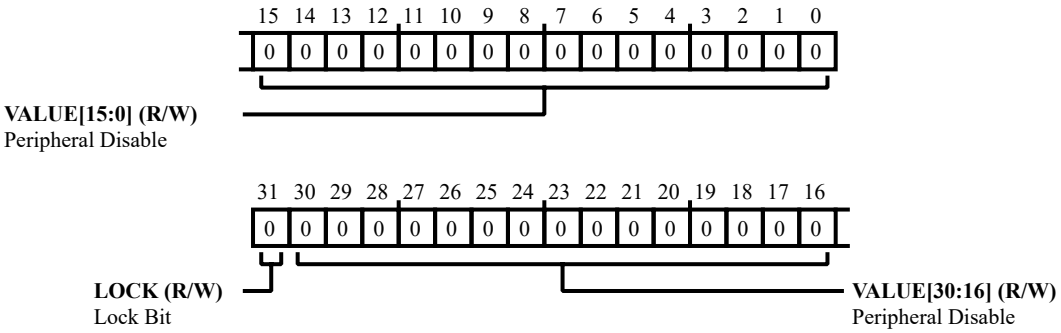


Figure 5-2: DPM_PER_DIS0 Register Diagram

Table 5-4: DPM_PER_DIS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
30:0 (R/W)	VALUE	Peripheral Disable. The DPM_PER_DIS0.VALUE bits are used to shut of clocks in individual peripherals.	
		0	SH0_FIR0
		1	SH0_IIR0
		2	SH0_IIR1
		3	SH0_IIR2
		4	SH0_IIR3
		5	DAI0
		6	DAI1
		7	Reserved
		8	Reserved
		9	MLB0
		10	EMAC0 (10/100/1000)
		12	MSHC0
		13	EMDMA0
		14	EMDMA1

Table 5-4: DPM_PER_DIS0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		15	Reserved
		16	CRYPTO ACCELERATOR-0 (EIP-150/PKP)
		17	CRYPTO ACCELERATOR-1 (EIP-93/SPE)
		18	SMPU-SMOY
		19	SMPU_SPIF (SPI2)
		20	SMPU-L2CTL-CL2-0
		21	SMPU-L2CTL-DL2-0
		22	SMPU-L2CTL-CL2-1
		23	SMPU-L2CTL-DL2-1
		24	SMPU-L2CTL-CL2-2
		25	Reserved
		26	Reserved
		27	SMPU-DMC0
		28	Reserved
		29	CAN0
		30	CAN1

Peripherals Disable Register 1

The DPM_PER_DIS1 register is used to shut off the clocks to peripherals that are not used in an application to save clock switching power.

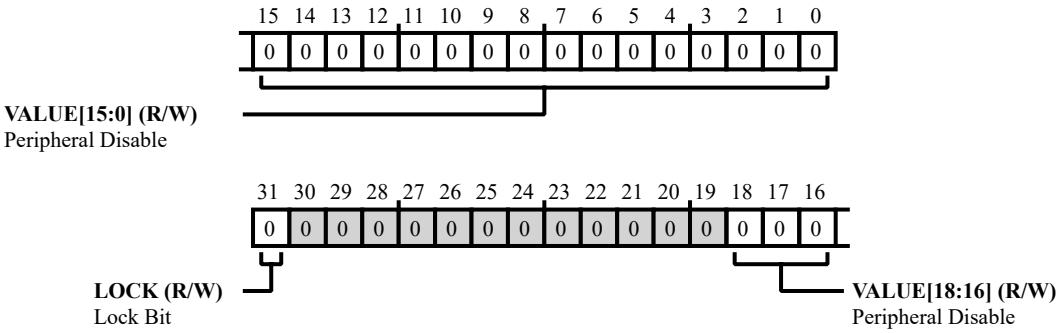


Figure 5-3: DPM_PER_DIS1 Register Diagram

Table 5-5: DPM_PER_DIS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
18:0 (R/W)	VALUE	Peripheral Disable. The DPM_PER_DIS1.VALUE bits are used to shut of clocks in individual peripherals.	
		0	TWI0
		1	TWI1
		2	TWI2
		3	TWI3
		4	TWI4
		5	TWI5
		6	xSPI_SYSCLK
		7	Reserved
		8	Reserved
		9	SH0_FIR1/FFT
		10	Reserved
		11	Reserved
		12	Reserved
		13	Reserved

Table 5-5: DPM_PER_DIS1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		14	XSPI_CLK010
		15	PDM
		16	XSPI1_sysclk
		17	XSPI1_clko08
		18	LPDDR4

Revision ID

The DPM_REVID register provides the revision of the DPM module.

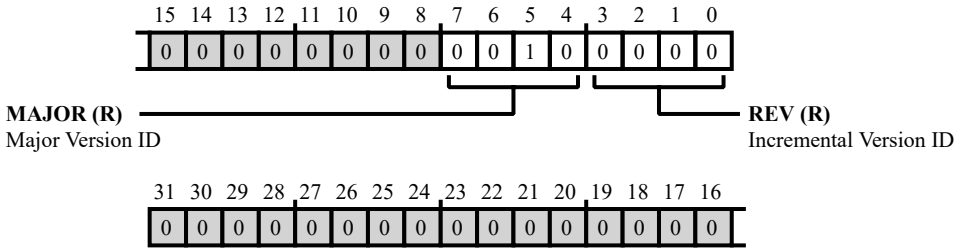


Figure 5-4: DPM_REVID Register Diagram

Table 5-6: DPM_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version ID.
3:0 (R/NW)	REV	Incremental Version ID.

Status Register

The DPM_STAT register contains bits that report the state of the module and various errors.

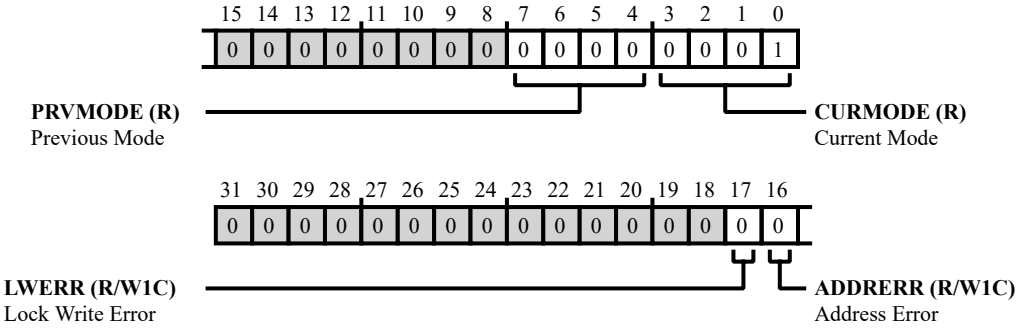


Figure 5-5: DPM_STAT Register Diagram

Table 5-7: DPM_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	LWERR	Lock Write Error. The DPM_STAT.LWERR bit indicates that a write transaction attempted an access to a write protected register. Triggers the DPMLV_PSLVERR interrupt.	
		0	Inactive
		1	Active
16 (R/W1C)	ADDRERR	Address Error. The DPM_STAT.ADDRERR bit indicates that a read or write transaction attempted an access to an unimplemented address or a write transaction attempted an access to a read only register or accesses are non aligned. Triggers the DPMLV_PSLVERR interrupt.	
		0	Inactive
		1	Active
7:4 (R/NW)	PRVMODE	Previous Mode. The DPM_STAT.PRVMODE bit field indicates the previous mode of the module.	
		0	Reset
		1	Full-On
		2	Reserved
		3	Reserved
		4	Reserved
		5-15	Reserved

Table 5-7: DPM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/NW)	CURMODE	Current Mode. The `DPM_STAT.CURMODE` bit field indicates the current mode of the module.	
		0	Reserved
		1	Full-On
		2	Reserved
		3	Reserved
		4-15	Reserved

6 Reset Control Unit (RCU)

Reset is the initial state of the processor at power-on or the run-time state of any core, as controlled by another core in the device using the RCU or as a result of a hardware or software triggered event. In this state, all control registers are set to their default values and functional units are idle. Exiting a full system hardware reset results with only the host core being released from reset and ready to boot (core 1(SHARC-FX) on the ADSP-2184x/SC84x processors).

The following table shows the cores that are released from reset. Exiting core n only reset starts with this core n being ready to execute the code from the software vector register (RCU_SVECT2).

Additional information on reset and booting can be found in the *Boot ROM and Booting the Processor* chapter.

Product	Core Released from Reset by RCU
ADSP-2184x	Core 1
ADSP-SC84x	Core 0

The Reset Control Unit (RCU) controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. Programs must guarantee that none of the reset functions puts the system into an undefined state or causes resources to stall. This functionality is important when only one of the cores is reset (programs must ensure that there is no pending system activity involving the core that is being reset). While core resets and software system resets are controlled directly in the RCU, hardware resets can come from the TRU or SEC.

RCU Features

The RCU module supports the following features:

- Hardware reset through the $\overline{\text{SYS_HWRST}}$ pin
- Software system reset through the RCU control (RCU_CTL) register
- Hardware system reset through:
  - TRU module
  - SEC module (System Fault Unit)
- Core reset through RCU Core Reset Output (RCU_CRCTL) register

RCU Functional Description

This section provides information on the function of RCU module.

Software reset using RCU registers

Setting the RCU_STAT.SWRST bit issues a software reset for all system units except the RTC module and RCU_BCODE, RCU_CRCTL and RCU_STAT registers.

Core reset RCU registers

A core can be individually reset by software, or by setting the RCU_CRCTL.CR[n] bit.

ADSP-2184x RCU Register List

The Reset Control Unit (RCU) controls how all the functional units in the processor enter and exit Reset. Differences in functional requirements and clocking constraints exist (units in different clock domains have to enter reset asynchronously, but units exit reset in a deterministic way), and these differences define how reset signals are generated. Reset signals propagate through all functional units asynchronously. For more information on RCU functionality, see the RCU register descriptions.

Table 6-1: ADSP-2184x RCU Register List

Name	Description
RCU_BCODE	Boot Code Register
RCU_CRCTL	Core Reset Outputs Control Register
RCU_CRSTAT	Core Reset Outputs Status Register
RCU_CTL	Control Register
RCU_MSG	Message Register
RCU_MSG_CLR	Message Clear Bits Register
RCU_MSG_SET	Message Set Bits Register
RCU_SIDIS	System Interface Disable Register
RCU_SISTAT	System Interface Status Register
RCU_SRRQSTAT	System Reset Request Status Register
RCU_STAT	Status Register
RCU_SVECT0	Software Vector Register 0
RCU_SVECT1	Software Vector Register 1
RCU_SVECT2	Software Vector Register 2
RCU_SVECT_LCK	SVECT Lock Register

ADSP-2184x RCU Trigger List

Table 6-2: ADSP-2184x RCU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
None			

Table 6-3: ADSP-2184x RCU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
147	RCU0_SYSRST0	RCU0 System Reset Receiver 0	Pulse
148	RCU0_SYSRST1	RCU0 System Reset Receiver 1	Pulse

RCU Definitions

To make the best use of the RCU, it is useful to understand the terms in this section.

The target or source defines the following are types of resets.

Hardware Reset (by target)

All functional units except a small subsection of debug interfaces are set to their default states. State is lost in all non-volatile storage.

System Reset (by target)

All functional units except the RCU, flash interface, and debug are set to their default states.

Core n Only Reset (by target)

Affects Core n only. The system software must guarantee that a bus requester cannot access the core in reset state.

Hardware Reset (by source)

The $\overline{\text{SYS_HWRST}}$ input signal is asserted active (pulled low).

System Reset (by source)

Software can trigger the reset by writing to the RCU_CTL register or by another functional unit such as the TRU or any of the generic reset inputs.

RCU Architectural Concepts

To understand the architecture of the RCU, it is important to consider the reset sources and how differing resets affect the functional units of the processor.

The RCU provides the hardware that controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. For example, units in different clock domains must enter reset asynchronously but exit reset in a deterministic way.

The program must guarantee that none of the reset functions put the system in an undefined state or cause resources to stall. This functionality is important when only one of the cores is reset. The program must guarantee that there is no pending system activity involving Core n before it is reset. For example, there must be no pending transactions to core 0 when the core 0 is reset and vice-versa.

The ***RCU Reset Sources*** table defines how reset sources affect the different functional units.

Table 6-4: RCU Reset Sources

Reset Source	Reset Type	Affected Functional Units
SYSCLK clock domain system reset by the Fault Unit (FMU) or TRU completer	System Reset	All functional units, except: • RTC (if present) • RCU_STAT • RCU_BCODE • the units on the VDDEXT power domain
`RCU_CTL.SYSRST` bit set (software triggered reset)	System Reset	All functional units, except: • RTC (if present) • RCU_STAT • RCU_BCODE • the units on the VDDEXT power domain
`RCU_CRCTL.CR[n]` bit set (software triggered reset)	Core Only Reset	Core n only

RCU Status and Error Signals

The `RCU_STAT` register reflects status and error information. There are three kinds of errors that can occur in the RCU. The reset out error is triggered when `RSTOUT` is both asserted and deasserted at the same time. The lock write error occurs if an attempt is made to write a lock RCU register. The address error occurs if a read-only register is written to or if an attempt is made to a reserved address within the RCU MMR address range.

Core n Reset In

Core n can be requested to stop its IO activity and acknowledge to the requester when the task is completed.

To reset core n:

1. Clear the `RCU_CRSTAT.CR[n]` bit.
2. Disable interrupts to core n.
3. Set (=1) `RCU_CRCTL.CR[n]` to reset core n.

4. Check the RCU_CRSTAT.CR[n] bit. If core n is in reset, continue.

Core n Reset Out

To take core n out of reset:

1. Clear the RCU_CRCTL.CR[n] bit.
2. Check the RCU_CRSTAT.CR[n] bit. If core n is in out of reset, continue.

ADSP-2184x RCU Register Descriptions

Reset Control Unit (RCU) contains the following registers.

Table 6-5: ADSP-2184x RCU Register List

Name	Description
RCU_BCODE	Boot Code Register
RCU_CRCTL	Core Reset Outputs Control Register
RCU_CRSTAT	Core Reset Outputs Status Register
RCU_CTL	Control Register
RCU_MSG	Message Register
RCU_MSG_CLR	Message Clear Bits Register
RCU_MSG_SET	Message Set Bits Register
RCU_SIDIS	System Interface Disable Register
RCU_SISTAT	System Interface Status Register
RCU_SRRQSTAT	System Reset Request Status Register
RCU_STAT	Status Register
RCU_SVECT0	Software Vector Register 0
RCU_SVECT1	Software Vector Register 1
RCU_SVECT2	Software Vector Register 2
RCU_SVECT_LCK	SVECT Lock Register

Boot Code Register

The RCU_BCODE register can be used to determine if and how core boots. This register is set to its default values by RESET.

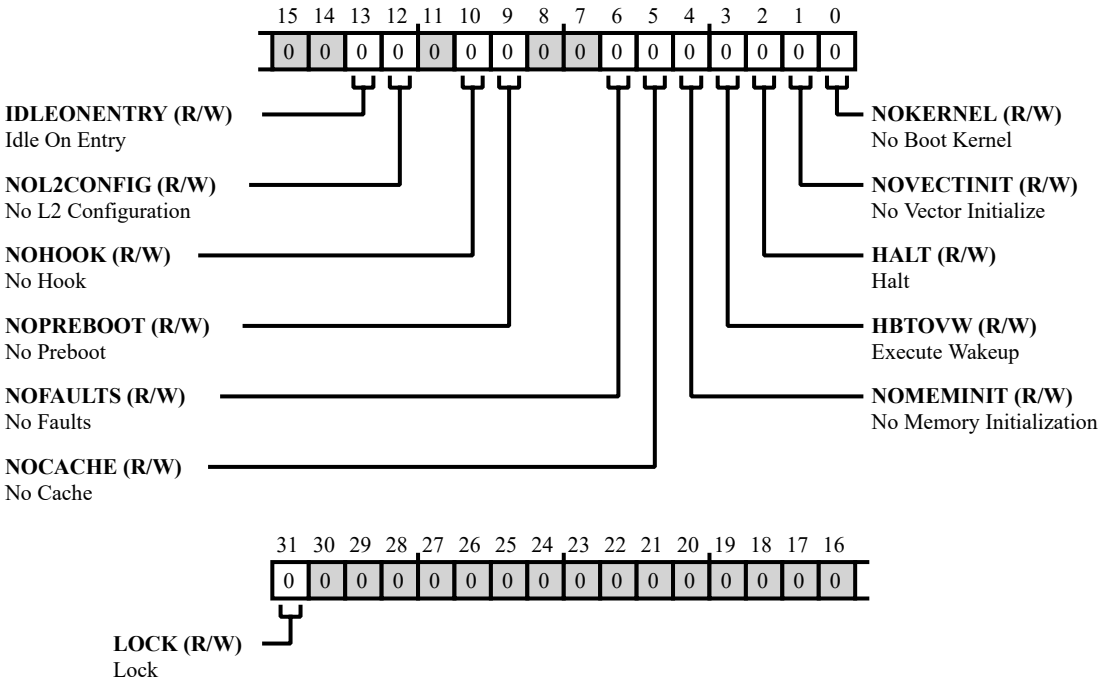


Figure 6-1: RCU_BCODE Register Diagram

Table 6-6: RCU_BCODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_BCODE.LOCK bit is set, the RCU_BCODE register is read only (locked).	
		0	Unlock
		1	Lock
13 (R/W)	IDLEONENTRY	Idle On Entry. The RCU_BCODE.IDLEONENTRY bit configures the RCU to enter the idle state at startup.	
		0	Do not enter idle state
		1	Enter idle state

Table 6-6: RCU_BCODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	NOL2CONFIG	No L2 Configuration. The `RCU_BCODE.NOL2CONFIG` bit configures the RCU to not perform the L2 memory configuration.	
		0	Configure L2 memory
		1	Do not configure L2 memory
10 (R/W)	NOHOOK	No Hook. The `RCU_BCODE.NOHOOK` bit configures the RCU to not perform the hook routine.	
		0	Perform hook routine
		1	Do not perform hook routine
9 (R/W)	NOPREBOOT	No Preboot. The `RCU_BCODE.NOPREBOOT` bit configures the RCU to not perform the customer preboot routine.	
		0	Perform preboot
		1	Do not perform preboot
6 (R/W)	NOFAULTS	No Faults. The `RCU_BCODE.NOFAULTS` bit configures the RCU to not perform fault initialization.	
		0	Perform fault initialization
		1	Do not perform fault initialization
5 (R/W)	NOCACHE	No Cache. The `RCU_BCODE.NOCACHE` bit configures the RCU to not perform a cache initialization and to not enable the cache.	
		0	Enable and initialize cache
		1	Do not initialize or enable cache
4 (R/W)	NOMEMINIT	No Memory Initialization. The `RCU_BCODE.NOMEMINIT` bit configures the RCU to not perform a memory initialization.	
		0	Perform memory initialization
		1	Do not perform memory initialization

Table 6-6: RCU_BCODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	HBTOVW	Execute Wakeup. The `RCU_BCODE.HBTOVW` bit configures the RCU to execute a wakeup.	
		0	Do not wakeup
		1	Execute wakeup
2 (R/W)	HALT	Halt. The `RCU_BCODE.HALT` bit configures the RCU to execute the no boot routine.	
		0	Do not execute routine
		1	Execute routine
1 (R/W)	NOVECTINIT	No Vector Initialize. The `RCU_BCODE.NOVECTINIT` bit configures the RCU to not vector to the application.	
		0	Vector
		1	Do not vector
0 (R/W)	NOKERNEL	No Boot Kernel. The `RCU_BCODE.NOKERNEL` bit configures the RCU to not execute the boot kernel.	
		0	Execute boot kernel
		1	Do not execute boot kernel

Core Reset Outputs Control Register

The RCU core reset control n registers (RCU_CRCTL) include a lock bit (RCU_CRCTL.LOCK) and a core reset bit (RCU_CRCTL.CR[n]) for each core reset signal on the product.

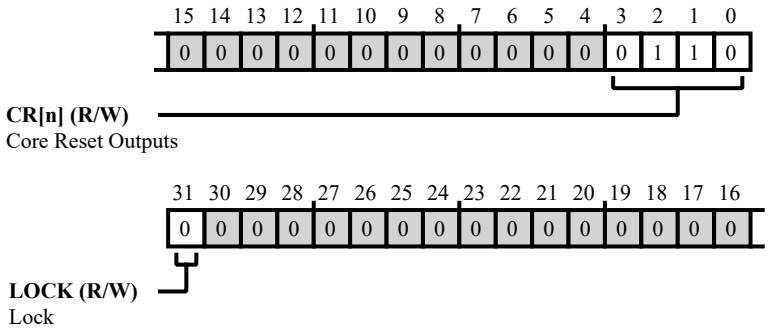


Figure 6-2: RCU_CRCTL Register Diagram

Table 6-7: RCU_CRCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_CRCTL.LOCK bit is set, the RCU_CRCTL register is read only (locked).	
		0	Unlock
		1	Lock
3:0 (R/W)	CR[n]	Core Reset Outputs. The RCU_CRCTL.CR[n] bits control CRES[1:0] core reset signals. The RCU_CRES[n] signals can be individually controlled. They are reset to their default value by a hard reset or a system reset. For each RCU_CRES[n], the selected RCU0_CRMSKi[n] bit is cleared.	
		0	RCU_CRES[3:0] Deasserted
		1	Reserved
		2	RCU_CRES[1] SHARC0 reset control bit asserted
		4	RCU_CRES[2] A55 reset control bit asserted
		8	RCU_CRES[3] DEBUG reset control bit asserted
		15	RCU_CRES[3:0] Asserted

Core Reset Outputs Status Register

The RCU core reset status register (RCU_CRSTAT) contains status bits, indicating which core reset signals have been asserted.

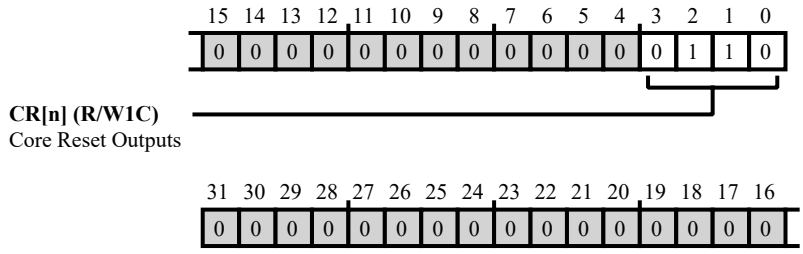


Figure 6-3: RCU_CRSTAT Register Diagram

Table 6-8: RCU_CRSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W1C)	CR[n]	Core Reset Outputs. The RCU_CRSTAT.CR[n] bits indicate which cores have been reset since the last time the bit was cleared. Bits masked by CORE_DISABLE_MASK[15:0] are permanently disabled and the corresponding CR bits set. CR bits are sticky, they need to be cleared by software.	
		0	RCU_CRES[1:0] deasserted. CR[n] corresponds to RCU_CRES[n].
		1	RCU_CRES[0] ARM A5 reset control bit asserted
		2	RCU_CRES[1] SHARC0 reset control bit asserted
		4	RCU_CRES[2] A55 reset control bit asserted
		8	RCU_CRES[3] DEBUG reset control bit asserted
		15	RCU_CRES[3:0] were asserted since the last time bits were cleared. CR[n] corresponds to RCU_CRES[n].

Control Register

The RCU control register (RCU_CTL) provides a register lock, enables for the core and system reset requests inputs and control for the Reset Output pin.

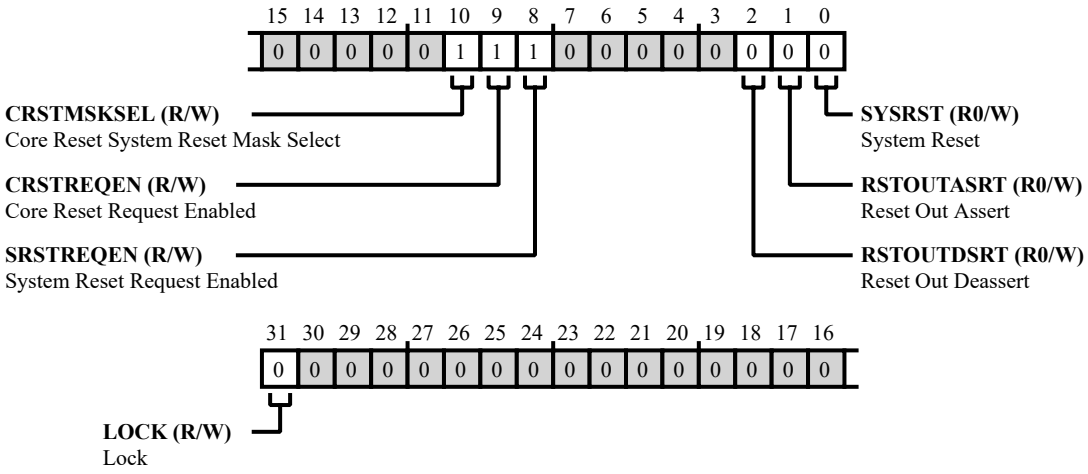


Figure 6-4: RCU_CTL Register Diagram

Table 6-9: RCU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_CTL.LOCK bit is set, the RCU_CTL register is read only (locked). This bit is cleared by a hard reset or any system reset event.	
		0	Unlock
		1	Lock
10 (R/W)	CRSTMSKSEL	Core Reset System Reset Mask Select. The RCU_CTL.CRSTMSKSEL bit selects the core reset system reset mask. This bit is cleared by a hard reset.	
9 (R/W)	CRSTREQEN	Core Reset Request Enabled. The RCU_CTL.CRSTREQEN bit controls whether the SYSCLK domain source(s) of reset is/are enabled to reset the core(s) when asserted. This bit is cleared by hard reset or any system reset event.	
		0	Disabled
		1	Enabled

Table 6-9: RCU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SRSTREQEN	System Reset Request Enabled. The RCU_CTL.SRSTREQEN bit controls whether the SYSCLK domain sources of reset are enabled to do a system reset when asserted. This bit is cleared by a hard reset.	
		0	Disabled
		1	Enabled
2 (R0/W)	RSTOUTDSRT	Reset Out Deassert. The RCU_CTL.RSTOUTDSRT bit controls the deassertion of the system reset pin.	
		0	No Action
		1	Deassert RSTOUT
1 (R0/W)	RSTOUTASRT	Reset Out Assert. The RCU_CTL.RSTOUTASRT bit controls assertion of the system reset pin.	
		0	No Action
		1	Assert RSTOUT
0 (R0/W)	SYSRST	System Reset. The RCU_CTL.SYSRST bit provides reset for all system units.	
		0	No Action
		1	System Reset

Message Register

The RCU_MSG is a general-purpose register. It is intended to provide flexibility for Boot ROM code and to pass predefined variables to the debugger. Please see the Booting chapter for product-specific details.

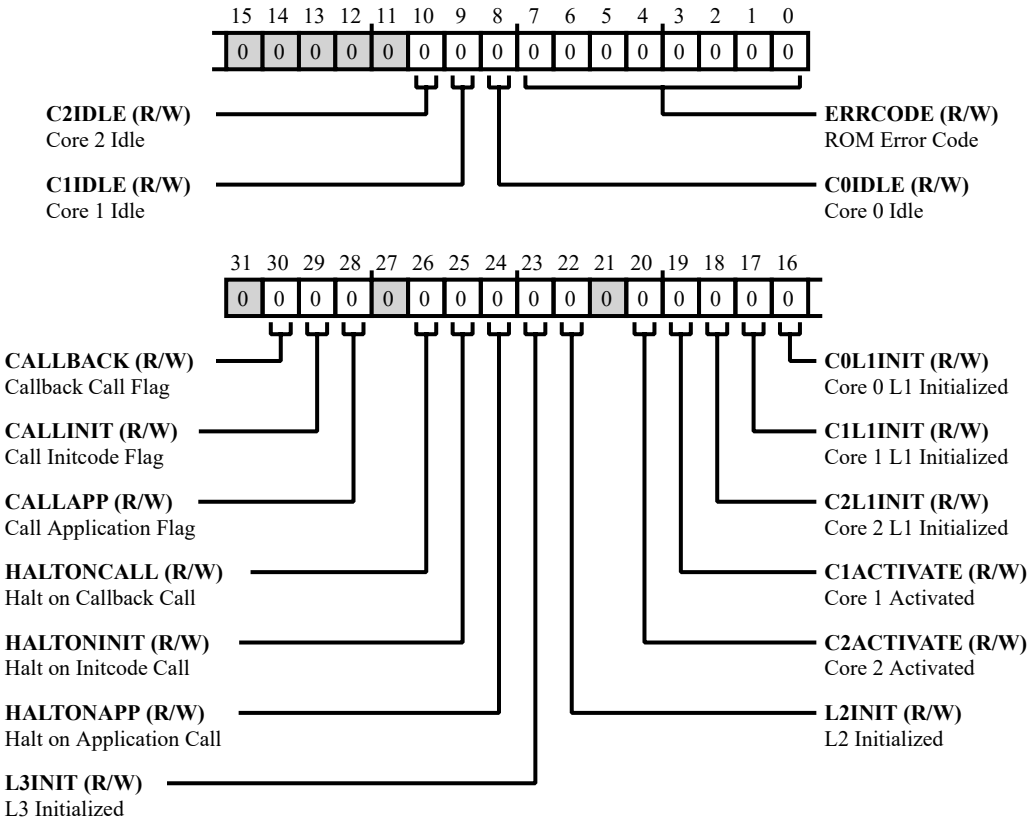


Figure 6-5: RCU_MSG Register Diagram

Table 6-10: RCU_MSG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	CALLBACK	Callback Call Flag. The RCU_MSG.CALLBACK bit indicates that a flag has been set by the boot code prior to a callback call.	
		0	Flag not set
		1	Flag set

Table 6-10: RCU_MSG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	CALLINIT	Call Initcode Flag. The `RCU_MSG.CALLINIT` bit indicates that a flag has been set by the boot code prior to an initcode call.	
		0	Flag not set
		1	Flag set
28 (R/W)	CALLAPP	Call Application Flag. The `RCU_MSG.CALLAPP` bit indicates that a flag has been set by the boot code prior to an application call.	
		0	Flag not set
		1	Flag set
26 (R/W)	HALTONCALL	Halt on Callback Call. The `RCU_MSG.HALTONCALL` bit generates an emulation exception prior to a callback call.	
		0	Do not generate exception
		1	Generate exception
25 (R/W)	HALTONINIT	Halt on Initcode Call. The `RCU_MSG.HALTONINIT` bit generates an emulation exception prior to an initcode call.	
		0	Do not generate exception
		1	Generate exception
24 (R/W)	HALTONAPP	Halt on Application Call. The `RCU_MSG.HALTONAPP` bit generates an emulation exception prior to an application call.	
		0	Do not generate exception
		1	Generate exception
23 (R/W)	L3INIT	L3 Initialized. The `RCU_MSG.L3INIT` bit indicates that the L3 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized
22 (R/W)	L2INIT	L2 Initialized. The `RCU_MSG.L2INIT` bit indicates that the L2 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized

Table 6-10: RCU_MSG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	C2ACTIVATE	Core 2 Activated. The RCU_MSG.C2ACTIVATE bit is used by tools for activation of Core 2.	
19 (R/W)	C1ACTIVATE	Core 1 Activated. The RCU_MSG.C1ACTIVATE bit is used by tools for activation of Core 1.	
18 (R/W)	C2L1INIT	Core 2 L1 Initialized. The RCU_MSG.C2L1INIT bit indicates that the core 2 L1 resource is initialized.	
17 (R/W)	C1L1INIT	Core 1 L1 Initialized. The RCU_MSG.C1L1INIT bit indicates that the core 1 L1 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized
16 (R/W)	C0L1INIT	Core 0 L1 Initialized. The RCU_MSG.C0L1INIT bit indicates that the core 0 L1 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized
10 (R/W)	C2IDLE	Core 2 Idle. The RCU_MSG.C2IDLE bit indicates that core 2 is in a safe idle state in ROM.	
9 (R/W)	C1IDLE	Core 1 Idle. The RCU_MSG.C1IDLE bit indicates that core 1 is in a safe idle state in ROM.	
8 (R/W)	C0IDLE	Core 0 Idle. The RCU_MSG.C0IDLE bit indicates that core 0 is in a safe idle state in ROM.	
7:0 (R/W)	ERRCODE	ROM Error Code. The RCU_MSG.ERRCODE bit indicates the error code of the ROM. It is valid only when in the error handler.	

Message Clear Bits Register

The RCU_MSG_CLR register is used to clear bits in RCU_MSG register. Reading this register returns 0x00000000.

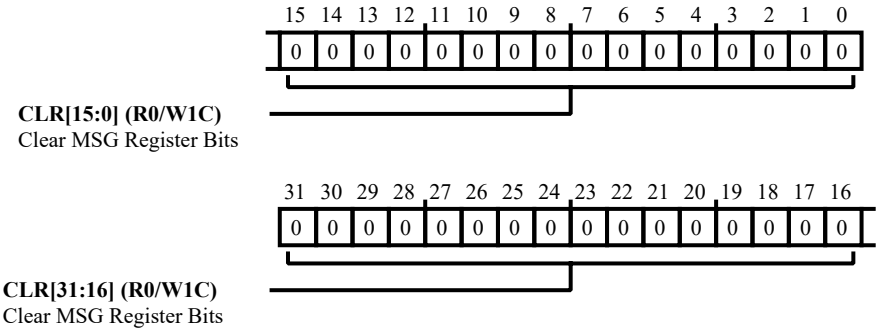


Figure 6-6: RCU_MSG_CLR Register Diagram

Table 6-11: RCU_MSG_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W1C)	CLR	Clear MSG Register Bits. The RCU_MSG_CLR.CLR bit resets MSG bit n.

Message Set Bits Register

The RCU_MSG_SET register is used to set bits in RCU_MSG register. Reading this register returns 0x00000000.

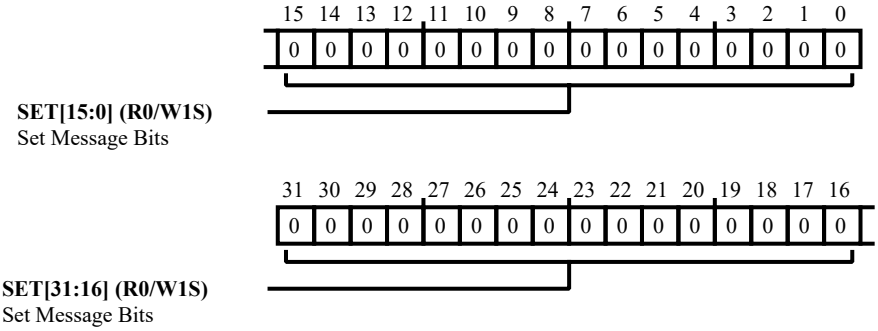


Figure 6-7: RCU_MSG_SET Register Diagram

Table 6-12: RCU_MSG_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W1S)	SET	Set Message Bits. The RCU_MSG_SET.SET bit sets MSG bit n.

System Interface Disable Register

The RCU system interface disable register (RCU_SIDIS) lets the RCU assert a system interface disable request to functional units in the processor. This register is set to its default values by a hard reset or any system reset event. For information on mapping between RCU_SIDIS bits and functional units, see the RCU functional description.

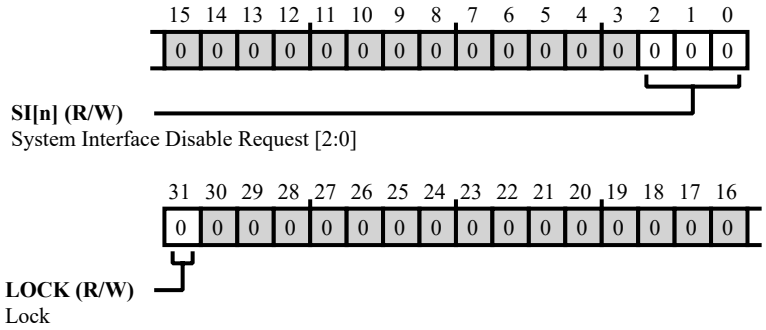


Figure 6-8: RCU_SIDIS Register Diagram

Table 6-13: RCU_SIDIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_SIDIS.LOCK bit is set, the RCU_SIDIS register is read only (locked).	
		0	Unlock
		1	Lock
2:0 (R/W)	SI[n]	System Interface Disable Request [2:0]. Each RCU_SIDIS.SI[n] bit corresponds to a functional unit in the processor that supports the system interface disable request-acknowledge protocol.	
		0	RCU_SI_DISABLE_REQ[2:0] deasserted
		1	RCU_SI_DISABLE_REQ[0] system interface disable request to SHARC0 asserted
		2	RCU_SI_DISABLE_REQ[1] system interface disable request to A55 asserted
		4	Reserved
		7	RCU_SI_DISABLE_REQ[2:0] asserted

System Interface Status Register

The RCU system interface status register (RCU_SISTAT) indicates whether a functional unit has or has not acknowledged an RCU unit disable request.

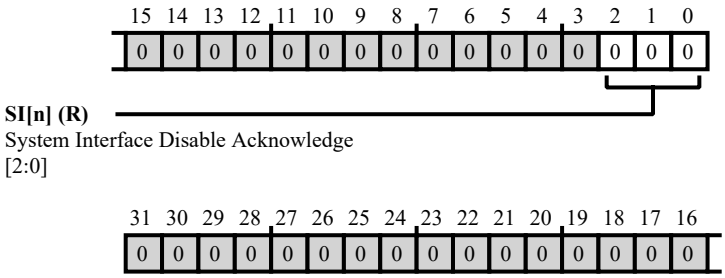


Figure 6-9: RCU_SISTAT Register Diagram

Table 6-14: RCU_SISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/NW)	SI[n]	System Interface Disable Acknowledge [2:0]. The RCU_SISTAT.SI[n] bit indicates whether a functional unit has or has not acknowledged an RCU unit disable request.	
		0	No Acknowledge
		1	RCU_SI_DISABLE_ACK[0] system interface disable acknowledge from SHARC0 asserted
		2	RCU_SI_DISABLE_ACK[1] system interface disable acknowledge from A55 asserted
		4	Reserved
		7	SI_DISABLE_ACK[2:0] asserted

System Reset Request Status Register

The RCU system reset request status register (RCU_SRRQSTAT) contains status bits, indicating which system reset request input triggered a system reset. This register is set to its default values by a hard reset.

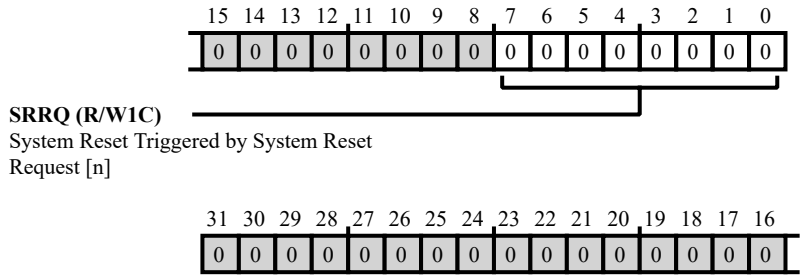


Figure 6-10: RCU_SRRQSTAT Register Diagram

Table 6-15: RCU_SRRQSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W1C)	SRRQ	System Reset Triggered by System Reset Request [n]. The RCU_SRRQSTAT.SRRQ bits are set by the assertion of the corresponding system reset request input and deasserted by writing "1" to the bit. The RCU_SRRQSTAT register is cleared by a hard reset.	
		1	RCU_SRRQ[0], System Reset Request from SEC asserted
		2	RCU_SRRQ[1], System Reset Request from TRGS_RCU0_SYSRST0 asserted
		4	RCU_SRRQ[2], System Reset Request from TRGS_RCU0_SYSRST1 asserted
		8	RCU_SRRQ[3], System Reset Request from CTI3 asserted

Status Register

The RCU status register (RCU_STAT) contains status bits for all RCU reset sources, reset status, and boot mode inputs. Status bits for reset sources are sticky and can cleared by software. Error status bits are cleared by any reset event.

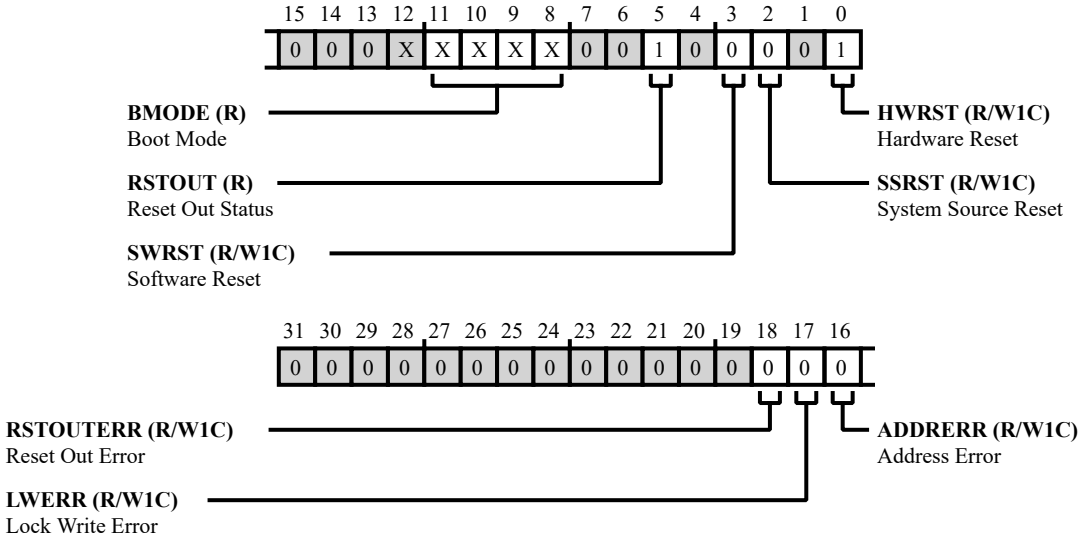


Figure 6-11: RCU_STAT Register Diagram

Table 6-16: RCU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W1C)	RSTOUTERR	Reset Out Error. The RCU_STAT.RSTOUTERR bit indicates (if set) that a write attempted to set the RCU_CTL.RSTOUTASRT and RCU_CTL.RSTOUTDSRT simultaneously. This condition triggers a bus error.	
		0	No Error
		1	Error Occurred
17 (R/W1C)	LWERR	Lock Write Error. The RCU_STAT.LWERR bit indicates (when set) there was an attempted write to an RCU register while the RCU_CTL.LOCK bit was set and the global lock bit is enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear	
		0	No Error
		1	Error Occurred

Table 6-16: RCU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	ADDRERR	Address Error. The RCU_STAT.ADDRERR bit indicates that the RCU generated an address error. This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
11:8 (R/NW)	BMODE	Boot Mode. The RCU_STAT.BMODE bits indicate the input on the boot mode pins.	
5 (R/NW)	RSTOUT	Reset Out Status. The RCU_STAT.RSTOUT bit indicates the assertion status of the system reset pin.	
		0	RSTOUT Deasserted
		1	RSTOUT Asserted
3 (R/W1C)	SWRST	Software Reset. The RCU_STAT.SWRST bit indicates that a system reset (which was triggered by software) has occurred since the last time a hardware reset occurred or since the RCU_STAT.SWRST bit was cleared by software.	
		0	Inactive
		1	Reset Occurred
2 (R/W1C)	SSRST	System Source Reset. The RCU_STAT.SSRST bit indicates that a system reset triggered by hardware in the system clock domain, clock A domain, or clock B domain has occurred since the last time a hardware reset occurred or since the RCU_STAT.SSRST bit was cleared by software.	
		0	Inactive
		1	Reset Occurred
0 (R/W1C)	HWRST	Hardware Reset. The RCU_STAT.HWRST bit indicates that a hardware reset has occurred.	
		0	Inactive
		1	Reset Occurred

Software Vector Register 0

The RCU_SVECT0 register contains the default location of the first instruction to execute after a reset.

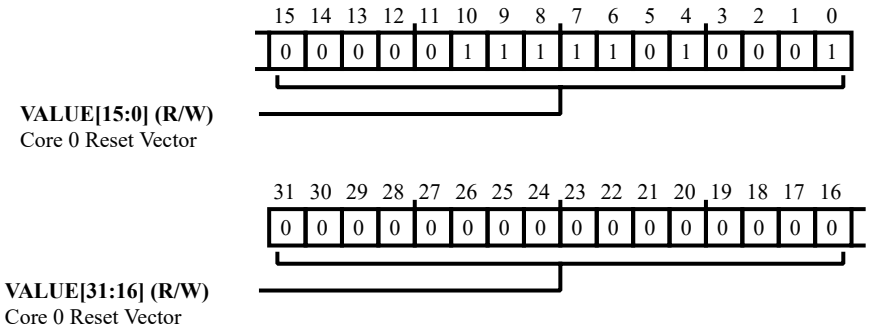


Figure 6-12: RCU_SVECT0 Register Diagram

Table 6-17: RCU_SVECT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Core 0 Reset Vector. Core0 jumps to the address in the RCU_SVECT0.VALUE bit field after reset.

Software Vector Register 1

The RCU_SVECT1 register contains the default location of the first instruction to execute after a reset.

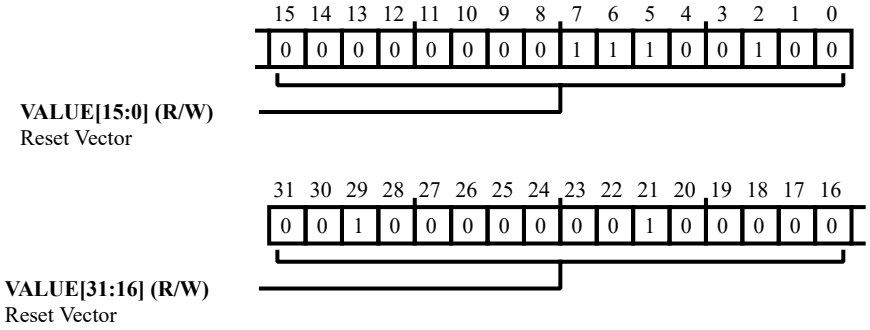


Figure 6-13: RCU_SVECT1 Register Diagram

Table 6-18: RCU_SVECT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Reset Vector. Core1 jumps to the address in the RCU_SVECT1.VALUE bit field after reset.

Software Vector Register 2

The RCU_SVECT2 register contains the default location of the first instruction to execute after a reset.

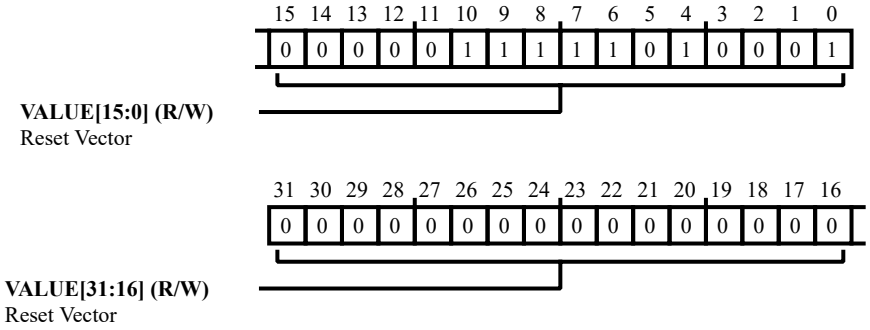


Figure 6-14: RCU_SVECT2 Register Diagram

Table 6-19: RCU_SVECT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Reset Vector. Core2 jumps to the address in the RCU_SVECT2.VALUE bit field after reset.

SVECT Lock Register

The RCU software vector lock register (RCU_SVECT_LCK) provides a register lock and software vector n enable bits for each processor core on the product. This register is set to its default values by a hard reset or any system reset event.

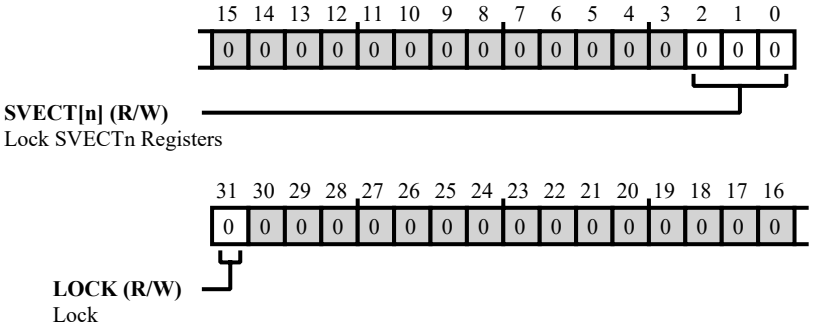


Figure 6-15: RCU_SVECT_LCK Register Diagram

Table 6-20: RCU_SVECT_LCK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_SVECT_LCK.LOCK bit is set, the RCU_SVECT_LCK register is read only (locked).	
		0	Unlock
		1	Lock
2:0 (R/W)	SVECT[n]	Lock SVECTn Registers. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_SVECT_LCK.SVECT[n] bit is set, the SVECT registers are read only (locked).	

7 System Event Controller (SEC) and Generic Interrupt Controller (GIC)

System event management is the responsibility of the system event controller (SEC). The SEC manages the configuration of all system event sources. The SEC also manages the propagation of system events to all connected cores and the system fault interface.

All the peripheral interrupts are routed using a single SEC interrupt to the desired core. The SEC allows programmability of the peripheral interrupt's priority, supporting up to 256 priority levels that are arbitrated within the SEC itself. The SEC also allows these interrupts to be grouped and masked by priority level and provides the flexibility to choose which core(s) the interrupt is routed to.

The SEC also supports self-nesting of interrupts, which is required when sharing a single interrupt request to an individual core, as this allows for a higher-priority peripheral interrupt to be passed to the core while it is currently servicing a lower-priority peripheral interrupt. For more information, refer to "Self-Nesting Mode for System Event Controller Interrupt (SECI)" in the *SHARC+ Core Programming Reference*.

For more information about the Arm GIC, visit the Arm Information Center.

SEC Features

The following list describes the system event controller features.

- Comprehensive system event source management including interrupt enable, fault enable, priority, core mapping, and source grouping.
- Fault management including fault action configuration, timeout, external indication, and system reset.
- Determinism where all system events have the same propagation delay and provide unique identification of a specific system event source.
- Distributed programming model where each system event source control and all status fields are independent of all others.
- Completer control port which provides access to all SEC registers for configuration, status, and interrupt or fault service model.

- Global locking supports a register level protection model to prevent writes to “locked” registers.

SEC Functional Description

The following sections provide a functional description of the SEC.

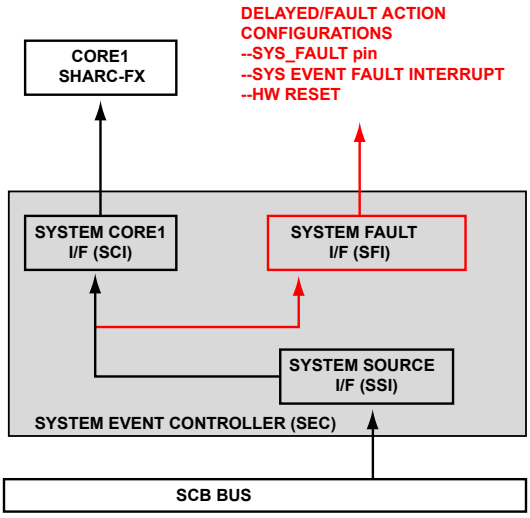


Figure 7-1: SEC Interrupt Signal Flow

ADSP-2184x SEC Register List

The System Event Controller (SEC) manages the system fault sources, including control features such as enable/disable and active/pending source status. For more information on SEC functionality, see the SEC register descriptions.

Table 7-1: ADSP-2184x SEC Register List

Name	Description
SEC_CACT[n]	SCI Active Register n
SEC_CCTL[n]	SCI Control Register n
SEC_CGMSK[n]	SCI Group Mask Register n
SEC_CPLVL[n]	SCI Priority Level Register n
SEC_CPMSK[n]	SCI Priority Mask Register n
SEC_CPND[n]	Core Pending Register n
SEC_CSID[n]	SCI Source ID Register n
SEC_CSTAT[n]	SCI Status Register n
SEC_END	Global End Register
SEC_FCOPP	Fault COP Period Register
SEC_FCOPP_CUR	Fault COP Period Current Register

Table 7-1: ADSP-2184x SEC Register List (Continued)

Name	Description
SEC_FCTL	Fault Control Register
SEC_FDLY	Fault Delay Register
SEC_FDLY_CUR	Fault Delay Current Register
SEC_FEND	Fault End Register
SEC_FSID	Fault Source ID Register
SEC_FSRDLY	Fault System Reset Delay Register
SEC_FSRDLY_CUR	Fault System Reset Delay Current Register
SEC_FSTAT	Fault Status Register
SEC_GCTL	Global Control Register
SEC_GSTAT	Global Status Register
SEC_RAISE	Global Raise Register
SEC_SCTL[n]	Source Control Register n
SEC_SSTAT[n]	Source Status Register n

ADSP-2184x SEC Interrupt List

Table 7-2: ADSP-2184x SEC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
0	SEC0_ERR	SEC0 Error	Level	

ADSP-2184x SEC Trigger List

Table 7-3: ADSP-2184x SEC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
116	SEC0_FAULT	SEC0 Fault	Edge

Table 7-4: ADSP-2184x SEC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

ADSP-2184x/ADSP-SC84x Interrupt List

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
0	SEC0_ERR	SEC0 Error	Level	
1	CANFD0_WU	CANFD0 Wakeup Interrupt	None	
2	CANFD0_IRQ	CANFD0 CAN Interrupt	None	
3	CANFD0_MSG	CANFD0 Message Receive/Transmit Interrupt	None	
4	CANFD1_WU	CANFD1 Wakeup Interrupt	None	
5	CANFD1_IRQ	CANFD1 CAN Interrupt	None	
6	CANFD1_MSG	CANFD1 Message Receive/Transmit Interrupt	None	
7	CGU0_EVT	CGU0 Event	Edge	
8	CGU1_EVT	CGU1 Event	Edge	
9	CNT0_STAT	CNT0 Status	Level	
10	C1_DATA_READ_ERR	Core 1 Data Read Interrupt		
11	C1_DATA_WRITE_ERR	Core 1 Data Write Interrupt		
12	C1_INST_READ_ERR	Core 1 Instruction Read Interrupt		
13	C0_PMUIRQ	Performance Monitoring Unit Interrupt		
14	LPDDR_LPDDR_DFI_ERR_2	LPDDR LPDDR4 dfi0 error internal	None	
15	LPDDR_LPDDR_SCRUB_INT	LPDDR LPDDR4 scrub interrupt	None	
16	GIC_PMU_INT	GIC Performance monitor interrupt		
21	FX_PFATAL_ERR	FX Fatal Error Interrupt		
32	C1_IDMA_READ_ERR	Core1 iDMA Read Interrupt	None	
33	C1_IDMA_WRITE_ERR	Core 1 iDMA Write Interrupt	None	
34	CRC0_ERR	CRC0 Error	Level	
35	CRC1_ERR	CRC1 Error	Level	
36	MDMA0_SRC	Memory DMA Stream 0 Source Channel		8
37	MDMA0_DST	Memory DMA Stream 0 Destination Channel		9
38	MDMA1_SRC	Memory DMA Stream 1 Source Channel		18
39	MDMA1_DST	Memory DMA Stream 1 Destination Channel		19
40	CRC0_DCNTEXP	CRC0 Datacount Expiration	Level	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
41	CRC1_DCNTEXP	CRC1 Datacount Expiration	Level	
42	CRC2_ERR	CRC2 Error	Level	
43	CRC3_ERR	CRC3 Error	Level	
44	MDMA4_SRC	Memory DMA Stream 0 Source Channel		45
45	MDMA4_DST	Memory DMA Stream 0 Destination Channel		46
46	MDMA5_SRC	Memory DMA Stream 1 Source Channel		47
47	MDMA5_DST	Memory DMA Stream 1 Destination Channel		48
48	CRC2_DCNTEXP	CRC2 Datacount Expiration	Level	
49	CRC3_DCNTEXP	CRC3 Datacount Expiration	Level	
50	MDMA0_SRC_ERR	Enhanced BW MDMA0 Channel 0 (Source) Error		
51	MDMA0_DST_ERR	Enhanced BW MDMA0 Channel 1 (Destination) Error		
52	MDMA1_SRC_ERR	Standard BW MDMA1 Channel 0 (Source) Error		
53	MDMA1_DST_ERR	Standard BW MDMA1 Channel 1 (Destination) Error		
54	MDMA4_SRC_ERR	Standard BW MDMA4 Channel 0 (Source) Error		
55	MDMA4_DST_ERR	Standard BW MDMA4 Channel 1 (Destination) Error		
56	MDMA5_SRC_ERR	Standard BW MDMA5 Channel 0 (Source) Error		
57	MDMA5_DST_ERR	Standard BW MDMA5 Channel 1 (Destination) Error		
58	CSETR1_BUFINTR	CSETR1 CoreSight Trace ETR		
59	CTI1_EVT0	CTI1 Core 1 CTI Event	Level	
64	DAI0_IRQH	DAI0 High Priority Interrupt	None	
65	DAI1_IRQH	DAI1 High Priority Interrupt	None	
66	DAI0_IRQL	DAI0 Low Priority Interrupt	None	
67	DAI1_IRQL	DAI1 Low Priority Interrupt	None	
68	EMAC0_DMA2_RX	EMAC0 DMAn Rx Channel Interrupt	None	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
70	EMAC0_DMA3_TX	EMAC0 DMAn Tx Channel Interrupt	None	
71	EMAC0_DMA4_TX	EMAC0 DMAn Tx Channel Interrupt	None	
72	EMAC0_DMA5_TX	EMAC0 DMAn Tx Channel Interrupt	None	
73	EMAC0_DMA6_TX	EMAC0 DMAn Tx Channel Interrupt	None	
74	EMAC0_DMA7_TX	EMAC0 DMAn Tx Channel Interrupt	None	
75	EMAC0_DMA0_RX	EMAC0 DMAn Rx Channel Interrupt	None	
82	EMAC0_DMA1_RX	EMAC0 DMAn Rx Channel Interrupt	None	
84	EMAC0_DMA3_RX	EMAC0 DMAn Rx Channel Interrupt	None	
86	EMAC0_DMA4_RX	EMAC0 DMAn Rx Channel Interrupt	None	
87	EMAC0_DMA5_RX	EMAC0 DMAn Rx Channel Interrupt	None	
88	EMAC0_DMA6_RX	EMAC0 DMAn Rx Channel Interrupt	None	
89	EMAC0_DMA7_RX	EMAC0 DMAn Rx Channel Interrupt	None	
94	EMAC0_STAT	EMAC0 Status	None	
95	EMAC0_PWR	EMAC0 Power Interrupt	None	
96	EMAC0_DMA0_TX	EMAC0 DMAn Tx Channel Interrupt	None	
97	EMAC0_DMA1_TX	EMAC0 DMAn Tx Channel Interrupt	None	
98	EMAC0_DMA2_TX	EMAC0 DMAn Tx Channel Interrupt	None	
99	EMAC0_MAC	EMAC0 MAC Interrupt	None	
106	EMDMA0_DONE	EMDMA0 Transfer Done	Edge	
107	EMDMA1_DONE	EMDMA1 Transfer Done	Edge	
108	FIR0_DMA	FIR0 Core 1 DMA	Edge	
109	FIR0_STAT	FIR0 Core 1 Status	Edge	
110	FIR1_DMA	FIR1 Core 1 DMA	Edge	
111	FIR1_STAT	FIR1 Core 1 Status	Edge	
112	FIR0_BUS_ERR	FIR0 Core 1 FIR Bus Error	Level	
113	FIR1_BUS_ERR	FIR1 Core 1 FIR Bus Error	Level	
114	FRACNPLL0_FRAC_LOCK	FRACNPLL0 Frac Pll Lock Interrupt	Edge	
115	FRACNPLL1_FRAC_LOCK	FRACNPLL1 Frac Pll Lock Interrupt	Edge	
116	FRACNPLL0_FRAC_UN-LOCK	FRACNPLL0 Frac Pll Unlock Interrupt	Edge	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
117	FRACNPLL1_FRAC_UN-LOCK	FRACNPLL1 Frac Pll Unlock Interrupt	Edge	
118	HADC0_EVT	HADC0 Event	Edge	
119	HSM_HSM_INTR_0	HSM Interrupt 0	None	
120	HSM_HSM_INTR_1	HSM Interrupt 1	None	
121	HSM_HSM_INTR_2	HSM Interrupt 2	None	
122	HSM_HSM_INTR_3	HSM Interrupt 3	None	
123	HSM_HSM_INTR_4	HSM Interrupt 4	None	
124	HSM_HSM_INTR_5	HSM Interrupt 5	None	
125	HSM_HSM_INTR_6	HSM Interrupt 6	None	
126	HSM_HSM_INTR_7	HSM Interrupt 7	None	
127	HSM_HSM_MISC_INTR_0	HSM Halt State Indication Interrupt	None	
128	HSM_HSM_MISC_INTR_1	HSM Two Bit Error Indication Interrupt from HSM OTP	None	
129	HSM_HSM_MISC_INTR_2	HSM Misc Interrupt 2	None	
130	HSM_HSM_MISC_INTR_3	HSM Misc Interrupt 3	None	
131	IIR0_DMA	IIR0 Core 1 DMA	Edge	
132	IIR0_STAT	IIR0 Core 1 Status	Edge	
133	IIR1_DMA	IIR1 Core 1 DMA	Edge	
134	IIR1_STAT	IIR1 Core 1 Status	Edge	
135	IIR2_DMA	IIR2 Core 1 DMA	Edge	
136	IIR2_STAT	IIR2 Core 1 Status	Edge	
137	IIR3_DMA	IIR3 Core 1 DMA	Edge	
138	IIR3_STAT	IIR3 Core 1 Status	Edge	
139	IIR0_BUS_ERR	IIR0 Core 1 IIR Bus Error	Level	
140	IIR1_BUS_ERR	IIR1 Core 1 IIR Bus Error	Level	
141	IIR2_BUS_ERR	IIR2 Core 1 IIR Bus Error	Level	
142	IIR3_BUS_ERR	IIR3 Core 1 IIR Bus Error	Level	
143	L2CTL0_ECC_ERR	L2CTL0 ECC Error	Level	
145	L2CTL0_EVT	L2CTL0 Scrub/Initialization Done	Level	
146	L2CTL1_ECC_ERR	L2CTL1 ECC Error	Level	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
148	L2CTL1_EVT	L2CTL1 Scrub/Initialization Done	Level	
152	LP0_DMA	LP0 DMA Channel		30
153	LP0_STAT	LP0 Status		
154	LP1_DMA	LP1 DMA Channel		36
155	LP1_STAT	LP1 Status		
156	LP0_DMA_ERR	LP0 DMA Data Error		
157	LP1_DMA_ERR	LP1 DMA Data Error		
158	LPDDR_LPDDR_ECC_CERR	LPDDR LPDDR4 ecc corrected err intr	None	
159	LPDDR_LPDDR_ECC_CFLT	LPDDR LPDDR4 ecc corrected err intr fault	None	
160	LPDDR_LPDDR_ECC_UCERR	LPDDR LPDDR4 ecc uncorrected err intr	None	
161	LPDDR_LPDDR_ECC_UCFLT	LPDDR LPDDR4 ecc uncorrected err intr fault	None	
163	LPDDR_LPDDR_DRT_TMP_LMT_INTR	LPDDR LPDDR4 derate temp limit intr	None	
164	LPDDR_LPDDR_DRT_TMP_LMT_FLT	LPDDR LPDDR4 derate temp limit intr fault	None	
165	LPDDR_LPDDR_DFI_ERR	LPDDR LPDDR4 dfi0 error PHY	None	
166	LPDDR_LPDDR_DDRPHY_INT_N	LPDDR LPDDR4 dwc ddrphy int	None	
167	MDMA2_SRC	Enhanced BW MDMA2 Channel 0 (Source)		39
168	MDMA2_DST	Enhanced BW MDMA2 Channel 1 (Destination)		40
169	MDMA3_SRC	Max BW MDMA3 Channel 0 (Source)		43
170	MDMA3_DST	Max BW MDMA3 Channel 1 (Destination)		44
171	MDMA7_SRC	Max BW MDMA7 Channel 0 (Source)		51
172	MDMA7_DST	Max BW MDMA7 Channel 1 (Destination)		52
173	MDMA6_SRC	Enhanced BW MDMA6 Channel 0 (Source)		49
174	MDMA6_DST	Enhanced BW MDMA6 Channel 1 (Destination)		50

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
175	MDMA2_SRC_ERR	Standard BW MDMA2 Channel 0 (Source) Error		
176	MDMA2_DST_ERR	Standard BW MDMA2 Channel 1 (Destination) Error		
177	MDMA3_SRC_ERR	Standard BW MDMA3 Channel 0 (Source) Error		
178	MDMA3_DST_ERR	Standard BW MDMA3 Channel 1 (Destination) Error		
179	MDMA6_SRC_ERR	Standard BW MDMA6 Channel 0 (Source) Error		
180	MDMA6_DST_ERR	Standard BW MDMA6 Channel 1 (Destination) Error		
181	MDMA7_SRC_ERR	Standard BW MDMA7 Channel 0 (Source) Error		
182	MDMA7_DST_ERR	Standard BW MDMA7 Channel 1 (Destination) Error		
183	MEC1_EEIRQ0	MEC1 ECC Error Interrupt Request	Level	
184	MEC1_EWIRQ0	MEC1 ECC Warning to Interrupt Request (Reserved)	Level	
185	MEC1_MEIRQ1	MEC1 Parity Error Interrupt Request	Level	
186	MEC1_MFIRQ00	MEC1 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
187	MEC1_MEIRQ0	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level	
188	MEC1_MFIRQ20	MEC1 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
189	MEC1_MEIRQ2	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level	
190	MEC1_PEIRQ0	MEC1 Parity Error Interrupt/Trigger Request Bus	Level	
191	MEC2_EEIRQ0	MEC2 ECC Error Interrupt/Trigger Request Bus	Level	
192	MEC2_EWIRQ0	MEC2 ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
193	MEC2_MWIRQ1	MEC2 Core Memory ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
194	MEC2_MEIRQ1	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level	
195	MEC2_MFIRQ00	MEC2 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
196	MEC2_MEIRQ0	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level	
197	MEC2_MFIRQ20	MEC2 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
198	MEC2_MEIRQ2	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level	
199	MEC2_PEIRQ0	MEC2 Parity Error Interrupt/Trigger Request Bus	Level	
200	MEC1_MWIRQ1	MEC1 MEC1 Core 1 Mem Warning to Core 1 (Reserved)	Level	
201	MEC0_EEIRQ0	MEC0 ECC Error Interrupt/Trigger Request Bus	Level	
202	MEC0_EWIRQ0	MEC0 ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
203	MEC0_MWIRQ1	MEC0 Core Memory ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
204	MEC0_MEIRQ1	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level	
205	MEC0_MFIRQ00	MEC0 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
206	MEC0_MEIRQ0	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level	
207	MEC0_MFIRQ20	MEC0 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
208	MEC0_MEIRQ2	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level	
209	MEC0_PEIRQ0	MEC0 Parity Error Interrupt/Trigger Request Bus	Level	
210	MREPAIR0_MEMREP	MREPAIR0 Memory Repair and BISR Register Clear Done Interrupt (Reserved)	None	
211	MLB0_INT0	MLB0 Interrupt 0		
212	MLB0_INT1	MLB0 Interrupt 1		

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
213	MLB0_STAT	MLB0 Status		
214	EMSI0_STAT	EMSI0 Mshc Interrupt	None	
215	EMSI0_WAKEUP	EMSI0 MSHC wake interrupt	None	
216	OTPC0_ERR	OTPC0 Dual-bit Error	Level	
217	PINT0_BLOCK	PINT0 Pin Interrupt Block 0	Level	
218	PINT1_BLOCK	PINT1 Pin Interrupt Block 1	Level	
219	PINT2_BLOCK	PINT2 Pin Interrupt Block 2	Level	
220	PINT3_BLOCK	PINT3 Pin Interrupt Block 3	Level	
221	PINT4_BLOCK	PINT4 Pin Interrupt Block 4	Level	
222	PINT5_BLOCK	PINT5 Pin Interrupt Block 5	Level	
223	PINT6_BLOCK	PINT6 Pin Interrupt Block 6	Level	
224	PINT7_BLOCK	PINT7 Pin Interrupt Block 7	Level	
225	PKIC0_IRQ	PKIC0 Interrupt	Level	
226	PKTE0_IRQ	PKTE0 Interrupt	Level	
227	PWM0_TRIP	PWM0 Trip	Level	
228	PWM0_SYNC	PWM0 PWMTMR Grouped	Edge	
229	SMPU_AGGR_INT	SMPU Aggregated Interrupt/Event		
230	SPI0_TXDMA	SPI0 TX DMA Channel	Level	22
231	SPI0_RXDMA	SPI0 RX DMA Channel	Level	23
232	SPI0_STAT	SPI0 Status	Level	
233	SPI0_ERR	SPI0 Error	Level	
234	SPI1_TXDMA	SPI1 TX DMA Channel	Level	24
235	SPI1_RXDMA	SPI1 RX DMA Channel	Level	25
236	SPI1_STAT	SPI1 Status	Level	
237	SPI1_ERR	SPI1 Error	Level	
238	SPI2_TXDMA	SPI2 TX DMA Channel	Level	26
239	SPI2_RXDMA	SPI2 RX DMA Channel	Level	27
240	SPI2_STAT	SPI2 Status	Level	
241	SPI2_ERR	SPI2 Error	Level	
242	SPI5_TXDMA	SPI5 TX DMA Channel	Level	59
243	SPI5_RXDMA	SPI5 RX DMA Channel	Level	60

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
244	SPI5_STAT	SPI5 Status	Level	
245	SPI5_ERR	SPI5 Error	Level	
246	SPI0_TXDMA_ERR	SPI0 TX DMA Channel Error	Level	
247	SPI0_RXDMA_ERR	SPI0 RX DMA Channel Error	Level	
248	SPI1_TXDMA_ERR	SPI1 TX DMA Channel Error	Level	
249	SPI1_RXDMA_ERR	SPI1 RX DMA Channel Error	Level	
250	SPI2_TXDMA_ERR	SPI2 TX DMA Channel Error	Level	
251	SPI2_RXDMA_ERR	SPI2 RX DMA Channel Error	Level	
252	SPI5_TXDMA_ERR	SPI5 TX DMA Error	Level	
253	SPI5_RXDMA_ERR	SPI5 RX DMA Error	Level	
254	SPORT0_A_DMA	SPORT0 Channel A DMA	Level	0
255	SPORT0_A_STAT	SPORT0 Channel A Status	Level	
256	SPORT0_B_DMA	SPORT0 Channel B DMA	Level	1
257	SPORT0_B_STAT	SPORT0 Channel B Status	Level	
258	SPORT1_A_DMA	SPORT1 Channel A DMA	Level	2
259	SPORT1_A_STAT	SPORT1 Channel A Status	Level	
260	SPORT1_B_DMA	SPORT1 Channel B DMA	Level	3
261	SPORT1_B_STAT	SPORT1 Channel B Status	Level	
262	SPORT2_A_DMA	SPORT2 Channel A DMA	Level	4
263	SPORT2_A_STAT	SPORT2 Channel A Status	Level	
264	SPORT2_B_DMA	SPORT2 Channel B DMA	Level	5
265	SPORT2_B_STAT	SPORT2 Channel B Status	Level	
266	SPORT3_A_DMA	SPORT3 Channel A DMA	Level	6
267	SPORT3_A_STAT	SPORT3 Channel A Status	Level	
268	SPORT3_B_DMA	SPORT3 Channel B DMA	Level	7
269	SPORT3_B_STAT	SPORT3 Channel B Status	Level	
270	SPORT4_A_DMA	SPORT4 Channel A DMA	Level	10
271	SPORT4_A_STAT	SPORT4 Channel A Status	Level	
272	SPORT4_B_DMA	SPORT4 Channel B DMA	Level	11
273	SPORT4_B_STAT	SPORT4 Channel B Status	Level	
274	SPORT5_A_DMA	SPORT5 Channel A DMA	Level	12

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
275	SPORT5_A_STAT	SPORT5 Channel A Status	Level	
276	SPORT5_B_DMA	SPORT5 Channel B DMA	Level	13
277	SPORT5_B_STAT	SPORT5 Channel B Status	Level	
278	SPORT6_A_DMA	SPORT6 Channel A DMA	Level	14
279	SPORT6_A_STAT	SPORT6 Channel A Status	Level	
280	SPORT6_B_DMA	SPORT6 Channel B DMA	Level	15
281	SPORT6_B_STAT	SPORT6 Channel B Status	Level	
282	SPORT7_A_DMA	SPORT7 Channel A DMA	Level	16
283	SPORT7_A_STAT	SPORT7 Channel A Status	Level	
284	SPORT7_B_DMA	SPORT7 Channel B DMA	Level	17
285	SPORT7_B_STAT	SPORT7 Channel B Status	Level	
286	DAI0_GBL_SPORT_INT0	DAI0 Global SPORT Interrupt 0	None	
287	DAI0_GBL_SPORT_INT1	DAI0 Global SPORT Interrupt 1	None	
288	DAI1_GBL_SPORT_INT0	DAI1 Global SPORT Interrupt 0	None	
289	DAI1_GBL_SPORT_INT1	DAI1 Global SPORT Interrupt 1	None	
290	SPORT0_A_DMA_ERR	SPORT0 Channel A DMA Error	Level	
291	SPORT0_B_DMA_ERR	SPORT0 Channel B DMA Error	Level	
292	SPORT1_A_DMA_ERR	SPORT1 Channel A DMA Error	Level	
293	SPORT1_B_DMA_ERR	SPORT1 Channel B DMA Error	Level	
294	SPORT2_A_DMA_ERR	SPORT2 Channel A DMA Error	Level	
295	SPORT2_B_DMA_ERR	SPORT2 Channel B DMA Error	Level	
296	SPORT3_A_DMA_ERR	SPORT3 Channel A DMA Error	Level	
297	SPORT3_B_DMA_ERR	SPORT3 Channel B DMA Error	Level	
298	SPORT4_A_DMA_ERR	SPORT4 Channel A DMA Error	Level	
299	SPORT4_B_DMA_ERR	SPORT4 Channel B DMA Error	Level	
300	SPORT5_A_DMA_ERR	SPORT5 Channel A DMA Error	Level	
301	SPORT5_B_DMA_ERR	SPORT5 Channel B DMA Error	Level	
302	SPORT6_A_DMA_ERR	SPORT6 Channel A DMA Error	Level	
303	SPORT6_B_DMA_ERR	SPORT6 Channel B DMA Error	Level	
304	SPORT7_A_DMA_ERR	SPORT7 Channel A DMA Error	Level	
305	SPORT7_B_DMA_ERR	SPORT7 Channel B DMA Error	Level	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
306	SPU0_INT	SPU0 Interrupt	Level	
307	SWU0_EVT	SWU0 Event SMC	None	
308	SWU1_EVT	SWU1 Event L2 Memory DMA Port 0	None	
309	SWU2_EVT	SWU2 Event L2 Memory Core Port 0	None	
310	SWU7_EVT	SWU7 Event SHARC0 Target Port 1	None	
311	SWU8_EVT	SWU8 Event SHARC0 Target Port 2	None	
312	SWU11_EVT	SWU11 Event SMMR	None	
313	SWU12_EVT	SWU12 Event SPI2/XSPI	None	
314	SWU13_EVT	SWU13 Event DMC0	None	
315	SWU3_EVT	SWU3 Event CL2_1	None	
316	SWU4_EVT	SWU4 Event DL2_1	None	
317	SWU5_EVT	SWU5 Event CL2_2	None	
318	SOFT8_INT	Software Interrupt 8		
319	SOFT9_INT	Software Interrupt 9		
320	SOFT0_INT	Software Interrupt 0		
321	SOFT1_INT	Software Interrupt 1		
322	SOFT2_INT	Software Interrupt 2		
323	SOFT3_INT	Software Interrupt 3		
324	SOFT4_INT	Software Interrupt 4		
325	SOFT5_INT	Software Interrupt 5		
326	SOFT6_INT	Software Interrupt 6		
327	SOFT7_INT	Software Interrupt 7		
328	TAPC_KEYFAIL	TAPC Test/User Key Fail Interrupt	Edge	
329	TIMER0_TMR00	TIMER0 Timer 0	Level	
330	TIMER0_TMR01	TIMER0 Timer 1	Level	
331	TIMER0_TMR02	TIMER0 Timer 2	Level	
332	TIMER0_TMR03	TIMER0 Timer 3	Level	
333	TIMER0_TMR04	TIMER0 Timer 4	Level	
334	TIMER0_TMR05	TIMER0 Timer 5	Level	
335	TIMER0_TMR06	TIMER0 Timer 6	Level	
336	TIMER0_TMR07	TIMER0 Timer 7	Level	

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
337	TIMER0_TMR08	TIMER0 Timer 8	Level	
338	TIMER0_TMR09	TIMER0 Timer 9	Level	
339	TIMER0_TMR10	TIMER0 Timer 10	Level	
340	TIMER0_TMR11	TIMER0 Timer 11	Level	
341	TIMER0_TMR12	TIMER0 Timer 12	Level	
342	TIMER0_TMR13	TIMER0 Timer 13	Level	
343	TIMER0_TMR14	TIMER0 Timer 14	Level	
344	TIMER0_TMR15	TIMER0 Timer 15	Level	
345	TIMER0_STAT	TIMER0 Status	Level	
346	HADC0_FIFO_EVT	HADC0 FIFO Event	Edge	
347	TMU0_FAULT	TMU0 Fault		
348	TMU0_ALERT	TMU0 Alert		
349	TRU0_RCV0	TRU0 Interrupt 0 - Core 0	Edge	
350	TRU0_RCV1	TRU0 Interrupt 1 - Core 0	Edge	
351	TRU0_RCV2	TRU0 Interrupt 2 - Core 0	Edge	
352	TRU0_RCV3	TRU0 Interrupt 3 - Core 0	Edge	
353	TRU0_RCV4	TRU0 Interrupt 4	Edge	
354	TRU0_RCV5	TRU0 Interrupt 5	Edge	
355	TRU0_RCV6	TRU0 Interrupt 6	Edge	
356	TRU0_RCV7	TRU0 Interrupt 7	Edge	
357	TRU0_RCV8	TRU0 Interrupt 8	Edge	
358	TRU0_RCV9	TRU0 Interrupt 9	Edge	
359	TRU0_RCV10	TRU0 Interrupt 10	Edge	
360	TRU0_RCV11	TRU0 Interrupt 11	Edge	
361	TWI0_DATA	TWI0 Data Interrupt	Level	
362	TWI1_DATA	TWI1 Data Interrupt	Level	
363	TWI2_DATA	TWI2 Data Interrupt	Level	
364	TWI3_DATA	TWI3 Data Interrupt	Level	
365	TWI4_DATA	TWI4 Data Interrupt	Level	
366	TWI5_DATA	TWI5 Data Interrupt	Level	
367	UART0_TXDMA	UART0 Transmit DMA	Level	20

Table 7-5: ADSP-2184x/ADSP-SC84x Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
368	UART0_RXDMA	UART0 Receive DMA	Level	21
369	UART0_STAT	UART0 Status	Level	
370	UART1_TXDMA	UART1 Transmit DMA	Level	34
371	UART1_RXDMA	UART1 Receive DMA	Level	35
372	UART1_STAT	UART1 Status	Level	
373	UART2_TXDMA	UART2 Transmit DMA	Level	37
374	UART2_RXDMA	UART2 Receive DMA	Level	38
375	UART2_STAT	UART2 Status	Level	
376	UART0_TXDMA_ERR	UART0 Transmit DMA Error	Level	
377	UART0_RXDMA_ERR	UART0 Receive DMA Error	Level	
378	UART1_TXDMA_ERR	UART1 Transmit DMA Error	Level	
379	UART1_RXDMA_ERR	UART1 Receive DMA Error	Level	
380	UART2_TXDMA_ERR	UART2 Transmit DMA Error	Level	
381	UART2_RXDMA_ERR	UART2 Receive DMA Error	Level	
382	WDOG0_EXP	WDOG0 Expiration	Level	
383	WDOG1_EXP	WDOG1 Expiration	Level	
384	WDOG2_EXP	WDOG2 Expiration	Level	
385	WDOG3_EXP	WDOG3 Expiration	Level	
386	XSPI0_IRQ	XSPI0 Interrupt Request	None	
387	XSPI1_IRQ	XSPI1 Interrupt Request	None	

SEC Definitions

The event controller uses the following definitions.

SCI

SEC core interface, core interface subblock of the SEC

SEA

SEC system event aggregator generates an aggregate request to the SEC based on the active events at a given time.

SID (Identification, unique)

Source numeric identifier for each system source connected to the SEC.

SFI

SEC Fault Interface, fault management subblock of the SEC.

SPR

SEC prioritizer determines the highest priority pending interrupt and the highest priority active interrupt. The SPR provides these interrupts in the appropriate registers of the SCI for the priority and nesting model of the SCI.

SSI

SEC source interface, system event source control, and status subblock of the SEC.

System Events

System source indications including interrupts and faults.

System Source

Point of origin of system event.

SEC System Event Aggregator (SEA)

System event sources request event service from the SEC. To reduce the number of required SSI channels in the SEC, the System Event Aggregator (SEA) acts as an intermediate aggregator between the source and the SEC. All unique system events have dedicated inputs to the SEA. The SEA generates an aggregate request to the SEC based on which events are active at a given time.

To determine the active event for a given SEC ID, read the `SEC_SSTAT[n].CHID` bit field. The *Combined SEC and GIC Interrupt List* table shows five SEAs connected to SEC ID 251-255. All the sources grouped over the SEA are error events for a set of peripherals like the FIR-IIR, SPI-UART-LP-EPPI, MDMA, SPORT, and SWU. An error event from a peripheral is only expected when that peripheral is enabled. To mask SEA events from sources, disable the source peripheral conditions. There are no registers to mask events from the SEA, but there is a provision to mask the complete group output at the SEC.

Events are handled with a fixed priority order, on a first-come, first-served basis. If multiple events corresponding to the same SEC ID are active at the same time, the priority level is order dependent with the lower SEA IDs having the higher priority. The SEA events are level-based, so they remain high until the events are cleared at the source ensuring no event is missed. There is no support for nesting in a group since they all have same the SEC ID. An SEA event must be handled completely before addressing the next event.

SEC Block Diagram

The *SEC Block Diagram* shows the event management architecture.

System sources connect to the SEC through the SSI. Each core has a dedicated SCI. The SFI provides fault action connections to the rest of the system.

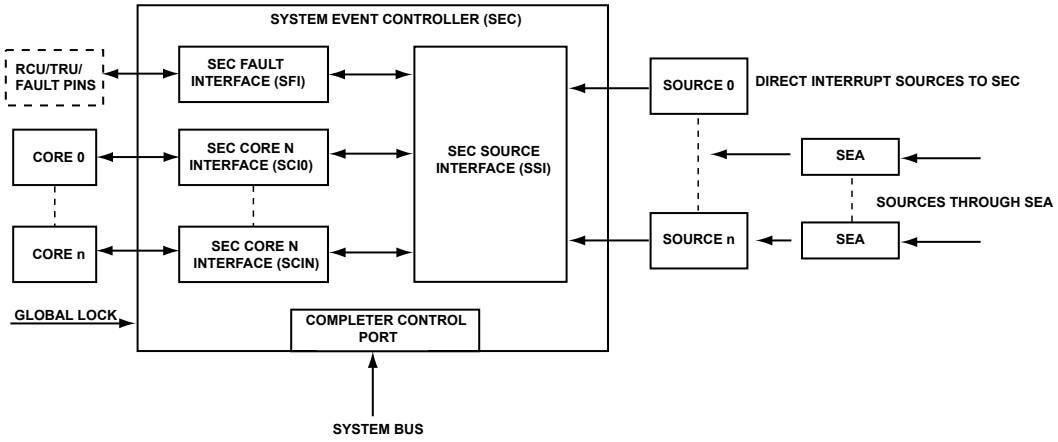


Figure 7-2: SEC Block Diagram

As shown in the figure, the SEC has three blocks for event management. The SFI monitors and manages any fault event triggered from various fault input sources. System interrupt sources are routed to the SFI through SEC source interface (SSI).

SEC Fault Interface (SFI)

The SFI manages fault events and associated actions. The fault management support provided in the SEC helps satisfy the safety requirements of various applications. The SSI provides the highest priority pending source that is enabled as a fault. The SFI captures this value and enables a countdown, and once the countdown expires, takes the prescribed action.

Fault actions which can be configured, as shown in *SFI Block Diagram*, include

- Trigger Output
- System Reset
- Fault Output
    - Computer Operating Properly (COP) mode
    - Fault Mode

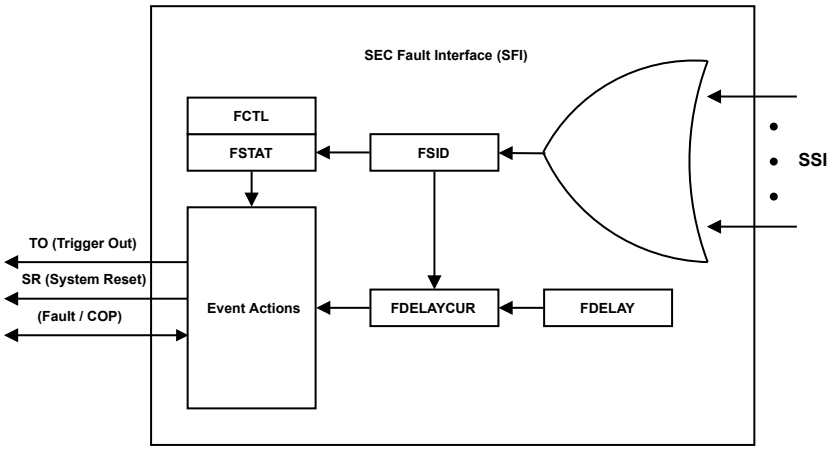


Figure 7-3: SFI Block Diagram

Fault Management

System sources can be enabled as fault sources in the SEC_SCTL[n] register. When a source enabled as a fault moves to pending, it is forwarded to the SFI as a fault indication. The pending bit (SEC_FSTAT.PND) indicates a source has signaled a fault assertion but it has not yet triggered the event actions (if delay is enabled). The SEC fault interface sets the SEC_FSTAT.PND bit when the fault source ID register (SEC_FSID) is updated on assertion of a fault source input. The system source pending triggers a fault pending and after a programmable delay the fault moves to active. Event actions then execute if appropriate action is not taken by the core. The SEC_FSTAT.ACT bit indicates that the SEC has received a fault source input, the delay has expired, and the fault actions are enabled.

The SEC_FSTAT.NPND bit indicates if one or more sources have signaled a fault assertion, but the input has not yet triggered the fault pending detection in the SEC fault interface. The SEC sets the SEC_FSTAT.NPND bit when the fault interface detects assertion of any enabled fault source input, while either the SEC_FSTAT.PND or SEC_FSTAT.ACT bits are set. The SEC clears the SEC_FSTAT.NPND bit when there are no fault sources waiting.

A fault indication from an external device can also be detected on sampling the fault signals. When a fault is detected the SEC_FSTAT.ACT and SEC_FSID.FEXT bits are set. The assertion of either signal results in a fault input detection.

The SEC_FEND register receives a fault end indication from the core. The core writes the SID of the fault to the SEC_FEND register. If the SID matches the value in the SEC_FSID register, the SEC_FSTAT.PND and SEC_FSTAT.ACT bits are cleared.

SEC Core Interface (SCI)

The SCI manages communication between the corresponding core and the SEC. The SEC prioritizer (SPR) of the SCI receives pending, active, and priority information from the SSI for each system event source assigned to this SCI. The SPR determines the highest priority pending system event and the SCI determines whether it propagates to the core. The SCI maintains the coherency for the system event service model implemented on the connected core.

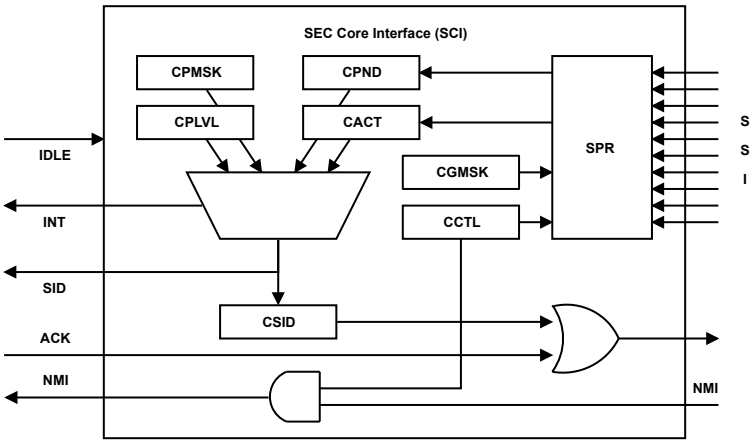


Figure 7-4: SCI Overview Block Diagram

SEC Source Interface (SSI)

The SSI manages all of the system event sources. It maintains the status of each source in the corresponding `SEC_SSTAT[n]` register. The corresponding `SEC_SCTL[n]` register manages the control of each source. A pending and enabled event passes its indication and priority to the SCI to which it is assigned for further processing.

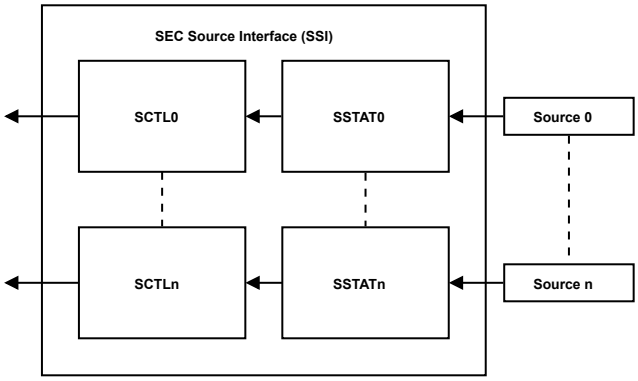


Figure 7-5: SSI Overview Block Diagram

SEC Architectural Concepts

The following sections describe SEC architectural features.

System Interrupt Groups

System sources can be assigned to groups using the `SEC_SCTL[n].GRP` bit field. Source groups allow fast context switching for system interrupts at each SCI. The `SEC_CGMSK[n]` register allows quick masking of interrupt groups of unlimited size with a single write operation.

System Interrupt Flow

An enabled and asserted system interrupt source is latched at the SSI and routed to the appropriate SCI based on the core target select (`SEC_SCTL[n].CTG`) bit field setting. The SEC priority ordering determines the highest priority pending system interrupt and the SCI updates the `SEC_CPND[n].SID` and `SEC_CACT[n].PRIO` bit

field values. The SCI compares the SEC_CPND[n] register value against the highest priority active source in the SEC_CACT[n] register).

The priority level register (SEC_CPLVL[n]) determines how many of the MSBs the SEC uses in the comparison. The priority mask register (SEC_CPMSK[n]) and the group mask register (SEC_CGMSK[n]) determines which pending interrupt sources participate. If the SEC_CPND[n] register value is a higher priority (lower value) than the priority of the SEC_CACT[n] register from the comparison based on the SEC_CPLVL[n] register, the system interrupt output is asserted. The source ID register (SEC_CSID[n]) is updated with the SEC_CPND[n].SID bit field value and forwarded to the connected core.

After the core provides an interrupt acknowledgment, the interrupt source is active, until the SEC completes interrupt service with a write to the SEC_END.SID bit field with the same value. Note the following:

- Interrupt acknowledgement occurs with an MMR write of the SEC_CSID[n] register or the core version of the SEC_CSID[n] register.
- Interrupt active status indication is SEC_SSTAT[n].ACT==1.

The following sequence shows the example flow for a single interrupt.

1. The SEC compares the SEC_CPND[n] register value to the SEC_CACT[n] register value. If the interrupt in the SEC_CPND[n] register is higher priority, continue.
2. The SEC copies the SEC_CPND[n] register value to the SEC_CSID[n] register and asserts the interrupt signal.
3. The core reads the SEC_CSID[n] register (or core version).
4. The core writes to the SEC_CSID[n] register (or core version, asserts the acknowledge signal).
5. The SEC deasserts the interrupt signal and clears the SEC_SSTAT[n].PND bit and sets the SEC_SSTAT[n].ACT bit of the source going active.
6. The core writes the SEC_CSID[n] of the active interrupt to the SEC_END register.
7. The SEC clears the SEC_SSTAT[n].ACT bit of the source being ended.

The following sequence shows the example flow for interrupt nesting where interrupt A is a lower priority and occurs earlier than interrupt B.

1. The SEC compares the SEC_CPND[n] (A) register value to the SEC_CACT[n] register and if the interrupt in the SEC_CPND[n] register is a higher priority, continue.
2. The SEC copies SEC_CPND[n] (A) register to the SEC_CSID[n] register and asserts the interrupt signal.
3. The core reads the SEC_CSID[n] (A) register (or core version).
4. The core writes to the SEC_CSID[n] register (or core version, asserts the acknowledge signal).
5. The SEC deasserts the INT signal and clears the SEC_SSTAT[n].PND bit and sets the SEC_SSTAT[n].ACT bit of the source (A) going active.

6. The SEC compares the SEC_CPND[n] (B) register value to the SEC_CACT[n] (A) register value. If the SEC_CACT[n] (B) register value is a higher priority, continue.
7. The SEC copies the SEC_CPND[n] (B) register value to SEC_CSID[n] register and asserts the interrupt signal.
8. The core reads the SEC_CSID[n] (B) register (or core version).
9. The core writes to the SEC_CSID[n] register (or core version, asserts the acknowledge signal).
10. The SEC deasserts the INT signal and clears the SEC_SSTAT[n].PND bit and sets the SEC_SSTAT[n].ACT bit of the source (B) going active.
11. The core writes the SEC_CSID[n] of the active interrupt (B) to the SEC_END register.
12. The SEC clears the SEC_SSTAT[n].ACT bit of the source (B) being ended.
13. The core writes the SEC_CSID[n] of the active interrupt (A) to the SEC_END register.
14. The SEC clears the SEC_SSTAT[n].ACT bit of the source (A) being ended.

System Interrupt Priorities

Each system interrupt source has its own programmable priority level which is configured using the SEC_SCTL[n].PRIO bit field. The SCI evaluates the priority of all pending sources to determine the source of the highest priority pending system interrupt for forwarding to the attached core. If more than one source of the pending system interrupt has the same priority setting, the SCI chooses the one with the lowest SID. For example, if SID 0, SID 1, and SID 2 are all pending and have the same priority setting, the SCI chooses SID 0 as the highest-priority source.

SEC Error

The processor includes an SEC error (SEC_GSTAT.ERR) as a source input to the SEC to allow for handling the error as an interrupt or fault.

SEC Programming Model

Implementing a system interrupt service model using the SEC requires, at a minimum:

- Proper configuration of a system interrupt source (for example a peripheral or DMA)
- A core interrupt or event service model

The core must be configured for response to system interrupts from the SEC. The SEC must be configured to enable and map the system interrupt source to the correct SCI and to forward interrupts to the connected core.

The system interrupt source must be configured to generate interrupt assertions. Alternatively, the processor can use software triggering for interrupt assertion. Software driven interrupts are generated by writing the source ID of the interrupt to be triggered to the SEC_RAISE register.

Programming Concepts

The following list provides the basic programming concepts necessary for configuring the SEC.

- Configuring an SSI as a system interrupt for a specific core.
- Configuring an SCI to provide system interrupts to the connected core (See Configuring a System Source to Interrupt a Core).
- Configuring an SSI as a system fault (See Configuring a System Source as a Fault).
- Configuring the SFI to manage system faults.

Programming Examples

This section provides example programming tasks that are typical for SEC usage.

Fault Management Interface Programming Model

The SFI interface can be programmed to manage fault events from system sources and associated actions such as issuing a system reset when watchdog expiration event occurs.

1. Set the `SEC_GCTL.EN` bit to enable the SEC.
2. Write to the `SEC_FCTL` register to configure specific fault actions.
    - Trigger Output. Set the `SEC_FCTL.TOEN` bit for the SEC to produce trigger outputs when a fault becomes active. The `SEC_FCTL.TES` bit can be programmed to select the event that directs the SEC to assert trigger output when a fault is pending or active. Configure receivers for SEC fault trigger generator output.

      NOTE: If the `SEC_FCTL.TOEN` and or the `SEC_FCTL.TES` bits =1 (Trigger Output Enabled and Trigger on Fault Pending), an external fault (if enabled by the `SEC_FCTL.FIEN` bit) will not issue a trigger since Fault Pending is bypassed for external faults.
    - System Reset. The Reset Control Unit (RCU) controls how the functional units enter and exit reset. Configure the `RCU_CTL.SRSTREQEN` bit. This bit controls whether the sources of reset are enabled to perform a system reset. To issue a system reset request when a fault becomes active, set the `SEC_FCTL.SREN` bit. The SEC fault system reset delay register (`SEC_FSRDLY`) can be programmed for the delay, if required, from a fault becoming active to system reset request assertion.
    - Fault Output. This configuration allows the SEC to indicate the fault status based on the `SEC_FCTL.CMS` bit configuration.
        - Computer Operating Normally (COP) mode. To configure fault output for COP mode, set the `SEC_FCTL.FOEN` bit to enable fault output. Set the `SEC_FCTL.CMS` bit to select COP mode to toggle the fault pin when no fault is active. Program the `SEC_FCOPP` period register with a desired width value for the COP toggled output pin.

- Fault mode. Set the SEC_FCTL.FOEN bit to enable fault output. The SEC_FCTL.CMS bit should be set to Fault mode to toggle the fault pin when a fault is active.

3. If required, program the Fault Input to sample fault inputs from external devices on fault pins. Configure the SEC_FCTL.FIEN bit to enable the SEC to sample a fault input from an external device.

   ADDITIONAL INFORMATION: The SEC_FCTL.FIEN bit should be set only while the SEC_FCTL.EN bit is low. If the SEC_FCTL.EN bit is already high and the SEC_FCTL.FIEN bit needs to be set, the SEC_FCTL.EN bit should be cleared first. Fault input can only be enabled when Fault mode is selected by the SEC_FCTL.CMS bit.

4. Program the required fault delay to the SEC_FDLY.COUNT bit field if a delay between fault source assertion and the fault response is required.

5. Configure the SEC_FCTL register to enable the SEC.

   ADDITIONAL INFORMATION: The SEC_FCTL.EN bit should be set only while the SEC_FSTAT.ACT bit is low.

6. Write to the control register of a specific source register using the SEC_SCTL[n] register to enable the source as a fault.

Configuring a System Source to Interrupt a Core

To configure a system source to interrupt a core, the SEC itself must be enabled with the source interface (SSI) and core interface (SCI) properly initialized. Specifically, the SCI must be set up to accept interrupt signaling from the SEC and pass them to the specified core, and the SSI must properly enable each of the peripheral interrupt sources to generate interrupt signals and optionally define a priority scheme that overrides the default priority settings. In summary:

1. Write to the SEC_GCTL register to enable the SEC.
2. Write to the appropriate SCI SEC_CCTL[n] register to enable SEC interrupts to be sent to that core.
3. Write to the appropriate SSI SEC_SCTL[n] register to enable that peripheral as an interrupt source and to set the core target field to map the source to the desired SCI.
4. (Optional) By default, all the SEC interrupts are grouped as a single priority level, so passing of peripheral interrupt requests from the SEC is based solely on the default enumerated source ID. By programming the SEC_CPLVL[n].PLVL register, interrupt sources can be grouped into priority levels within the SEC such that arbitration is first performed by source ID within a grouped priority level before proceeding to the next priority level, thus providing the flexibility to have lower-priority interrupt sources considered before higher-priority sources.

   ADDITIONAL INFORMATION: The SEC_CPMSK[n] and SEC_CGMSK[n] registers must also can be programmed to mask the interrupts based on the customized levels and grouping.

Core/SEC Handshake Requirements to Ensure Proper Interrupt Handling

A specific handshake with the SEC is required to handle interrupts associated with an individual core. The handshake ensures that nested interrupts are properly tracked and that new peripheral interrupts being raised within the SEC are either passed immediately to the core or held off and queued within the SEC for later servicing.

Use the following procedure to write a custom dispatcher inside the Interrupt Service Routine. Note that the core needs to read and acknowledge the SEC_CSID[n] register by writing the same value. The core must also write to the SEC_END register after the ISR execution completes.

1. Read the SEC_CSID[n] register to obtain the source ID of the peripheral interrupt request.
2. Write the read value back to the SEC_CSID[n] register to send the acknowledge signal to the SEC that the core has accepted and begun processing the interrupt request.
3. Execute the actual ISR (typically a call to a specific handler function from a look-up table based on the peripheral source ID). Write to the SEC_GCTL register to enable the SEC.
4. Write the value of the SEC_CSID[n] register of the active interrupt (read in step 1 above) to the SEC_END register to signal to the SEC that the interrupt has now been serviced.
5. Return from interrupt.

This procedure allows a higher-priority interrupt raised by the SEC to be serviced by the core after step 2. The SEC knows what it passed to the core because of the write to the SEC_CSID[n] register. After the core acknowledges that write, the SEC knows whether newly raised peripheral interrupts are a higher priority than the highest-priority interrupt currently being processed by the core.

- When a higher priority, the SEC pushes the current SEC_CSID[n] value to an internal stack, writes the new SEC_CSID[n] value, and asserts a new SEC interrupt request.
- When a lower priority, the SEC queues the interrupt until the core writes to the SEC_END register with the source ID of the higher-priority interrupt, confirming that it was fully processed.

At this point the SEC_CSID[n] value is popped from the internal stack and any pending peripheral interrupt requests are arbitrated before the SEC writes the new SEC_CSID[n] value and asserts a new interrupt request. At the same time the core self-nests the latched SEC interrupt requests as needed. When a higher-priority interrupt is presented to the core the write to the SEC_END register in the SEC handler epilog code guarantees that each nested level has the required handshake to signal to the SEC block that each individual source ID interrupt request is fully serviced. See the *SHARC+ Core Programming Reference* for more details about SEC handler code.

Configuring a System Source as a Fault

Use the following procedure configure a system source as a fault.

1. Write to the SEC_GCTL register to enable the SEC.
2. Write to the SEC_FCTL register to configure specific fault actions.
3. Write to the SEC_FDLY bit field to specify fault delay.

4. Write to the control register of a specific source to enable the source as a fault.

Configuring the WDOG Expiry Event to Issue a System Reset

Use the following procedure to configure the WDOG timer to issue a system reset.

1. Configure the SEC_GCTL register to enable the SEC.

```
*pREG_SEC0_GCTL = BITM_SEC_GCTL_EN;
```

2. Configure the SEC_FCTL register to choose the Fault response mode. In the following code example, the system reset is issued.

```
*pREG_RCU0_CTL |= BITM_RCU_CTL_SRSTREQEN;
*pREG_SEC0_FCTL |= BITM_SEC_FCTL_SREN;
```

3. Configure the SEC_SCTL[n].SEN and SEC_SCTL[n].FEN bits in the registers to determine how the fault source is handled. To configure the WDOG as the fault source, program the register. The program can configure any interrupt as the fault source by programming the corresponding register.

```
 *pREG_SEC0_SCTL3 = BITM_SEC_SCTL_FEN|BITM_SEC_SCTL_SEN;
```

ADDITIONAL INFORMATION: The SEC ID corresponding to WDOG0 is 3, as indicated in the ADSP-2184x Interrupt List .

4. Write to the enable bit.

```
*pREG_SEC0_FCTL |= BITM_SEC_FCTL_EN;
```

SEC Programming Restrictions

Setting the SEC_FCTL.EN bit while the SEC_FSTAT.ACT bit is high can result in unpredictable behavior. To avoid this issue, set the SEC_FCTL.EN bit while the SEC_FSTAT.ACT bit is low. The SEC_FSTAT.ACT bit is only set when the SEC_FCTL.EN bit is high. Therefore, the problem can only occur if the SEC_FCTL.EN bit transitions from 1 to 0 and then to 1 again.

Writing to SEC_FEND to end a fault with both the SEC_FCTL.FOEN bit and the SEC_FCTL.FIEN bit set can result in erroneous external fault detection. If this operation (ending a fault) and configuration (fault input and fault output enabled) are required by the application, clear the SEC_FCTL.FOEN bit prior to writing to SEC_FEND. The recommended sequence for ending a fault with the SEC_FCTL.FIEN or SEC_FCTL.FOEN==1 is as follows:

1. Clear the SEC_FCTL.FOEN bit.
2. Write to the SEC_FEND register.
3. Set the SEC_FCTL.FOEN bit.

ADSP-2184x SEC Register Descriptions

System Event Controller (SEC) contains the following registers.

Table 7-6: ADSP-2184x SEC Register List

Name	Description
SEC_CACT[n]	SCI Active Register n
SEC_CCTL[n]	SCI Control Register n
SEC_CGMSK[n]	SCI Group Mask Register n
SEC_CPLVL[n]	SCI Priority Level Register n
SEC_CPMSK[n]	SCI Priority Mask Register n
SEC_CPND[n]	Core Pending Register n
SEC_CSID[n]	SCI Source ID Register n
SEC_CSTAT[n]	SCI Status Register n
SEC_END	Global End Register
SEC_FCOPP	Fault COP Period Register
SEC_FCOPP_CUR	Fault COP Period Current Register
SEC_FCTL	Fault Control Register
SEC_FDLY	Fault Delay Register
SEC_FDLY_CUR	Fault Delay Current Register
SEC_FEND	Fault End Register
SEC_FSID	Fault Source ID Register
SEC_FSRDLY	Fault System Reset Delay Register
SEC_FSRDLY_CUR	Fault System Reset Delay Current Register
SEC_FSTAT	Fault Status Register
SEC_GCTL	Global Control Register
SEC_GSTAT	Global Status Register
SEC_RAISE	Global Raise Register
SEC_SCTL[n]	Source Control Register n
SEC_SSTAT[n]	Source Status Register n

SCI Active Register n

The SEC SCI active interrupt register (SEC_CACT[n]) contains the source ID and priority of the highest priority active interrupt detected by the SEC prioritizer.

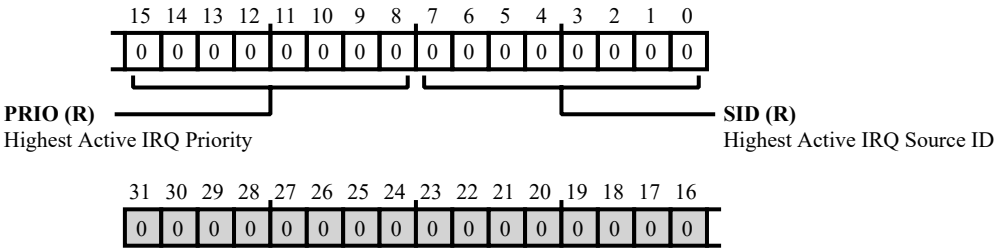


Figure 7-6: SEC_CACT[n] Register Diagram

Table 7-7: SEC_CACT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/NW)	PRIO	Highest Active IRQ Priority. The SEC_CACT[n].PRIO indicates the priority value of the highest priority active interrupt for core n.
7:0 (R/NW)	SID	Highest Active IRQ Source ID. The SEC_CACT[n].SID identifies the source ID value of the highest priority active interrupt for core n.

SCI Control Register n

The SEC control register (SEC_CCTL[n]) contains SCI control bits for all system sources.

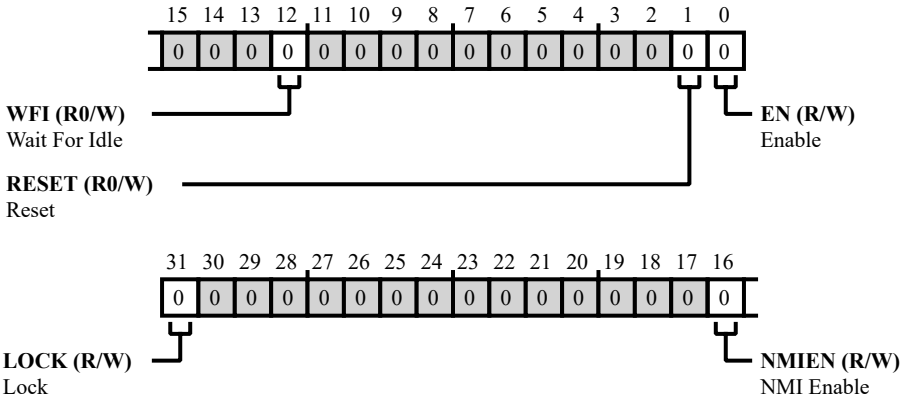


Figure 7-7: SEC_CCTL[n] Register Diagram

Table 7-8: SEC_CCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CCTL[n].LOCK bit is enabled, the SEC_CCTL[n] register is read only.	
		0	Unlock
		1	Lock
16 (R/W)	NMIEN	NMI Enable. The SEC_CCTL[n].NMIEN bit controls NMI propagation to the core. When the SEC_CCTL[n].NMIEN bit is enabled, the SCI allows NMIs to propagate to the core for servicing.	
		0	Disable
		1	Enable
12 (R0/W)	WFI	Wait For Idle. When set, the SEC_CCTL[n].WFI bit forces the SCI to wait for indication of core idle before the SCI resumes activity.	
		0	No Action
		1	Wait for Idle

Table 7-8: SEC_CCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R0/W)	RESET	Reset. When set, the `SEC_CCTL[n].RESET` bit resets all SCI registers to their default values.	
		0	No Action
		1	Reset
0 (R/W)	EN	Enable. The `SEC_CCTL[n].EN` bit controls operation of the SCI. Clearing the `SEC_CCTL[n].EN` bit halts the execution of the SCI without resetting status registers. (The INT signal to a core is not affected.) Setting the `SEC_CCTL[n].EN` bit enables the SCI to begin or resume operation with the current configuration and status.	
		0	Disable
		1	Enable

SCI Group Mask Register n

The SEC SCI group mask register (SEC_CGMSK[n]) contains selections for a group mask, an ungroup mask, and a register lock. This register contains the system interrupt group masks for the connected core. The core uses the SEC_CGMSK[n].UGRP and SEC_CGMSK[n].GRP fields to mask (disable) interrupts from the specified groups.

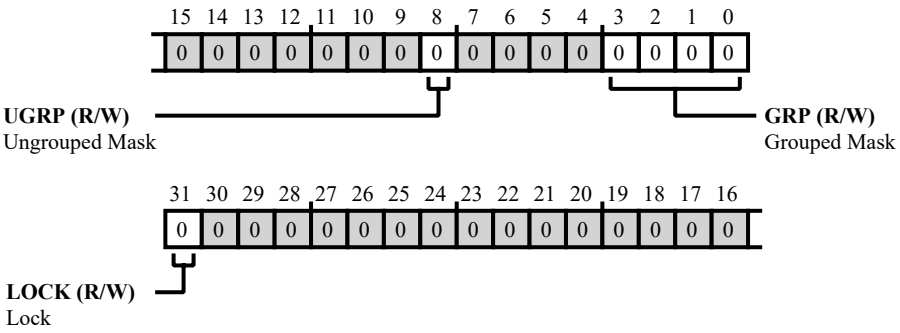


Figure 7-8: SEC_CGMSK[n] Register Diagram

Table 7-9: SEC_CGMSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CGMSK[n].LOCK bit is enabled, the SEC_CGMSK[n] register is read only.	
		0	Unlock
		1	Lock
8 (R/W)	UGRP	Ungrouped Mask. The SEC_CGMSK[n].UGRP bit masks interrupts (if set) for the ungrouped interrupt sources for core n.	
		0	Unmask Ungrouped Sources
		1	Mask Ungrouped Sources
3:0 (R/W)	GRP	Grouped Mask. The SEC_CGMSK[n].GRP field selects a group of interrupt sources to mask for core n. (For more information about interrupt source groups, see the SEC_SCTL[n] register description.)	
		0	No groups masked
		1	Mask group 0
		2	Mask group 1
		3	Mask groups 0, 1

Table 7-9: SEC_CGMSK[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	Mask group 2
		5	Mask groups 0, 2
		6	Mask groups 1, 2
		7	Mask groups 0, 1, 2
		8	Mask group 3
		9	Mask groups 0, 3
		10	Mask groups 1, 3
		11	Mask groups 0, 1, 3
		12	Mask groups 2, 3
		13	Mask groups 0, 2, 3
		14	Mask groups 1, 2, 3
		15	Mask groups 0, 1, 2, 3

SCI Priority Level Register n

The SEC SCI priority level register (SEC_CPLVL[n]) contains selections for priority levels and a register lock. This register is used to divide the total number of priority levels into sub-levels. The sub-level priority resolution provides the tie breaker for simultaneously pending interrupts assigned to the same level.

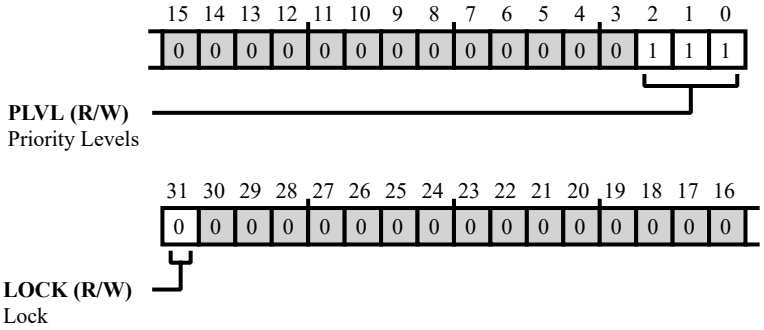


Figure 7-9: SEC_CPLVL[n] Register Diagram

Table 7-10: SEC_CPLVL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CPLVL[n].LOCK bit is enabled, the SEC_CPLVL[n] register is read only.	
		0	Unlock
		1	Lock
2:0 (R/W)	PLVL	Priority Levels. The SEC_CPLVL[n].PLVL field serves to divide the total number of interrupt priority levels into sub-levels. The sub-level priority resolution provides the tie breaker for simultaneously pending interrupts assigned to the same interrupt level. The sub-level priority value specifies the number of MSBs (minus 1) designated to interrupt levels, while the remaining LSBs are designated for sub-level specification. For example, if the SEC_CPLVL[n].PLVL field is set to two, the result is four priority levels are specified, because only the two MSBs are used for preemption evaluation. The remaining bits of the priority setting are used for sub-level prioritization.	
		0	1 MSBs (2 priority levels)
		1	2 MSBs (4 priority levels)
		2	3 MSBs (8 priority levels)
		3	4 MSBs (16 priority levels)
		4	5 MSBs (32 priority levels)
		5	6 MSBs (64 priority levels)

Table 7-10: SEC_CPLVL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	7 MSBs (128 priority levels)
		7	8 MSBs (256 priority levels)

SCI Priority Mask Register n

The SEC SCI priority mask register (SEC_CPMSK[n]) contains the SCI priority mask for core n and includes a register lock.

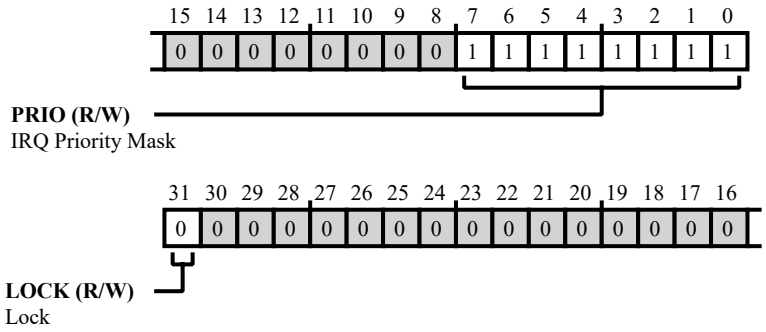


Figure 7-10: SEC_CPMSK[n] Register Diagram

Table 7-11: SEC_CPMSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CPMSK[n].LOCK bit is enabled, the SEC_CPMSK[n] register is read only.	
		0	Unlock
		1	Lock
7:0 (R/W)	PRIO	IRQ Priority Mask. The SEC_CPMSK[n].PRIO contains the system interrupt priority mask for core n. The core uses the SEC_CPMSK[n].PRIO field to mask (block) interrupts below the specified level.	
		0	Priority level 0 (highest)
		1-254	
		255	Priority level 255 (lowest)

Core Pending Register n

The SCI pending interrupt register (SEC_CPND[n]) contains the source ID and priority of the highest priority pending interrupt detected by the SEC prioritizer.

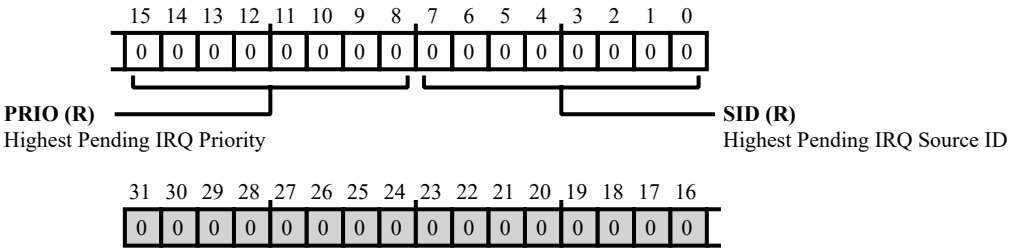


Figure 7-11: SEC_CPND[n] Register Diagram

Table 7-12: SEC_CPND[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/NW)	PRIO	Highest Pending IRQ Priority. The SEC_CPND[n].PRIO indicates the priority value of the highest priority pending interrupt for core n.
7:0 (R/NW)	SID	Highest Pending IRQ Source ID. The SEC_CPND[n].SID identifies the source ID value of the highest priority pending interrupt for core n.

SCI Source ID Register n

The SCI source ID register (SEC_CSID[n]) contains the source ID of the interrupt last issued to core n. The SEC_CSID[n] register value is loaded by the SCI when a system interrupt indication is sent to core n. The SCI does not change the SEC_CSID[n] until after the interface receives an interrupt acknowledge from core n. Writing to the SEC_CSID[n] register generates an interrupt acknowledge, but does not update the value in the register.

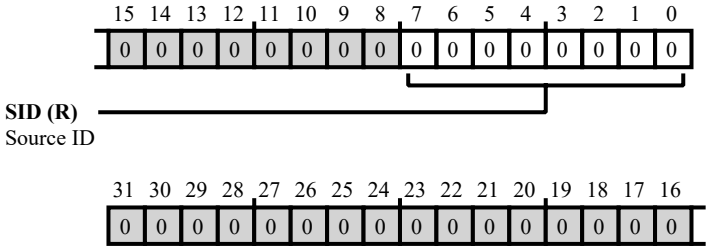


Figure 7-12: SEC_CSID[n] Register Diagram

Table 7-13: SEC_CSID[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	SID	Source ID. The SEC_CSID[n].SID bit is the source ID of the interrupt last issued to core n.

SCI Status Register n

The SCI status register (SEC_CSTAT[n]) contains status bits, indicating the operational status of the SCI.

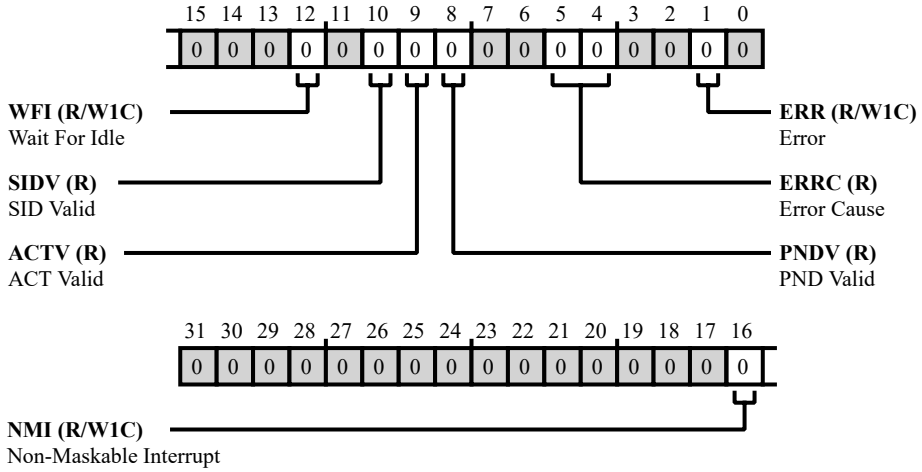


Figure 7-13: SEC_CSTAT[n] Register Diagram

Table 7-14: SEC_CSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	NMI	Non-Maskable Interrupt. The SEC_CSTAT[n].NMI bit indicates whether an NMI has occurred since the bit was last cleared.	
		0	No NMI Occurred
		1	NMI Occurred
12 (R/W1C)	WFI	Wait For Idle. The SEC_CSTAT[n].WFI bit indicates (if set) that the SCI is temporarily disabled, pending a core idle indication. This bit is set when SEC_CCTL[n].WFI is set.	
		0	Not Waiting
		1	Waiting
10 (R/NW)	SIDV	SID Valid. The SEC_CSTAT[n].SIDV bit indicates (if set) that the current value in the SEC_CSID[n] register is valid. The SCI sets the SEC_CSTAT[n].SIDV bit when the updating the SEC_CSID[n] register with a new value. The SEC_CSTAT[n].SIDV bit is cleared when the SEC_CSID[n] register is written. This status indication may be used to extract all pending interrupts in a single interrupt service routine.	
		0	Invalid
		1	Valid

Table 7-14: SEC_CSTAT[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	ACTV	ACT Valid. The SEC_CSTAT[n].ACTV bit indicates (if set) that the current value in the SEC_CACT[n] register is valid. The SCI sets the SEC_CSTAT[n].ACTV bit when updating the SEC_CACT[n] registers with a new value. The SEC_CSTAT[n].ACTV bit is cleared when the SEC_CSID[n] register is written.	
		0	Invalid
		1	Valid
8 (R/NW)	PNDV	PND Valid. The SEC_CSTAT[n].PNDV bit indicates (if set) that the current value in the SEC_CPND[n] register is valid. The SCI sets the SEC_CSTAT[n].PNDV bit when updating the SEC_CPND[n] register with a new value. The SEC_CSTAT[n].PNDV bit is cleared when the SEC_CSID[n] register is written.	
		0	Invalid
		1	Valid
5:4 (R/NW)	ERRC	Error Cause. The SEC_CSTAT[n].ERRC bits are updated on assertion of the SEC_CSTAT[n].ERR bit to indicate the SCI error type. SEC_CSTAT[n].ERRC is only updated on the assertion of SEC_CSTAT[n].ERR. Subsequent errors while SEC_CSTAT[n].ERR is asserted do not update SEC_CSTAT[n].ERRC.	
		0	Reserved
		1	Acknowledge Error. SCI has received an acknowledge without a pending, unacknowledged interrupt present.
		2	Reserved
		3	Reserved
1 (R/W1C)	ERR	Error. The SEC_CSTAT[n].ERR bit indicates that an error has occurred in the SCI. When SEC_CSTAT[n].ERR is set, the SCI updates the SEC_CSTAT[n].ERRC field to the value of the corresponding error cause.	
		0	No Error
		1	Error Occurred

Global End Register

The SEC global end register (SEC_END) contains a source ID interrupt service end field (SEC_END.SID). When a core has finished servicing an interrupt, the core writes the SEC_END.SID field in the SEC_END register. This write causes the SEC to clear the SEC_SSTAT[n].ACT bit in the SEC_SSTAT[n] register of the corresponding interrupt.

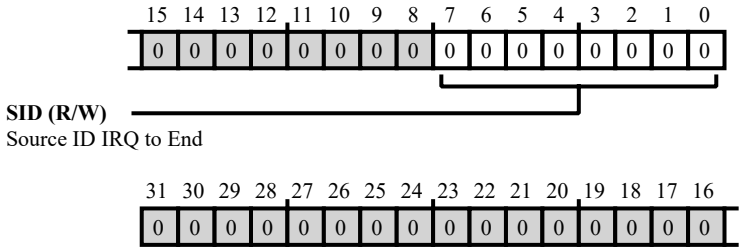


Figure 7-14: SEC_END Register Diagram

Table 7-15: SEC_END Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	SID	Source ID IRQ to End. The SEC_END.SID bit field contains the source ID interrupt service end value.

Fault COP Period Register

The SEC fault COP period register (SEC_FCOPP) contains the width value (count in (SEC) clock cycles) for the high and low phase of the computer operating properly (COP) toggled output on the COP pin. Note that the actual high/low phase value is the SEC_FCOPP.COUNT programmed value plus 1.

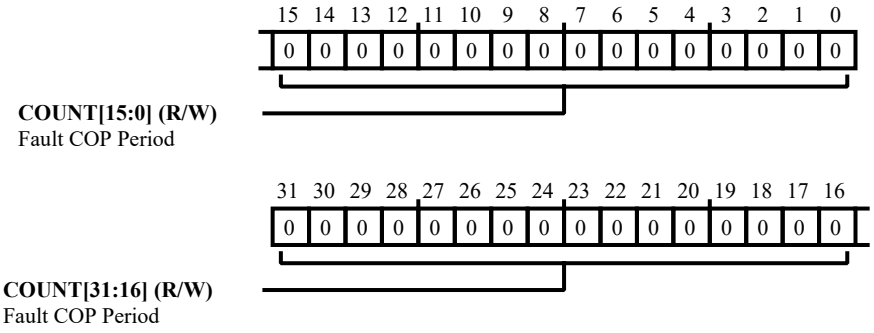


Figure 7-15: SEC_FCOPP Register Diagram

Table 7-16: SEC_FCOPP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	COUNT	Fault COP Period. The SEC_FCOPP.COUNT bit field is the width value for the high and low phase of the computer operating properly (COP) toggled output on the COP pin.

Fault COP Period Current Register

The SEC fault COP period current register (SEC_FCOPP_CUR) contains the active count (in (SEC) clock periods) for the current phase (high or low) of the computer operating properly (COP) toggled output on the COP pin. The SEC loads the SEC_FCOPP_CUR register from the SEC_FCOPP register when the SEC_FCOPP_CUR.COUNT field is cleared and the SEC is in COP mode (SEC_FCTL.CMS bit =1). The SEC decrements the SEC_FCOPP_CUR count each (SEC) clock cycle while SEC_FCTL.CMS is set and the SEC_FSTAT.ACT bit is not set.

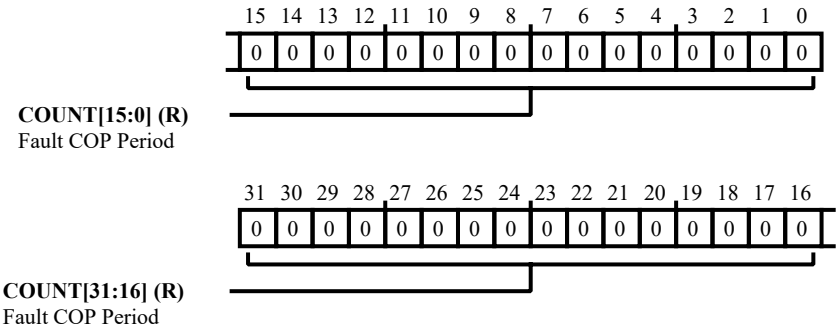


Figure 7-16: SEC_FCOPP_CUR Register Diagram

Table 7-17: SEC_FCOPP_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	COUNT	Fault COP Period. The SEC_FCOPP_CUR.COUNT bit field is the active count for the current phase (high or low) of the computer operating properly (COP) toggled output on the COP pin.

Fault Control Register

The SEC fault control register (SEC_FCTL) contains fault control bits for all SEC channels. This register controls the operation of the System Fault Management Interface (SFI).

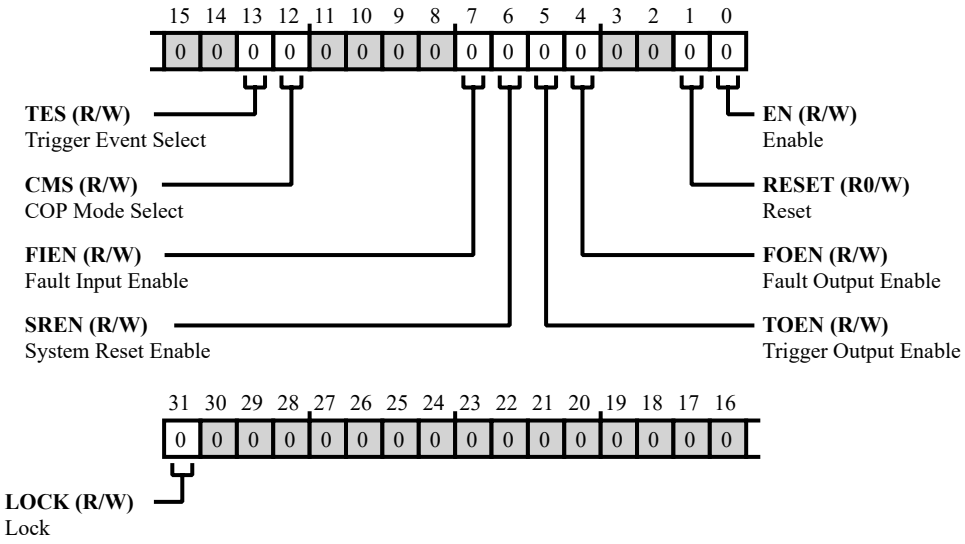


Figure 7-17: SEC_FCTL Register Diagram

Table 7-18: SEC_FCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_FCTL.LOCK bit is enabled, the SEC_FCTL register is read only.	
		0	UnLock
		1	Lock
13 (R/W)	TES	Trigger Event Select. The SEC_FCTL.TES bit selects the event that directs the SEC to assert trigger output. In fault pending mode, the SEC asserts trigger output when a fault is pending. In fault active mode, the SEC asserts trigger output when a fault is active.	
		0	Fault Active Mode
		1	Fault Pending Mode

Table 7-18: SEC_FCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	CMS	COP Mode Select. The SEC_FCTL.CMS selects the SEC mode for handling fault input. In COP mode, the SEC toggles the COP pin to indicate that no fault is active and ceases toggling the pin to indicate that a fault is active. In fault mode, the SEC deasserts the fault pin (=0) and fault_b pin (=1) when no fault is active and asserts the fault pin (=1) and fault_b pin (=0) when a fault is active. Not all processors feature both the fault and fault_b pins. Refer to the product data sheet for details.	
		0	Fault Mode
		1	COP Mode
7 (R/W)	FIEN	Fault Input Enable. The SEC_FCTL.FIEN bit enables the SEC to sample fault input. If SEC_FCTL.FIEN is set (=1), a fault indication from an external device sets the SEC_FSTAT.ACT bit and SEC_FSID.FEXT bit.	
		0	Disable
		1	Enable
6 (R/W)	SREN	System Reset Enable. The SEC_FCTL.SREN bit enables the SEC to issue a system reset request when a fault becomes active.	
		0	Disable
		1	Enable
5 (R/W)	TOEN	Trigger Output Enable. The SEC_FCTL.TOEN bit enables the SEC to produce trigger output when a fault becomes active.	
		0	Disable
		1	Enable
4 (R/W)	FOEN	Fault Output Enable. The SEC_FCTL.FOEN bit enables the SEC to indicate fault status, according to the SEC_FCTL.CMS bit configuration.	
		0	Disable
		1	Enable
1 (R0/W)	RESET	Reset. Setting the SEC_FCTL.RESET bit resets ALL SEC registers to their default values.	
		0	No Action
		1	Reset

Table 7-18: SEC_FCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Enable. The `SEC_FCTL.EN` bit controls the operational state of the SEC. Clearing the `SEC_FCTL.EN` bit halts the execution of the SEC without resetting status registers. Setting the `SEC_FCTL.EN` bit enables the SEC to begin or resume operation with the current configuration and status.	
		0	Disable
		1	Enable

Fault Delay Register

The SEC fault delay register (SEC_FDLY) contains the number (SEC_FDLY.COUNT field) of (SEC) clock periods to delay from fault pending to fault active, when actions are enabled.

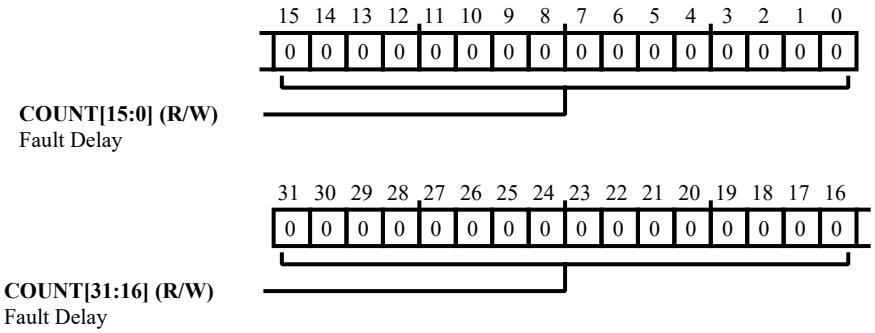


Figure 7-18: SEC_FDLY Register Diagram

Table 7-19: SEC_FDLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	COUNT	Fault Delay. The SEC_FDLY.COUNT bit field is the number of (SEC) clock periods to delay from fault pending to fault active, when actions are enabled.

Fault Delay Current Register

The SEC fault delay current register (SEC_FDLY_CUR) contains the active count (SEC_FDLY_CUR.COUNT field) in (SEC) clock periods for the delay from fault pending to fault active, when actions are enabled. The count is loaded from the SEC_FDLY register when a fault becomes pending (SEC_FSTAT.PND bit is set). The SEC decrements the value in SEC_FDLY_CUR each (SEC) clock cycle while the SEC_FSTAT.PND bit is set.

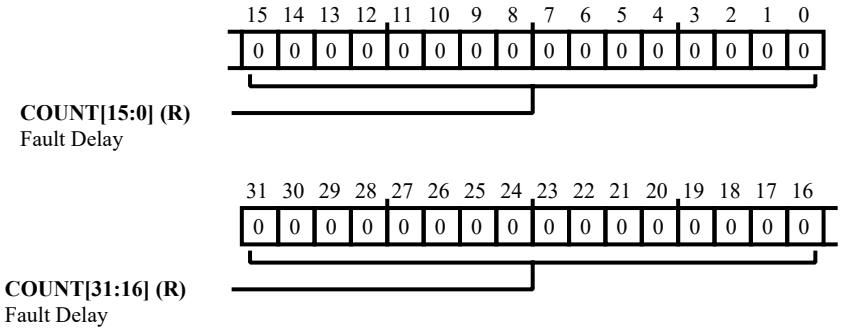


Figure 7-19: SEC_FDLY_CUR Register Diagram

Table 7-20: SEC_FDLY_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	COUNT	Fault Delay. The SEC_FDLY_CUR.COUNT bit field is the active count in (SEC) clock periods for the delay from fault pending to fault active, when actions are enabled.

Fault End Register

The SEC fault end register (SEC_FEND) contains fault source ID and internal/external fields. This register receives fault end indication from a core.

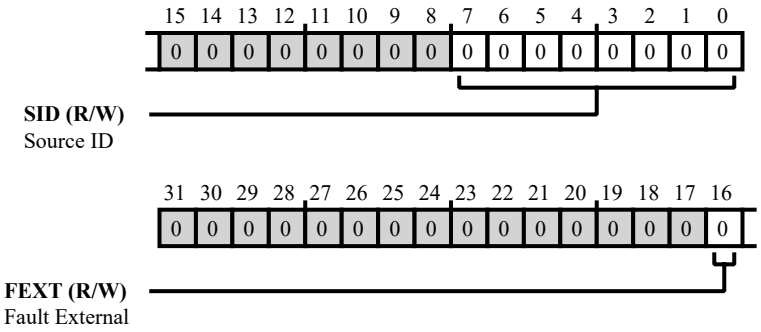


Figure 7-20: SEC_FEND Register Diagram

Table 7-21: SEC_FEND Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	FEXT	Fault External. Setting the SEC_FEND.FEXT bit, when the SEC_FEND.SID field is cleared, clears an active fault from an external source.	
		0	Fault Internal
		1	Fault External
7:0 (R/W)	SID	Source ID. The SEC_FEND.SID identifies a fault to be ended as indicated to the SEC by the core. The core loads the SEC_FEND.SID field value. If the SEC_FEND.SID value matches the SEC_FSID.SID value, the SEC_FSTAT.PND bit and SEC_FSTAT.ACT bit are cleared.	

Fault Source ID Register

The SEC fault source ID register (SEC_FSID) contains a fault source ID and internal/external fields.

NOTE:These bits are not reset by system reset so that a fault that automatically triggers a system reset to avoid a fault may be analyzed after the reset occurs.

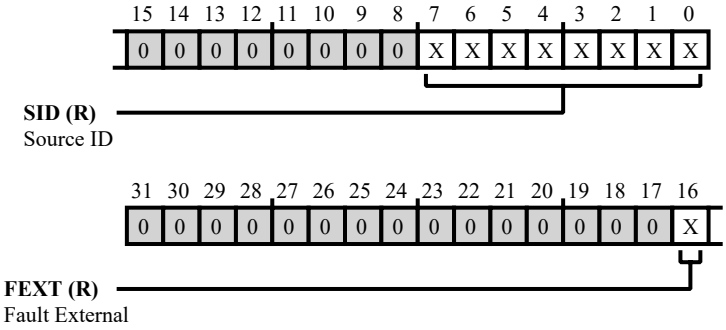


Figure 7-21: SEC_FSID Register Diagram

Table 7-22: SEC_FSID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	FEXT	Fault External. The SEC_FSID.FEXT bit indicates that the last active fault was asserted by an external device. The SEC sets the SEC_FSID.FEXT bit when the SEC_FSTAT.ACT bit is set by the fault input pins. The SEC_FSID.FEXT bit is cleared when the SEC_FSTAT.ACT bit is set by an internal fault or when the external fault is ended. When the SEC_FSID.FEXT bit is set, the SEC_FSID.SID is cleared.	
		0	Fault Internal
		1	Fault External
7:0 (R/NW)	SID	Source ID. The SEC_FSID.SID identifies the fault assertion detected by the SEC fault interface. The SEC loads the SEC_FSID.SID field value when a system fault indication is asserted. The SEC fault interface does not change the SEC_FSID.SID value until the fault is no longer pending or active, as indicated by the SEC_FSTAT.PND bit and SEC_FSTAT.ACT bit being cleared in the SEC_FSTAT register.	

Fault System Reset Delay Register

The SEC fault system reset delay register (SEC_FSRDLY) contains the number (SEC_FSRDLY.COUNT field) of (SEC) clock periods for the delay from a fault becoming active to system reset request assertion, if enabled.

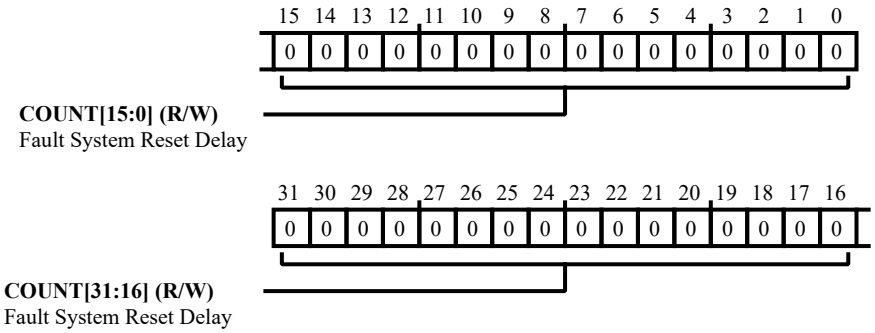


Figure 7-22: SEC_FSRDLY Register Diagram

Table 7-23: SEC_FSRDLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	COUNT	Fault System Reset Delay. The SEC_FSRDLY.COUNT bit field is the number of (SEC) clock periods for the delay from a fault becoming active to system reset request assertion.

Fault System Reset Delay Current Register

The SEC fault system reset delay current register (SEC_FSRDLY_CUR) contains the active count (SEC_FSRDLY_CUR.COUNT field) in (SEC) clock periods for the delay from fault active to system reset assertion, if enabled. The count is loaded from the SEC_FSRDLY register when a fault becomes active (SEC_FSTAT.ACT bit is set). The SEC decrements the value in SEC_FSRDLY_CUR each (SEC) clock cycle while the SEC_FSTAT.ACT bit is set.

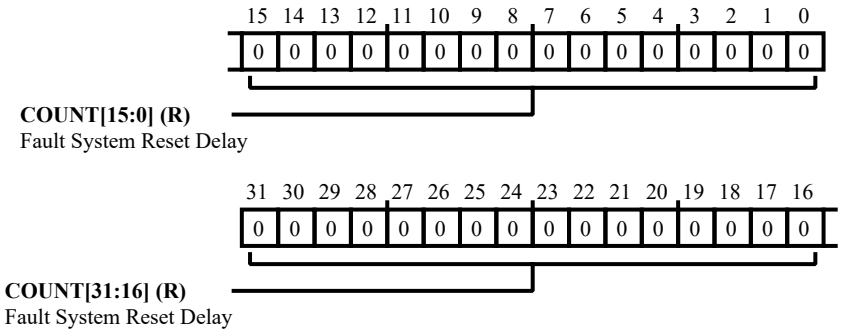


Figure 7-23: SEC_FSRDLY_CUR Register Diagram

Table 7-24: SEC_FSRDLY_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	COUNT	Fault System Reset Delay. The SEC_FSRDLY_CUR.COUNT bit field is the active count in (SEC) clock periods for the delay from fault active to system reset assertion.

Fault Status Register

The SEC fault status register (SEC_FSTAT) indicates the operational status of the SFI.

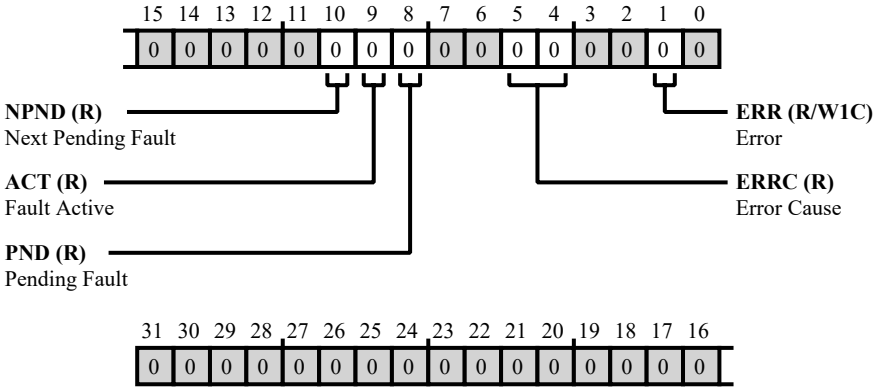


Figure 7-24: SEC_FSTAT Register Diagram

Table 7-25: SEC_FSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/NW)	NPND	Next Pending Fault. The SEC_FSTAT.NPND bit indicates that one or more sources have signaled fault assertion, but the input has not yet triggered the fault pending detection in the SEC fault interface. The SEC sets the SEC_FSTAT.NPND bit when the fault interface detects assertion of any enabled fault source input, while either the SEC_FSTAT.PND or SEC_FSTAT.ACT bits are set. The SEC clears the SEC_FSTAT.NPND bit when there are no fault sources waiting.	
		0	Not Pending
		1	Pending
9 (R/NW)	ACT	Fault Active. The SEC_FSTAT.ACT bit indicates that the SEC has received a fault source input, the current fault delay count (in the SEC_FDLY_CUR register) has expired, and the fault actions are enabled. The SEC also sets the SEC_FSTAT.ACT bit on fault input detection if the SEC_FCTL.FIEN bit is set. The SEC_FSTAT.ACT bit is cleared by writing the ID value of the asserted fault from SEC_FSID register to the SEC_FEND register.	
		0	No Fault
		1	Active Fault

Table 7-25: SEC_FSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	PND	Pending Fault. The SEC_FSTAT.PND bit indicates a fault source has signaled a fault assertion to the SEC, but the SEC has not yet triggered the event actions due to the delay selected with the SEC_FDLY register. The SEC fault interface sets the SEC_FSTAT.PND bit when the SEC_FSID is updated on assertion of a fault source input. The SEC_FSTAT.PND bit is only set when the SEC_FSTAT.ACT bit is cleared. The SEC updates the SEC_FSID register with the SID value when the SEC_FSTAT.PND bit is set. The SEC_FSTAT.PND bit is cleared *either* by the SEC fault interface when the current delay count in the SEC_FDLY_CUR register expires *or* by writing the SEC_FSID.SID field value (which indicates the ID of the asserted fault) to the SEC_FEND register.	
		0	Not Pending
		1	Pending
5:4 (R/NW)	ERRC	Error Cause. When the SEC_FSTAT.ERR bit is asserted, the SEC updates SEC_FSTAT.ERRC field to convey the interrupt source error type. When the error type is source overflow, the status indicates that a source signal assertion occurred or an SEC raise operation was attempted while pending was already set. The source overflow is detected when the source is set for edge only. When the error type is end error, the status indicates that an end was received for a source while the SEC_FSTAT.ACT bit was not set.	
		0	Source Overflow Error
		1	Reserved
		2	End Error
		3	Reserved
1 (R/W1C)	ERR	Error. The SEC_FSTAT.ERR bit indicates an SEC fault interface error. When SEC_FSTAT.ERR is set, the SEC updates the SEC_FSTAT.ERRC field to indicate the corresponding error cause. When multiple errors occur, the SEC_FSTAT register captures the status for the first error and does not capture subsequent errors until the status is cleared.	
		0	No Error
		1	Error Occurred

Global Control Register

The SEC global control register (SEC_GCTL) provides register locking, reset, and enable for the SEC module.

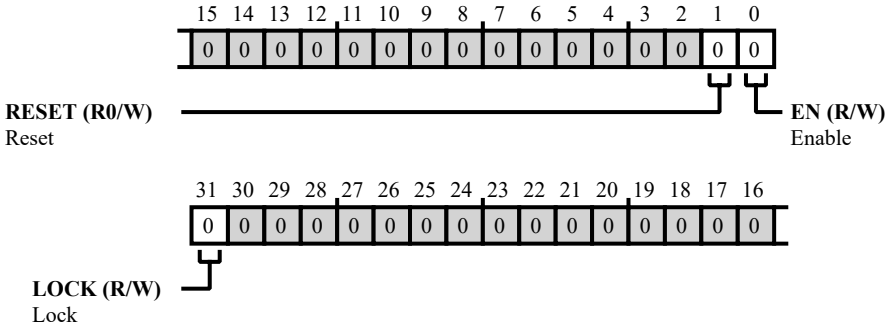


Figure 7-25: SEC_GCTL Register Diagram

Table 7-26: SEC_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_GCTL.LOCK bit is enabled, the SEC_GCTL register is read only.	
		0	Unlock
		1	Lock
1 (R0/W)	RESET	Reset. The SEC_GCTL.RESET bit is write-1-action and triggers a soft reset to all SEC registers.	
		0	No Action
		1	Reset
0 (R/W)	EN	Enable. The SEC_GCTL.EN bit is read/write and must be set for the SEC to begin/resume SEC operation with the current configuration and status. Clearing the SEC_GCTL.EN bit halts the execution of the SFI and all SCIs. All SSIs remain active, along with all error detection, without resetting status registers.	
		0	Disable
		1	Enable

Global Status Register

The SEC global status register (SEC_GSTAT) contains global status bits for the SEC.

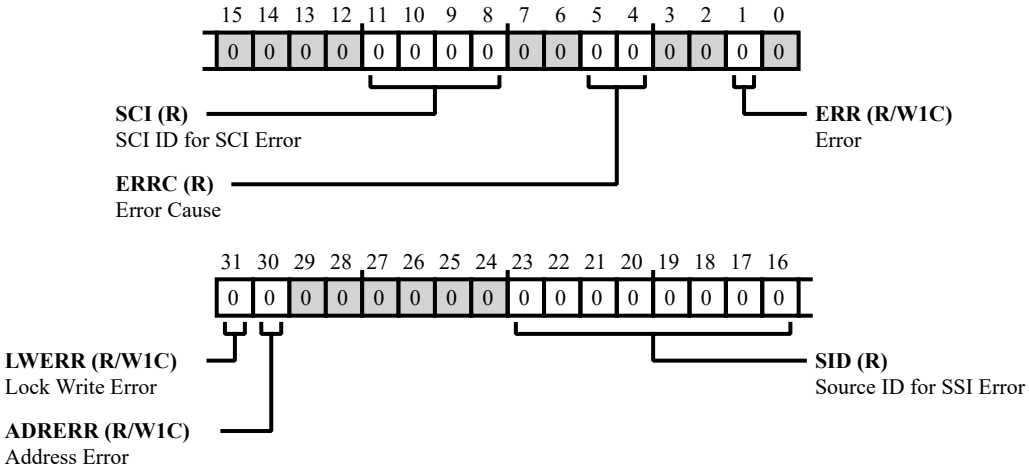


Figure 7-26: SEC_GSTAT Register Diagram

Table 7-27: SEC_GSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Lock Write Error. The SEC_GSTAT.LWERR bit indicates (when set) there was an attempted write to an SEC register while the SEC_GCTL.LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
30 (R/W1C)	ADRERR	Address Error. The SEC_GSTAT.ADRERR bit indicates that the SEC generated and address error. This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
23:16 (R/NW)	SID	Source ID for SSI Error. The SEC_GSTAT.SID bits indicate the source ID that generated the last SSI error conveyed in the SEC_GSTAT.ERRC field.	
11:8 (R/NW)	SCI	SCI ID for SCI Error. The SEC_GSTAT.SCI bits indicate the number for the specific SCI that generated the last SCI error conveyed in the SEC_GSTAT.ERRC field.	

Table 7-27: SEC_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	ERRC	Error Cause. When the SEC updates the `SEC_GSTAT.ERR` bit, the SEC updates the `SEC_GSTAT.ERRC` bits to indicate the error type. Note that for SCI errors, the error status represents an OR of all the errors from each SCI. Note that for SSI errors, the error status indicates an error is active for any SSI input. This error is an OR of all the interrupt source errors.	
		0	SFI Error
		1	SCI Error
		2	SSI Error
		3	Reserved
1 (R/W1C)	ERR	Error. The `SEC_GSTAT.ERR` bit indicates an error has occurred in the SEC. When the SEC asserts this bit (=1), the SEC updates the `SEC_GSTAT.ERRC` field to indicate the corresponding error cause. Even if multiple errors occur, only the first error is captured on assertion of this bit. This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred

Global Raise Register

The SEC global raise register (SEC_RAISE) contains a source ID event set-to-pending field (SEC_RAISE.SID). When a source ID value is written to this field, the SEC raises the source's event status to pending.

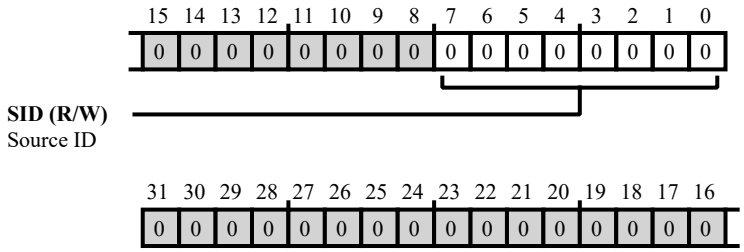


Figure 7-27: SEC_RAISE Register Diagram

Table 7-28: SEC_RAISE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	SID	Source ID. The SEC_RAISE.SID bit field is the source ID of event that is set to pending status.

Source Control Register n

The SEC source control register (SEC_SCTL[n]) contains control bits to configure the SEC event sources. This register controls the configuration of the corresponding SEC event source.

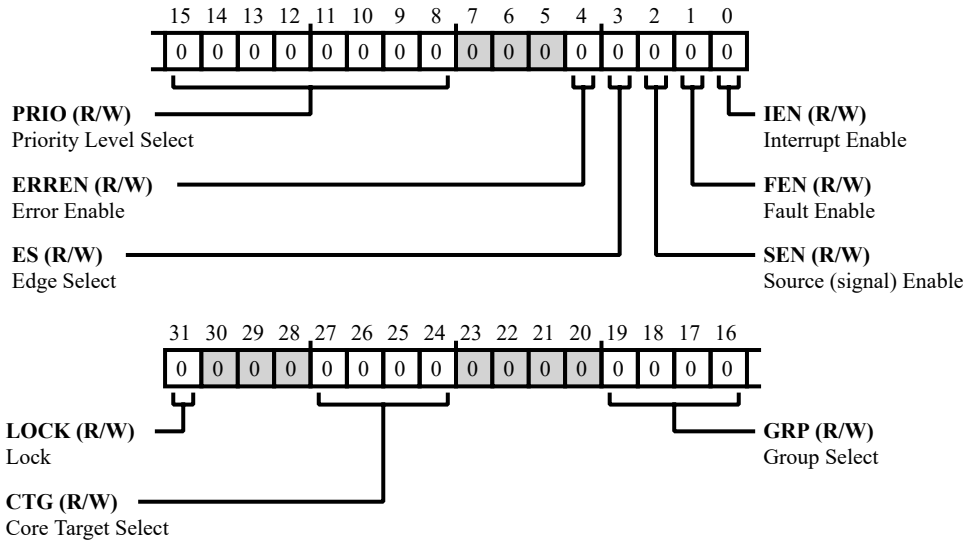


Figure 7-28: SEC_SCTL[n] Register Diagram

Table 7-29: SEC_SCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_SCTL[n].LOCK bit is enabled, the SEC_SCTL[n] register is read only.	
		0	Unlock
		1	Lock
27:24 (R/W)	CTG	Core Target Select. The SEC_SCTL[n].CTG bits selects the specific SEC core interface to which the interrupt is mapped. Each system interrupt is mapped uniquely to one specific SEC core interface and (as a result) to a specific core.	
		1	CORE1 SHARC0
		2	Reserved

Table 7-29: SEC_SCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GRP	Group Select. The SEC_SCTL[n].GRP bits each select a specific group for the interrupt. Each system interrupt can be assigned to any combination of groups supported by the SEC_SCTL[n].GRP field. For example, consider the situation where SEC_SCTL[n].GRP0 represents interrupt group 0, SEC_SCTL[n].GRP1 represents interrupt group 1, and so on. One group might be used for all enabled interrupts (for example, group 0) and an additional group might be used for all wakeup interrupts (for example, group 1). This approach supports a model of all interrupts and just the wakeup subset. Before going to idle or sleep, all non-wakeup interrupts can be masked off to allow only wakeup interrupts to be enabled for service. Selecting no group (all SEC_SCTL[n].GRP bits = 0) places the interrupt source in the category of "ungrouped".	
15:8 (R/W)	PRIO	Priority Level Select. The SEC_SCTL[n].PRIO bits sets the relative priority for an interrupt request. A pending interrupt request forwards its SEC_SCTL[n].PRIO value to the SEC core interface.	
4 (R/W)	ERREN	Error Enable. The SEC_SCTL[n].ERREN bit permits the SEC_SSTAT[n].ERR status bit to be set on error detection. If SEC_SCTL[n].ERREN is cleared, no errors are detected.	
		0	Disable
		1	Enable
3 (R/W)	ES	Edge Select. The SEC_SCTL[n].ES bit selects the operational and sensitivity mode of the SEC source interface input.	
		0	Level Sensitive
		1	Edge Sensitive
2 (R/W)	SEN	Source (signal) Enable. The SEC_SCTL[n].SEN bit controls whether the system event source input signal may affect the pending status of the source. Clearing the SEC_SCTL[n].SEN bit disables the source input signal from affecting the pending status. Setting SEC_SCTL[n].SEN enables the source input signal to affect the pending status.	
		0	Disable
		1	Enable

Table 7-29: SEC_SCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	FEN	Fault Enable. The `SEC_SCTL[n].FEN` bit controls whether the SEC may forward an interrupt request to the SEC fault interface as a fault source. This bit does not affect the ability of an interrupt source to set an interrupt as pending. The `SEC_SCTL[n].FEN` bit only affects whether the pending request may be forwarded to the SEC fault interface.	
		0	Disable
		1	Enable
0 (R/W)	IEN	Interrupt Enable. The `SEC_SCTL[n].IEN` bit controls whether the SEC may forward an interrupt request to a core for servicing. This bit does not affect the ability of an interrupt source to set an interrupt as pending.	
		0	Disable
		1	Enable

Source Status Register n

The SEC event source status register (SEC_SSTAT[n]) contains bits indicating the status of the corresponding event source n. An event source may be: pending, active, active and pending, or neither pending nor active.

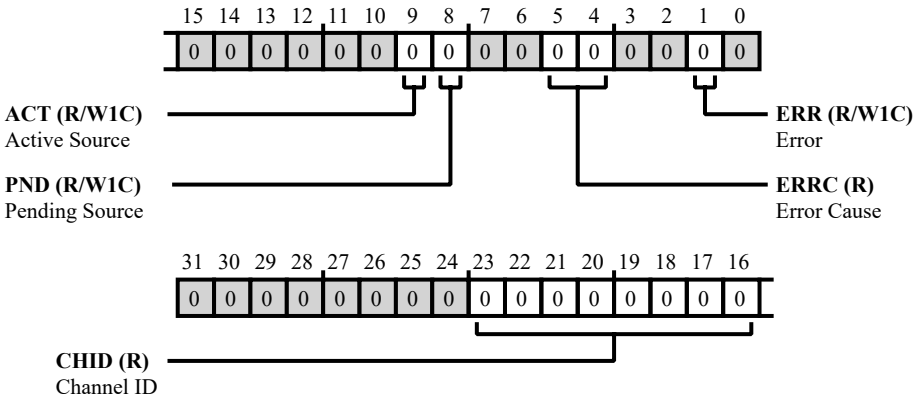


Figure 7-29: SEC_SSTAT[n] Register Diagram

Table 7-30: SEC_SSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/NW)	CHID	Channel ID. The SEC_SSTAT[n].CHID bits indicate the ID of the specific source (from a set of sources sharing one SEC source interface input) that asserted the SEC source interface input. An SEC source interface input may support multiple system sources, in which case the assertion must be qualified by an identifier to determine the channel that generated the assertion. The SEC_SSTAT[n].CHID field provides this value in the form of a numeric reference that is mapped to a specific interrupt source. The prioritization for simultaneously asserted sources is according to ID, with 0 being the highest priority. The SEC_SSTAT[n].CHID is captured when the SEC source interface input is acknowledged.	
9 (R/W1C)	ACT	Active Source. The SEC_SSTAT[n].ACT bit indicates the source has been accepted by a core for servicing, but the service is not yet complete. An SEC_SSTAT[n].ACT bit is set by the SEC when the specific system interrupt is acknowledged by the core through the SEC core interface. An SEC_SSTAT[n].ACT bit is cleared by the SEC when the core provides interrupt service end indication for the specific system interrupt through the SEC core interface.	
		0	Not Active
		1	Active

Table 7-30: SEC_SSTAT[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	PND	Pending Source. The SEC_SSTAT[n].PND bit indicates the source has signaled an event request, but the event request has not been (or is not currently being) serviced. A SEC_SSTAT[n].PND bit is set by the SEC on detection of an assertion of the corresponding system source input. A SEC_SSTAT[n].PND bit is cleared by the SEC when the specific system event is acknowledged by the core through the SEC core interface or by a W1C operation.	
		0	Not Pending
		1	Pending
5:4 (R/NW)	ERRC	Error Cause. When the SEC_SSTAT[n].ERR bit is asserted, the SEC updates SEC_SSTAT[n].ERRC field to convey the interrupt source error type. When the error type is source overflow, the status indicates that a source signal assertion occurred or an SEC raise operation was attempted while pending was already set. The source overflow is detected when the source is set for edge only. When the error type is end error, the status indicates that an end was received for a source while the SEC_SSTAT[n].ACT bit was not set.	
		0	Source Overflow Error
		1	Reserved
		2	End Error
		3	Reserved
1 (R/W1C)	ERR	Error. The SEC_SSTAT[n].ERR bit indicates an error for a specific system interrupt source. When the SEC_SSTAT[n].ERR bit is set, the SEC updates the SEC_SSTAT[n].ERRC field to the value of the corresponding error cause. Even if multiple errors occur, only the first error is captured on assertion of the SEC_SSTAT[n].ERR bit.	
		0	No Error
		1	Error Occurred

GIC Overview

The generic interrupt controller (GIC) provides an interface to the dual Cortex A55 core in the processor and collects up to 419 interrupt requests from all processor system sources. In addition, the GIC also supports sixteen software generated interrupts and nine private peripheral interrupts that are internal to the Cortex A55 core and not connected to the SECs. All GIC interrupts are also connected to the SEC in the same order.

Each interrupt can be configured as a normal or a secure interrupt. Software force registers and software priority masking are also supported. The “Register Descriptions” section in this chapter provides brief descriptions of these Arm-based registers. For complete information refer to the *Arm® Generic Interrupt Controller Architecture version 3.0 and 4.0 Architecture Specification*.

GIC Functional Description

The GIC interrupt controller consists of three interfaces;

- Distributor interface
- Redistributor interface
- CPU interface

The distributor and the redistributor interfaces are used to configure interrupts; the CPU interface is used to handle interrupts.

GIC Distributer Interface

The distributor registers are memory-mapped. They are used to configures SPIs. The distributor block provides a programming interface to perform the following tasks.

- Interrupt prioritization and distribution of SPIs
- Enable and disable SPIs
- Set the priority level of each SPI
- Set each SPI to be level-sensitive or edge-triggered
- Assign each SPI to an interrupt group
- Control the active and pending state of SPIs.

GIC Redistributer Interface

There is one redistributor per connected core. The redistributors provide a programming interface to perform the following tasks.

- Enable and disable SGIs and PPIs
- Set the priority level of SGIs and PPIs
- Set each PPI to be level-sensitive or edge-triggered
- Assign each SGI and PPI to an interrupt group
- Control the state of SGIs and PPIs.

GIC CPU Interface

Each core contains a CPU interface that includes system registers that are used during interrupt handling. The CPU interfaces provide a programming interface to perform the following tasks.

- Provide general control and configuration to enable interrupt handling
- Acknowledge an interrupt
- Perform a priority drop and deactivation of interrupts
- Set an interrupt priority mask for the processor
- Define the preemption policy for the processor
- Determine the highest priority pending interrupt for the processor

In Arm CoreLink GICv3, the CPU interface registers are accessed as system registers: ICC_*_ELn, where “n” specifies the Exception level: EL1-EL3.

Refer to the GIC Register Descriptions in this chapter for details on the distributor and the redistributor interface registers. Refer to the *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and 4.0* for details on the CPU interface registers.

GIC Block Diagram

The ***GIC Block Diagram*** shows the event management architecture.

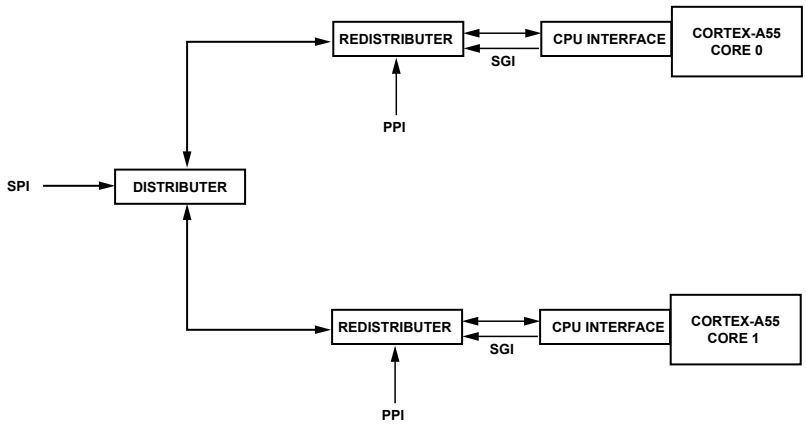


Figure 7-30: GIC Block Diagram

GIC Performance Monitoring Unit (GIC PMU)

The GIC contains a performance monitoring unit (PMU) for counting key GIC events from the distributor. Redistributor events are not tracked by the PMU. The delivery of PPI and SGI interrupts are counted by recording calls to the core interrupt service routine.

The PMU has five counters with snapshot capability and overflow interrupt. For all the GIC events, PMU count configuration and register information, please refer to the *Arm® CoreLink™ GIC 600 Generic Interrupt Controller Technical Reference Manual*.

Overflow interrupt

The overflow interrupt is enabled on a per counter basis by enabling the relevant bit of the register. Similarly, the overflow interrupt enable is disabled by corresponding writes to the register. When enabled, the interrupt activates when there is a write to the register for any counter or overflow on any enabled counter.

Snapshot

Each PMU counter, , has a corresponding snapshot register. On a snapshot event, all five counters are copied to their backup registers so that all consistent data is copied out over a longer period.

The snapshot events are as follows:

- Write of 1 to the GICP_ bit
- An overflow of an enabled counter when the bit is set

8 Trigger Routing Unit (TRU)

The TRU provides system-level sequence control without core intervention. The TRU maps trigger generators to trigger receivers. Receiver endpoints can be configured to respond to triggers in various ways. Multiple TRUs may be provided in a multiprocessor system to create a trigger network. Common applications enabled by the TRU include:

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes
- Software triggering
- Synchronization of concurrent activities

TRU Features

The TRU supports the following features:

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes. Once a DMA channel completes data transfer, it can act as a trigger generator and signal an internal trigger pulse to the programmed trigger receiver which can also be another DMA channel. The receiver trigger connected to the DMA channel kicks off the DMA transfer automatically. None of this requires core intervention once the initialization is done.
- Software triggers. The best use of triggers is to minimize core intervention. It is also possible to initiate a trigger pulse to a trigger receiver, in the software.
- Synchronization of concurrent activities. A single trigger generator can initiate a trigger pulse to multiple trigger receiver so that several system level activities can be synchronized on an internally or externally generated event.
- Configuration protection through register-level lock bits and global lock indication

TRU Functional Description

The following sections provide a description of the TRU.

ADSP-2184x TRU Register List

The Trigger Routing Unit (TRU) provides simple sequence control of distributed modules without the penalties associated with core intervention (for example, interrupt overhead). The TRU receives trigger inputs from all trigger generator inputs (MTI) and the TRU trigger receiver register (TRU_GEN). Based on these inputs, the TRU logic generates trigger outputs that initiate receiver operations in the processor core and peripherals. A set of registers governs TRU operations. For more information on TRU functionality, see the TRU register descriptions.

Table 8-1: ADSP-2184x TRU Register List

Name	Description
TRU_ERRADDR	Error Address Register
TRU_GCTL	Global Control Register
TRU_GEN	Generator Trigger Register
TRU_RSR[n]	Receiver Select Register
TRU_STAT	Status Information Register

ADSP-2184x TRU Interrupt List

Table 8-2: ADSP-2184x TRU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
349	TRU0_RCV0	TRU0 Interrupt 0 - Core 0	Edge	
350	TRU0_RCV1	TRU0 Interrupt 1 - Core 0	Edge	
351	TRU0_RCV2	TRU0 Interrupt 2 - Core 0	Edge	
352	TRU0_RCV3	TRU0 Interrupt 3 - Core 0	Edge	
353	TRU0_RCV4	TRU0 Interrupt 4	Edge	
354	TRU0_RCV5	TRU0 Interrupt 5	Edge	
355	TRU0_RCV6	TRU0 Interrupt 6	Edge	
356	TRU0_RCV7	TRU0 Interrupt 7	Edge	
357	TRU0_RCV8	TRU0 Interrupt 8	Edge	
358	TRU0_RCV9	TRU0 Interrupt 9	Edge	
359	TRU0_RCV10	TRU0 Interrupt 10	Edge	
360	TRU0_RCV11	TRU0 Interrupt 11	Edge	

ADSP-2184x Trigger List

Table 8-3: ADSP-2184x Trigger List Generators

Trigger ID	Name	Description	Sensitivity
0		Reserved	
1	CANFD0_IPD_REQ	CANFD0 DMA Request	None
2	CANFD1_IPD_REQ	CANFD1 DMA Request	None
3	CGU0_EVT	CGU0 Event	Edge
4	CGU1_EVT	CGU1 Event	Edge
5	CNT0_STAT	CNT0 Status	Level
6	CNT0_UD	CNT0 Count Up and Direction	Level
7	CNT0_DG	CNT0 Count Down and Gate	Level
8	CNT0_TO	CNT0 Output to Timer Block	Level
9	C1_SID_ACK	Core 1 System Interface Disable Acknowledge	
10	C1_IDMA	SHARC-FX (Core 1) iDMA Trigger Initiator	
11	MDMA0_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	
12	MDMA0_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	
13	MDMA1_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	
14	MDMA1_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	
15	MDMA4_SRC	Standard BW DMA Channel 0 Source (CRC IN)	
16	MDMA4_DST	Standard BW DMA Channel 0 Source (CRC OUT)	
17	MDMA5_SRC	Standard BW DMA Channel 0 Source (CRC IN)	
18	MDMA5_DST	Standard BW DMA Channel 0 Source (CRC OUT)	
19	ECT_EVT0	Embedded Cross Trigger Event n	
20	ECT_EVT1	Embedded Cross Trigger Event n	
21	ECT_EVT2	Embedded Cross Trigger Event n	
22	ECT_EVT3	Embedded Cross Trigger Event n	
23	ECT_EVT4	Embedded Cross Trigger Event n	

Table 8-3: ADSP-2184x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
24	ECT_EVT5	Embedded Cross Trigger Event n	
25	ECT_EVT6	Embedded Cross Trigger Event n	
26	ECT_EVT7	Embedded Cross Trigger Event n	
27	EMAC0_MCGR_DMA_REQ0	EMAC0 MCGR DMA request	Edge
28	EMAC0_MCGR_DMA_REQ1	EMAC0 MCGR DMA request	Edge
29	EMAC0_MCGR_DMA_REQ2	EMAC0 MCGR DMA request	Edge
30	EMAC0_MCGR_DMA_REQ3	EMAC0 MCGR DMA request	Edge
31	EMAC0_DMA0_RX	EMAC0 DMAn Rx Channel Interrupt	None
32	EMAC0_DMA1_RX	EMAC0 DMAn Rx Channel Interrupt	None
33	EMAC0_DMA2_RX	EMAC0 DMAn Rx Channel Interrupt	None
34	EMAC0_DMA3_RX	EMAC0 DMAn Rx Channel Interrupt	None
35	EMAC0_DMA4_RX	EMAC0 DMAn Rx Channel Interrupt	None
36	EMAC0_DMA5_RX	EMAC0 DMAn Rx Channel Interrupt	None
37	EMAC0_DMA6_RX	EMAC0 DMAn Rx Channel Interrupt	None
38	EMAC0_DMA7_RX	EMAC0 DMAn Rx Channel Interrupt	None
51	EMDMA0_DONE	EMDMA0 DMA Done	Edge
52	EMDMA1_DONE	EMDMA1 DMA Done	Edge
53	FIR0_DMA	FIR0 Core 1 DMA	Edge
54	FIR1_DMA	FIR1 Core 1 DMA	Edge
55	FRACNPLL0_FRACPLL_TRIGM	FRACNPLL0 Frac Pll Lock Trigger	Edge
56	FRACNPLL1_FRACPLL_TRIGM	FRACNPLL1 Frac Pll Lock Trigger	Edge
57	HADC0_EOC	HADC0 HADC0 End of Conversion	Edge
58	HSM_MISC_HSM_ALRM_0	HSM_MISC Input Interrupt 0 to HSM from TRU Responder Triggers	None
59	HSM_MISC_HSM_ALRM_1	HSM_MISC Input Interrupt 1 to HSM from TRU Responder Triggers	None
60	IIR0_DMA	IIR0 Core 1 DMA	Edge
61	IIR1_DMA	IIR1 Core 1 DMA	Edge
62	IIR2_DMA	IIR2 Core 1 DMA	Edge
63	IIR3_DMA	IIR3 Core 1 DMA	Edge

Table 8-3: ADSP-2184x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
64	L2CTL0_EVT	L2CTL0 L2 Memory Event	Level
65	L2CTL1_EVT	L2CTL1 L2 Memory Event	Level
67	LP0_DMA	LP0 DMA Channel	
68	LP1_DMA	LP1 DMA Channel	
69	Cn_EVENTIACK	Event Input Request Acknowledge	
70	Cn_EVENTOREQ	Event Output Request	
71	A55_COREP_ACCEPT0	MISC_A55 A55[0] Preq Accept	Edge
72	A55_COREP_DENY0	MISC_A55 A55[0] Preq Deny	Edge
73	CLUSTERP_ACCEPT	MISC_A55 Cluster Preq Accept	Edge
74	CLUSTERP_DENY	MISC_A55 Cluster Preq Deny	Edge
75	COREP_ACCEPT2	MISC_A55 A55[2] Preq Accept	Edge
76	COREP_DENY2	MISC_A55 A55[2] Preq Deny	Edge
77	MDMA2_SRC	Enh BW DMA Channel 0	
78	MDMA2_DST	Enh BW DMA Channel 1	
79	MDMA3_SRC	Max BW DMA Channel 0	
80	MDMA3_DST	Max BW DMA Channel 1	
81	MDMA6_SRC	Enh BW DMA Channel 0	
82	MDMA6_DST	Enh BW DMA Channel 1	
83	MDMA7_SRC	Max BW DMA Channel 0	
84	MDMA7_DST	Max BW DMA Channel 1	
85	MEC1_EEIRQ0	MEC1 ECC Error Interrupt Request	Level
86	MEC1_MEIRQ2	MEC1 Memory Error Interrupt Request	Level
87	MEC1_PEIRQ0	MEC1 Parity Error Interrupt Request	Level
88	MEC2_EEIRQ0	MEC2 ECC Error Interrupt/Trigger Request Bus	Level
89	MEC2_MEIRQ2	MEC2 Memory Error Interrupt Request	Level
90	MEC2_PEIRQ0	MEC2 Parity Error Interrupt/Trigger Request Bus	Level
91	MEC1_MEIRQ1	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level
92	MEC2_MEIRQ1	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level

Table 8-3: ADSP-2184x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
93	MEC1_MEIRQ0	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level
94	MEC2_MEIRQ0	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level
95	MEC0_EEIRQ0	MEC0 ECC Error Interrupt/Trigger Request Bus	Level
96	MEC0_MEIRQ2	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level
97	MEC0_PEIRQ0	MEC0 Parity Error Interrupt Request	Level
98	MEC0_MEIRQ1	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level
99	MEC0_MEIRQ0	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level
100		Reserved	
101		Reserved	
102		Reserved	
103		Reserved	
104		Reserved	
105		Reserved	
106		Reserved	
107		Reserved	
108	PINT0_BLOCK	PINT0 Pin Interrupt Block	Level
109	PINT1_BLOCK	PINT1 Pin Interrupt Block	Level
110	PINT2_BLOCK	PINT2 Pin Interrupt Block	Level
111	PINT3_BLOCK	PINT3 Pin Interrupt Block	Level
112	PINT4_BLOCK	PINT4 Pin Interrupt Block	Level
113	PINT5_BLOCK	PINT5 Pin Interrupt Block	Level
114	PINT6_BLOCK	PINT6 Pin Interrupt Block	Level
115	PINT7_BLOCK	PINT7 Pin Interrupt Block	Level
116	SEC0_FAULT	SEC0 Fault	Edge
117	SPI0_TXDMA	SPI0 TX DMA Channel	Edge
118	SPI0_RXDMA	SPI0 RX DMA Channel	Edge
119	SPI1_TXDMA	SPI1 TX DMA Channel	Edge
120	SPI1_RXDMA	SPI1 RX DMA Channel	Edge

Table 8-3: ADSP-2184x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
121	SPI2_TXDMA	SPI2 TX DMA Channel	Edge
122	SPI2_RXDMA	SPI2 RX DMA Channel	Edge
123	SPI5_TXDMA	SPI5 TX DMA Channel	Edge
124	SPI5_RXDMA	SPI5 RX DMA Channel	Edge
125	SPORT0_A_DMA	SPORT0 Channel A DMA	Edge
126	SPORT0_B_DMA	SPORT0 Channel B DMA	Edge
127	SPORT1_A_DMA	SPORT1 Channel A DMA	Edge
128	SPORT1_B_DMA	SPORT1 Channel B DMA	Edge
129	SPORT2_A_DMA	SPORT2 Channel A DMA	Edge
130	SPORT2_B_DMA	SPORT2 Channel B DMA	Edge
131	SPORT3_A_DMA	SPORT3 Channel A DMA	Edge
132	SPORT3_B_DMA	SPORT3 Channel B DMA	Edge
133	SPORT4_A_DMA	SPORT4 Channel A DMA	Edge
134	SPORT4_B_DMA	SPORT4 Channel B DMA	Edge
135	SPORT5_A_DMA	SPORT5 Channel A DMA	Edge
136	SPORT5_B_DMA	SPORT5 Channel B DMA	Edge
137	SPORT6_A_DMA	SPORT6 Channel A DMA	Edge
138	SPORT6_B_DMA	SPORT6 Channel B DMA	Edge
139	SPORT7_A_DMA	SPORT7 Channel A DMA	Edge
140	SPORT7_B_DMA	SPORT7 Channel B DMA	Edge
141	DAI0_GBL_SPORT_TRG_O0	DAI0 SPORT GROUP0 Trigger Output	None
142	DAI0_GBL_SPORT_TRG_O1	DAI0 SPORT GROUP1 Trigger Output	None
143	DAI1_GBL_SPORT_TRG_O0	DAI1 SPORT GROUP2 Trigger Output	None
144	DAI1_GBL_SPORT_TRG_O1	DAI1 SPORT Group3 Trigger Output	None
145		Reserved	
146	SWU1_EVT	SWU1 Event	None
147	SWU2_EVT	SWU2 Event	None
148	SWU7_EVT	SWU7 Event	None

Table 8-3: ADSP-2184x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
149	SWU8_EVT	SWU8 Event	None
150	SWU11_EVT	SWU11 Event	None
151	SWU12_EVT	SWU12 Event	None
152	SWU13_EVT	SWU13 Event	None
153	SWU3_EVT	SWU3 Event	None
154	SWU4_EVT	SWU4 Event	None
155	SWU5_EVT	SWU5 Event	None
156		Reserved	
157	SWU1_DBG	SWU1 Debug	Edge
158	SWU2_DBG	SWU2 Debug	Edge
159	SWU7_DBG	SWU7 Debug	Edge
160	SWU8_DBG	SWU8 Debug	Edge
161	SWU11_DBG	SWU11 Debug	Edge
162	SWU12_DBG	SWU12 Debug	Edge
163	SWU13_DBG	SWU13 Debug	Edge
164	SWU3_DBG	SWU3 Debug	Edge
165	SWU4_DBG	SWU4 Debug	Edge
166	SWU5_DBG	SWU5 Debug	Edge
167	SOFT0_MST	Software-driven Trigger 0	
168	SOFT1_MST	Software-driven Trigger 1	
169	SOFT2_MST	Software-driven Trigger 2	
170	SOFT3_MST	Software-driven Trigger 3	
171	SOFT4_MST	Software-driven Trigger 4	
172	SOFT5_MST	Software-driven Trigger 5	
173	TIMER0_TMR00_GEN	TIMER0 Timer 0 Trigger Initiator	Edge
174	TIMER0_TMR01_GEN	TIMER0 Timer 1 Trigger Initiator	Edge
175	TIMER0_TMR02_GEN	TIMER0 Timer 2 Trigger Initiator	Edge
176	TIMER0_TMR03_GEN	TIMER0 Timer 3 Trigger Initiator	Edge
177	TIMER0_TMR04_GEN	TIMER0 Timer 4 Trigger Initiator	Edge
178	TIMER0_TMR05_GEN	TIMER0 Timer 5 Trigger Initiator	Edge
179	TIMER0_TMR06_GEN	TIMER0 Timer 6 Trigger Initiator	Edge

Table 8-3: ADSP-2184x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
180	TIMER0_TMR07_GEN	TIMER0 Timer 7 Trigger Initiator	Edge
181	TIMER0_TMR08_GEN	TIMER0 Timer 8 Trigger Initiator	Edge
182	TIMER0_TMR09_GEN	TIMER0 Timer 9 Trigger Initiator	Edge
183	TIMER0_TMR10_GEN	TIMER0 Timer 10 Trigger Initiator	Edge
184	TIMER0_TMR11_GEN	TIMER0 Timer 11 Trigger Initiator	Edge
185	TIMER0_TMR12_GEN	TIMER0 Timer 12 Trigger Initiator	Edge
186	TIMER0_TMR13_GEN	TIMER0 Timer 13 Trigger Initiator	Edge
187	TIMER0_TMR14_GEN	TIMER0 Timer 14 Trigger Initiator	Edge
188	TIMER0_TMR15_GEN	TIMER0 Timer 15 Trigger Initiator	Edge
189	TMU0_FAULT	TMU0 TM0 Fault Event	
190	TMU0_ALERT	TMU0 TM0 Alert Event	
191	UART0_TXDMA	UART0 Transmit DMA	Edge
192	UART0_RXDMA	UART0 Receive DMA	Edge
193	UART1_TXDMA	UART1 Transmit DMA	Edge
194	UART1_RXDMA	UART1 Receive DMA	Edge
195	UART2_TXDMA	UART2 Transmit DMA	Edge
196	UART2_RXDMA	UART2 Receive DMA	Edge
197	WDOG0_EXP	WDOG0 Expiration	Level
198	WDOG1_EXP	WDOG1 Expiration	Level
199	WDOG2_EXP	WDOG2 Expiration	Level
200	WDOG3_EXP	WDOG3 Expiration	Level

Table 8-4: ADSP-2184x Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
0		Reserved	
1		Reserved	
2	CNT0_UD	CNT0 Count Up and Direction	Pulse
3	CNT0_DG	CNT0 Count Down and Gate	Pulse
11	MDMA0_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	Pulse
12	MDMA0_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
13	MDMA1_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	Pulse
14	MDMA1_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	Pulse
15	MDMA4_SRC	Standard BW DMA Channel 0 Source (CRC IN)	Pulse
16	MDMA4_DST	Standard BW DMA Channel 0 Source (CRC OUT)	Pulse
17	MDMA5_SRC	Standard BW DMA Channel 1 Source (CRC IN)	Pulse
18	MDMA5_DST	Standard BW DMA Channel 1 Source (CRC OUT)	Pulse
19	ECT_EVT0	Embedded Cross Trigger Event n	Pulse
20	ECT_EVT1	Embedded Cross Trigger Event n	Pulse
21	ECT_EVT2	Embedded Cross Trigger Event n	Pulse
22	ECT_EVT3	Embedded Cross Trigger Event n	Pulse
23	ECT_EVT4	Embedded Cross Trigger Event n	Pulse
24	ECT_EVT5	Embedded Cross Trigger Event n	Pulse
25	ECT_EVT6	Embedded Cross Trigger Event n	Pulse
26	EMAC0_DMA0_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
27	EMAC0_DMA1_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
28	EMAC0_DMA2_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
29	EMAC0_DMA3_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
30	EMAC0_DMA4_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
31	EMAC0_DMA5_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
32	EMAC0_DMA6_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
33	EMAC0_DMA7_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
34	EMAC0_DMA0_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
35	EMAC0_DMA1_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
36	EMAC0_DMA2_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
37	EMAC0_DMA3_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
38	EMAC0_DMA4_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
39	EMAC0_DMA5_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
40	EMAC0_DMA6_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
41	EMAC0_DMA7_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
42	EMAC0_DMA0_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
43	EMAC0_DMA1_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
44	EMAC0_DMA2_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
45	EMAC0_DMA3_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
46	EMAC0_DMA4_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
47	EMAC0_DMA5_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
48	EMAC0_DMA6_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
49	EMAC0_DMA7_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
50	EMAC0_DMA0_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
51	EMAC0_DMA1_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
52	EMAC0_DMA2_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
53	EMAC0_DMA3_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
54	EMAC0_DMA4_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
55	EMAC0_DMA5_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
56	EMAC0_DMA6_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
57	EMAC0_DMA7_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
58	EMAC0_MCGR_DMA_ACK0	EMAC0 MCGR DMA acknowledge	Pulse
59	EMAC0_MCGR_DMA_ACK1	EMAC0 MCGR DMA acknowledge	Pulse
60	EMAC0_MCGR_DMA_ACK2	EMAC0 MCGR DMA acknowledge	Pulse
61	EMAC0_MCGR_DMA_ACK3	EMAC0 MCGR DMA acknowledge	Pulse
98	FIR0_TRGI	FIR0 Core 1 FIR Wait on Trigger Input	Pulse
99	FIR1_TRGI	FIR1 Core 1 FIR Wait on Trigger Input	Pulse
100	FRACNPLL0_FRACPLL_TRIGS	FRACNPLL0 Frac Pll Start Trigger	Pulse
101	FRACNPLL1_FRACPLL_TRIGS	FRACNPLL1 Frac Pll Start Trigger	Pulse
102	IIR0_TRGI	IIR0 Core 1 IIR Wait on Trigger Input	Pulse
103	IIR1_TRGI	IIR1 Core 1 IIR Wait on Trigger Input	Pulse
104	IIR2_TRGI	IIR2 Core 1 IIR Wait on Trigger Input	Pulse
105	IIR3_TRGI	IIR3 Core 1 IIR Wait on Trigger Input	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
106	HSM_MISC_HSM_ALRM_0	HSM_MISC Input Interrupt 0 to HSM from TRU Responder Triggers	Pulse
107	HSM_MISC_HSM_ALRM_1	HSM_MISC Input Interrupt 1 to HSM from TRU Responder Triggers	Pulse
108	HSM_MISC_HSM_ALRM_2	HSM_MISC Input Interrupt 2 to HSM from TRU Responder Triggers	Pulse
109	HSM_MISC_HSM_ALRM_3	HSM_MISC Input Interrupt 3 to HSM from TRU Responder Triggers	Pulse
110	HSM_MISC_HSM_ALRM_4	HSM_MISC Input Interrupt 4 to HSM from TRU Responder Triggers	Pulse
111	HSM_MISC_HSM_ALRM_5	HSM_MISC Input Interrupt 5 to HSM from TRU Responder Triggers	Pulse
112	HSM_MISC_HSM_ALRM_6	HSM_MISC Input Interrupt 6 to HSM from TRU Responder Triggers	Pulse
113	HSM_MISC_HSM_ALRM_7	HSM_MISC Input Interrupt 7 to HSM from TRU Responder Triggers	Pulse
114	LP0_DMA	LP0 DMA Channel	Pulse
115	LP1_DMA	LP1 DMA Channel	Pulse
116	Cn_EVENTIREQ	Event Input Request for Core Wake-up From WFE State	Pulse
117	Cn_EVENTOACK	Event Output Acknowledgment for Core to Wakeup	Pulse
120	MDMA2_SRC	Enh BW DMA Channel 0	Pulse
121	MDMA2_DST	Enh BW DMA Channel 1	Pulse
122	MDMA3_SRC	Max BW DMA Channel 0	Pulse
123	MDMA3_DST	Max BW DMA Channel 1	Pulse
124	MDMA6_SRC	Enh BW DMA Channel 0	Pulse
125	MDMA6_DST	Enh BW DMA Channel 1	Pulse
126	MDMA7_SRC	Max BW DMA Channel 0	Pulse
127	MDMA7_DST	Max BW DMA Channel 1	Pulse
128	PCG0_HWA	PCG0 PCG-A Hardware trigger control	Pulse
129	PCG0_HWB	PCG0 PCG-B Hardware trigger control	Pulse
130	PCG0_HWC	PCG0 PCG-C Hardware trigger control	Pulse
131	PCG0_HWD	PCG0 PCG-D Hardware trigger control	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
132	PCG0_HWE	PCG0 PCG-E Hardware trigger control	Pulse
133	PCG0_HWF	PCG0 PCG-F Hardware trigger control	Pulse
134	PCG0_HWG	PCG0 PCG-G Hardware trigger control	Pulse
135	PCG0_HWH	PCG0 PCG-H Hardware trigger control	Pulse
136	PORTA_TOGGLE	Port Toggle Trigger	Pulse
137	PORTB_TOGGLE	Port Toggle Trigger	Pulse
138	PORTC_TOGGLE	Port Toggle Trigger	Pulse
139	PORTD_TOGGLE	Port Toggle Trigger	Pulse
140	PORTE_TOGGLE	Port Toggle Trigger	Pulse
141	PORTF_TOGGLE	Port Toggle Trigger	Pulse
142	PORTG_TOGGLE	Port Toggle Trigger	Pulse
143	PORTH_TOGGLE	Port Toggle Trigger	Pulse
145	PWM0_TRIP	PWM0 PWM Trip1B Trigger In	Pulse
146	PWM0_SYNC	PWM0 PWMTMR Grouped	Pulse
147	RCU0_SYSRST0	RCU0 System Reset Receiver 0	Pulse
148	RCU0_SYSRST1	RCU0 System Reset Receiver 1	Pulse
149	SPI0_TXDMA	SPI0 TX DMA Channel	Pulse
150	SPI0_RXDMA	SPI0 RX DMA Channel	Pulse
151	SPI1_TXDMA	SPI1 TX DMA Channel	Pulse
152	SPI1_RXDMA	SPI1 RX DMA Channel	Pulse
153	SPI5_TXDMA	SPI5 TX DMA Channel	Pulse
154	SPI5_RXDMA	SPI5 RX DMA Channel	Pulse
155	SPI2_TXDMA	SPI2 TX DMA Channel	Pulse
156	SPI2_RXDMA	SPI2 RX DMA Channel	Pulse
157	SPORT0_A_DMA	SPORT0 Channel A DMA	Pulse
158	SPORT0_B_DMA	SPORT0 Channel B DMA	Pulse
159	SPORT1_A_DMA	SPORT1 Channel A DMA	Pulse
160	SPORT1_B_DMA	SPORT1 Channel B DMA	Pulse
161	SPORT2_A_DMA	SPORT2 Channel A DMA	Pulse
162	SPORT2_B_DMA	SPORT2 Channel B DMA	Pulse
163	SPORT3_A_DMA	SPORT3 Channel A DMA	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
164	SPORT3_B_DMA	SPORT3 Channel B DMA	Pulse
165	SPORT4_A_DMA	SPORT4 Channel A DMA	Pulse
166	SPORT4_B_DMA	SPORT4 Channel B DMA	Pulse
167	SPORT5_A_DMA	SPORT5 Channel A DMA	Pulse
168	SPORT5_B_DMA	SPORT5 Channel B DMA	Pulse
169	SPORT6_A_DMA	SPORT6 Channel A DMA	Pulse
170	SPORT6_B_DMA	SPORT6 Channel B DMA	Pulse
171	SPORT7_A_DMA	SPORT7 Channel A DMA	Pulse
172	SPORT7_B_DMA	SPORT7 Channel B DMA	Pulse
173	DAI0_GBL_SPORT_TRG_I0	DAI0 SPORT GROUP0 Trigger Input	Pulse
174	DAI0_GBL_SPORT_TRG_I1	DAI0 SPORT GROUP1 Trigger Input	Pulse
175	DAI1_GBL_SPORT_TRG_I0	DAI1 SPORT GROUP2 Trigger Input	Pulse
176	DAI1_GBL_SPORT_TRG_I1	DAI1 SPORT GROUP3 Trigger Input	Pulse
177		Reserved	
178	SWU1_EN	SWU1 Enable	Pulse
179	SWU2_EN	SWU2 Enable	Pulse
180	SWU7_EN	SWU7 Enable	Pulse
181	SWU8_EN	SWU8 Enable	Pulse
182	SWU11_EN	SWU11 Enable	Pulse
183	SWU12_EN	SWU12 Enable	Pulse
184	SWU13_EN	SWU13 Enable	Pulse
185	SWU3_EN	SWU3 Enable Event CL2_1	Pulse
186	SWU4_EN	SWU4 Enable Event DL2_1	Pulse
187	SWU5_EN	SWU5 Enable Event CL2_2	Pulse
188	TIMER0_TMR00_RCV0	TIMER0 Timer 0 Trigger Responder 0	Pulse
189	TIMER0_TMR00_RCV1	TIMER0 Timer 0 Trigger Responder 1	Pulse
190	TIMER0_TMR01_RCV0	TIMER0 Timer 1 Trigger Responder 0	Pulse
191	TIMER0_TMR01_RCV1	TIMER0 Timer 1 Trigger Responder 1	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
192	TIMER0_TMR02_RCV0	TIMER0 Timer 2 Trigger Responder 0	Pulse
193	TIMER0_TMR02_RCV1	TIMER0 Timer 2 Trigger Responder 1	Pulse
194	TIMER0_TMR03_RCV0	TIMER0 Timer 3 Trigger Responder 0	Pulse
195	TIMER0_TMR03_RCV1	TIMER0 Timer 3 Trigger Responder 1	Pulse
196	TIMER0_TMR04_RCV0	TIMER0 Timer 4 Trigger Responder 0	Pulse
197	TIMER0_TMR04_RCV1	TIMER0 Timer 4 Trigger Responder 1	Pulse
198	TIMER0_TMR05_RCV0	TIMER0 Timer 5 Trigger Responder 0	Pulse
199	TIMER0_TMR05_RCV1	TIMER0 Timer 5 Trigger Responder 1	Pulse
200	TIMER0_TMR06_RCV0	TIMER0 Timer 6 Trigger Responder 0	Pulse
201	TIMER0_TMR06_RCV1	TIMER0 Timer 6 Trigger Responder 1	Pulse
202	TIMER0_TMR07_RCV0	TIMER0 Timer 7 Trigger Responder 0	Pulse
203	TIMER0_TMR07_RCV1	TIMER0 Timer 7 Trigger Responder 1	Pulse
204	TIMER0_TMR08_RCV0	TIMER0 Timer 8 Trigger Responder 0	Pulse
205	TIMER0_TMR08_RCV1	TIMER0 Timer 8 Trigger Responder 1	Pulse
206	TIMER0_TMR09_RCV0	TIMER0 Timer 9 Trigger Responder 0	Pulse
207	TIMER0_TMR09_RCV1	TIMER0 Timer 9 Trigger Responder 1	Pulse
208	TIMER0_TMR10_RCV0	TIMER0 Timer 10 Trigger Responder 0	Pulse
209	TIMER0_TMR10_RCV1	TIMER0 Timer 10 Trigger Responder 1	Pulse
210	TIMER0_TMR11_RCV0	TIMER0 Timer 11 Trigger Responder 0	Pulse
211	TIMER0_TMR11_RCV1	TIMER0 Timer 11 Trigger Responder 1	Pulse
212	TIMER0_TMR12_RCV0	TIMER0 Timer 12 Trigger Responder 0	Pulse
213	TIMER0_TMR12_RCV1	TIMER0 Timer 12 Trigger Responder 1	Pulse
214	TIMER0_TMR13_RCV0	TIMER0 Timer 13 Trigger Responder 0	Pulse
215	TIMER0_TMR13_RCV1	TIMER0 Timer 13 Trigger Responder 1	Pulse
216	TIMER0_TMR14_RCV0	TIMER0 Timer 14 Trigger Responder 0	Pulse
217	TIMER0_TMR14_RCV1	TIMER0 Timer 14 Trigger Responder 1	Pulse
218	TIMER0_TMR15_RCV0	TIMER0 Timer 15 Trigger Responder 0	Pulse
219	TIMER0_TMR15_RCV1	TIMER0 Timer 15 Trigger Responder 1	Pulse
220	TRU0_RCV0	TRU0 Interrupt Request n	Pulse
221	TRU0_RCV1	TRU0 Interrupt Request n	Pulse
222	TRU0_RCV2	TRU0 Interrupt Request n	Pulse

Table 8-4: ADSP-2184x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
223	TRU0_RCV3	TRU0 Interrupt Request n	Pulse
224	TRU0_RCV4	TRU0 Interrupt Request 4	Pulse
225	TRU0_RCV5	TRU0 Interrupt Request 5	Pulse
226	TRU0_RCV6	TRU0 Interrupt Request 6	Pulse
227	TRU0_RCV7	TRU0 Interrupt Request 7	Pulse
228	TRU0_RCV8	TRU0 Interrupt Request n	Pulse
229	TRU0_RCV9	TRU0 Interrupt Request n	Pulse
230	TRU0_RCV10	TRU0 Interrupt Request n	Pulse
231	TRU0_RCV11	TRU0 Interrupt Request n	Pulse
232	UART0_TXDMA	UART0 Transmit DMA	Pulse
233	UART0_RXDMA	UART0 Receive DMA	Pulse
234	UART1_TXDMA	UART1 Transmit DMA	Pulse
235	UART1_RXDMA	UART1 Receive DMA	Pulse
236	UART2_TXDMA	UART2 Transmit DMA	Pulse
237	UART2_RXDMA	UART2 Receive DMA	Pulse

TRU Definitions

The following definitions are helpful when using the TRU module.

Trigger Generator

A trigger generator is any system module that provides trigger event indication to the TRU. Trigger generator modules define trigger events and conditions for assertion.

Trigger Generator ID

Trigger generators are assigned a unique numeric ID according to their physical connection to the TRU. Trigger receiver ID 0 is reserved and defined as null.

Trigger Receiver

A trigger receiver is any system module that receives a trigger event indication from the TRU. Trigger receiver modules define a trigger event response.

TRU Block Diagram

The trigger generator and the generator trigger register (GTR) generate trigger assertions. Each trigger receiver has a dedicated receiver select register (RSR) that specifies the unique trigger generator from which it receives the trigger indication.

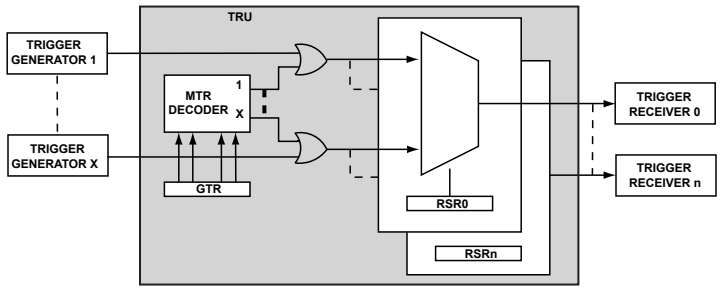


Figure 8-1: TRU Block Diagram

TRU Architectural Concepts

The TRU supports a simple trigger-in/trigger-out model for modules that comply with the triggering functional model. The TRU is the controller of the trigger system. Trigger outputs from trigger generators are mapped to trigger inputs of trigger receivers through a set of programmable registers (`TRU_RSR[n]`).

System modules are trigger generator only, trigger receiver only, or trigger generator and trigger receiver.

All of the trigger input and output signals are connected to a trigger routing unit (TRU) which manages the connections of triggers between modules.

In multi-processor systems, multiple TRU units are provided. These TRUs are networked together. Generic Trigger Ports (GTPs) are provided to forward trigger events from one TRU unit to another, forming a pathway from trigger generator to trigger receivers wherever they might lie in the system.

TRU Programming Model

Implementing sequence control using the TRU requires, at a minimum, proper configuration of a trigger receiver, a trigger generator, and the TRU module itself. The only requirement for the configuration procedure is that the trigger generator is configured and enabled as the last step.

Complete the following other steps:

- Configure the trigger receiver for response to triggers
- Configure the TRU to map the trigger generator to the trigger receiver through the `TRU_RSR[n]` registers
- Configure the trigger generator to generate trigger assertions
- Alternatively, use software triggering for trigger assertion. Writing the trigger generator ID to the GTR register generates software triggers.

Programming Concepts

The following concepts aid in programming the TRU.

- ***Trigger Sequence Configuration.*** A simple sequence consists of one trigger generator and one trigger receiver. More complex trigger sequences consist of several trigger receivers functioning as trigger receiver and trigger generator. Additionally, trigger sequences can loopback to the original generator forming a perpetual sequence.
- ***Software Triggering.*** Writing a trigger generator ID to the GTR generates a trigger within the TRU from the trigger generator ID specified.
- ***Synchronization.*** The TRU can be used to coarsely synchronize events by mapping multiple trigger receivers to the same trigger generator or by generating multiple trigger generator assertions simultaneously through the GTR.
- ***Configuration Protection.*** The `TRU_RSR[n].LOCK` bit and the `TRU_GCTL.LOCK` bit enable register level write-protection when the global lock is asserted in the SPU.

Programming Examples

The following examples shows the steps to create a single trigger.

Configuring a Simple Trigger Sequence

The following example shows the steps to create a simple trigger.

1. Enable the Global Lock feature bit on the SPU (set the `SPU_CTL.GLCK` bit).
2. Set the `TRU_GCTL.LOCK` bit to 1 so that the `TRU_GCTL` register becomes read-only.
3. Set the `TRU_GCTL.GENL` bit to 1 so that the `TRU_GEN` register becomes read-only.

TRU Event Control

The TRU is a major part of event control solutions. It is the center of the trigger functional model and can extend to support the interrupt and fault management models as well.

TRU Status and Error Signals

The TRU does not have dedicated status and error output signals other than the MMR interface. Receiver errors are reported to the generator over the standard peripheral bus protocol.

ADSP-2184x TRU Register Descriptions

Trigger Routing Unit (TRU) contains the following registers.

Table 8-5: ADSP-2184x TRU Register List

Name	Description
TRU_ERRADDR	Error Address Register
TRU_GCTL	Global Control Register
TRU_GEN	Generator Trigger Register
TRU_RSR[n]	Receiver Select Register
TRU_STAT	Status Information Register

Error Address Register

The TRU error address register (TRU_ERRADDR) holds the address from the memory-mapped register access generating an access error of TRU registers.

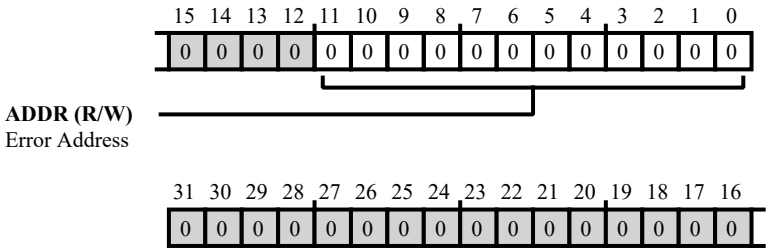


Figure 8-2: TRU_ERRADDR Register Diagram

Table 8-6: TRU_ERRADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	ADDR	Error Address. The TRU_ERRADDR.ADDR holds the address from the memory-mapped register access generating an access error of TRU registers. These errors occur on access to the TRU_RSR[n] or TRU_GEN registers when these registers are locked or on access to an invalid address. See the TRU_RSR[n] and TRU_GEN register descriptions for more information about locking. The TRU_ERRADDR register holds the address of the first error to occur. In the event of multiple errors occurring, the TRU_ERRADDR register contains the address of the first error. To re-enable the TRU_ERRADDR register for update, both status bits (TRU_STAT.LWERR and TRU_STAT.ADDRERR) in the TRU_STAT register must be cleared.

Global Control Register

The TRU global control register (TRU_GCTL) provides register locking, TRU reset, and TRU enable.

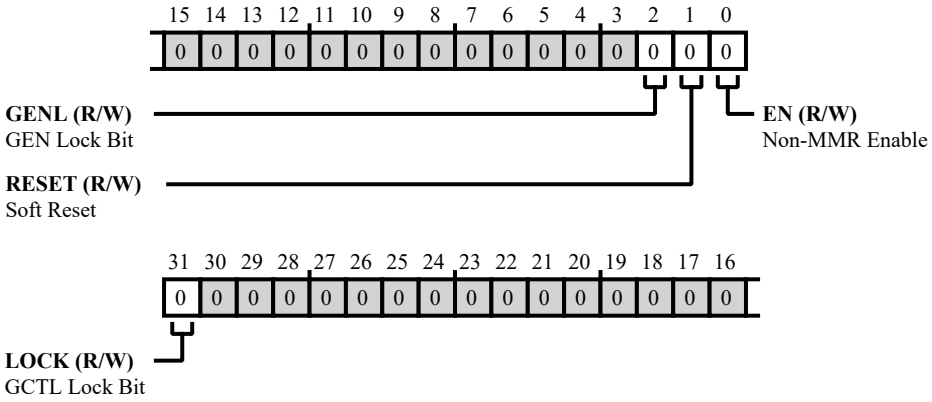


Figure 8-3: TRU_GCTL Register Diagram

Table 8-7: TRU_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	GCTL Lock Bit. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_GCTL.LOCK bit is enabled, the TRU_GCTL register is read only.	
		0	Read Write
		1	Read Only
2 (R/W)	GENL	GEN Lock Bit. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_GCTL.GENL bit is enabled, the TRU_GEN register is read only.	
		0	Read Write
		1	Read Only
1 (R/W)	RESET	Soft Reset. The TRU_GCTL.RESET bit is write-1-action and triggers a soft reset to all TRU registers.	
		0	No Action
		1	Soft Reset

Table 8-7: TRU_GCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Non-MMR Enable. The `TRU_GCTL.EN` bit is read/write and must be set for the TRU to propagate trigger events. All TRU register read/write operations continue to operate independent of the `TRU_GCTL.EN` bit.	
		0	No Trigger Events
		1	Propagate Trigger Events

Generator Trigger Register

The TRU generator trigger register (TRU_GEN) permits trigger generation through software by writing a trigger generator ID value to one of the four fields in the TRU_GEN register. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_GCTL.LOCK bit is set, the TRU_GEN register is read only. Note this register is primarily used for debug to trigger a TRU output

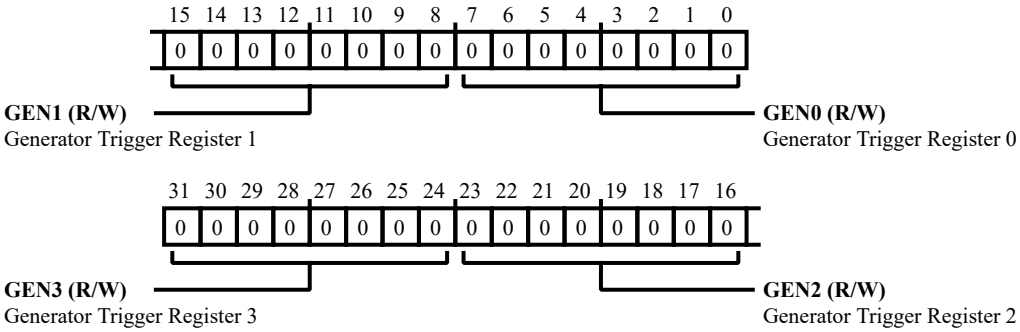


Figure 8-4: TRU_GEN Register Diagram

Table 8-8: TRU_GEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	GEN3	Generator Trigger Register 3. The TRU_GEN.GEN3 bit field is the trigger generator ID value for generator 3.	
		0	No Generator Specified
		1-182	Range of Valid Generators
23:16 (R/W)	GEN2	Generator Trigger Register 2. The TRU_GEN.GEN2 bit field is the trigger generator ID value for generator 2.	
		0	No Generator Specified
		1-182	Range of Valid Generators
15:8 (R/W)	GEN1	Generator Trigger Register 1. The TRU_GEN.GEN1 bit field is the trigger generator ID value for generator 1.	
		0	No Generator Specified
		1-182	Range of Valid Generators
7:0 (R/W)	GEN0	Generator Trigger Register 0. The TRU_GEN.GEN0 bit field is the trigger generator ID value for generator 0.	
		0	No Generator Specified
		1-182	Range of Valid Generators

Receiver Select Register

The TRU receiver select registers (TRU_RSR[n]) each provide trigger receiver selection and register locking.

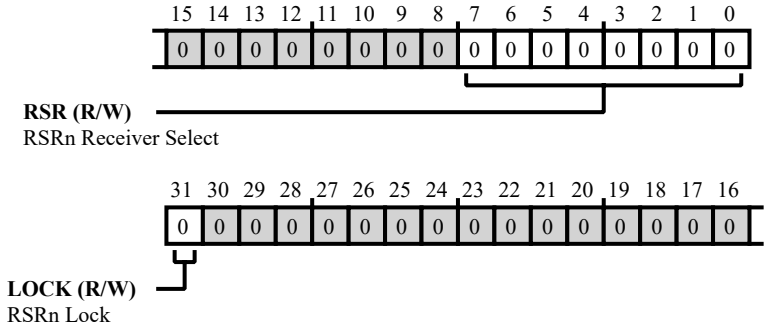


Figure 8-5: TRU_RSR[n] Register Diagram

Table 8-9: TRU_RSR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	RSRn Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_RSR[n].LOCK bit is enabled, the TRU_RSR[n] register is read only.	
		0	Unlock Register
		1	Lock Register
7:0 (R/W)	RSR	RSRn Receiver Select. The TRU_RSR[n] register selects the trigger generator ID to which the trigger receiver responds. For example, when a TRU_RSR[n] register is set to respond to trigger generator ID n, a trigger that is generated by trigger generator ID n results in a trigger out to the receiver.	
		0	No Generator Specified
		1-182	Range of Valid Generators

Status Information Register

The TRU status register (TRU_STAT) contains the status of TRU_GEN and TRU_RSR[n] register writes and status of bus read/write errors.

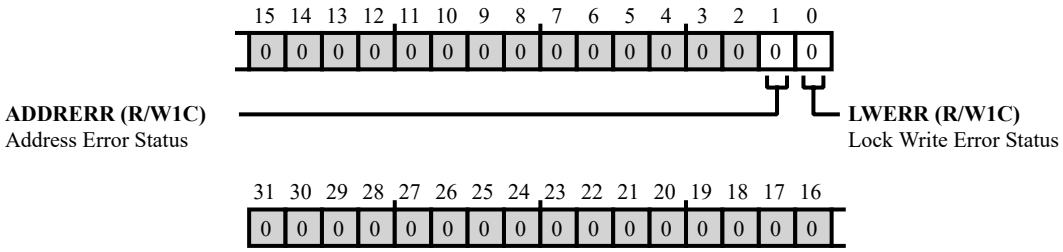


Figure 8-6: TRU_STAT Register Diagram

Table 8-10: TRU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	ADDRERR	Address Error Status. The TRU_STAT.ADDRERR bit is set when an invalid address is provided for an MMR access while the TRU is selected. Writing a one to this bit clears the error indication. The TRU_ERRADDR register also is updated when an address error occurs during an MMR access while the TRU is selected.	
		0	No error
		1	Error occurred
0 (R/W1C)	LWERR	Lock Write Error Status. If TRU_STAT.LWERR is set, a lock write error has occurred. Writing a one to this bit clears the error indication.	
		0	No error
		1	Error occurred

9 L2 System Memory

L2 system memories have significant bandwidth for core accesses, but it is important to note that L2 responds slower to core accesses than L1 memories. L2 SRAM is the ideal storage for multiple processor cores to share data and instruction resources, such as semaphores, shared buffers, and code libraries. Due to sophisticated data integrity protection and write protection, L2 SRAM is also ideal for data and instructions critical for safe operation of the application.

L2 System Memory Features

The L2 system memory features include:

- Operation at `SYSCLK` frequency
- ECC protection of SRAM area
- ECC memory refresh

There are three instance of L2 system memory: L2CTL0, L2CTL1 and LCTL2.

- L2CTL0 contains 2M byte of RAM grouped into eight banks, 256K bytes each and 192K bytes of boot ROM.
- L2CTL1 contains 2M byte of RAM grouped into eight banks, 256K bytes each.
- SHARC core data ports can access L2 memory through direct path.

L2 System Memory Functional Description

The L2 system memory manages all of the L2 SRAM and ROM memory banks. The system memory interface arbitrates competing accesses, write protection, and ensures SRAM data integrity. The L2 system memory domain is a unified instruction and data memory. It can hold any mixture of code and data required by the system design.

The following sections provide a functional description of the L2 system memory.

ADSP-2184x L2CTL Register List

The L2 memory controller (L2CTL) includes the controls to manage each L2 memory bank independently. A set of registers governs L2CTL operations. For more information on L2CTL functionality, see the L2CTL register descriptions.

Table 9-1: ADSP-2184x L2CTL Register List

Name	Description
L2CTL_CTL	Control Register
L2CTL_EADDR0	Error Type 0 Address Register
L2CTL_EADDR1	Error Type 1 Address Register
L2CTL_EADDR2	Error Type 2 Address Register
L2CTL_EADDR3	Error Type 3 Address Register
L2CTL_EADDR4	Error Type 4 Address Register
L2CTL_ERRADDR0	ECC Error Address 0 Register
L2CTL_ERRADDR1	ECC Error Address 1 Register
L2CTL_ERRADDR2	ECC Error Address 2 Register
L2CTL_ERRADDR3	ECC Error Address 3 Register
L2CTL_ERRADDR4	ECC Error Address 4 Register
L2CTL_ERRADDR5	ECC Error Address 5 Register
L2CTL_ERRADDR6	ECC Error Address 6 Register
L2CTL_ERRADDR7	ECC Error Address 7 Register
L2CTL_ERRADDR8	ECC Error Address 8 Register
L2CTL_ET0	Error Type 0 Register
L2CTL_ET1	Error Type 1 Register
L2CTL_ET2	Error Type 2 Register
L2CTL_ET3	Error Type 3 Register
L2CTL_ET4	Error Type 4 Register
L2CTL_INIT	Initialization Register
L2CTL_ISTAT	Initialization Status Register
L2CTL_PCTL	Power Control Register
L2CTL_REVID	Revision ID Register
L2CTL_RPCR0	Read Priority Count Register
L2CTL_RPCR1	Read Priority Count Register
L2CTL_SADR	Scrub Start Address Register

Table 9-1: ADSP-2184x L2CTL Register List (Continued)

Name	Description
L2CTL_SCNT	Scrub Count Register
L2CTL_SCTL	Scrub Control Register
L2CTL_STAT	Status Register
L2CTL_STAT_1	L2 Port Error Status Register
L2CTL_WPCR0	Write Priority Count Register
L2CTL_WPCR1	Write Priority Count Register

ADSP-2184x L2CTL Interrupt List

Table 9-2: ADSP-2184x L2CTL Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
143	L2CTL0_ECC_ERR	L2CTL0 ECC Error	Level	
145	L2CTL0_EVT	L2CTL0 Scrub/Initialization Done	Level	
146	L2CTL1_ECC_ERR	L2CTL1 ECC Error	Level	
148	L2CTL1_EVT	L2CTL1 Scrub/Initialization Done	Level	

ADSP-2184x L2CTL Trigger List

Table 9-3: ADSP-2184x L2CTL Trigger List Generators

Trigger ID	Name	Description	Sensitivity
64	L2CTL0_EVT	L2CTL0 L2 Memory Event	Level
65	L2CTL1_EVT	L2CTL1 L2 Memory Event	Level

Table 9-4: ADSP-2184x L2CTL Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

L2 System Memory Block Diagram

The ***ADSP-2184x/ADSP-SC84x Complete L2 System Memory Block Diagram*** shows the complete L2 system memory, including the memory block instances L2CTL0, L2CTL1 and L2CTL2. The L2CTL0 block contains boot ROM code for Arm and eight banks of L2 RAM containing 256 Kbytes each. The Arm reset ISR (location 0x00000000) is mapped to this block.

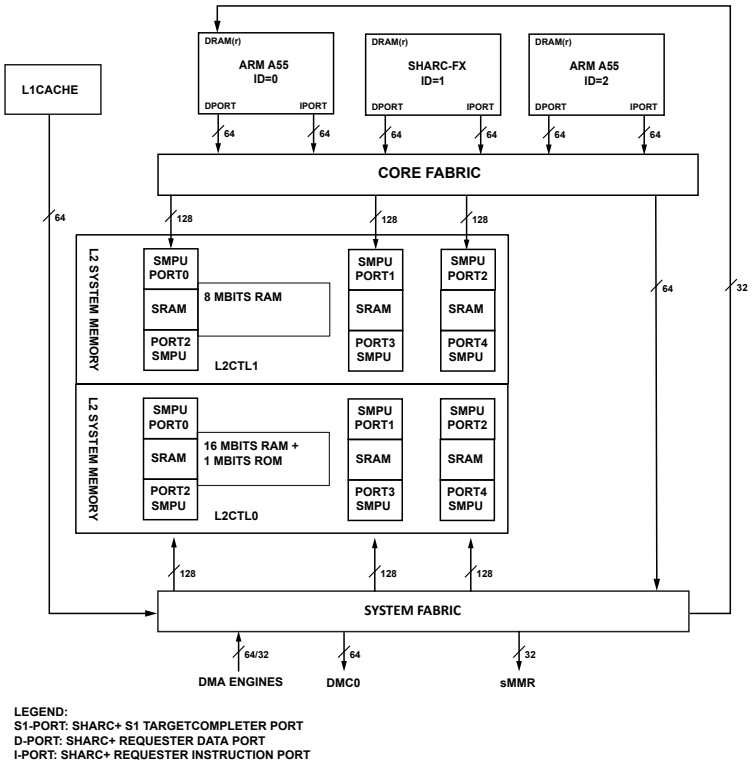


Figure 9-1: ADSP-2184x Complete L2 System Block Diagram

L2 System Memory Architectural Concepts

The following sections describe the architecture features of the L2 system memory.

Access Characteristics

The L2 system memory interface converts all 8-bit, 16-bit, 32-bit, and 64-bit accesses to 128-bit accesses. Additionally, it converts 8-bit, 16-bit, 32-bit, and 64-bit bursts to an equivalent internal 128-bit access. For example, the L2 system memory interface converts a 128-bit address-aligned burst of 8-bit accesses of burst length 16 to a single 128-bit access.

Read/Write Latency and Throughput

The L2 memory design is optimized for burst accesses at the crossbar interface. The L2 system memory buffers and converts write data of 8/16/32/64-bit to an equivalent 128-bit access. This conversion creates modulo-32-bit writes if the starting addresses are 32-bit aligned. A single 8-bit or 16-bit access, or a non-32-bit address-aligned 8-bit or 16-bit burst access to an ECC-enabled bank creates an extra latency of two SYSCLK cycles. No extra latency is seen if the ECC is disabled.

NOTE: Continuous 8/16-bit core access to an ECC-enabled L2 bank is not recommended from a throughput perspective.

L2 Memory Controller Block Diagram (Instance)

As shown in the ***L2 System Memory Block Diagram***, the L2 controller has five ports that interface to system crossbars. Port 0, port 1, and port 2 are 128-bit interfaces dedicated to core and MDMA3 traffic. Port 3 and port 4 are 128-bit interfaces that connect through DMA. For L2 SRAM, all ports (0/1/2/3/4) have a read channel and a write channel; for L2 ROM, all ports (0/1/2/3/4) have read channels only. The SRAM is organized in multiple banks; each bank has 256K Bytes. The 192KB ROM is divided into three banks of 64KB.

Within each bank, data is organized into words, with each word comprising 128 bits of data and 28 bits of ECC checksum. ROM memory is not protected by the ECC scheme. When the L2 controller accesses RAM and ROM cells, it always reads and writes whole 128-bit words. Despite this, the L2 controller supports 8-, 16-, 32-, and 64-bit reads and writes from cores and system by applying respective data masks.

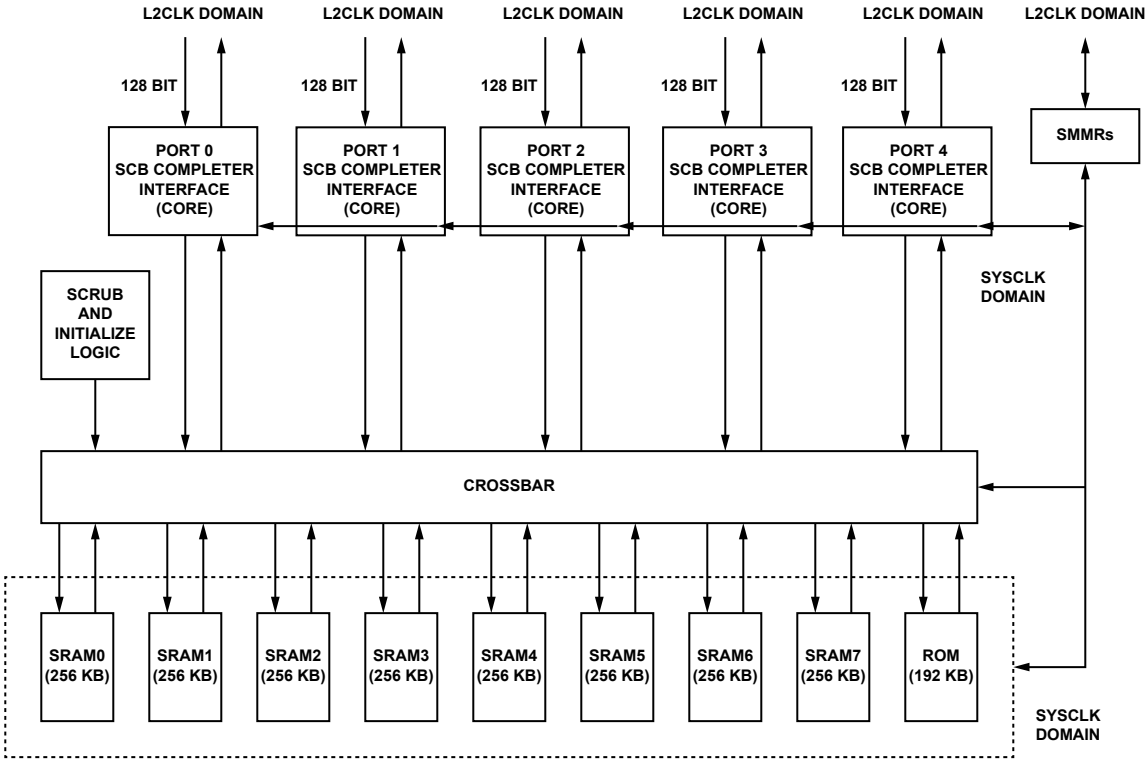


Figure 9-2: L2 System Memory Block Diagram

Arbitration and Priority

Each bank of L2 RAM or ROM has an arbiter which receives requests from the five crossbar ports.

Each arbiter follows a fixed priority scheme for giving grants when more than one channel requests the same bank. The arbiter also supports priority elevation through urgent priority requests.

NOTE: Attempting a write access to both L2 ROM spaces returns an error.

The ***Fixed Priority*** table shows the priority for fixed priority mode (with urgent priority disabled) for each SCB channel.

If two cores (or the 64-bit Max BW DMA) simultaneously try to access L2 for the same instance (both read or both write), even to different banks, software allows only one controller access at a time. One access port can support one read and write at the same time. However, if one core issues a write and the other issues a read, then access can proceed simultaneously. There is no extra latency inside L2, when the accesses are to different banks (assuming pending DMA traffic is also to a non-conflicting bank).

When a core and DMA both access the same bank via the same port (both read or both writes), the best access rate that DMA can achieve is one 128-bit access in every three SYSCLK cycles during the conflict period. This access rate is achieved by programming the read priority count register (`L2CTL_RPCR0.RPC0`) bit and the write priority count register (`L2CTL_WPCR0.WPC0`) to 0, while programming the `L2CTL_RPCR0.RPC1` and the `L2CTL_WPCR0.WPC1` bits to 1.

The arbiters also support priority elevation for a particular channel that has been starved of grants for many SYSCLK cycles. If a channel does not get a grant for *N* cycles after its request, then that channel can elevate the priority of its request by issuing an urgent priority request. This request causes that channel to become the highest priority controller for the next grant cycle (pipelined arbitration for urgent priority). The number of cycles *N*, after which the priority is elevated, can be programmed for each channel separately using the (`L2CTL_RPCR0`) and (`L2CTL_WPCR0`) registers.

Programming the bits in the (`L2CTL_RPCR0`) and (`L2CTL_WPCR0`) registers appropriately achieves the best grant rate for DMA. This grant rate of one in three SYSCLK cycles during the conflict period is achievable under the following conflict conditions:

- An access conflict between the core and DMA to the same memory bank in the fixed priority arbitration scheme with core activity always prioritized over DMA activity
- An access conflict within the pipelined implementation of urgent priority

To disable urgent priority requests, set the `L2CTL_CTL.DISURP` bit. This bit disables the urgent priority requests for all port channels. Each channel can also be prevented from raising the urgent priority request through the priority count register for the specific channel. However, there is no support for disabling urgent priority for a specific memory bank arbiter.

The ***Fixed Priority With Priority Elevation*** table provides the various priority levels for the L2 system memory.

Table 9-5: Fixed Priority With Priority Elevation

Channel	Priority Level
L2 Scrub/Initialization Request	21 (highest)
Port 0 Read Channel Urgent Request	20
Port 0 Write Channel Urgent Request	19
Port 1 Read Channel Urgent Request	18
Port 1 Write Channel Urgent Request	17
Port 2 Read Channel Urgent Request	16

Table 9-5: Fixed Priority With Priority Elevation (Continued)

Channel	Priority Level
Port 2 Write Channel Urgent Request	15
Port 3 Read Channel Urgent Request	14
Port 3 Write Channel Urgent Request	13
Port 4 Read Channel Urgent Request	12
Port 4 Write Channel Urgent Request	11
Port 0 Read Channel Normal Request	10
Port 0 Write Channel Normal Request	9
Port 1 Read Channel Normal Request	8
Port 1 Write Channel Normal Request	7
Port 2 Read Channel Normal Request	6
Port 2 Write Channel Normal Request	5
Port 3 Read Channel Normal Request	4
Port 3 Write Channel Normal Request	3
Port 4 Read Channel Normal Request	2
Port 4 Write Channel Normal Request	1 (lowest)

Data Integrity

The following sections provide information on how the L2 system memory ensures data integrity.

ECC Algorithm

Hsaio encoding calculates the ECC syndrome. A 7-bit syndrome is generated during write operation and stored as a 7-bit parity field along with the 32 data bits. Each data bit contributes to three parity bits according. Each parity bit represents the XOR value of 13 or 14 data bits according to the following mapping:

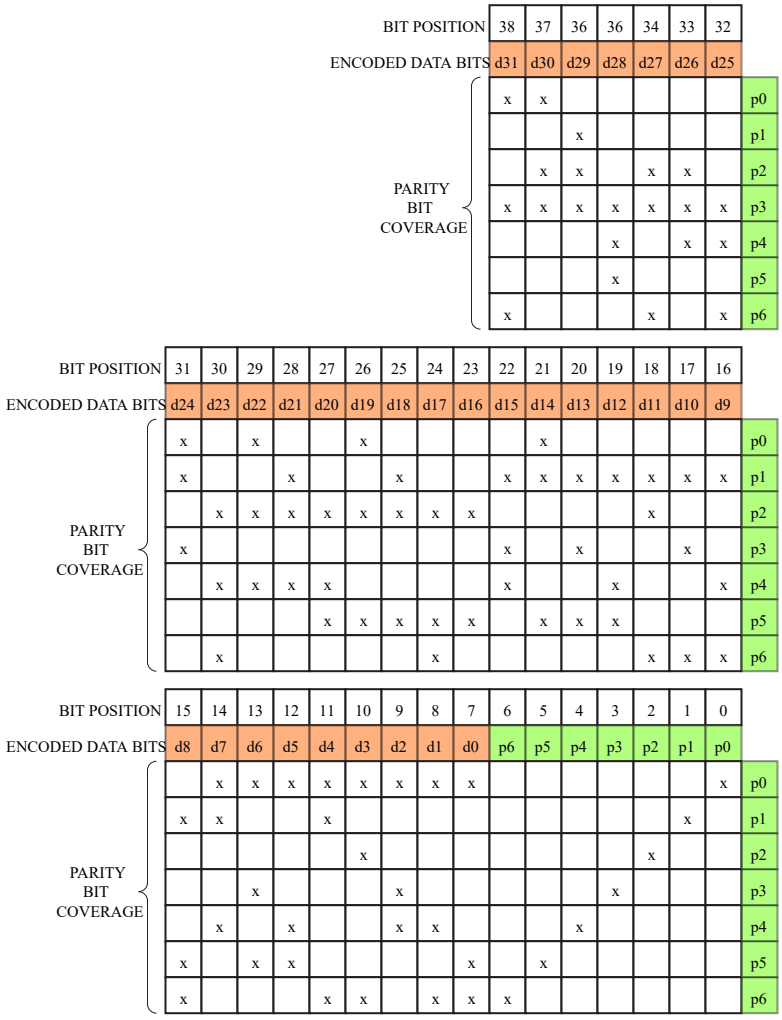


Figure 9-3: Hsaio Parity Bit Mapping

During read operation, the parity bits become part of the syndrome equation. The new syndrome bits are now the XOR values of the 13 or 14 data bits plus the respective stored parity bit. If any of the seven syndrome bits is set, an error situation is detected. An OR gate cross of the 7 bits reports the error, without specifying the type of the error.

If a single parity bit failed, the new 7-bit syndrome has 1 bit that is set. If a single data bit failed, the new syndrome has 3 bits that are set, because all three related parity bits fail. So, an XOR gate cross of all seven syndrome bits detects a single-bit error, indicating that an odd number of syndrome bits is set.

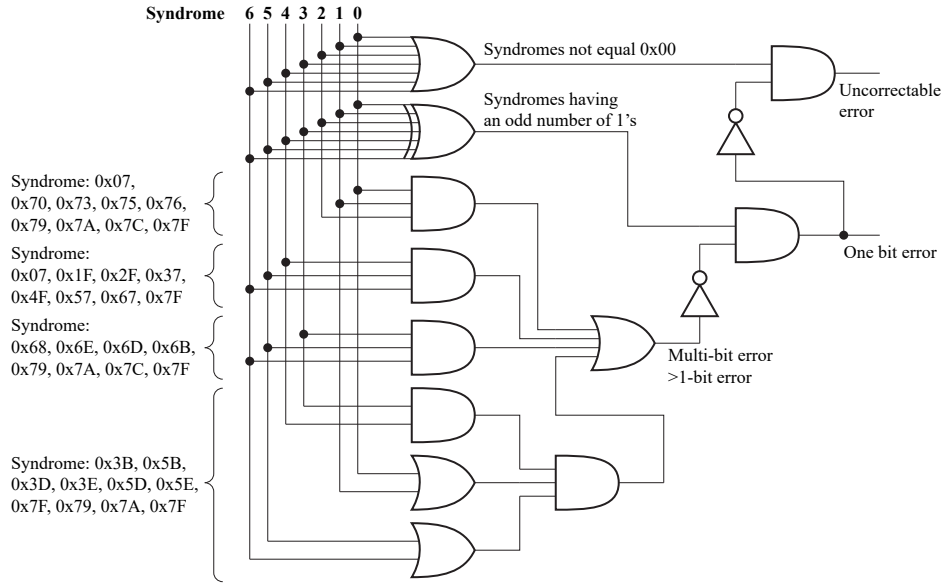


Figure 9-4: Hsiao Error Reports

The XOR gate detects single-bit errors and does not flag any dual-bit error. But the gate does flag 50% of the other multi-bit errors undesirably. Extra logic is implemented to increase the detection rate of multi-bit errors to 68.7% as shown in the figure.

If a single-bit error is detected, the failing bit can be localized and corrected. If all three syndrome bits corresponding to a specific data bit are 1, a data error is assumed. The respective data bit is toggled on its way to the system bus.

ECC Hardware Control

After reset, ECC protection is enabled. The boot code initializes all L2 SRAM data and checksum cells. ECC protection adds some cycle penalty when 8-bit and 16-bit values write L2 memory. Disable ECC protection for individual SRAM banks by setting the `L2CTL_CTL.BK0EDIS` through `L2CTL_CTL.BK7EDIS` disable bits. Due to caching mechanisms of the processor cores and data bursting of the DMA channels, 8-bit and 16-bit write accesses are uncommon. Typically, only two-dimensional DMA operations or un-cached 8-bit and 16-bit store instruction can trigger these writes.

ECC Error Management

The L2 system memory flags 2-bit and multi-bit errors to the system by:

- Raising the ECC_ERR interrupt
- Reporting a read error to the system bus
- Setting the sticky `L2CTL_STAT.ECCERR7-L2CTL_STAT.ECCERR0` status flag
- Latching the address of the failing operation into the respective `L2CTL_ERRADDR7-L2CTL_ERRADDR0` register.

There is one error status bit and one error address register per L2 SRAM bank.

Typically, ECC_ERR events are declared as system faults in the system event controller (SEC). Whether these faults are reported, the interrupt service routine can consult the L2CTL_STAT register and the L2CTL_ERRADDR0 through L2CTL_ERRADDR7 registers to determine whether:

- The data at the failing L2 address was critical enough to require an immediate reboot of the system
- The data at the failing L2 address was less critical or can be restored

The L2CTL_STAT.ECCERR0 through L2CTL_STAT.ECCERR7 flags are cleared with a W1C operation.

Memory Refresh

If data in L2 SRAM contains single-bit errors, the data is corrected on its way to the system buses. The corrected value is not written back to the SRAM location. To prevent any risk of accumulation of single-bit errors over time and to minimize likelihood of multi-bit errors, the L2 system memory provides a special memory refresh mechanism.

If there are dual-bit or multi-bit errors, the ECC_ERR interrupt is raised, and data is not written back to memory.

- TheL2CTL_SADR register with the start address of the scrub
- The L2CTL_SCNT register with the total number of 128-bit addresses to be scrubbed starting from the L2CTL_SADR address

Program the number of cycles between each scrub using the L2CTL_SCTL register. During the scrub, the L2 system memory issues a read, followed by a write-back operation if there is a single-bit ECC error. Once the L2 system memory completes scrubbing the programmed memory region, it generates an interrupt and starts again from the L2CTL_SADR address unless the L2CTL_SCTL.SEN bit is disabled. If the L2CTL_SCTL.SEN bit is cleared before completing the programmed address range, the scrub stops after completing any already issued scrub access. The scrub read/writes always start from the full 128-bit equivalent of the address written into the L2CTL_SADR register. A 128-bit value is always read, and the 128-bit value is written back. The scrub access has the highest priority. Programs can configure 8-bit, 16-bit, or 32-bit addresses in the L2CTL_SADR register. The lower 4 bits of this register are treated as *do-not-care* values because the internal memory array is always accessed when using 128 bits.

Memory refresh operation is meaningless when the L2CTL_CTL.BK0EDIS through L2CTL_CTL.BK7EDIS disable bits are set.

Power Modes

Each L2 memory bank supports low-power modes.

If the bank is not in use, put each L2 system memory bank into the following low-power modes to save power.

Deep Sleep Mode

Set the L2CTL_PCTL.BK7DS through L2CTL_PCTL.BK0DS control bits to enter this mode. This mode preserves the memory contents.

Shut Down Mode

Set the L2CTL_PCTL.BK7SD through L2CTL_PCTL.BK0SD control bits to enter this mode. The memory banks do not retain the data (existing data is lost).

NOTE: It may take up to 11 SYSCLK cycles to deactivate the power mode after writing to the L2CTL_PCTL register. Therefore, L2 access must be issued after this delay.

Access to a memory bank in shut down or deep sleep mode may result in an unpredictable behavior. Ensure that initialization/refresh and memory access to such banks are not issued.

There is no deep sleep or shut down mode for ROM memories.

L2 System Memory Event Control

The following sections describe event control features of the L2 system memory, such as error response.

ECC Error Interrupt

A bus error is signaled under any of the following conditions.

- A write access to ROM address space
- A read/write access to reserved address space
- An ECC multi-bit error in an ECC-enabled bank. A non-modulo, 32-bit write to an ECC-enabled bank can also potentially create a bus error response due to an ECC multi-bit error. This response is because the L2 system memory implements a 32-bit ECC, and therefore a non-modulo, 32-bit write results in a read. This read can create multi-bit errors even if the memory was initialized.

Bus error notifications are stored in the L2CTL_STAT register, and the addresses that generated the error on a given port are stored in the L2CTL_EADDR0/L2CTL_EADDR1 register of that port. The details of the error are stored in the L2CTL_ET0/L2CTL_ET1 register of the port.

ADSP-2184x L2CTL Register Descriptions

L2 Memory Controller (L2CTL) contains the following registers.

Table 9-6: ADSP-2184x L2CTL Register List

Name	Description
L2CTL_CTL	Control Register
L2CTL_EADDR0	Error Type 0 Address Register
L2CTL_EADDR1	Error Type 1 Address Register
L2CTL_EADDR2	Error Type 2 Address Register
L2CTL_EADDR3	Error Type 3 Address Register

Table 9-6: ADSP-2184x L2CTL Register List (Continued)

Name	Description
L2CTL_EADDR4	Error Type 4 Address Register
L2CTL_ERRADDR0	ECC Error Address 0 Register
L2CTL_ERRADDR1	ECC Error Address 1 Register
L2CTL_ERRADDR2	ECC Error Address 2 Register
L2CTL_ERRADDR3	ECC Error Address 3 Register
L2CTL_ERRADDR4	ECC Error Address 4 Register
L2CTL_ERRADDR5	ECC Error Address 5 Register
L2CTL_ERRADDR6	ECC Error Address 6 Register
L2CTL_ERRADDR7	ECC Error Address 7 Register
L2CTL_ERRADDR8	ECC Error Address 8 Register
L2CTL_ET0	Error Type 0 Register
L2CTL_ET1	Error Type 1 Register
L2CTL_ET2	Error Type 2 Register
L2CTL_ET3	Error Type 3 Register
L2CTL_ET4	Error Type 4 Register
L2CTL_INIT	Initialization Register
L2CTL_ISTAT	Initialization Status Register
L2CTL_PCTL	Power Control Register
L2CTL_REVID	Revision ID Register
L2CTL_RPCR0	Read Priority Count Register
L2CTL_RPCR1	Read Priority Count Register
L2CTL_SADR	Scrub Start Address Register
L2CTL_SCNT	Scrub Count Register
L2CTL_SCTL	Scrub Control Register
L2CTL_STAT	Status Register
L2CTL_STAT_1	L2 Port Error Status Register
L2CTL_WPCR0	Write Priority Count Register
L2CTL_WPCR1	Write Priority Count Register

Control Register

The L2CTL_CTL register includes a write protection bit, enables L2 banks, and selects mapping of banks (as ECC RAM or data RAM).

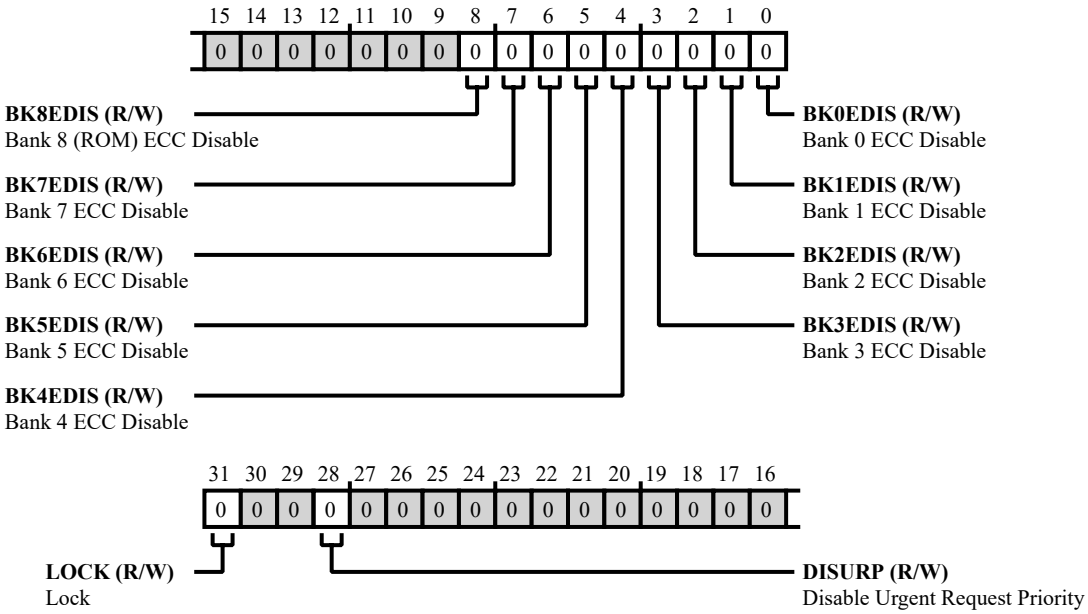


Figure 9-5: L2CTL_CTL Register Diagram

Table 9-7: L2CTL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the L2CTL_CTL.LOCK bit is set, the L2CTL_CTL register is read only (locked).	
		0	Unlock
		1	Lock
28 (R/W)	DISURP	Disable Urgent Request Priority. The L2CTL_CTL.DISURP disables urgent request priority mode for all L2 banks.	
		0	Enable URP
		1	Disable URP

Table 9-7: L2CTL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	BK8EDIS	Bank 8 (ROM) ECC Disable. The `L2CTL_CTL.BK8EDIS` bit disables L2 bank 8 (ROM) ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
7 (R/W)	BK7EDIS	Bank 7 ECC Disable. The `L2CTL_CTL.BK7EDIS` bit disables L2 bank 7 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
6 (R/W)	BK6EDIS	Bank 6 ECC Disable. The `L2CTL_CTL.BK6EDIS` bit disables L2 bank 6 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
5 (R/W)	BK5EDIS	Bank 5 ECC Disable. The `L2CTL_CTL.BK5EDIS` bit disables L2 bank 5 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
4 (R/W)	BK4EDIS	Bank 4 ECC Disable. The `L2CTL_CTL.BK4EDIS` bit disables L2 bank 4 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
3 (R/W)	BK3EDIS	Bank 3 ECC Disable. The `L2CTL_CTL.BK3EDIS` bit disables L2 bank 3 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC

Table 9-7: L2CTL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	BK2EDIS	Bank 2 ECC Disable. The `L2CTL_CTL.BK2EDIS` bit disables L2 bank 2 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
1 (R/W)	BK1EDIS	Bank 1 ECC Disable. The `L2CTL_CTL.BK1EDIS` bit disables L2 bank 1 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
0 (R/W)	BK0EDIS	Bank 0 ECC Disable. The `L2CTL_CTL.BK0EDIS` bit disables L2 bank 0 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC

Error Type 0 Address Register

The L2CTL_EADDR0 register holds the address that created an access error on the L2 port 0 bus interface. This register is be updated only if the corresponding error status bit (L2CTL_STAT.ERR0) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR0 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

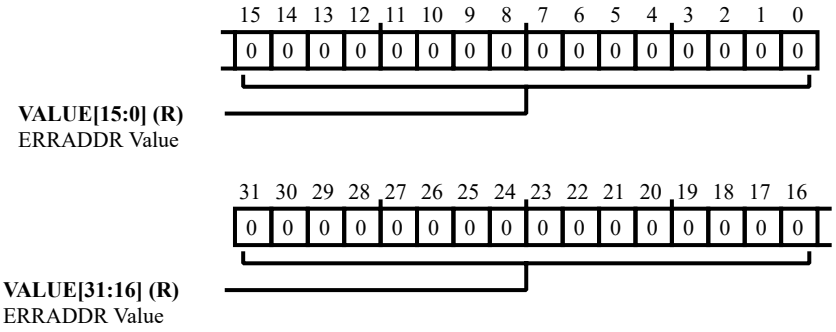


Figure 9-6: L2CTL_EADDR0 Register Diagram

Table 9-8: L2CTL_EADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR0.VALUE bits hold the address causing the bus error.

Error Type 1 Address Register

The L2CTL_EADDR1 register holds the address that created an access error on the L2 port 1 bus interface. This register is be updated only if the corresponding error status bit (L2CTL_STAT.ERR1) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR1 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

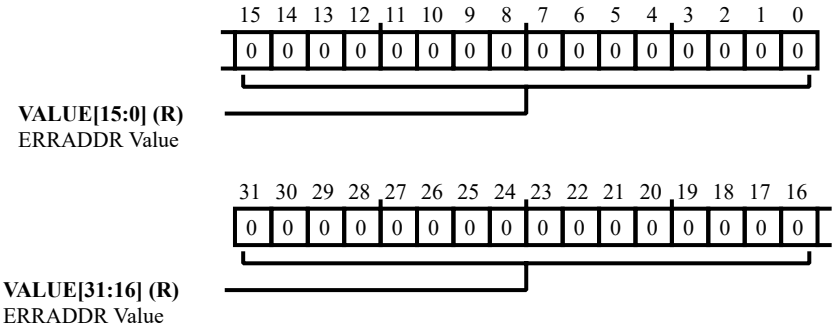


Figure 9-7: L2CTL_EADDR1 Register Diagram

Table 9-9: L2CTL_EADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR1.VALUE bits hold the address causing the bus error.

Error Type 2 Address Register

The L2CTL_EADDR2 register holds the address that created an access error on the L2 port 2 bus interface (cores). This register is be updated only if the corresponding error status bit (L2CTL_STAT_1.ERR2) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR2 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

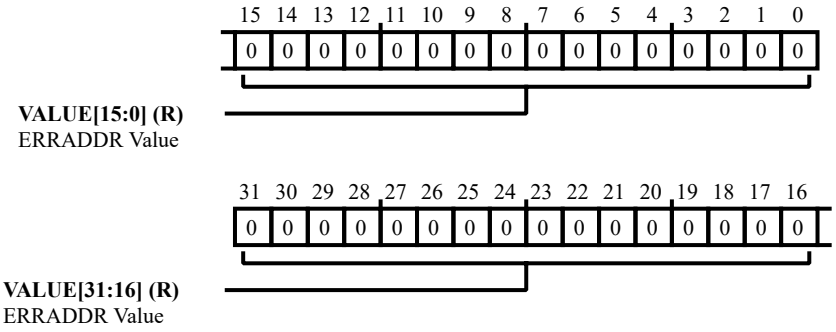


Figure 9-8: L2CTL_EADDR2 Register Diagram

Table 9-10: L2CTL_EADDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR2.VALUE bits hold the address causing the bus error.

Error Type 3 Address Register

The L2CTL_EADDR3 register holds the address that created an access error on the L2 port 3 bus interface (DMA). This register is be updated only if the corresponding error status bit (L2CTL_STAT_1.ERR3) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR3 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

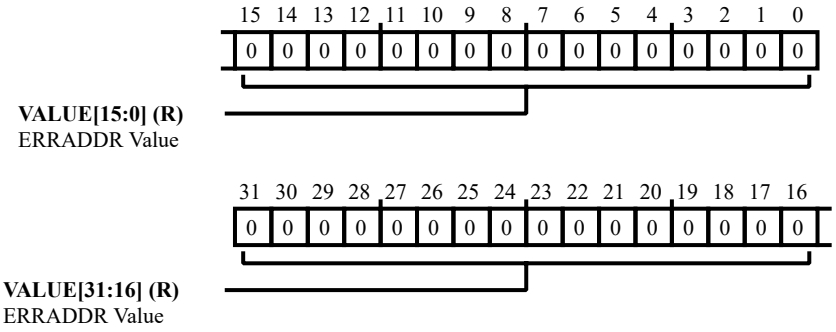


Figure 9-9: L2CTL_EADDR3 Register Diagram

Table 9-11: L2CTL_EADDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR3.VALUE bits hold the address causing the bus error.

Error Type 4 Address Register

The L2CTL_EADDR4 register holds the address that created an access error on the L2 port 4 bus interface (DMA). This register is be updated only if the corresponding error status bit (L2CTL_STAT_1.ERR4) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR4 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

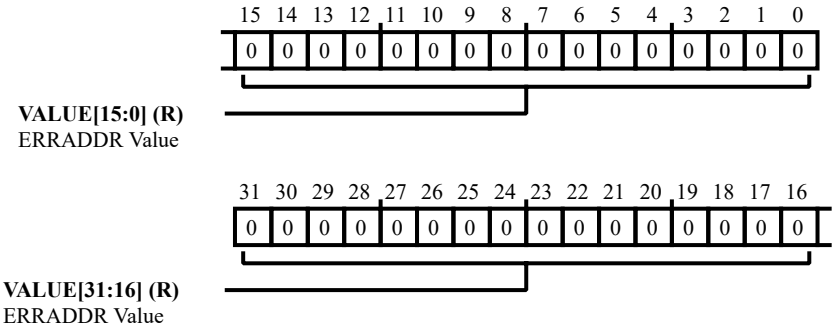


Figure 9-10: L2CTL_EADDR4 Register Diagram

Table 9-12: L2CTL_EADDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR4.VALUE bits hold the address causing the bus error.

ECC Error Address 0 Register

The L2CTL_ERRADDR0 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR0) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

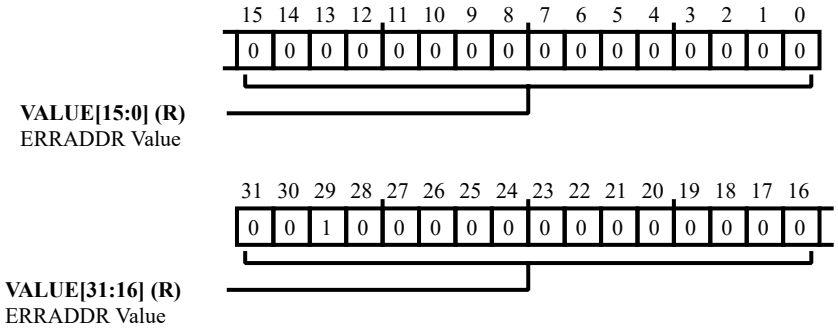


Figure 9-11: L2CTL_ERRADDR0 Register Diagram

Table 9-13: L2CTL_ERRADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR0.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 1 Register

The L2CTL_ERRADDR1 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR1) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

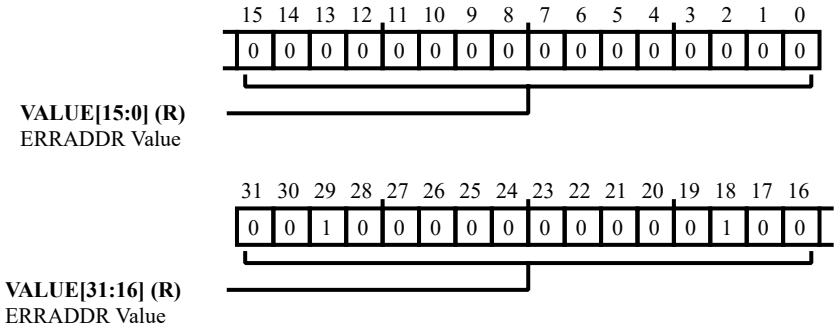


Figure 9-12: L2CTL_ERRADDR1 Register Diagram

Table 9-14: L2CTL_ERRADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR1.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 2 Register

The L2CTL_ERRADDR2 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR2) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

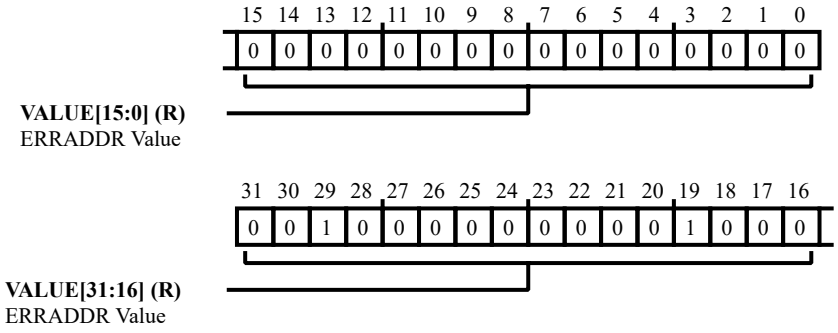


Figure 9-13: L2CTL_ERRADDR2 Register Diagram

Table 9-15: L2CTL_ERRADDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR2.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 3 Register

The L2CTL_ERRADDR3 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR3) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

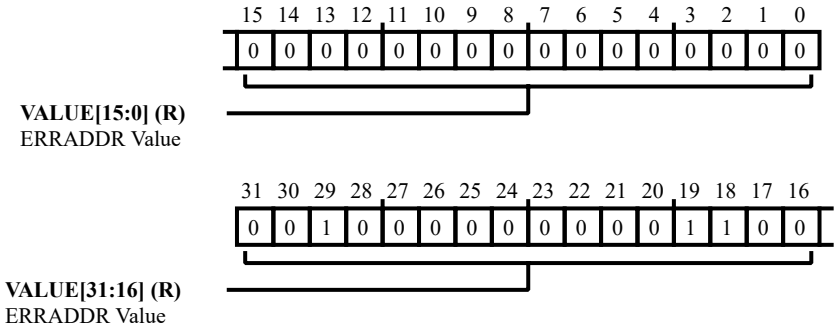


Figure 9-14: L2CTL_ERRADDR3 Register Diagram

Table 9-16: L2CTL_ERRADDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR3.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 4 Register

The L2CTL_ERRADDR4 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR4) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

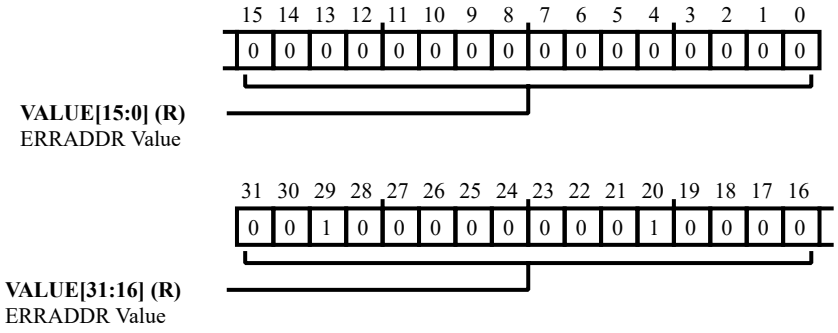


Figure 9-15: L2CTL_ERRADDR4 Register Diagram

Table 9-17: L2CTL_ERRADDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR4.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 5 Register

The L2CTL_ERRADDR5 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR5) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

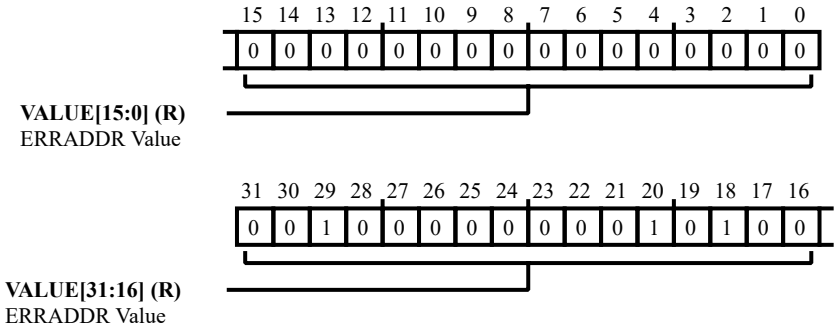


Figure 9-16: L2CTL_ERRADDR5 Register Diagram

Table 9-18: L2CTL_ERRADDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR5.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 6 Register

The L2CTL_ERRADDR6 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR6) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

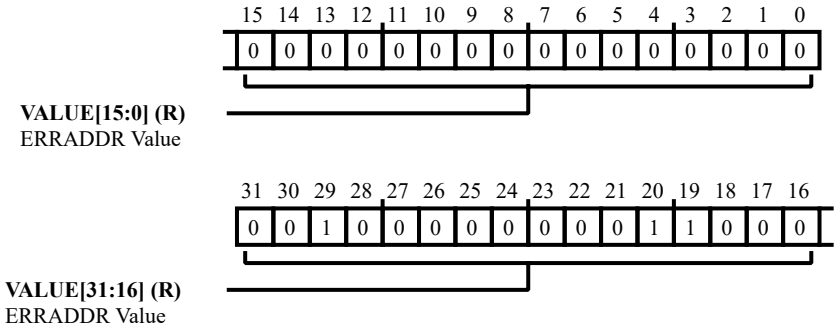


Figure 9-17: L2CTL_ERRADDR6 Register Diagram

Table 9-19: L2CTL_ERRADDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR6.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 7 Register

The L2CTL_ERRADDR7 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR7) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

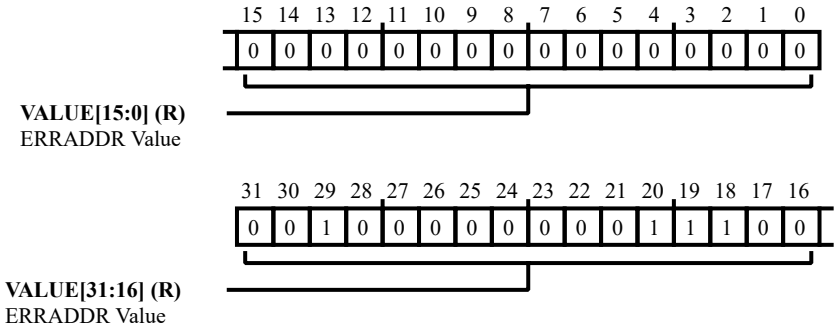


Figure 9-18: L2CTL_ERRADDR7 Register Diagram

Table 9-20: L2CTL_ERRADDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR7.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 8 Register

The L2CTL_ERRADDR8 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR8) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

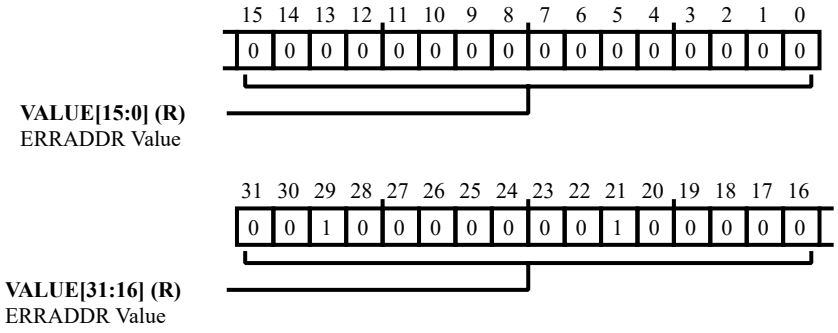


Figure 9-19: L2CTL_ERRADDR8 Register Diagram

Table 9-21: L2CTL_ERRADDR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR8.VALUE bits hold the address containing the ECC double-bit error.

Error Type 0 Register

The L2CTL_ET0 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 0. This register is updated only if the corresponding error status bit L2CTL_STAT.ERR0 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET0 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET0 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR0).

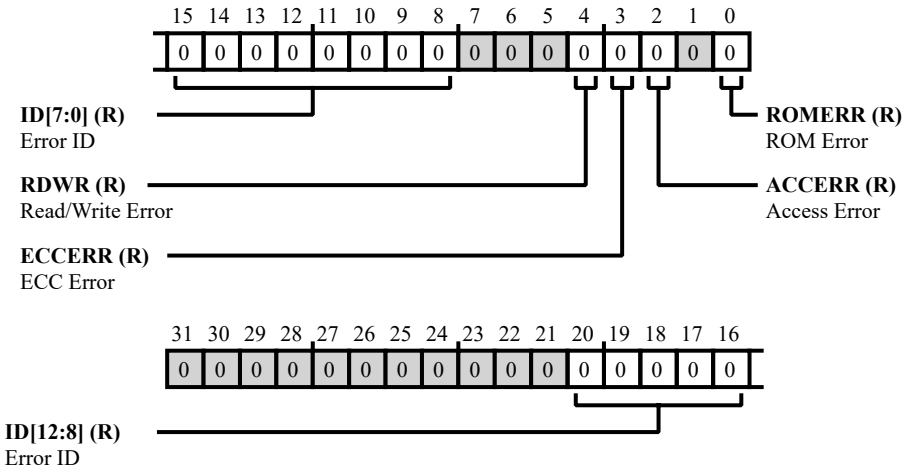


Figure 9-20: L2CTL_ET0 Register Diagram

Table 9-22: L2CTL_ET0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	Error ID. The L2CTL_ET0.ID bits hold the bus requester ID of the access that caused an error.	
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET0.RDWR bit indicates whether a read or write access caused an error.	
		0	Read Access created Error
		1	Write Access created Error
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET0.ECCERR bit indicates whether the access had an ECC double-bit error.	
2 (R/NW)	ACCERR	Access Error. The L2CTL_ET0.ACCERR bit indicates whether the access went to a restricted bank.	

Table 9-22: L2CTL_ET0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	ROMERR	ROM Error. The `L2CTL_ET0.ROMERR` bit indicates whether a write access went to a ROM area.

Error Type 1 Register

The L2CTL_ET1 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 1. This register is updated only if the corresponding error status bit L2CTL_STAT.ERR1 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET1 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET1 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR1).

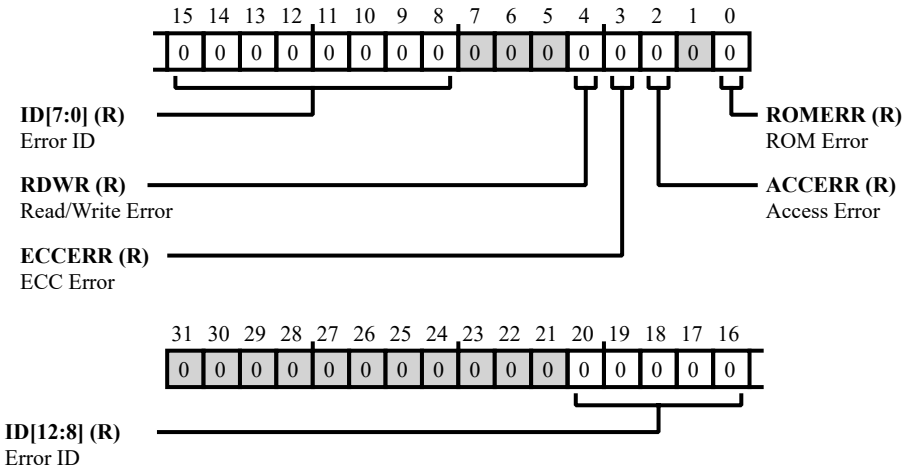


Figure 9-21: L2CTL_ET1 Register Diagram

Table 9-23: L2CTL_ET1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	Error ID. The L2CTL_ET1.ID bits hold the bus requester ID of the access that caused an error.	
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET1.RDWR bit indicates whether a read or write access caused an error.	
		0	Read access created error
		1	Write access created error
3 (R/NW)	ECCERR	ECC Error. If the L2CTL_ET1.ECCERR bit =1, the access had an ECC double-bit error.	
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET1.ACCERR bit =1, the access went to a restricted bank.	
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET1.ROMERR bit =1, a write access went to a ROM area.	

Error Type 2 Register

The L2CTL_ET2 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 2 (cores). This register is updated only if the corresponding error status bit L2CTL_STAT_1.ERR2 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET2 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET2 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR2).

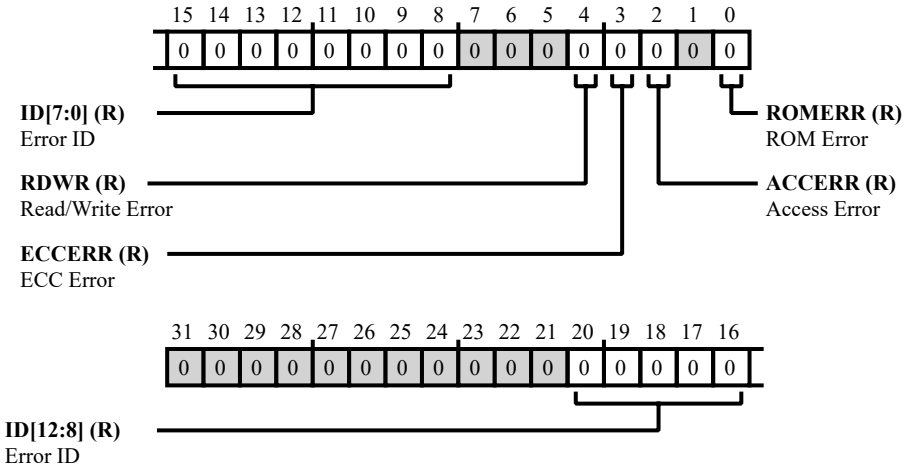


Figure 9-22: L2CTL_ET2 Register Diagram

Table 9-24: L2CTL_ET2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ID	Error ID. The L2CTL_ET2.ID bits hold the bus requester ID of the access that caused an error.
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET2.RDWR bit indicates whether a read or write access caused an error.
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET2.ECCERR bit indicates whether the access had an ECC double-bit error.
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET2.ACCERR bit =1, the access went to a restricted bank.
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET2.ROMERR bit =1, a write access went to a ROM area.

Error Type 3 Register

The L2CTL_ET3 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 3 (DMA). This register is updated only if the corresponding error status bit L2CTL_STAT_1.ERR3 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET3 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET3 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR3).

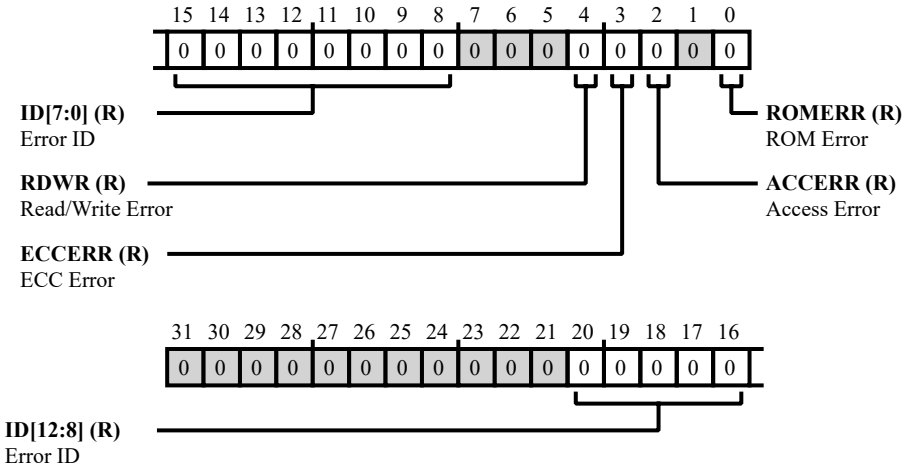


Figure 9-23: L2CTL_ET3 Register Diagram

Table 9-25: L2CTL_ET3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ID	Error ID. The L2CTL_ET3.ID bits hold the bus requester ID of the access that caused an error.
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET3.RDWR bit indicates whether a read or write access caused an error.
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET3.ECCERR bit indicates whether the access had an ECC double-bit error.
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET3.ACCERR bit =1, the access went to a restricted bank.
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET3.ROMERR bit =1, a write access went to a ROM area.

Error Type 4 Register

The L2CTL_ET4 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 4 (DMA). This register is updated only if the corresponding error status bit L2CTL_STAT_1.ERR4 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET4 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET4 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR4).

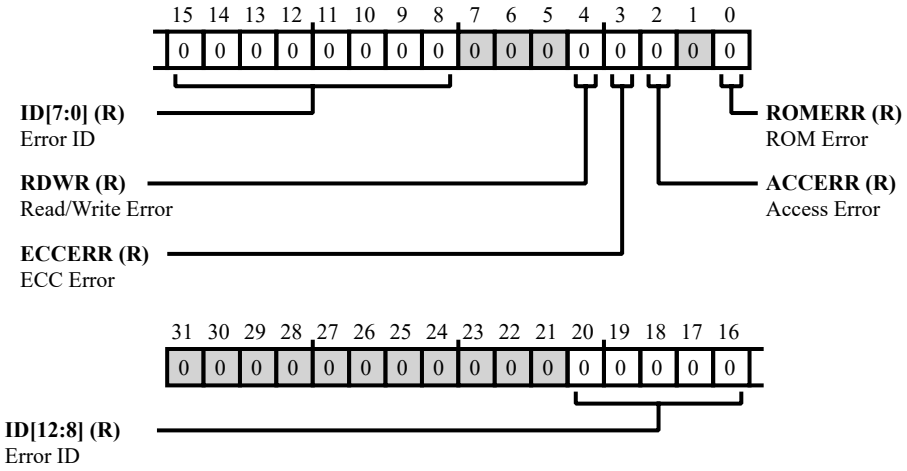


Figure 9-24: L2CTL_ET4 Register Diagram

Table 9-26: L2CTL_ET4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ID	Error ID. The L2CTL_ET4.ID bits hold the bus requester ID of the access that caused an error.
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET4.RDWR bit indicates whether a read or write access caused an error.
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET4.ECCERR bit indicates whether the access had an ECC double-bit error.
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET4.ACCERR bit =1, the access went to a restricted bank.
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET4.ROMERR bit =1, a write access went to a ROM area.

Initialization Register

The L2CTL_INIT register initializes memory banks with 128'b0 and ECC bits corresponding to 128'b0. Any writes to the bits in this register while initialization is occurring to any of the banks is ignored. All bits are W1A (Write 1 for Action).

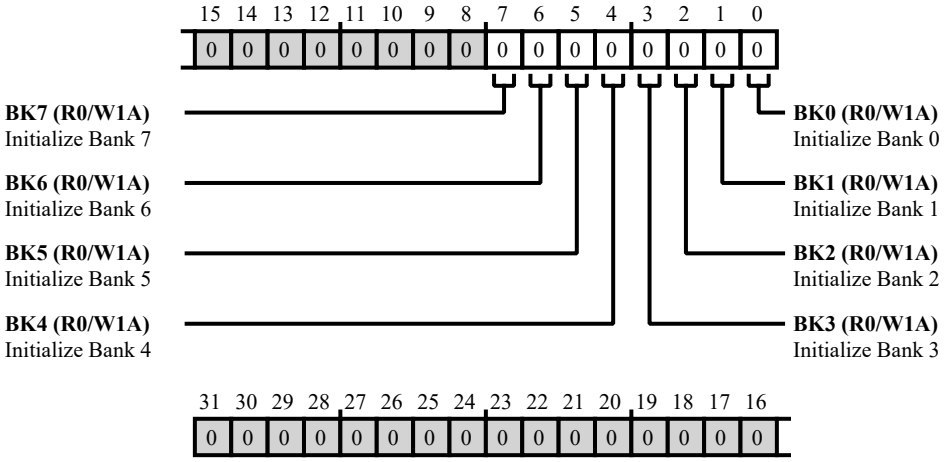


Figure 9-25: L2CTL_INIT Register Diagram

Table 9-27: L2CTL_INIT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R0/W1A)	BK7	Initialize Bank 7. The write-1 to the L2CTL_INIT.BK7 bit initializes bank 7 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
6 (R0/W1A)	BK6	Initialize Bank 6. The write-1 to the L2CTL_INIT.BK6 bit initializes bank 6 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
5 (R0/W1A)	BK5	Initialize Bank 5. The write -1 to the L2CTL_INIT.BK5 bit initializes bank 5 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
4 (R0/W1A)	BK4	Initialize Bank 4. The write-1 to the L2CTL_INIT.BK4 bit initializes bank 4 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.

Table 9-27: L2CTL_INIT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R0/W1A)	BK3	Initialize Bank 3. The write-1 to the `L2CTL_INIT.BK3` bit initializes bank 3 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
2 (R0/W1A)	BK2	Initialize Bank 2. The write-1 to the `L2CTL_INIT.BK2` bit initializes bank 2 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
1 (R0/W1A)	BK1	Initialize Bank 1. The write-1 to the `L2CTL_INIT.BK1` bit initializes bank 1 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
0 (R0/W1A)	BK0	Initialize Bank 0. The write-1 to the `L2CTL_INIT.BK0` bit initializes bank 0 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.

Initialization Status Register

The L2CTL_ISTAT register holds the status of the RAM bank initialization. If set, the corresponding bank is initialized.

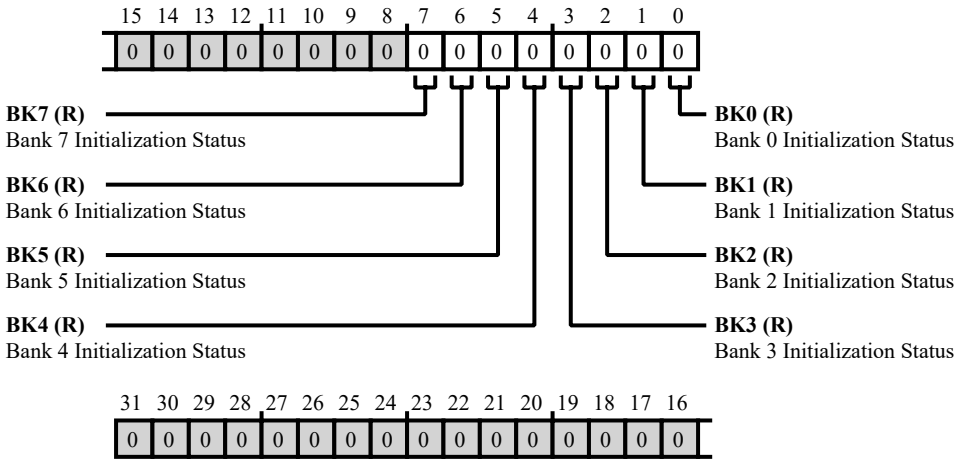


Figure 9-26: L2CTL_ISTAT Register Diagram

Table 9-28: L2CTL_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	BK7	Bank 7 Initialization Status. The L2CTL_ISTAT.BK7 bits hold the bank 7 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
6 (R/NW)	BK6	Bank 6 Initialization Status. The L2CTL_ISTAT.BK6 bits hold the bank 6 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
5 (R/NW)	BK5	Bank 5 Initialization Status. The L2CTL_ISTAT.BK5 bits hold the bank 5 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
4 (R/NW)	BK4	Bank 4 Initialization Status. The L2CTL_ISTAT.BK4 bits hold the bank 4 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
3 (R/NW)	BK3	Bank 3 Initialization Status. The L2CTL_ISTAT.BK3 bits hold the bank 3 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
2 (R/NW)	BK2	Bank 2 Initialization Status. The L2CTL_ISTAT.BK2 bits hold the bank 2 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.

Table 9-28: L2CTL_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	BK1	Bank 1 Initialization Status. The `L2CTL_ISTAT.BK1` bits hold the bank 1 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
0 (R/NW)	BK0	Bank 0 Initialization Status. The `L2CTL_ISTAT.BK0` bits hold the bank 0 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.

Power Control Register

The L2CTL_PCTL register has the various control settings for selectively enabling the deep sleep and shut down power saving features.

NOTE: The corresponding L2 bank should not be accessed if the power control feature is enabled for that bank. An access to a bank in deep sleep/shut down may create unpredictable behavior.

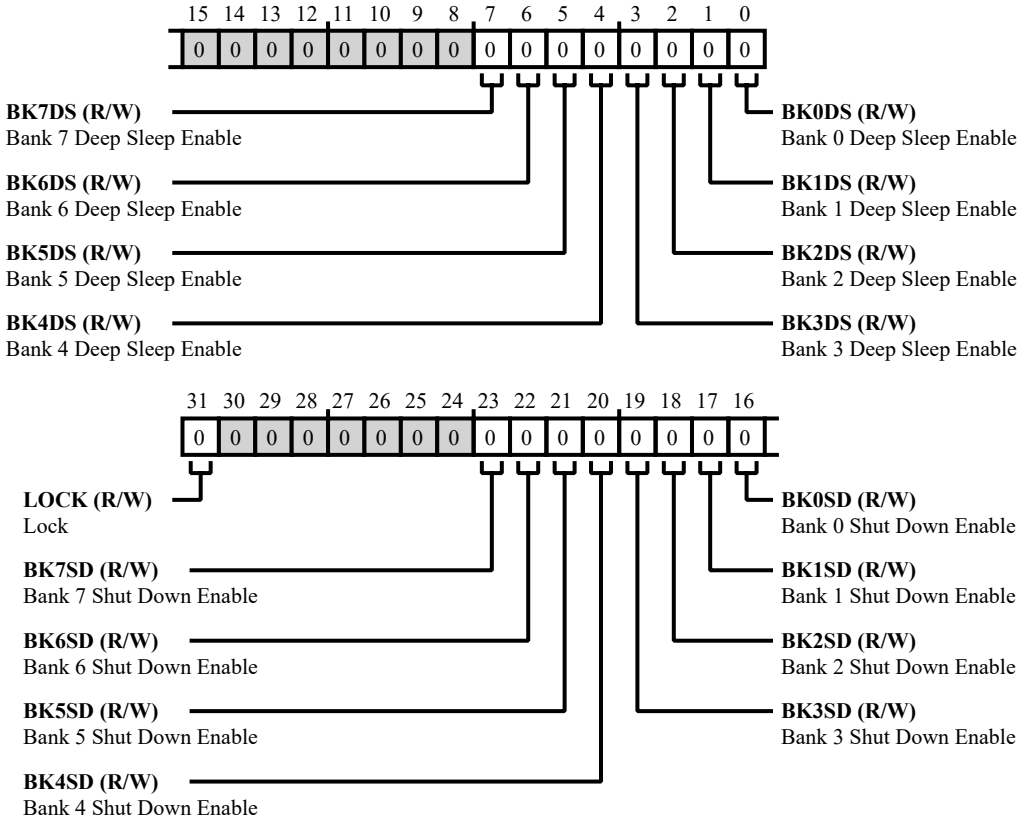


Figure 9-27: L2CTL_PCTL Register Diagram

Table 9-29: L2CTL_PCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the L2CTL_PCTL.LOCK bit is set, the L2CTL_PCTL register is read only (locked).	
		0	Unlock
		1	Lock
23 (R/W)	BK7SD	Bank 7 Shut Down Enable. The L2CTL_PCTL.BK7SD bits enables bank 7 shut down.	

Table 9-29: L2CTL_PCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
22 (R/W)	BK6SD	Bank 6 Shut Down Enable. The L2CTL_PCTL.BK6SD bits enables bank 6 shut down.
21 (R/W)	BK5SD	Bank 5 Shut Down Enable. The L2CTL_PCTL.BK5SD bits enables bank 5 shut down.
20 (R/W)	BK4SD	Bank 4 Shut Down Enable. The L2CTL_PCTL.BK4SD bits enables bank 4 shut down.
19 (R/W)	BK3SD	Bank 3 Shut Down Enable. The L2CTL_PCTL.BK3SD bits enables bank 3 shut down.
18 (R/W)	BK2SD	Bank 2 Shut Down Enable. The L2CTL_PCTL.BK2SD bits enables bank 2 shut down.
17 (R/W)	BK1SD	Bank 1 Shut Down Enable. The L2CTL_PCTL.BK1SD bits enables bank 1 shut down.
16 (R/W)	BK0SD	Bank 0 Shut Down Enable. The L2CTL_PCTL.BK0SD bits enables bank 0 shut down.
7 (R/W)	BK7DS	Bank 7 Deep Sleep Enable. The L2CTL_PCTL.BK7DS bits enables bank 7 deep sleep.
6 (R/W)	BK6DS	Bank 6 Deep Sleep Enable. The L2CTL_PCTL.BK6DS bits enables bank 6 deep sleep.
5 (R/W)	BK5DS	Bank 5 Deep Sleep Enable. The L2CTL_PCTL.BK5DS bits enables bank 5 deep sleep.
4 (R/W)	BK4DS	Bank 4 Deep Sleep Enable. The L2CTL_PCTL.BK4DS bits enables bank 4 deep sleep.
3 (R/W)	BK3DS	Bank 3 Deep Sleep Enable. The L2CTL_PCTL.BK3DS bits enables bank 3 deep sleep.
2 (R/W)	BK2DS	Bank 2 Deep Sleep Enable. The L2CTL_PCTL.BK2DS bits enables bank 2 deep sleep.
1 (R/W)	BK1DS	Bank 1 Deep Sleep Enable. The L2CTL_PCTL.BK1DS bits enables bank 1 deep sleep.
0 (R/W)	BK0DS	Bank 0 Deep Sleep Enable. The L2CTL_PCTL.BK0DS bits enables bank 0 deep sleep.

Revision ID Register

The L2CTL_REVID register provides the L2 Revision ID.

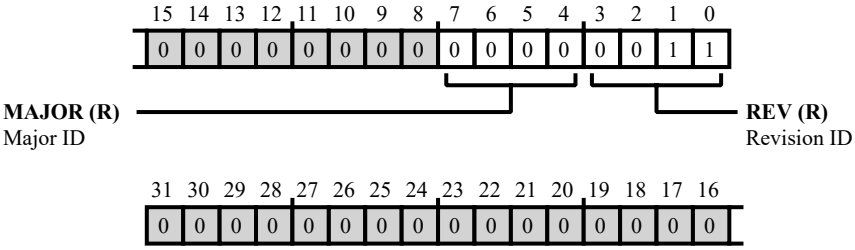


Figure 9-28: L2CTL_REVID Register Diagram

Table 9-30: L2CTL_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major ID. The L2CTL_REVID.MAJOR bit indicates L2 Major ID.
3:0 (R/NW)	REV	Revision ID. The L2CTL_REVID.REV bit indicates the L2 Revision ID.

Read Priority Count Register

The L2CTL_RPCR0 register stores the count value to be used for priority elevation for bus read channels of Port0 to Port3. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number (stored in RPCx) of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request. Default value wait cycle for each channel is 15.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus read channel. A best practice is to program this register before initiating an L2 access.

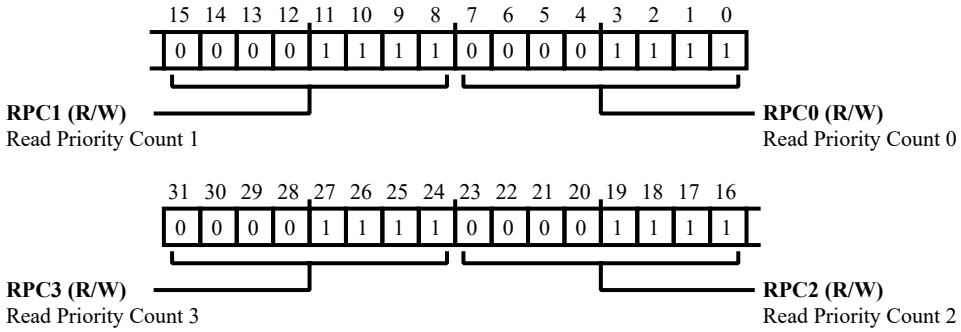


Figure 9-29: L2CTL_RPCR0 Register Diagram

Table 9-31: L2CTL_RPCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RPC3	Read Priority Count 3. The L2CTL_RPCR0.RPC3 bits hold the priority count for L2 bus read channel 3.
23:16 (R/W)	RPC2	Read Priority Count 2. The L2CTL_RPCR0.RPC2 bits hold the priority count for L2 bus read channel 2.
15:8 (R/W)	RPC1	Read Priority Count 1. The L2CTL_RPCR0.RPC1 bits hold the priority count for L2 bus read channel 1.
7:0 (R/W)	RPC0	Read Priority Count 0. The L2CTL_RPCR0.RPC0 bits hold the priority count for L2 bus read channel 0.

Read Priority Count Register

The L2CTL_RPCR1 register stores the count value to be used for priority elevation for bus read channel 4. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number (stored in RPC4) of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request. Default value of RPC4 is 15.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus read channel. A best practice is to program this register before initiating an L2 access.

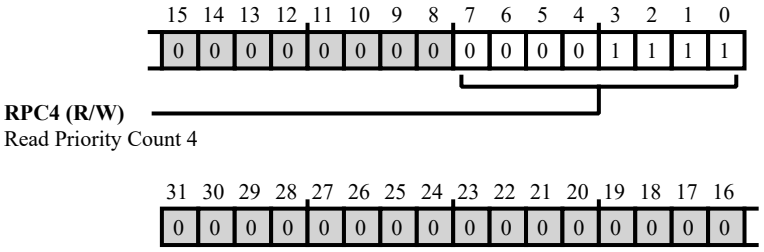


Figure 9-30: L2CTL_RPCR1 Register Diagram

Table 9-32: L2CTL_RPCR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RPC4	Read Priority Count 4. The L2CTL_RPCR1.RPC4 bits hold the priority count for L2 bus read channel 4.

Scrub Start Address Register

The L2CTL_SADR register stores the scrub start address value. Writes to this register can be prevented by setting the L2CTL_SCTL.LOCK bit and the global lock.

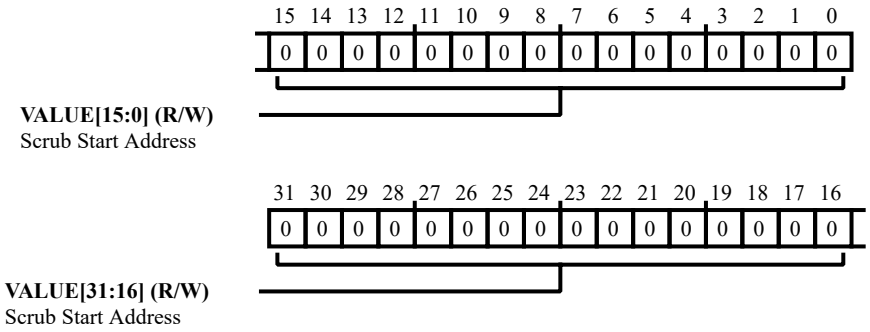


Figure 9-31: L2CTL_SADR Register Diagram

Table 9-33: L2CTL_SADR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Scrub Start Address. The L2CTL_SADR.VALUE bits hold the scrub start address. The writes to this register can be prevented by setting the L2_SCTL.LOCK and the global lock.

Scrub Count Register

The L2CTL_SCNT register determines the number of 128-bit locations scrubbed starting from the start address (L2CTL_SADR register). Writes to the L2CTL_SCNT register can be prevented by setting the L2CTL_CTL.LOCK bit and the global lock.

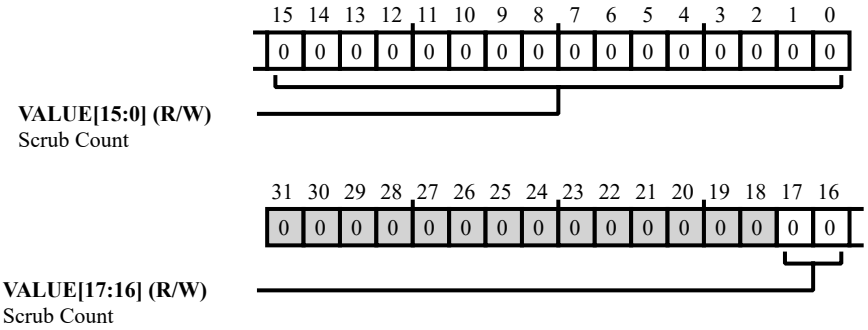


Figure 9-32: L2CTL_SCNT Register Diagram

Table 9-34: L2CTL_SCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:0 (R/W)	VALUE	Scrub Count. The L2CTL_SCNT.VALUE bits determines the number of 128-bit locations scrubbed starting from the start address. Ensure value programmed is less than the total addressable 128-bit L2 RAM locations available.

Scrub Control Register

The L2CTL_SCTL register holds the automatic scrub related controls. This is a read/write register. Memory scrub can be performed by programming the L2CTL_SADR (scrub start address) register with the start address of the scrub and the L2CTL_SCNT register with the total number of 128 -bit addresses to be scrubbed starting from the L2CTL_SADR register. The number of cycles between each scrub can be programmed with the L2CTL_SCTL.SRT bit.

During the scrub the controller issues a read followed by write back if there is a single bit ECC error. Once the L2 controller completes scrubbing the memory region mentioned using the L2CTL_SADR and the L2CTL_SCNT registers, an interrupt is generated and scrubbing re-starts from the start unless the scrub enable bit is disabled in the control register. If scrub enable is cleared before completing the address range used in the L2CTL_SADR and L2CTL_SCNT registers, the scrub stops after completing any already issued scrub access.

The scrub read/writes always start from the full 128-bit equivalent of the address written into the L2CTL_SADR register. A 128-bit value is always read and the 128-bit value is written back. The scrub access has the highest priority. Programs can configure 8-, 16-, or 32-bit addresses in the L2CTL_SADR register but the lower 3 bits are treated as don't care because the internal memory array is always accessed in a 128-bit mode.

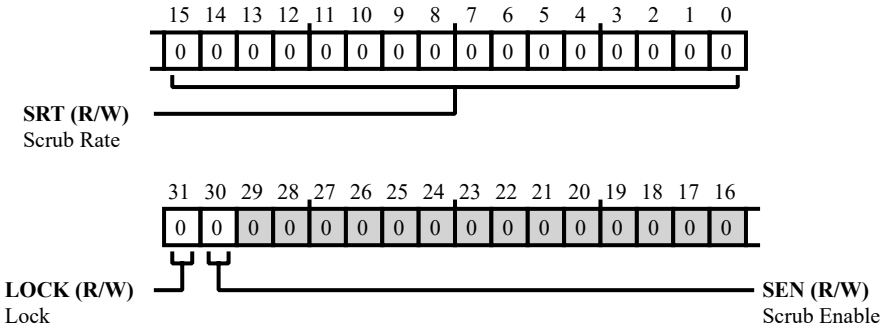


Figure 9-33: L2CTL_SCTL Register Diagram

Table 9-35: L2CTL_SCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the L2CTL_SCTL.LOCK bit is set, the L2CTL_SCTL register is read only (locked).	
		0	Unlock
		1	Lock
30 (R/W)	SEN	Scrub Enable. The L2CTL_SCTL.SEN bits enable automatic scrub.	

Table 9-35: L2CTL_SCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	SRT	Scrub Rate. The `L2CTL_SCTL.SRT` bits determines the number of clock cycles that elapsed between two automatic scrubs.

Status Register

The L2CTL_STAT register indicates ECC error status, refresh register status, and bus error status of Port0 and Port1.

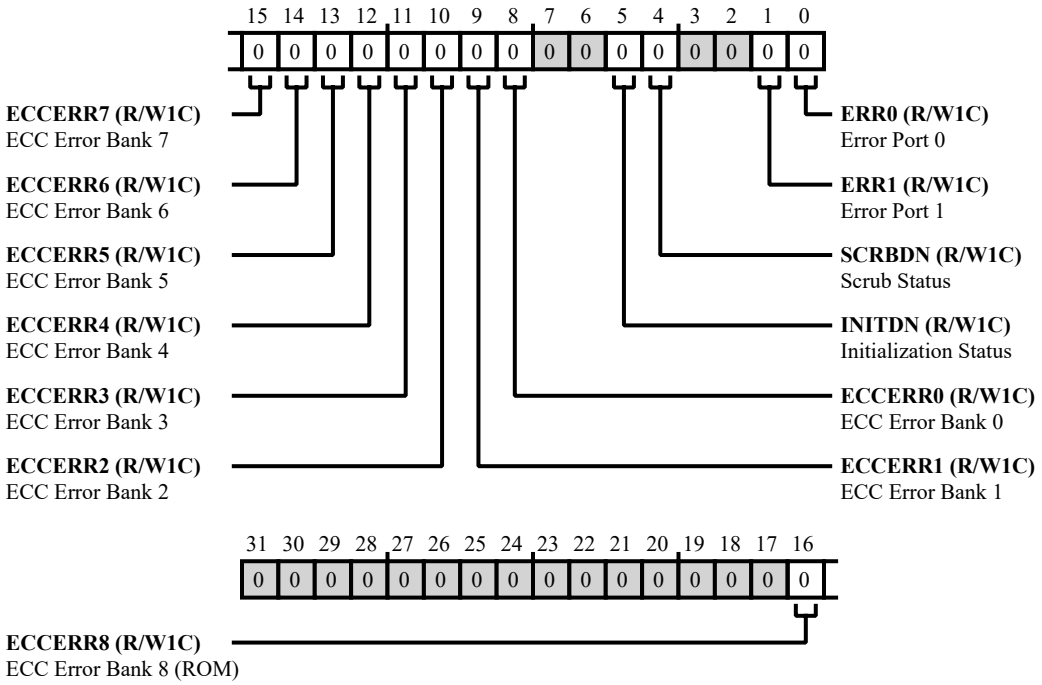


Figure 9-34: L2CTL_STAT Register Diagram

Table 9-36: L2CTL_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	ECCERR8	ECC Error Bank 8 (ROM). The L2CTL_STAT.ECCERR8 bit indicates that an ECC double-bit error occurred inside L2 bank 8 (ROM).	
		0	No Status
		1	ECC Double Bit Error
15 (R/W1C)	ECCERR7	ECC Error Bank 7. The L2CTL_STAT.ECCERR7 bit indicates that an ECC double-bit error occurred inside L2 bank 7.	
		0	No Status
		1	ECC Double Bit Error

Table 9-36: L2CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W1C)	ECCERR6	ECC Error Bank 6. The `L2CTL_STAT.ECCERR6` bit indicates that an ECC double-bit error occurred inside L2 bank 6.	
		0	No Status
		1	ECC Double Bit Error
13 (R/W1C)	ECCERR5	ECC Error Bank 5. The `L2CTL_STAT.ECCERR5` bit indicates that an ECC double-bit error occurred inside L2 bank 5.	
		0	No Status
		1	ECC Double Bit Error
12 (R/W1C)	ECCERR4	ECC Error Bank 4. The `L2CTL_STAT.ECCERR4` bit indicates that an ECC double-bit error occurred inside L2 bank 4.	
		0	No Status
		1	ECC Double Bit Error
11 (R/W1C)	ECCERR3	ECC Error Bank 3. The `L2CTL_STAT.ECCERR3` bit indicates that an ECC double-bit error occurred inside L2 bank 3.	
		0	No Status
		1	ECC Double Bit Error
10 (R/W1C)	ECCERR2	ECC Error Bank 2. The `L2CTL_STAT.ECCERR2` bit indicates that an ECC double-bit error occurred inside L2 bank 2.	
		0	No Status
		1	ECC Double Bit Error
9 (R/W1C)	ECCERR1	ECC Error Bank 1. The `L2CTL_STAT.ECCERR1` bit indicates that an ECC double-bit error occurred inside L2 bank 1.	
		0	No Status
		1	ECC Double Bit Error

Table 9-36: L2CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	ECCERR0	ECC Error Bank 0. The `L2CTL_STAT.ECCERR0` bit indicates that an ECC double-bit error occurred inside L2 bank 0.	
		0	No Status
		1	ECC Double Bit Error
5 (R/W1C)	INITDN	Initialization Status. The `L2CTL_STAT.INITDN` bit indicates whether initialization has been completed.	
		0	Initialization Not Complete
		1	Initialization Completed
4 (R/W1C)	SCRBDN	Scrub Status. The `L2CTL_STAT.SCRBDN` bit indicates whether a round of memory scrub has completed.	
		0	Scrub Not Complete
		1	Scrub Completed
1 (R/W1C)	ERR1	Error Port 1. The `L2CTL_STAT.ERR1` indicates whether the L2CTL has detected a bus access error on L2s bus port 1.	
		0	No Error
		1	Bus Access Error
0 (R/W1C)	ERR0	Error Port 0. The `L2CTL_STAT.ERR0` indicates whether the L2CTL has detected a bus access error on L2s bus port 0.	
		0	No Error
		1	Bus Access Error

L2 Port Error Status Register

The L2CTL_STAT_1 register stores the AXI Error Status from Port2 onwards. The register bits are sticky - once set it has to be explicitly reset by writing a 1 to that particular bit.

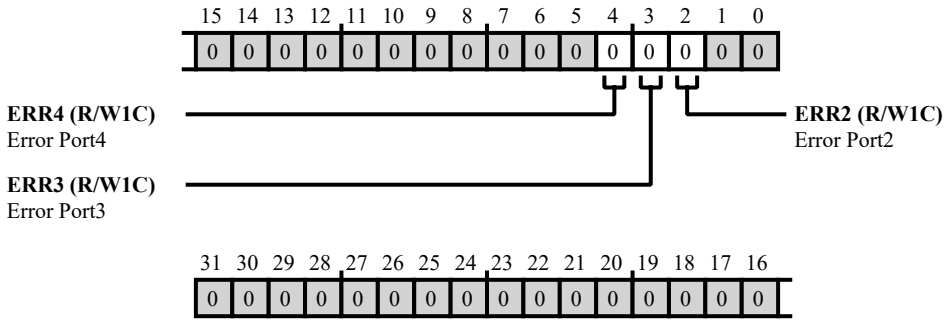


Figure 9-35: L2CTL_STAT_1 Register Diagram

Table 9-37: L2CTL_STAT_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W1C)	ERR4	Error Port4. The L2CTL_STAT_1.ERR4 indicates whether the L2CTL has detected a bus access error on L2s bus port 4.
3 (R/W1C)	ERR3	Error Port3. The L2CTL_STAT_1.ERR3 indicates whether the L2CTL has detected a bus access error on L2s bus port 3.
2 (R/W1C)	ERR2	Error Port2. The L2CTL_STAT_1.ERR2 indicates whether the L2CTL has detected a bus access error on L2s bus port 2.

Write Priority Count Register

The L2CTL_WPCR0 register stores the count value to be used for priority elevation for bus write channels. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus write channel. A best practice is to program this register before initiating an L2 access.

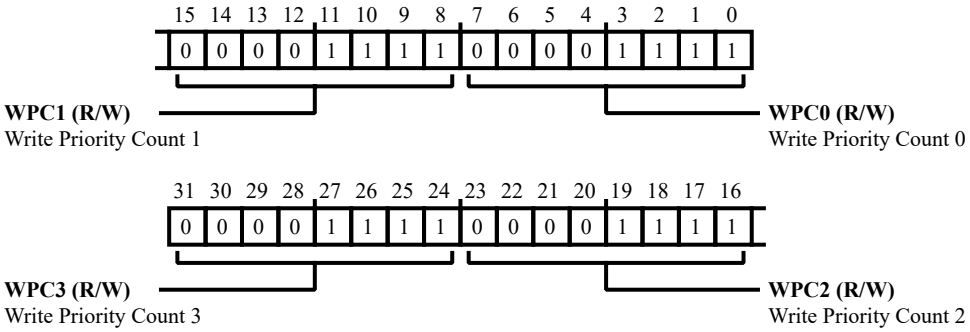


Figure 9-36: L2CTL_WPCR0 Register Diagram

Table 9-38: L2CTL_WPCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	WPC3	Write Priority Count 3. The L2CTL_WPCR0.WPC3 bits hold the priority count for L2 bus write channel 3.
23:16 (R/W)	WPC2	Write Priority Count 2. The L2CTL_WPCR0.WPC2 bits hold the priority count for L2 bus write channel 2.
15:8 (R/W)	WPC1	Write Priority Count 1. The L2CTL_WPCR0.WPC1 bits hold the priority count for L2 bus write channel 1.
7:0 (R/W)	WPC0	Write Priority Count 0. The L2CTL_WPCR0.WPC0 bits hold the priority count for L2 bus write channel 0.

Write Priority Count Register

The L2CTL_WPCR1 register stores the count value to be used for priority elevation for bus write channels. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus write channel. A best practice is to program this register before initiating an L2 access.

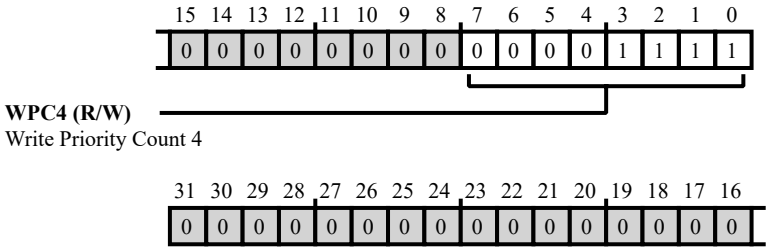


Figure 9-37: L2CTL_WPCR1 Register Diagram

Table 9-39: L2CTL_WPCR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	WPC4	Write Priority Count 4. The L2CTL_WPCR1.WPC4 bits hold the priority count for L2 bus write channel 4.

10 DDR Pre-fetch Buffer (DDRPFB)

The DDR Pre-fetch buffer is designed to reduce the latency for fetching instructions or reading data from DDR memory. The Pre-fetch buffer works in the background to reduce the latency of a subsequent data or instruction access by pre-fetching the next data for next address (using the principle of locality). The DDR pre-fetch buffer is integrated in the DDR path and uses the following features.

- SMART pre-fetching decision which is a direct access or pre-fetch based on the pattern seen in the previous read transactions issued
- ID based direct access where in a multi-controller scenario with more than four controllers avoids a slow controller (for example the SPORT) to flush out a buffer line that is being used by a fast requester (for example HSMDMA)
- Range based return zero feature

The *Pre-fetch Buffer Block Diagram* figure shows the functional blocks within the DDRPFB.

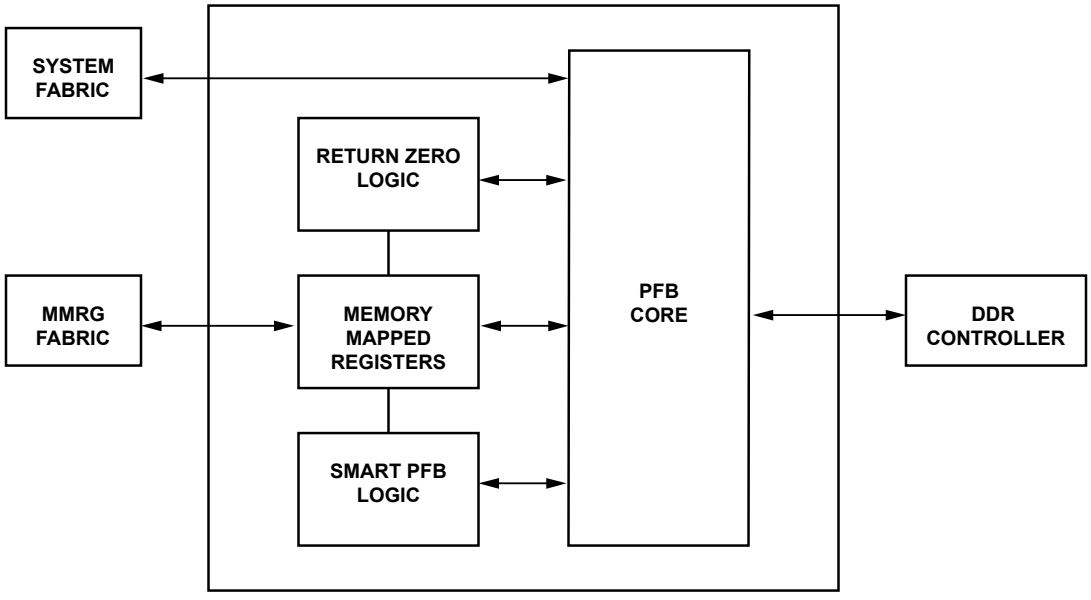


Figure 10-1: Pre-fetch Buffer Block Diagram

DDRPFB Features

The DDRPFB supports the following features:

- Smart Pre-fetch
- Direct DMC access
- Return Zero
- Controller ID based direct access
- Range based pre-fetch

Smart Pre-fetch

Smart pre-fetch is used for the worst case scenario where the controller fetches data at non continuous addresses which results no hits in the DDRPFB. To avoid fetching excess data for such non continuous read transactions, smart pre-fetching choses to start to fetch and/or pre-fetch requests from the DDRPFB to the DMC or gives direct access for such read requests. The decision is made based on the history of read transactions received at the input of the prefetch buffer. The `PFB_CTL1.SMART_FEATURE_EN` bit is used to enable this feature.

Data Invalidation

When the `PFB_CTL0.DATA_INVALIDATION` bit is set (=1), the complete pre-fetch buffer is invalidated. Clear this bit manually after setting it to make use of pre-fetch buffer.

Direct Access

In direct access, there are no pre-fetch accesses and the accesses are forwarded to the DMC directly. There is a one cycle latency while the pre-fetch buffer decides whether the access is direct or not. The read data from the DMC is directly forwarded to the controller without any latency impact and is not stored in the buffer.

Return Zero

This feature allows the program to configure a range of addresses range in L3 memory to always returns zero. To enable this feature, set (=1) the `PFB_CTL0.RETURN_ZERO` bit. The address range which require all zeros to be read is configured using the `PFB_ZERO_START.VAL` and `PFB_ZERO_END.VAL` registers. All the requests which that fall between the start and end addresses (that are aligned to 256 bytes) are returned with all zeros.

Controller ID Based Direct Access

This section explains how to configure a specific SCB controller for direct access or DDR pre-fetch access. Set the `PFB_CTL1.RQSTR_ID_FEATURE_EN` bit = 1 to use this feature. The following registers and bits are used to manage ID based direct access.

- The SCB controller ID is the ID of the SCB requester that is requesting data from the DDR. For ID information, see .

- The PFB_RQSTR_ID_EN.VAL bit field has four bits. It can take all 16 combinations. If this bit is cleared (=0) the corresponding SCB requester has direct access.
- The PFB_RQSTR0_ID, PFB_RQSTR1_ID, PFB_RQSTR2_ID and PFB_RQSTR3_ID registers are four IDs that are preprogrammed with the SCB ID. The configuration of these registers determine whether the DDRPFB gives a normal PFB enabled access or a direct access (if the corresponding PFB_RQSTR_ID_EN.VAL bit is set). If the SCB ID of the requester matches with the DDRPFB_MSTRx_ID, then the DDR pre-fetch operation is enabled.
- The PFB_RQSTR0_ID_MSK, PFB_RQSTR1_ID_MSK, PFB_RQSTR2_ID_MSK and PFB_RQSTR3_ID_MSK: When the PFB_RQSTR_ID_EN bit is set for some DDRPFB_MSTRx_ID, then the PFB enabled access is provided for a partial ID match of the SCB ID with the corresponding preprogrammed DDRPFB_MSTRx_ID. Clearing a bit masks the corresponding bit and the masked bits are not checked for a match.

The *DDRPFB Access Configurations* table shows the register configurations for different access types.

Table 10-1: DDRPFB Access Configurations

SCB Requester ID	DDRPFB_MSTR_ID_EN[0]	DDRPFB_MSTR0_ID	DDRPFB_MSTR0_ID_MASK	Access Type
XXXX	0	XXXX	XXXX	Direct access Irrespective of SCB requester ID, PFB_RQSTR0_ID and PFB_RQSTR0_ID_MSK, direct access is given to all the requests.
MMMM	1	NNNN	0000	DDRPFB enabled access DDRPFB does not check any bits of the SCB requester ID if it matches with PFB_RQSTR0_ID. Therefore, DDRPFB performs prefetch operation for all IDs. This is an example where DDR prefetch is enabled, though none of the bits of SCB ID matches with PFB_RQSTR0_ID.
MMXX	1	MMXX	1100	DDRPFB enabled access DDRPFB looks for only two MSBs from SCB ID matching the two MSBs of PFB_RQSTR0_ID. If they match, the DDRPFB performs a prefetch operation. Otherwise, direct access is given to such requests. In this example the two MSBs matching with the two MSBs of PFB_RQSTR0_ID configures a normal DDRPFB enabled access.
MMXX	1	NNXX	1100	Direct access The PFB looks for only the two MSBs from the SCB ID that match with the two MSBs of PFB_RQSTR0_ID. If they match, the PFB performs a normal operation. Otherwise, direct access is given to such requests. In this example, since the two MSBs do not match with the two MSBs of PFB_RQSTR0_ID, it is a direct access.

Range Based Pre-fetch

Range based prefetch allows an entire address range to be divided into 16 address ranges to selectively disable the pre-fetch buffer (while pre-fetching is enabled) for this address range using the PFB_RANGE_SELECT register. For example, the *Range Based Pre-fetch* table shows that if PFB_RANGE_SELECT [9] = 0, then the PFB is disabled for address range 90000000 to 9FFFFFFF.

The PFB_CTL1.RANGE_SHIFT bit field is used to configure pre-fetching 16 sections of the address map based on the most significant nibble in the input address. In the example PFB_CTL1.RANGE_SHIFT is 28.

Table 10-2: Range Based Pre-fetch

Sr. No.		PFB Operation
1	PFB_RANGE_SELECT [ARADDR_IN[31:28]] = 1 && PFB_CTL0.PREFETCH_EN = 1	PFB enabled operation
2	PFB_RANGE_SELECT [ARADDR_IN[31:28]] = 0 && PFB_CTL0.PREFETCH_EN = 1	PFB disabled operation
3	PFB_RANGE_SELECT [ARADDR_IN[31:28]] = X && PFB_CTL0.PREFETCH_EN = 0	PFB disabled operation

Table 10-3: Range Based Address Range Selections

Input bit	Address range for Pre-fetch
RANGE_SELECT[0]	00000000 – 0FFFFFFF
RANGE_SELECT[1]	10000000 – 1FFFFFFF
RANGE_SELECT[2]	20000000 – 2FFFFFFF
RANGE_SELECT[3]	30000000 – 3FFFFFFF
RANGE_SELECT[4]	40000000 – 4FFFFFFF
RANGE_SELECT[5]	50000000 – 5FFFFFFF
RANGE_SELECT[6]	60000000 – 6FFFFFFF
RANGE_SELECT[7]	70000000 – 7FFFFFFF
RANGE_SELECT[8]	80000000 – 8FFFFFFF
RANGE_SELECT[9]	90000000 – 9FFFFFFF
RANGE_SELECT[10]	A0000000 – AFFFFFFF
RANGE_SELECT[11]	B0000000 – BFFFFFFF
RANGE_SELECT[12]	C0000000 – CFFFFFFF
RANGE_SELECT[13]	D0000000 – DFFFFFFF
RANGE_SELECT[14]	E0000000 – EFFFFFFF
RANGE_SELECT[15]	F0000000 – FFFFFFFF

Programming Notes

The following guidelines should be noted when using range based pre-fetch.

- PFB_RANGE_SELECT.VAL is used to change the prefetch range
- When PFB_RANGE_SELECT.VAL = 5'bN, then ARADDR_IN[N+3:N] are used for range selection (here, N is programmable number)
  - When PFB_CTL1.RANGE_SHIFT = 28, logic behavior is same as above table
  - When PFB_CTL1.RANGE_SHIFT = 26, then ARADDR_IN[29:26] are used for range selection

For finer control of a small range the configurations in the following table can be used.

Sr. No.		PFB Operation
1	PFB_RANGE_SELECT [ARADDR_IN[N+3:N]] = 1 && PFB_CTL0.PREFETCH_EN = 1	PFB enabled operation
2	PFB_RANGE_SELECT [ARADDR_IN[N+3:N]] = 0 && PFB_CTL0.PREFETCH_EN = 1	PFB disabled operation
3	PFB_RANGE_SELECT [ARADDR_IN[N+3:N]] = X(don't care) && PFB_CTL0.PREFETCH_EN = 0	PFB disabled operation

ADSP-2184x PFB Register List

PFB

Table 10-4: ADSP-2184x PFB Register List

Name	Description
PFB_BCR_END	Prefetch buffer fourth control register
PFB_BCR_START	Prefetch buffer third control register
PFB_CTL0	Control Register 0
PFB_CTL1	Control Register 1
PFB_RANGE_SELECT	Range Select Register
PFB_RQSTR0_ID	Requester ID 0 Register
PFB_RQSTR0_ID_MSK	Requester ID 0 Mask Register
PFB_RQSTR1_ID	Requester ID 1 Register
PFB_RQSTR1_ID_MSK	Requester ID 1 Mask Register
PFB_RQSTR2_ID	Requester ID 2 Register
PFB_RQSTR2_ID_MSK	Requester ID 2 Mask Register
PFB_RQSTR3_ID	Requester ID 3 Register

Table 10-4: ADSP-2184x PFB Register List (Continued)

Name	Description
PFB_RQSTR3_ID_MSK	Requester ID 3 Mask Register
PFB_RQSTR_ID_EN	Requester ID Enable Register
PFB_STS0	Prefetch Buffer First Status Register
PFB_ZERO_END	Return Zero End Address Register
PFB_ZERO_START	Return Zero Start Address Register

DDRPFB Programming Model

The following sections describe how to program the DDRPFB.

Enabling the Pre-fetch Buffer and PFB Idle State

Set the PFB_CTL0.PREFETCH_EN bit (=1) to enable the DDR pre-fetch buffer.

When the PFB_STS0.PFB_IDLE bit = 1 there are no pending transactions between the DDRPFB and the DMC. When this bit = 0, there is at least one pending transaction between the DDRPFB and the DMC.

General Programming Guidelines

The following are few guidelines which need to be followed for proper functioning of DDR PFB.

- The PFB_STS0.PFB_IDLE indicates whether the DDRPFB is IDLE or not and can be polled to add delay to allow pending requests from the PFB to completer get completed.
- To issue a DMC reset during operation, the following steps are used to maintain data coherence and prevent system hangs.
  1. Block all requesters that are currently accessing DDR
  2. Check that the PFB is in the IDLE state by checking PFB_STS0.PFB_IDLE
  3. Issue a DMC reset
  4. Invalidate the PFB
  5. Unblock all requesters that need access to DDR
- During DDR initialization the PFB should be disabled. If the program needs to have the PFB enabled during DDR initialization, perform a PFB invalidation after DDR initialization is completed.
- The maximum possible value of PFB_CTL1.RANGE_SHIFT is ‘b11100. Any value that is more than ‘b11100 has no significance because the address bus width is 32-bits.

Guidelines For Achieving Maximum Performance

To achieve maximum performance, please note the following:

- Each DDR PFB line can hold 256 bytes of data. To achieve maximum performance the starting address of sequential memory accesses should be 256 bytes aligned
- Enable DDRPFB for requesters that support burst mode
- Burst mode with a burse size of 128 bits provide good performance compared to other combinations of burst size less than 128 bits
- Use the Requester ID based direct access feature for the SPORTs and IIR when there are more than four requesters acting at a time.
- Multiple requesters can be combined as a one requester when they are fetching data in interleaved fashion. This is accomplished by placing data in DDR in such a way that (for example), address A is for requester 1, A+x for requester 2, A+2x for again requester 1 and A +3x for requester 2. So in this example for requester 1 addresses A + 2nx are used and for requester 2 addresses A + (2n+1)x are used, where n = 0,1,2, and so on.
- Avoiding wrap access – Data defined in source code can be placed in memory in such a way that all requests from requester A55 is cache line aligned so that all requests from cache onwards are incremental accesses.
- The decision should be made on what is most suitable with respect to performance out of the various DDRPFB configurations, including DDRPFB disabled. In most cases PFB = enabled and the smart feature = enabled has better performance. For few use-cases PFB = enabled, smart feature disabled work best.

ADSP-2184x PFB Register Descriptions

Prefetch buffer mmr registers (PFB) contains the following registers.

Table 10-5: ADSP-2184x PFB Register List

Name	Description
PFB_BCR_END	Prefetch buffer fourth control register
PFB_BCR_START	Prefetch buffer third control register
PFB_CTL0	Control Register 0
PFB_CTL1	Control Register 1
PFB_RANGE_SELECT	Range Select Register
PFB_RQSTR0_ID	Requester ID 0 Register
PFB_RQSTR0_ID_MSK	Requester ID 0 Mask Register
PFB_RQSTR1_ID	Requester ID 1 Register
PFB_RQSTR1_ID_MSK	Requester ID 1 Mask Register
PFB_RQSTR2_ID	Requester ID 2 Register
PFB_RQSTR2_ID_MSK	Requester ID 2 Mask Register
PFB_RQSTR3_ID	Requester ID 3 Register
PFB_RQSTR3_ID_MSK	Requester ID 3 Mask Register

Table 10-5: ADSP-2184x PFB Register List (Continued)

Name	Description
PFB_RQSTR_ID_EN	Requester ID Enable Register
PFB_STS0	Prefetch Buffer First Status Register
PFB_ZERO_END	Return Zero End Address Register
PFB_ZERO_START	Return Zero Start Address Register

Prefetch buffer fourth control register

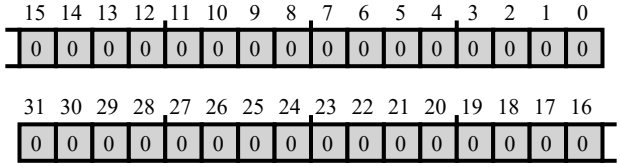


Figure 10-2: PFB_BCR_END Register Diagram

Table 10-6: PFB_BCR_END Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	BCR_END	End address of range in which fetch and prefetch requests launched at 4K boundary crossing.

Prefetch buffer third control register

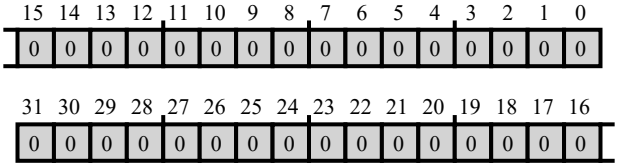


Figure 10-3: PFB_BCR_START Register Diagram

Table 10-7: PFB_BCR_START Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	BCR_START	Start address of range in which fetch and prefetch requests launched at 4K boundary crossing.

Control Register 0

The PFB_CTL0 register disables prefetching for up to sixteen address ranges. Bits[31:16] bits control the range selection for the instruction cache, and bits [15:0] control the range selection for the data cache.

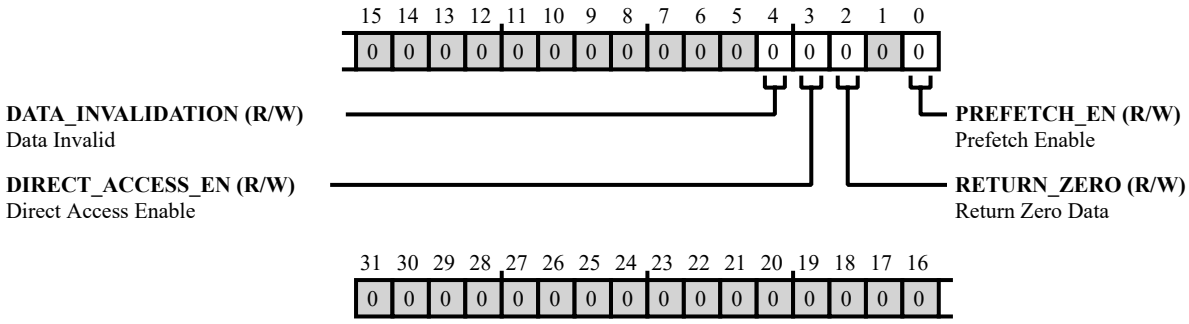


Figure 10-4: PFB_CTL0 Register Diagram

Table 10-8: PFB_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	DATA_INVALIDATION	Data Invalid. When the PFB_CTL0.DATA_INVALIDATION bit is set the prefetch buffer is invalid. The program must clear this bit manually.	
		0	Buffer valid
		1	Buffer invalid
3 (R/W)	DIRECT_ACCESS_EN	Direct Access Enable. When the PFB_CTL0.DIRECT_ACCESS_EN bit is set (=1) the prefetch buffer passes requests as is from a requester to a completer. No excess data is fetched and stored in prefetch buffer. When the PFB_CTL0.DIRECT_ACCESS_EN bit is cleared (=0) the prefetch buffer behaves as if it is disabled.	
		0	Direct access disabled
		1	Direct access enabled
2 (R/W)	RETURN_ZERO	Return Zero Data. When the PFB_CTL0.RETURN_ZERO bit is set, the prefetch buffer returns zero data for the address range configured in the PFB_ZERO_START and PFB_ZERO_END registers. The addresses in the PFB_ZERO_START and PFB_ZERO_END registers should be 128 byte aligned.	
		0	Return zero disabled
		1	Return zero data in selected address range

Table 10-8: PFB_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PREFETCH_EN	Prefetch Enable. When the `PFB_CTL0.PREFETCH_EN` bit is set (=1), the prefetch buffer is enabled. Otherwise, the prefetch buffer is disabled.	
		0	Prefetch disabled
		1	Prefetch enabled

Control Register 1

The PFB_CTL1 register enables smart prefetch, range shifting, and direct access.

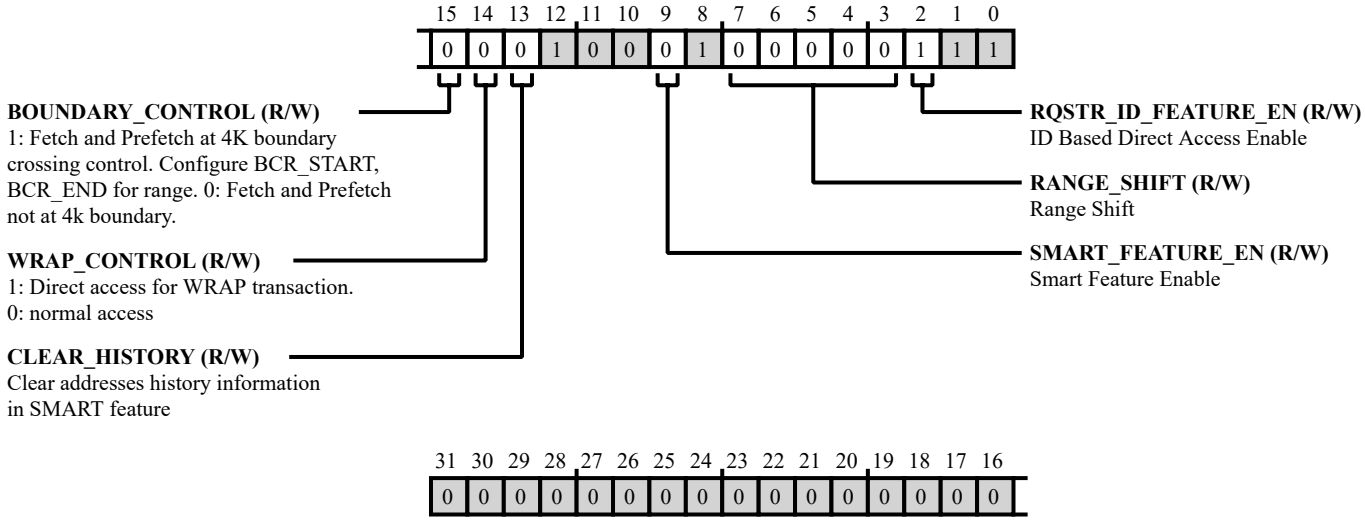


Figure 10-5: PFB_CTL1 Register Diagram

Table 10-9: PFB_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	BOUNDARY_CONTROL	1: Fetch and Prefetch at 4K boundary crossing control. Configure BCR_START, BCR_END for range. 0: Fetch and Prefetch not at 4k boundary..	
14 (R/W)	WRAP_CONTROL	1: Direct access for WRAP transaction. 0: normal access.	
13 (R/W)	CLEAR_HISTORY	Clear addresses history information in SMART feature.	
9 (R/W)	SMART_FEATURE_EN	Smart Feature Enable. Setting the PFB_CTL1.SMART_FEATURE_EN bit enables the SMART feature.	
		0	Smart prefetch disabled
		1	Smart prefetch enabled
7:3 (R/W)	RANGE_SHIFT	Range Shift. The PFB_CTL1.RANGE_SHIFT bit field determines which bits of the read address (that appear at the input of prefetch buffer) are used for the range select operation. By default this is the four MSBs are used. When this bit field is programmed to 5hN then [N+3:N] address bits are used for the range select operation.	

Table 10-9: PFB_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	RQSTR_ID_FEATURE_EN	ID Based Direct Access Enable. The `PFB_CTL1.RQSTR_ID_FEATURE_EN` bit, when set (=1) enables the ID based direct access feature.	
		0	ID based direct access disabled
		1	ID based direct access enabled

Range Select Register

The PFB_RANGE_SELECT register determines whether the prefetch buffer is enabled or disabled in the specified address range.

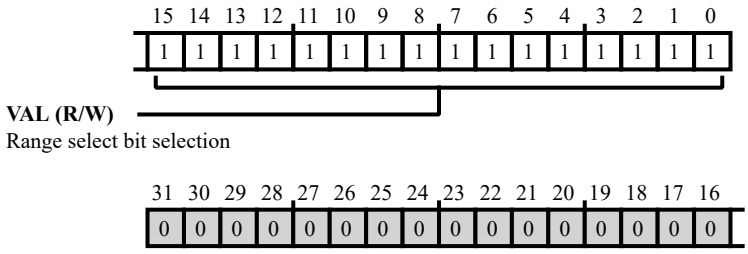


Figure 10-6: PFB_RANGE_SELECT Register Diagram

Table 10-10: PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VAL	Range select bit selection. Each bit in this bit field determines prefetch buffer should be enabled or disabled in that address range. This bit field is used in conjunction with RANGE_SHIFT bit field.

Requester ID 0 Register

The PFB_RQSTR0_ID register is used to configure the first requester ID for controller ID based direct access operations.

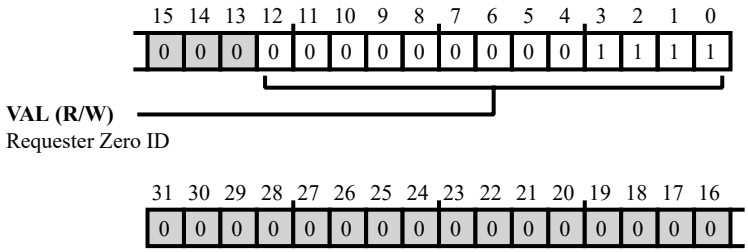


Figure 10-7: PFB_RQSTR0_ID Register Diagram

Table 10-11: PFB_RQSTR0_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Zero ID. The PFB_RQSTR0_ID.VAL bit field identifies the first requester ID to be considered for ID based direct access.

Requester ID 0 Mask Register

The PFB_RQSTR0_ID_MSK register is used to mask the first requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

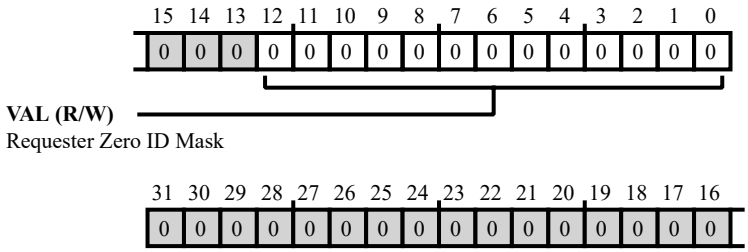


Figure 10-8: PFB_RQSTR0_ID_MSK Register Diagram

Table 10-12: PFB_RQSTR0_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Zero ID Mask. The PFB_RQSTR0_ID_MSK.VAL bit field masks the first requester ID to exclude from ID based direct access.

Requester ID 1 Register

The PFB_RQSTR1_ID register is used to configure the first requester ID for controller ID based direct access operations.

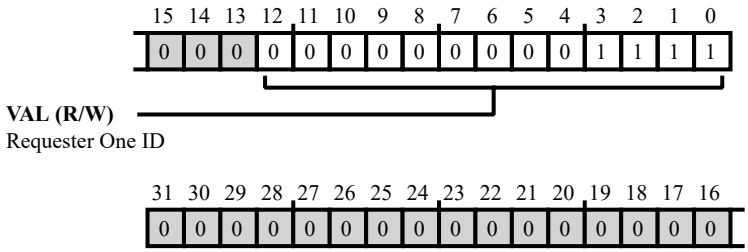


Figure 10-9: PFB_RQSTR1_ID Register Diagram

Table 10-13: PFB_RQSTR1_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester One ID. The PFB_RQSTR1_ID.VAL bit field identifies the second requester ID to be considered for ID based direct access.

Requester ID 1 Mask Register

The PFB_RQSTR1_ID_MSK register is used to mask the second requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

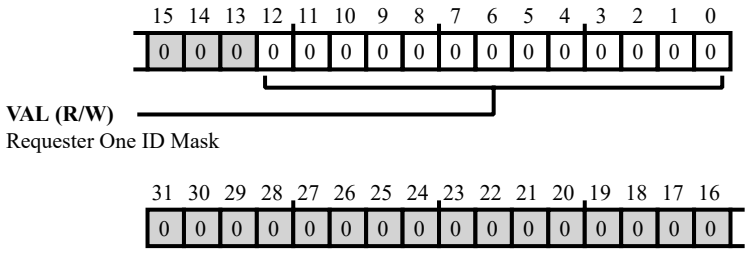


Figure 10-10: PFB_RQSTR1_ID_MSK Register Diagram

Table 10-14: PFB_RQSTR1_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester One ID Mask. The PFB_RQSTR1_ID_MSK.VAL bit field masks the second requester ID to exclude from ID based direct access.

Requester ID 2 Register

The PFB_RQSTR2_ID register is used to configure the third requester ID for controller ID based direct access operations.

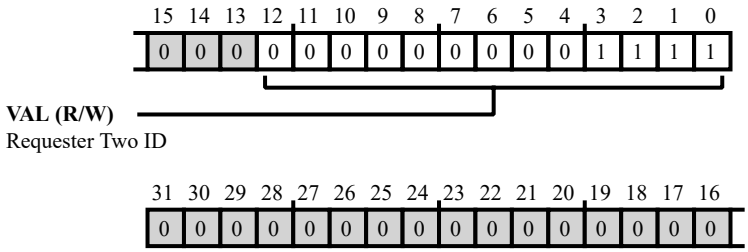


Figure 10-11: PFB_RQSTR2_ID Register Diagram

Table 10-15: PFB_RQSTR2_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Two ID. The PFB_RQSTR2_ID.VAL bit field identifies the third requester ID to be considered for ID based direct access.

Requester ID 2 Mask Register

The PFB_RQSTR2_ID_MSK register is used to mask the third requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

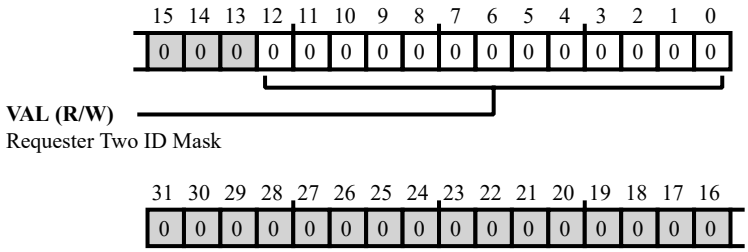


Figure 10-12: PFB_RQSTR2_ID_MSK Register Diagram

Table 10-16: PFB_RQSTR2_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Two ID Mask. The PFB_RQSTR2_ID_MSK.VAL bit field masks the third requester ID to exclude from ID based direct access.

Requester ID 3 Register

The PFB_RQSTR3_ID register is used to configure the fourth requester ID for controller ID based direct access operations.

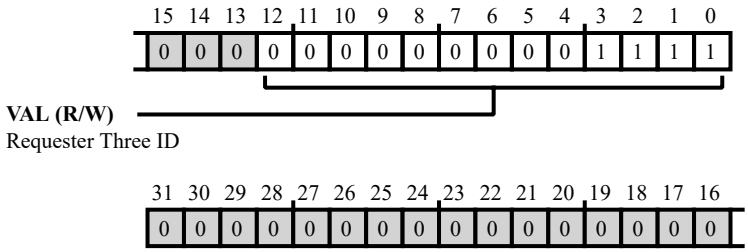


Figure 10-13: PFB_RQSTR3_ID Register Diagram

Table 10-17: PFB_RQSTR3_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Three ID. The PFB_RQSTR3_ID.VAL bit field identifies the fourth requester ID to be considered for ID based direct access.

Requester ID 3 Mask Register

The PFB_RQSTR3_ID_MSK register is used to mask the fourth requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

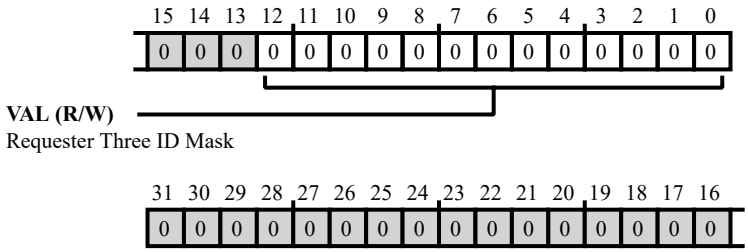


Figure 10-14: PFB_RQSTR3_ID_MSK Register Diagram

Table 10-18: PFB_RQSTR3_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Three ID Mask. The PFB_RQSTR3_ID_MSK.VAL bit field masks the fourth requester ID to exclude from ID based direct access.

Requester ID Enable Register

The PFB_RQSTR_ID_EN register is used to configure controller based direct access. As many as four controller IDs can be programmed for direct access. For more information, see "Controller ID Based Direct Access".

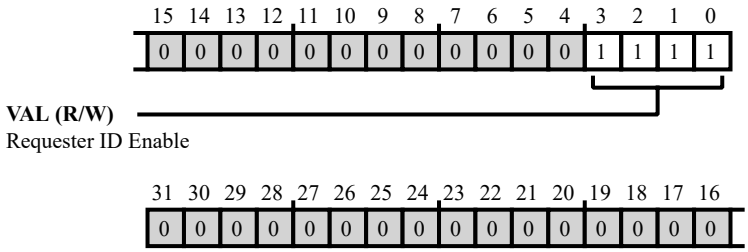


Figure 10-15: PFB_RQSTR_ID_EN Register Diagram

Table 10-19: PFB_RQSTR_ID_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	VAL	Requester ID Enable. The PFB_RQSTR_ID_EN.VAL bits are used to configure controller based direct access. As many as four controller IDs can be programmed for direct access. For more information, see "Controller ID Based Direct Access".

Prefetch Buffer First Status Register

The PFB_STS0 register indicates whether all fetch and prefetch operations from the PFB to the DDR are complete.

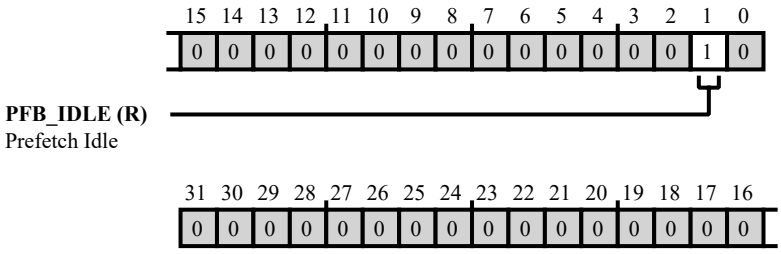


Figure 10-16: PFB_STS0 Register Diagram

Table 10-20: PFB_STS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	PFB_IDLE	Prefetch Idle. The PFB_STS0.PFB_IDLE bit, when set (=1), indicates all fetch and prefetch operations from the PFB to the DDR are complete.

Return Zero End Address Register

The PFB_ZERO_END register is used with the PFB_ZERO_START register to configure an address range in L3 memory to always returns zero.

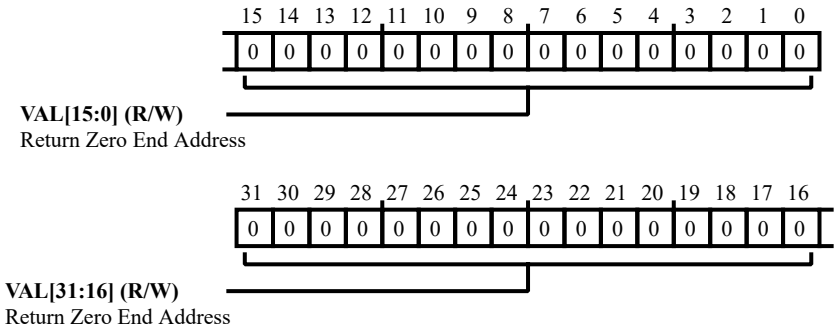


Figure 10-17: PFB_ZERO_END Register Diagram

Table 10-21: PFB_ZERO_END Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VAL	Return Zero End Address. The PFB_ZERO_END.VAL bit field is used to configure the end address (that are aligned to 256 bytes) that are returned with all zeros.

Return Zero Start Address Register

The PFB_ZERO_START register is used with the PFB_ZERO_END register to configure a range of addresses in L3 memory to always returns zero.

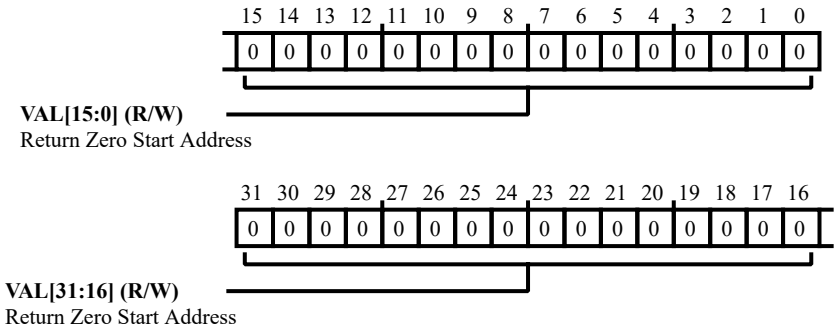


Figure 10-18: PFB_ZERO_START Register Diagram

Table 10-22: PFB_ZERO_START Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VAL	Return Zero Start Address. The PFB_ZERO_START.VAL bit field is used to configure the start address (that are aligned to 256 bytes) that are returned with all zeros.

11 One-Time Programmable Memory Controller (OTPC)

This chapter describes the operation of the OTP controller. The OTP module is a complete system integrating an OTP memory core with a programming controller, charge pump, and voltage regulator. A built-in Hamming Code Error Correction (ECC), and a fully implemented double-redundant program or read scheme protect the OTP data.

CAUTION: OTP memory does not support burst transfers, which are required to support cache line fills. As such, OTP memory should not be made cacheable. If it is, the OTP controller returns an error when a read access is attempted.

OTPC Features

The OTP memory and controller have the following features:

- Built-in redundant read mode
- Built-in integrated power supply
- Built-in Hamming Code Error Correction (ECC)
- Full word serial (single bit at a time) programming with internal VPP

NOTE: The OTP module is protected by the System Memory Protection Unit (SMPU) and is always secure by default. Non secure access to the OTP by any module will fail unless allowed by the SMPU.

OTPC Functional Description

ADSP-2184x OTPC Register List

The One-Time-Programmable Memory controller (OTPC) supports programming the OTP memory. A set of registers governs OTPC operations. For more information on OTPC functionality, see the OTPC register descriptions.

Table 11-1: ADSP-2184x OTPC Register List

Name	Description
OTPC_SECU_STATE	OTP Security State Register
OTPC_STAT	OTP Status Register

ADSP-2184x OTPC Interrupt List

Table 11-2: ADSP-2184x OTPC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
216	OTPC0_ERR	OTPC0 Dual-bit Error	Level	

Error Correction

The OTP memory features a Hamming error correction implementation. Signal bit errors are automatically corrected, and dual-bit errors are detected. Refer to OTPC Interrupt Signals.

ECC is always enabled. ECC applies to each 16-bit segment. Because of this functionality, each 16-bit location can only be written to once. Writing to a 16-bit location a second time results in unexpected behavior.

OTP Layout

This section details the memory layout of the OTP memory.

Table 11-3: ADSP-2184x/SC84x OTP Layout

Name	32-bit Aligned Byte Address	Size (bits)	Description
xspi0_phyconfig0	0x3C	32	xSPI0 PHY Configu Word 0
xspi0_phyconfig1	0x40	32	xSPI0 PHY Configu Word 1
xspi0_phyconfig2	0x44	32	xSPI0 PHY Configu Word 2
xspi0_phyconfig3	0x48	32	xSPI0 PHY Configu Word 3
xspi1_phyconfig0	0x4C	32	xSPI1 PHY Configu Word 0
xspi1_phyconfig1	0x50	32	xSPI1 PHY Configu Word 1
xspi1_phyconfig2	0x54	32	xSPI1 PHY Configu Word 2
xspi1_phyconfig3	0x58	32	xSPI1 PHY Configu Word 3
huk	0x5C	256	Hardware Unique Key
rkek	0x7C	128	Root Key encryption key
dek	0x8C	128	Local encryption key

Table 11-3: ADSP-2184x/SC84x OTP Layout (Continued)

Name	32-bit Aligned Byte Address	Size (bits)	Description
oem_public_key	0x9C	512	OEM Public Key
pvt_128key0	0xDC	128	Customer Privatekey 0 128bits
pvt_128key1	0xEC	128	Customer Privatekey 1 128bits
pvt_128key2	0xFC	128	Customer Privatekey 2 128bits
pvt_128key3	0x10C	128	Customer Privatekey 3 128bits
ek	0x11C	256	Endorsement Key
secure_emu_key0	0x13C	256	Secure Emulation Key0
secure_emu_key1	0x15C	256	Secure Emulation Key1
reserved	0x180	544	reserved
xspi_sram_custom_header	0x1CC	32	xSPI SRAM Custom Header
emu_key0_disable	0x1D0	16	Secure emulation key0 disable
emu_key1_disable	0x1D0	16	Secure emulation key1 disable
public_key0	0x1D4	512	Customer Public Key0
public_key1	0x214	512	Customer Public Key1
watchdog_count	0x254	32	Watchdog Timeout Count
reserved	0x258	32	reserved
cgu0_ctl_WEN	0x25C	1	CGU0 Config ctl_WEN
cgu0_div_WEN	0x25C	1	CGU0 Config div_WEN
cgu0_reserved0	0x25C	1	CGU0 Config reserved0
cgu0_div_DSEL	0x25C	5	CGU0 Config div_DSEL
cgu0_div_CSEL	0x25C	5	CGU0 Config div_CSEL
cgu0_div_S0SEL	0x25C	3	CGU0 Config div_S0SEL
cgu0_div_SYSSEL	0x25C	5	CGU0 Config div_SYSSEL
cgu0_div_S1SEL	0x25C	3	CGU0 Config div_S1SEL
cgu0_div_OSEL	0x25C	7	CGU0 Config div_OSEL
cgu0_ctl_DF	0x25C	1	CGU0 Config ctl_DF
cgu0_ctl_MSEL	0x260	7	CGU0 Config ctl_MSEL
cgu0_ctl_S0SELEXEN	0x260	1	CGU0 Config ctl_S0SELEXEN
cgu0_ctl_S1SELEXEN	0x260	1	CGU0 Config ctl_S1SELEXEN
cgu0_auto_disable	0x260	1	CGU0 Disable Auto Alignment
cgu0_clkoutsel_CLKOUTSEL	0x260	5	CGU0 Config clkoutsel_CLKOUTSEL

Table 11-3: ADSP-2184x/SC84x OTP Layout (Continued)

Name	32-bit Aligned Byte Address	Size (bits)	Description
cgu0_clkoutsel_WEN	0x260	1	CGU0 Config clkoutsel_WEN
cgu0_divex_S0SELEX	0x260	8	CGU0 Config divex_S0SELEX
cgu0_divex_S1SELEX	0x260	8	CGU0 Config divex_S1SELEX
cgu0_reserved1	0x264	32	CGU0 Config reserved1
cgu1_ctl_WEN	0x268	1	CGU1 Config ctl_WEN
cgu1_div_WEN	0x268	1	CGU1 Config div_WEN
cgu1_reserved0	0x268	1	CGU1 Config reserved0
cgu1_div_DSEL	0x268	5	CGU1 Config div_DSEL
cgu1_div_CSEL	0x268	5	CGU1 Config div_CSEL
cgu1_div_S0SEL	0x268	3	CGU1 Config div_S0SEL
cgu1_div_SYSSEL	0x268	5	CGU1 Config div_SYSSEL
cgu1_div_S1SEL	0x268	3	CGU1 Config div_S1SEL
cgu1_div_OSEL	0x268	7	CGU1 Config div_OSEL
cgu1_ctl_DF	0x268	1	CGU1 Config ctl_DF
cgu1_ctl_MSEL	0x26C	7	CGU1 Config ctl_MSEL
cgu1_ctl_S0SELEXEN	0x26C	1	CGU1 Config ctl_S0SELEXEN
cgu1_ctl_S1SELEXEN	0x26C	1	CGU1 Config ctl_S1SELEXEN
cgu1_auto_disable	0x26C	1	CGU1 Disable Auto Alignment
cgu1_clkoutsel_CLKOUTSEL	0x26C	5	CGU1 Config clkoutsel_CLKOUTSEL
cgu1_clkoutsel_WEN	0x26C	1	CGU1 Config clkoutsel_WEN
cgu1_divex_S0SELEX	0x26C	8	CGU1 Config divex_S0SELEX
cgu1_divex_S1SELEX	0x26C	8	CGU1 Config divex_S1SELEX
cgu1_reserved1	0x270	32	CGU1 Config reserved1
cdu_cfg_wen	0x274	1	CDU Config Enable
cdu_cfg_clkin	0x274	1	CDU Clock Input
cdu_cfg0_clk_en	0x274	1	CDU CFG0 Clock Enable
cdu_cfg0_clk_sel	0x274	2	CDU CFG0 Clock SEL
cdu_cfg1_clk_en	0x274	1	CDU CFG1 Clock Enable
cdu_cfg1_clk_sel	0x274	2	CDU CFG1 Clock SEL
cdu_cfg2_clk_en	0x274	1	CDU CFG2 Clock Enable
cdu_cfg2_clk_sel	0x274	2	CDU CFG2 Clock SEL

Table 11-3: ADSP-2184x/SC84x OTP Layout (Continued)

Name	32-bit Aligned Byte Address	Size (bits)	Description
cdu_cfg3_clk_en	0x274	1	CDU CFG3 Clock Enable
cdu_cfg3_clk_sel	0x274	2	CDU CFG3 Clock SEL
cdu_cfg4_clk_en	0x274	1	CDU CFG4 Clock Enable
cdu_cfg4_clk_sel	0x274	2	CDU CFG4 Clock SEL
cdu_cfg5_clk_en	0x274	1	CDU CFG5 Clock Enable
cdu_cfg5_clk_sel	0x274	2	CDU CFG5 Clock SEL
cdu_cfg6_clk_en	0x274	1	CDU CFG6 Clock Enable
cdu_cfg6_clk_sel	0x274	2	CDU CFG6 Clock SEL
cdu_cfg7_clk_en	0x274	1	CDU CFG7 Clock Enable
cdu_cfg7_clk_sel	0x274	2	CDU CFG7 Clock SEL
cdu_cfg8_clk_en	0x274	1	CDU CFG8 Clock Enable
cdu_cfg8_clk_sel	0x274	2	CDU CFG8 Clock SEL
cdu_cfg9_clk_en	0x274	1	CDU CFG9 Clock Enable
cdu_cfg9_clk_sel	0x274	2	CDU CFG9 Clock SEL
cdu_cfg10_clk_en	0x278	1	CDU CFG10 Clock Enable
cdu_cfg10_clk_sel	0x278	2	CDU CFG10 Clock SEL
cdu_cfg11_clk_en	0x278	1	CDU CFG11 Clock Enable
cdu_cfg11_clk_sel	0x278	2	CDU CFG11 Clock SEL
cdu_cfg12_clk_en	0x278	1	CDU CFG12 Clock Enable
cdu_cfg12_clk_sel	0x278	2	CDU CFG12 Clock SEL
cdu_cfg13_clk_en	0x278	1	CDU CFG13 Clock Enable
cdu_cfg13_clk_sel	0x278	2	CDU CFG13 Clock SEL
cdu_cfg14_clk_en	0x278	1	CDU CFG14 Clock Enable
cdu_cfg14_clk_sel	0x278	2	CDU CFG14 Clock SEL
cdu_reserved0	0x278	17	CDU Reserved 0
flashStartAddress	0x27c	32	Flash Start Address Override
altflashStartAddress	0x280	32	Alterate Flash Start Address Override
spiControllerBootCmd	0x284	32	SPI Controller Boot Command Override
xspiControllerBootCmd	0x288	32	XSPI Controller Boot Command Override
xspiControllerBootCmd1	0x28c	32	XSPI Boot command1 Override
xspi1ControllerBootCmd	0x290	32	XSPI1 Controller Boot Command Override

Table 11-3: ADSP-2184x/SC84x OTP Layout (Continued)

Name	32-bit Aligned Byte Address	Size (bits)	Description
xspi1ControllerBootCmd1	0x294	32	XSPI1 Boot command1 Override
emmcControllerBootCmd0	0x298	32	eMMC Boot Command 0 Override
emmcControllerBootCmd1	0x29c	32	eMMC Boot Command 1 Override
emmcSDBootCmd0	0x2A0	32	eMMC SD Boot Command 0 Override
emmcSDBootCmd1	0x2A4	32	eMMC SD Boot Command 1 Override
spiTargetBootCmd	0x2A8	32	SPI Target Boot Command Override
lpBootCmd	0x2AC	32	LinkPort Boot Command Override
uartBootCmd	0x2B0	32	UART Boot Command Override
altTargetBootCmd	0x2B4	32	Alternate Target Boot Command Override
bcfg_pubkey0Inv	0x2B8	1	Boot Config pubkey0Inv
bcfg_reserved2	0x2B8	15	Boot Config reserved2
bcfg_pubkey1Inv	0x2B8	1	Boot Config pubkey1Inv
bcfg_reserved3	0x2B8	15	Boot Config reserved3
bcfg_privkey0Inv	0x2BC	1	Boot Config privkey0Inv
bcfg_reserved4	0x2BC	15	Boot Config reserved4
bcfg_privkey1Inv	0x2BC	1	Boot Config privkey1Inv
bcfg_reserved5	0x2BC	15	Boot Config reserved5
bcfg_privkey2Inv	0x2C0	1	Boot Config privkey2Inv
bcfg_reserved6	0x2C0	15	Boot Config reserved6
bcfg_privkey3Inv	0x2C0	1	Boot Config privkey3Inv
bcfg_reserved7	0x2C0	15	Boot Config reserved7
bcfg_dmcEn	0x2C4	1	Boot Config dmcEn
bcfg_reserved8	0x2C4	15	Boot Config reserved8
bcfg_dmcInv	0x2C4	1	Boot Config dmcInv
bcfg_reserved9	0x2C4	15	Boot Config reserved9
bcfg_xspisramEn	0x2C8	1	Boot Config xspiramEn
bcfg_reserved8	0x2C8	15	Boot Config reserved8
bcfg_xspisramInv	0x2C8	1	Boot Config xspiramInv
bcfg_reserved9	0x2C8	15	Boot Config reserved9
bcfg_wdtDis	0x2CC	1	Boot Config wdtDis
bcfg_reserved10	0x2CC	15	Boot Config reserved10

Table 11-3: ADSP-2184x/SC84x OTP Layout (Continued)

Name	32-bit Aligned Byte Address	Size (bits)	Description
bcfg_wdtResetEn	0x2CC	1	Boot Config wdtResetEn
bcfg_reserved11	0x2CC	15	Boot Config reserved11
bcfg_altMstrBootEn	0x2D0	1	Boot Config altMstrBootEn
bcfg_reserved12	0x2D0	15	Boot Config reserved12
bcfg_altMstrBootDis	0x2D0	1	Boot Config altMstrBootDis
bcfg_reserved13	0x2D0	15	Boot Config reserved13
bcfg_altTargetBootEn	0x2D4	1	Boot Config altTargetBootEn
bcfg_reserved14	0x2D4	15	Boot Config reserved14
bcfg_altTargetBootDis	0x2D4	1	Boot Config altTargetBootDis
bcfg_reserved15	0x2D4	15	Boot Config reserved15
bcfg_benchmark_En	0x2D8	1	Boot Config Benchmarking enable
bcfg_reserved16	0x2D8	15	Boot Config reserved16
bcfg_fastcguDis	0x2D8	1	Boot Config fastcguDis
bcfg_reserved17	0x2D8	15	Boot Config reserved17
bcfg_dmcDecDis	0x2DC	1	Boot Config dmcDecDis
bcfg_reserved18	0x2DC	15	Boot Config reserved17
bcfg_secBootHSMDis	0x2DC	16	Boot Config secBootHSMDis
antiroll_nv_cntr	0x2E4	512	AntiRollback NV Counter
gp1	0x324	320	General Purpose 1
reserved	0x34c	24	reserved
bootModeDisable	0x34c	8	Boot Mode Disable Bits
xspi_sram_phyconfig0	0x350	32	xSPI SRAM Phyconfig0
xspi_sram_phyconfig1	0x354	32	xSPI SRAM Phyconfig1
xspi_sram_phyconfig2	0x358	32	xSPI SRAM Phyconfig2
xspi_sram_phyconfig3	0x35c	32	xSPI SRAM Phyconfig3
xspi_sram_remapConfig	0x360	2	XSPI sram remapConfig
xspi_sram_devicetype	0x360	1	XSPI sram devicetype
xspi_sram_Addressbytes	0x360	2	XSPI sram Addressbytes
xspi_sram_bytesperaddr	0x360	1	XSPI sram bytesperaddr
xspi_sram_byteswap	0x360	1	XSPI sram byteswap
xspi_sram_devnum	0x360	1	XSPI sram devnum

Table 11-3: ADSP-2184x/SC84x OTP Layout (Continued)

Name	32-bit Aligned Byte Address	Size (bits)	Description
xspi_sram_csda_mindelay	0x360	8	XSPI sram csda_mindelay
xspi_sram_clockDiv	0x360	8	XSPI sram clockDiv
xspi_sram_readopcode1	0x360	8	XSPI sram readopcode1
xspi_sram_readopcode2	0x364	8	XSPI sram readopcode2
xspi_sram_writeopcode1	0x364	8	XSPI sram writeopcode1
xspi_sram_writeopcode2	0x364	8	XSPI sram writeopcode2
xspi_sram_readlatency	0x364	8	XSPI sram readlatency
xspi_sram_writelatency	0x368	8	XSPI sram writelatency
xspi_sram_busmode	0x368	8	XSPI sram busmode
xspi_sram_targetsel	0x368	2	XSPI sram targetsel
xspi_sram_wen_enable	0x368	1	XSPI sram wen_enable
xspi_sram_wen_opcodeexenable	0x368	1	XSPI sram wen_opcodeexenable
xspi_sram_wen_opcodeex	0x368	8	XSPI sram wen_opcodeex
xspi_sram_reserved0	0x368	4	reserved0
xspi_sram_dev_active_max	0x36c	32	XSPI sram dev_active_max
stageID	0x370	48	StageID
reserved	0x374	16	reserved

OTPC Event Control

The following sections provide information on OTP events and error management.

OTPC Interrupt Signals

When making 32-bit accesses to OTP memory, a double-bit error in any 16-bit segment triggers the OTPC_INT interrupt. The OTPC also has the OTPC dual bit error (OTPC0_ERR) with the SEC ID of 5.

OTPC Status and Error Signals

The OTP controller does not produce error signals.

OTP API Overview

The ROM/OTP SSLDD library provides a set of functions to facilitate OTP field access. The OTP memory is broken up into a set of specialized fields that are described in this section. The API removes the requirement of understanding the details of the layout or OTP access procedures.

All OTP accesses are made through the provided API.

OTP Programming

The OTP programming API provides a simple access, abstracting particulars of the OTP controller.

Any fields that contain zero or null pointers are skipped.

All addresses are assumed to be byte addresses unless otherwise noted.

OTP Reading

This API provides a unified source for retrieving OTP data fields.

All addresses are assumed to be byte addresses, unless otherwise noted.

A list of APIs follow:

`bool adi_rom_otp_get( OTPCMD cmd, uint32_t* data);`	OTP Get Field

OTP Get Field

Retrieves indicated data from OTP memory.

Name	OTP Get Field	
Prototype	`bool adi_rom_otp_get( OTPCMD cmd, uint32_t* data);`	
Argument	cmd	Indicates what data to fetch, based on the`OTPCMD` enum.
Argument	data	memory location to write the data to
Return Value	bool	true for a successful read
Stack Requirements	valid stack	

```
bool res = adi_rom_otp_get(otpcmd_info,data);
```

The data specified by the `OTPCMD` enum parameter is fetched from OTP memory and placed in the location specified by data. The `OTPCMD` enum contains entries for each field defined in OTP memory, for the most current list, refer to the OTP header file.

ADSP-2184x OTPC Register Descriptions

OTP Memory Controller (OTPC) contains the following registers.

Table 11-4: ADSP-2184x OTPC Register List

Name	Description
OTPC_SECU_STATE	OTP Security State Register
OTPC_STAT	OTP Status Register

OTP Security State Register

The OTPC_SECU_STATE register indicates the secure state. The register is updated only after the part is rebooted and locked.

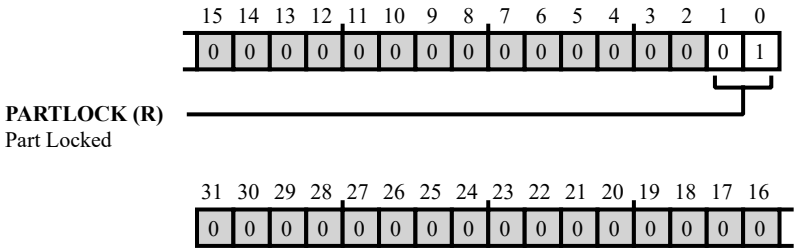


Figure 11-1: OTPC_SECU_STATE Register Diagram

Table 11-5: OTPC_SECU_STATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PARTLOCK	Part Locked. The OTPC_SECU_STATE.PARTLOCK field indicates a locked part.	
		0	OPEN part
		1	Locked part
		2	Reserved
		3	Reserved

OTP Status Register

The OTPC_STAT register bits indicate errors and flag status and control the protection bits.

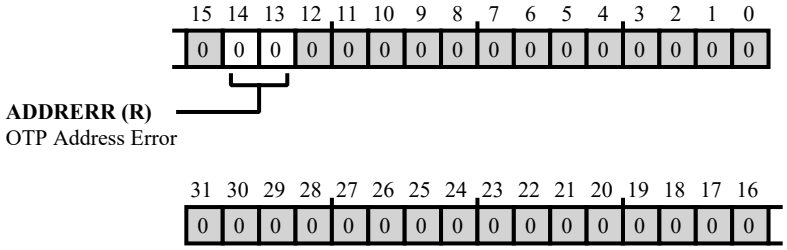


Figure 11-2: OTPC_STAT Register Diagram

Table 11-6: OTPC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14:13 (R/NW)	ADDRERR	OTP Address Error. The OTPC_STAT.ADDRERR bit field indicates errors which occur when the OTP programming address is out of range or tries to access protected space.	
		0	No error - proper OTP address
		1	OTP address out of range
		2	8-bit OTP address
		3	Protected OTP address

12 System Memory Protection Unit (SMPU)

The SMPU provides a flexible way of protecting memory regions against read or write access from any or all bus requester in the system. In addition, it can guard against memory access depending on security privileges of the system bus requester.

SMPU Features

The system memory protection unit has the following features.

- After reset, the default state of the system is fully open. The SMPUs admit any access to memory spaces by any controller.
- On reset, secure transactions are only allowed; non secure transactions are disabled.
- Each SMPU instance can be configured to monitor multiple regions. Each can be individually enabled.
- Each region can be configured with its own protection settings.
- Provides general read or write protection.
- Read and write transactions are restricted or allowed depending on the transaction ID.

On the ADSP-SC84x/ADSP-2184x processor, 15 SMPU instances are available to protect the L2, external memory (DMC) interface.

Table 12-1: SMPU Instances

Module	SMPU Instance	Controller Allowed	L2 Memory Address Range
L2-Core Port 0 (CL2_0)	2	Data port of both the SHARC-FX and Arm and IIR accelerator port	0x20400000 - 0x205FFFFF (L2 SRAM Banks 0-19) 0x20200000 – 0x2021FFFF (L2 Boot ROM 0 and 1)
L2 DMA Port 0 (DL2_0)	3	HSMDMA	0x20400000 - 0x208FFFFF (L2 SRAM Banks 0-19) 0x20200000 – 0x2021FFFF (L2 Boot ROM 0 and 1)
L2-Core Port 1 (CL2_1)	4	Data port of the SHARC-FX and Arm and IIR accelerator port	0x20600000 - 0x208FFFFF (L2 SRAM Banks 16-19)
L2 DMA Port 1 (DL2_1)	5	Peripheral and remaining DMA	0x20400000 - 0x208FFFFF (L2 SRAM Banks 0-19) 0x20200000 – 0x2021FFFF (L2 Boot ROM 0 and 1)

Table 12-1: SMPU Instances (Continued)

Module	SMPU Instance	Controller Allowed	L2 Memory Address Range
L2-Core Port 2 (CL2_2)	6	Instruction port of both SHARC-FX and FIR accelerator port	0x20400000 - 0x208FFFFF (L2 SRAM Banks 0-19) 0x20200000 – 0x2021FFFF (L2 Boot ROM 0 and 1)
DMC0	9	All allowed requesters	Entire range
SPI2/xSPI Flash Address Space	11	All allowed requesters	Entire range
OTP Memory	12	All allowed requesters	Entire range

All the SMPU instances can be configured up to eight regions.

- Up to eight outstanding read or write transactions supported on SMPU instances for Core_L2 and DMC0. Up to four outstanding read or write transactions supported on the SMPU instances for DMA_L2 and SPI flash address space.

SMPU Functional Description

The following sections provide details on the function of the SMPU module. If the region security settings allow transactions to go through, the ID in the ID-based region protection settings can still filter the transactions.

For the memory that an SMPU protects, programs can configure region-based settings with the `SMPU_RCTL[n]` registers. (There can be multiple SMPUs in a system.) The `SMPU_RCTL[n]` registers define the ID-based protection for memory regions.

If the target address does not reside in any configured memory region, the transaction permission resorts back to the global configuration setting.

Protection Units

Each SMPU provides two protection units, A and B for ID-based matching in the region-based memory protection. This feature provides a degree of flexibility for the user to match against multiple IDs.

Instruction Fetches

When the core executes instructions from memory, this operation is also considered a memory transaction. If the SMPU is configured to protect a memory region from read accesses that contain instructions, the core cannot fetch and execute these instructions.

Using Cache

When the processor uses both cache and the SMPU, there are a few issues to be aware of. If the SMPU is configured to protect a memory region from write accesses, instruction fetches from a core are still possible since instructions are not updated and replaced during run time.

NOTE: The debugger typically replaces an instruction with a breakpoint instruction for software breakpoints. If a memory region is protected against write accesses, software breakpoints are not possible unless the SMPU is configured with the appropriate system controller ID of the debugger. The configuration allows it to perform a write-access.

In the case where memory is used for data, a read access or cache fill is not possible if the memory is blocked from read accesses. If read accesses are allowed but write accesses are disallowed, then there is an issue with coherency. The cache is filled but when the cache is updated and must be written back to the SMPU protected memory, the write-access is blocked.

In general, exercise caution when using both the SMPU and cache.

Speculative Reads

If speculative reads are enabled (`SMPU_CTL.RSDIS` =0), the SMPU forwards the read transaction directly to the memory before checking the protection setting corresponding to the addressed memory region. This functionality saves one clock cycle in the clock domain of the SMPU. The SMPU checks the protection setting while the read transaction occurs with the memory. If the protection setting dictates that the target memory address is blocked, the SMPU blocks the read to the controller.

If speculative reads are disabled (`SMPU_CTL.RSDIS` =1), the SMPU checks the protection settings first and forwards the transaction to memory only if it passes the configured protection settings. This functionality incurs a one-cycle latency per read.

NOTE: Reads affect certain memory operations such as automatic clearing of the memory (that is, FIFOs). When the SMPU protects this type of memory, disable read speculation since the blocking can occur without the read transaction reaching the target memory.

ADSP-2184x SMPU Register List

The System Memory Protection Unit (SMPU) provides selective protection of the processor's memory resources. The SMPU includes a set of processor events that can be monitored during program execution. A set of registers governs SMPU operations. For more information on SMPU functionality, see the SMPU register descriptions.

Table 12-2: ADSP-2184x SMPU Register List

Name	Description
`SMPU_BADDR`	Bus Error Address Register
`SMPU_BDTLS`	Bus Error Details Register
`SMPU_CTL`	SMPU Control Register
`SMPU_EXACADD[n]`	Exclusive Access IDn Address
`SMPU_EXACSTAT[n]`	Exclusive Access Status
`SMPU_IADDR`	Interrupt Address Register

Table 12-2: ADSP-2184x SMPU Register List (Continued)

Name	Description
SMPU_IDTLS	Interrupt Details Register
SMPU_RADDR[n]	Region n Address Register
SMPU_RCTL[n]	Region n Control Register
SMPU_REVID	SMPU Revision ID Register
SMPU_RIDA[n]	Region n ID A Register
SMPU_RIDB[n]	Region n ID B Register
SMPU_RIDMSKA[n]	Region n ID Mask A Register
SMPU_RIDMSKB[n]	Region n ID Mask B Register
SMPU_SECURECTL	SMPU Control Secure Accesses Register
SMPU_SECURERCTL[n]	Region n Control Secure Accesses Register
SMPU_STAT	SMPU Status Register

SMPU Interrupts

The SMPU has one interrupt with the SEC ID = 142.

See the SEC chapter for complete information on interrupt generation and use.

Memory Writes

A write transaction to address n is prevented when the following is true.

Address n is in memory region m and memory region m is write-protected (SMPU_RCTL[n].WPROTEN =1) and ID is not a match. (See ID Comparison). The block occurs because the memory region is configured for write-protection and the ID comparison does not result in a match. If an ID comparison results in a match, the write transaction is allowed through.

Memory Reads

A read transaction from address n is prevented when the following is true:

- Address n is in memory region and memory region m is read-protected (SMPU_RCTL[n].RPROTEN =1) and
- ID is not a match

The block occurs because the memory region is configured for read protection and the ID comparison does not result in a match. If the ID comparison results in a match, the read transaction is permitted. (See ID Comparison).

ID Comparison

ID comparison automatically occurs during region-based memory protection. ID matches allow the transaction to bypass the configured memory protection for that region. The following sections describe the calculation of a write ID match and read ID match.

Write Transaction

The state of the following values determines the ID value that is compared with the ID of an incoming write transaction:

- The SMPU_RCTL[n].WIDCINV bit
- The SMPU_RIDA[n].ID and SMPU_RIDB[n].ID bit fields
- The SMPU_RIDMSKA[n].MSK and SMPU_RIDMSKB[n].MSK bit fields

Write IDA match = ((ID of incoming write transaction AND SMPU_RIDMSKA[n].MSK) == (SMPU_RIDA[n].ID AND SMPU_RIDMSKA[n].MSK))

Write IDB match = ((ID of incoming write transaction AND SMPU_RIDMSKB[n].MSK) == (SMPU_RIDB[n].ID AND SMPU_RIDMSKB[n].MSK))

Write ID match = (Write IDA match OR Write IDB match) XOR SMPU_RCTL[n].WIDCINV bit

Read Transaction

The state of the following values determines the ID value that is compared with the ID of an incoming read transaction:

- The SMPU_RCTL[n].RIDCINV bit
- The SMPU_RIDA[n].ID and SMPU_RIDB[n].ID bit fields
- The SMPU_RIDMSKA[n].MSK and SMPU_RIDMSKB[n].MSK bit fields

Read IDA match = ((ID of incoming read transaction AND SMPU_RIDMSKA[n].MSK) == (SMPU_RIDA[n].ID AND SMPU_RIDMSKA[n].MSK))

Read IDB match = ((ID of incoming read transaction AND SMPU_RIDMSKB[n].MSK) == (SMPU_RIDB[n].ID AND SMPU_RIDMSKB[n].MSK))

Read ID match = (Read IDA match OR Read IDB match) XOR SMPU_RCTL[n].RIDCINV

In the two cases described above, the incoming transaction (either write or read) ID is AND'ed with the configured mask value in protection unit A. It is then compared to the value of the configured ID value which is also AND'ed with the configured mask value in protection unit A. The mask provides a method to allow a group of IDs to match. This process is also performed for protection unit B. The two outcomes (from A and B) are then OR'ed together.

Depending on the setting of the SMPU_RCTL[n].RIDCINV or the SMPU_RCTL[n].WIDCINV bits, the ID match comparison is inverted or not. The final result after applying the inversion, SMPU_RCTL[n].RIDCINV, or SMPU_RCTL[n].WIDCINV, determines whether the transaction bypasses the protection.

Usage

The masks, SMPU_RIDMSKA[n] and SMPU_RIDMSKB[n], are AND'ed with both the incoming transaction ID and the configured ID in SMPU_RIDA[n].ID and SMPU_RIDB[n].ID, respectively. By default the masks are zero. If ID-based region protection is enabled by setting the SMPU_RCTL[n].WPROTEN or SMPU_RCTL[n].RPROTEN bit fields and the masks are not set, the ID comparison essentially compares zeros. The comparison allows all transactions to bypass (if the region-based security setting is also configured in a way to allow transactions to go through for the region). To have the ID-based region protection to function, the mask registers and ID registers must also be set.

System IDs

The ***System controller IDs*** table provides the IDs for the system controllers. An x means that the bit can be a 0 or a 1. There are multiple IDs associated with that system controller.

Table 12-3: System Controller IDs

ASIB Name	IID	SIID	ID
SP0A	0	0	11'b0000000x000
SP0B	0	1	11'b0000000x001
SP1A	0	2	11'b0000000x010
SP1B	0	3	11'b0000000x011
SP2A	0	4	11'b0000000x100
SP2B	0	5	11'b0000000x101
SP3A	0	6	11'b0000000x110
SP3B	0	7	11'b0000000x111
CRC0_CH0	1	6	11'b0010000x110
CRC0_CH1	1	7	11'b0010000x111
MLB	2	4	11'b00100000100
UART0_TX	3	0	11'b0011000x000
UART0_RX	3	4	11'b0011000x100
SPI0TX	4	0	11'b0100000x000
SPI0RX	4	1	11'b0100000x001
SPI1TX	4	2	11'b0100000x010
SPI1RX	4	3	11'b0100000x011

Table 12-3: System Controller IDs (Continued)

ASIB Name	IID	SIID	ID
SPI2TX	4	5	11'b0100000x101
SPI2RX	4	4	11'b0100000x100
UART1_TX	3	1	11'b0011000x001
UART1_RX	3	2	11'b0011000x010
CRYPTO	6	2	11'b01100000010
ACC_CH0	6	0	11'b0110xxxx000
ACC_CH1	6	1	11'b0110xxxx001
EMDMA0_CH0	7	0	11'b01110000000
EMDMA0_CH1	7	1	11'b01110000001
EMDMA1_CH0	8	1	11'b10000000001
EMDMA1_CH1	8	0	11'b10000000000
MSMDMA_CH0	9	0	11'b1001000x000
MSMDMA_CH1	9	1	11'b1001000x001
DBG	9	2	11'b10010000010
CRC1_CH0	10	4	11'b1010000x100
CRC1_CH1	10	5	11'b1010000x101
SH0_DPORT	11	0	11'b1011xxxx000
SH0_IPORT	11	1	11'b10110000001
HSMDMA_CH0	12	0	11'b1100000x000
HSMDMA_CH1	12	1	11'b1100000x001
SP4A	13	0	11'b1101000x000
SP4B	13	1	11'b1101000x001
SP5A	13	2	11'b1101000x010
SP5B	13	3	11'b1101000x011
SP6A	13	4	11'b1101000x100
SP6B	13	5	11'b1101000x101
SP7A	13	6	11'b1101000x110
SP7B	13	7	11'b1101000x111
SH0_MMR	14	0	11'b1110xxxx000

Memory Region

Memory regions can start at address 0x00000000 or at any address that is a multiple of its size. The ***Supported Memory Region Size and Alignment*** table shows the memory region sizes that the processor supports and the alignment of the memory region. (X values are do-not-care).

SMPU supports a maximum of eight regions.

Table 12-4: Supported Memory Region Size and Alignment

Size	SMPU_RCTLn.SIZE	Address	Possible Values for N
4KB	0b00000	0xXXXXX000	-
8KB	0b00001	0xXXXXN000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
16KB	0b00010	0xXXXXN000	0x0, 0x4, 0x8, 0xC
32KB	0b00011	0xXXXXN000	0x0, 0x8
64KB	0b00100	0xXXXX0000	-
128KB	0b00101	0xXXXN0000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
256KB	0b00110	0xXXXN0000	0x0, 0x4, 0x8, 0xC
512KB	0b00111	0xXXXN0000	0x0, 0x8
1MB	0b01000	0xXXX00000	-
2MB	0b01001	0xXXN00000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
4MB	0b01010	0xXXN00000	0x0, 0x4, 0x8, 0xC
8MB	0b01011	0xXXN00000	0x0, 0x8
16MB	0b01100	0xXX000000	-
32MB	0b01101	0xXN000000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
64MB	0b01110	0xXN000000	0x0, 0x4, 0x8, 0xC
128MB	0b01111	0xXN000000	0x0, 0x8
256MB	0b10000	0xX0000000	-
512MB	0b10001	0xN0000000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
1GB	0b10010	0xN0000000	0x0, 0x4, 0x8, 0xC
2GB	0b10011	0xN0000000	0x0, 0x8
4GB	0b10100	0x00000000	-

For the case where the region size is selected as 4 GB, the region address must be at address 0x00000000.

NOTE: If a memory region address is not aligned to its size, the memory region start address protected by the SMPU is the configured address with the corresponding least significant bits masked. For example, if the size is configured for 16 KB (`SMPU_RCTL[n].SIZE` =0b00010), and the base address is configured for `SMPU_RADDR[n].BADDR`=0x00005018, the actual base address used by the SMPU is 0x00004000. When `SMPU_RADDR[n].BADDR` is read back, the program reads 0x00005000. This functionality is

because only bits [11:0] are reserved as 0's. Programs must use care when setting the base address as it is not always the true base address.

SMPU Definitions

To make the best use of the SMPU, it is useful to understand the terms in this section.

Global Protection

Guarding of the entire memory space for the SMPU instantiation.

Region-Based Protection

Guarding individual segments of memory inside the memory space for the SMPU instantiation.

ID Match

A successful comparison of the ID associated with the incoming transaction and the ID and MASK configured in the SMPU.

SMPU Block Diagram

The ***SMPU Top-Level Block Diagram*** shows the SMPU block.

As seen in the diagram, the SMPU sits between the memory port (SCB requester port) and the SCB fabric (SCB completer port). It acts as a gateway analyzing the transaction requests. It either rejects the transaction request or allows access based on the user-programmed configuration of the SMPU.

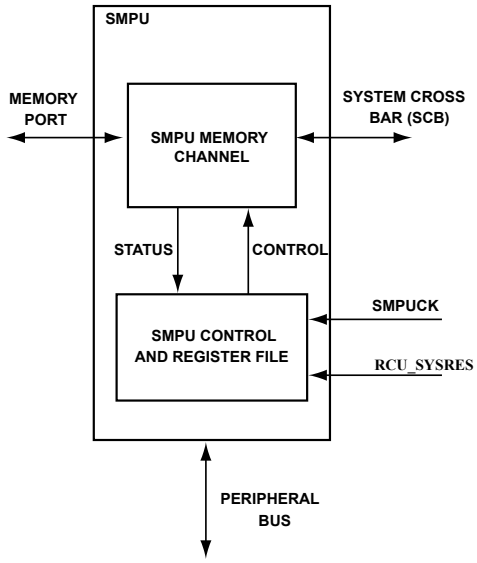


Figure 12-1: SMPU Top-Level Block Diagram

SMPU Architectural Concepts

The following sections provide brief descriptions of the architecture of the SMPU module.

Default Setting

At reset, the default state of the system is fully open. The SMPUs admit any access to memory spaces.

Latency

The SMPU adds latency to all the transactions to the memory except reads when read speculation is enabled (SMPU_CTL.RSDIS =0). In this case, read accesses are always forwarded to the memory and read responses are generated according to the SMPU settings. If read speculation is disabled (SMPU_CTL.RSDIS =1), reads are blocked if they cause a security or protection violation. The SMPU generates the SCB read response that corresponds to a blocked transaction.

If read speculation is enabled, the SMPU adds 1 clock cycle latency to the read transaction. If read speculation is disabled, the SMPU adds 2 clock cycles latency to the read transaction.

SMPU Operating Modes

The SMPU does not have any strict modes of operation. However, it can be configured for region-based protection where a controller with a particular ID can be blocked or allowed based on settings in the SMPU_RCTL[n] register.

Region-based protection is programmed with registers:

- SMPU_RCTL[n]
- SMPU_RADDR[n]
- SMPU_RIDA[n]
- SMPU_RIDMSKA[n]
- SMPU_RIDB[n]
- SMPU_RIDMSKB[n]

SMPU Interrupt Signals

There is one interrupt signal associated with the SMPU. If interrupts are enabled, the SMPU_STAT.IRQ bit is set. The SMPU_IRQ signal is asserted when the SMPU detects a memory access violation. The target address triggering the interrupt is found in the SMPU_IADDR register. The SMPU_IDTLS register provides further details about the cause of the interrupt.

Write errors are prioritized over read errors.

Protection violations (an ID-based violation) can trigger the SMPU interrupt and can be enabled independently. The protection violation interrupt is enabled by setting the SMPU_CTL.PINTEN bit.

The SMPU interrupt is asserted for any of the following conditions:

If a second memory access violation occurs while the SMPU_STAT.IRQ bit is set, the SMPU_STAT.IOVR (interrupt overrun) bit is set. The SMPU_IADDR and the SMPU_IDTLS registers are not updated until the SMPU_STAT.IRQ bit is cleared. Any information on the subsequent interrupt is lost. Once the SMPU_STAT.IRQ bit and the SMPU_STAT.IOVR bit are cleared, any new memory access violations can trigger an interrupt and its details can be captured.

NOTE: When a blocked access occurs, the SMPU triggers an interrupt when interrupt generation is enabled. The SMPU can also be configured to generate a bus error that propagates back to the system controller. The system controller can also trigger an interrupt due to this bus error.

NOTE: On the processor, each SMPU instance has an interrupt. All of the SMPU interrupts are OR'ed and mapped to a single SMPU interrupt on the SEC. While servicing the SMPU interrupt, check all of the SMPU_STAT registers to determine which triggered the interrupt. The interrupt service routine clears the SMPU_STAT.IRQ bit of all of the SMPU_STAT registers for which the interrupt is triggered.

SMPU Status and Error Signals

If bus errors are enabled (SMPU_CTL.PBEDIS =0), the SMPU generates and returns a bus error to the controller initiating the blocked access. This bit also sets the SMPU_STAT.BERR bit. The SMPU_BADDR and SMPU_BDTLS registers can be read to get the address and details of the transaction that caused the SMPU to generate the error.

Write errors are prioritized over read errors.

A bus error status is returned to the system controller if:

- an ID-based violation happened and the SMPU_CTL.PBEDIS bit =0

If a second memory access violation occurs while the SMPU_STAT.BERR bit is set, the SMPU_STAT.BEOVR bit (bus error overrun) is set. The SMPU_BADDR and the SMPU_BDTLS registers are not updated until the SMPU_STAT.IRQ bit is cleared. The information about the transaction that caused the SMPU_STAT.BEOVR bit to be set is lost.

NOTE: If both the protection violation interrupt is not enabled (SMPU_CTL.PINTEN =0) and the protection bus error is disabled (SMPU_CTL.PBEDIS =1), the SMPU blocks invalid transactions. However, it does not provide any status or interrupt information indicating that a transaction is blocked.

ADSP-2184x SMPU Register Descriptions

The System Memory Protection Unit (SMPU) contains the following registers.

Table 12-5: ADSP-2184x SMPU Register List

Name	Description
SMPU_BADDR	Bus Error Address Register
SMPU_BDTLS	Bus Error Details Register
SMPU_CTL	SMPU Control Register
SMPU_EXACADD[n]	Exclusive Access IDn Address
SMPU_EXACSTAT[n]	Exclusive Access Status
SMPU_IADDR	Interrupt Address Register
SMPU_IDTLS	Interrupt Details Register
SMPU_RADDR[n]	Region n Address Register
SMPU_RCTL[n]	Region n Control Register
SMPU_REVID	SMPU Revision ID Register
SMPU_RIDA[n]	Region n ID A Register
SMPU_RIDB[n]	Region n ID B Register
SMPU_RIDMSKA[n]	Region n ID Mask A Register
SMPU_RIDMSKB[n]	Region n ID Mask B Register
SMPU_SECURECTL	SMPU Control Secure Accesses Register
SMPU_SECURERCTL[n]	Region n Control Secure Accesses Register
SMPU_STAT	SMPU Status Register

Bus Error Address Register

Programs read the SMPU_BADDR and the SMPU_BDTLS registers to determine the cause of a bus error. Write errors are prioritized over read errors.

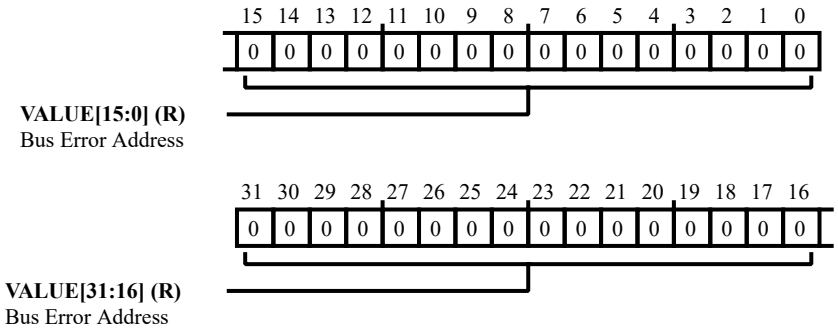


Figure 12-2: SMPU_BADDR Register Diagram

Table 12-6: SMPU_BADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Bus Error Address. The SMPU_BADDR.VALUE bit field contains the address of the bus error.

Bus Error Details Register

The SMPU_BDTLS register indicates the ID of the bus error transaction, whether the transaction that caused the last bus error was a read, a write, secure or non-secure.

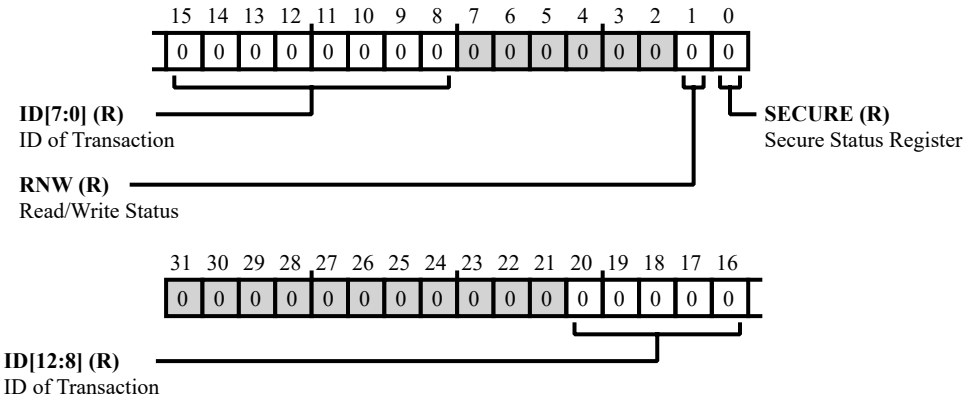


Figure 12-3: SMPU_BDTLS Register Diagram

Table 12-7: SMPU_BDTLS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	ID of Transaction. The SMPU_BDTLS.ID bit field provides the ID of the transaction that caused the bad address error.	
1 (R/NW)	RNW	Read/Write Status. The SMPU_BDTLS.RNW bit indicates whether the last transaction that caused the bad address error was a read or write.	
		0	Transaction that caused last bus error was a write
		1	Transaction that caused last bus error was a read
0 (R/NW)	SECURE	Secure Status Register. The SMPU_BDTLS.SECURE bit indicates whether the last transaction that caused the bad address error was secure or non-secure.	
		0	Transaction that caused last bus error was non-secure
		1	Transaction that caused last bus error was secure

SMPU Control Register

The SMPU_CTL register provides access to the locking control, error interrupts and SMPU violations.

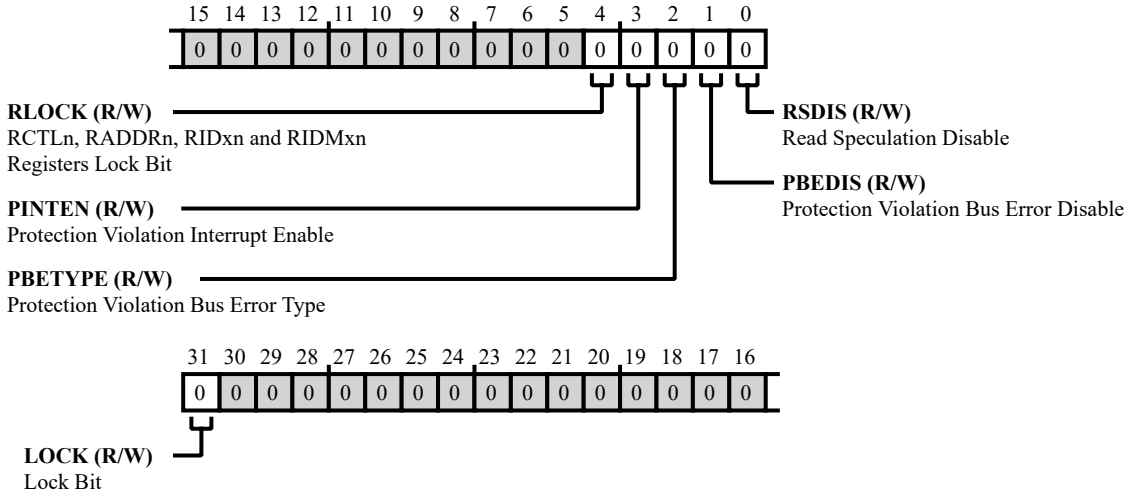


Figure 12-4: SMPU_CTL Register Diagram

Table 12-8: SMPU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit. When the SMPU_CTL.LOCK bit is set and the global lock signal is asserted from the SPU, the SMPU_CTL register is write-protected. Write-protection is disabled only when the global lock signal becomes deasserted again.	
		0	CTL Global Lock Disable. The SMPU_CTL register is not write-protected.
		1	CTL Global Lock Enable. The SMPU_CTL register is write-protected.
4 (R/W)	RLOCK	RCTLn, RADDRn, RIDxn and RIDMxn Registers Lock Bit. When the SMPU_CTL.RLOCK bit is set, all the registers associated with region-based control (SMPU_RCTL[n], SMPU_RADDR[n], SMPU_RIDA[n], SMPU_RIDB[n], SMPU_RIDMSKA[n] and SMPU_RIDMSKB[n]) are write-protected when the global lock signal is active from the SPU. Write access is allowed again when the global lock signal is deasserted.	
		0	Region Registers Write-Protect Enable. All region registers are not write-protected.
		1	Region Registers Write-Protect Disable. All region registers are write-protected.

Table 12-8: SMPU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	PINTEN	Protection Violation Interrupt Enable. The SMPU_CTL.PINTEN bit controls whether or not an interrupt is generated when a protection violation occurs.	
		0	Protection Violation IRQ Disable. The protection violation interrupt is disabled.
		1	Protection Violation IRQ Enable. The protection violation interrupt is enabled.
2 (R/W)	PBETYPE	Protection Violation Bus Error Type. The SMPU_CTL.PBETYPE bit controls whether a protection violation produces a decode error or a completer error.	
		0	Decode Error Type. Decode error for transactions that violate the configured protection.
		1	Completer Error Type. Completer error for transactions which violate the configured protection.
1 (R/W)	PBEDIS	Protection Violation Bus Error Disable. If set, the SMPU_CTL.PBEDIS bit blocks protection violations, but does not cause a bus error.	
		0	Bus Error Generation Enable. Transactions which violate the configured protection are blocked and cause a bus error.
		1	Bus Error Generation Disable. Transactions which violate the configured protection are blocked but do not cause a bus error.
0 (R/W)	RSDIS	Read Speculation Disable. The SMPU_CTL.RSDIS bit controls whether or not the read addresses are checked before being sent to the completer.	
		0	Read Speculation Enable. Read addresses are sent to the completer without checking.
		1	Read Speculation Disable. Read addresses are checked before being sent to the completer.

Exclusive Access IDn Address

The SMPU_EXACADD[n] register provides the address ID of an exclusive access.

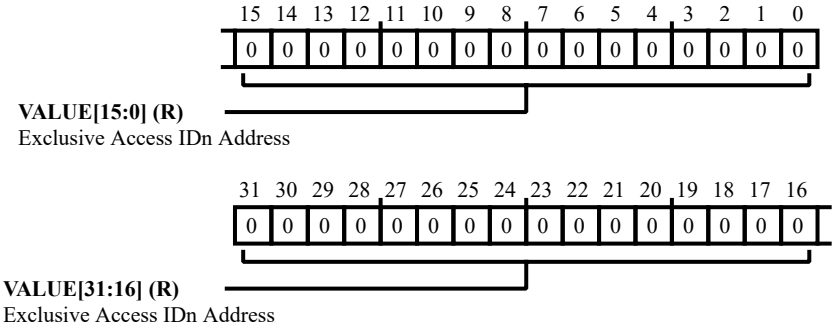


Figure 12-5: SMPU_EXACADD[n] Register Diagram

Table 12-9: SMPU_EXACADD[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Exclusive Access IDn Address.

Exclusive Access Status

The SMPU_EXACSTAT[n] register provides the exclusive access ID, read size and read length as well as the indication that the access was valid.

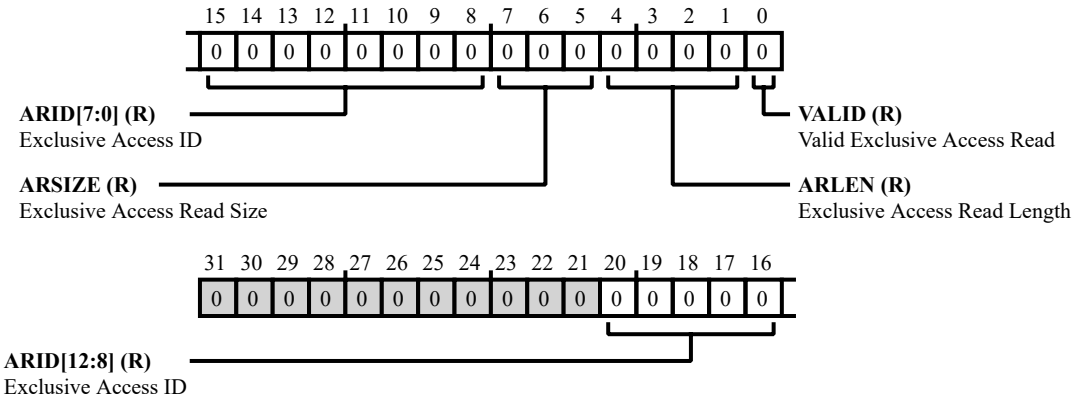


Figure 12-6: SMPU_EXACSTAT[n] Register Diagram

Table 12-10: SMPU_EXACSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ARID	Exclusive Access ID.
7:5 (R/NW)	ARSIZE	Exclusive Access Read Size.
4:1 (R/NW)	ARLEN	Exclusive Access Read Length.
0 (R/NW)	VALID	Valid Exclusive Access Read.

Interrupt Address Register

The SMPU_IADDR register indicates an attempt to make a read or write access to unimplemented addresses or accesses are non-aligned. The SMPU issues a bus error for this condition.

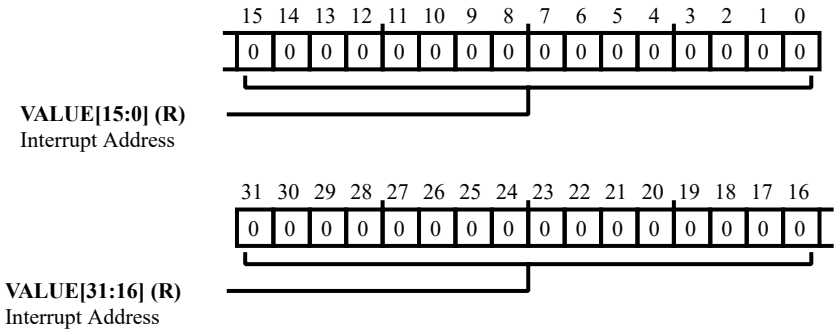


Figure 12-7: SMPU_IADDR Register Diagram

Table 12-11: SMPU_IADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Interrupt Address. The SMPU_IADDR.VALUE bit field is the address where an attempt to access an unimplemented address or a non-aligned access has occurred.

Interrupt Details Register

The SMPU_IDTLS register provides the ID of the last signaled interrupt, whether the interrupt was caused by a read or write, and whether the transaction that caused the last signaled interrupt was secure.

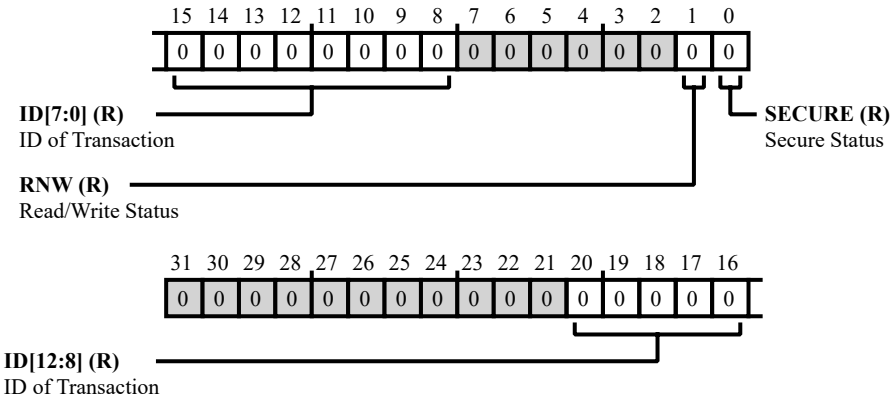


Figure 12-8: SMPU_IDTLS Register Diagram

Table 12-12: SMPU_IDTLS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	ID of Transaction. The SMPU_IDTLS.ID bit field provides the ID of the transaction that caused the interrupt.	
1 (R/NW)	RNW	Read/Write Status. The SMPU_IDTLS.RNW bit indicates whether the last transaction that caused the interrupt was a read or write.	
		0	Transaction that caused last signaled interrupt was a write
		1	Transaction that caused last signaled interrupt was a read
0 (R/NW)	SECURE	Secure Status. The SMPU_IDTLS.SECURE bit indicates whether the last transaction that caused the interrupt was secure or non-secure.	
		0	Transaction that caused last signaled interrupt was non-secure
		1	Transaction that caused last signaled interrupt was secure

Region n Address Register

The SMPU_RADDR[n] register is used to define the base address for a memory region to be protected.

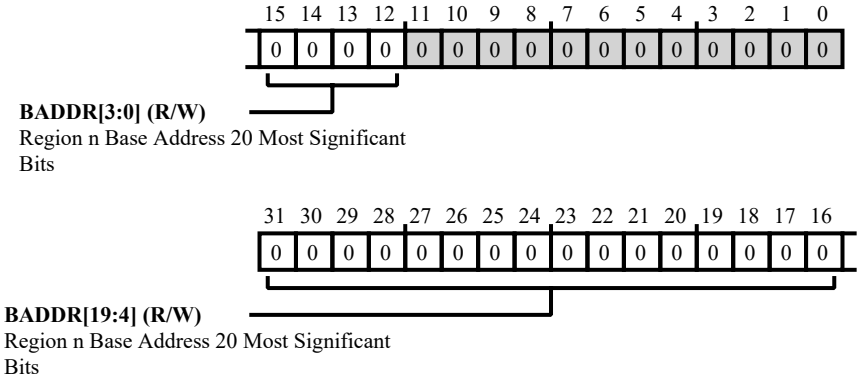


Figure 12-9: SMPU_RADDR[n] Register Diagram

Table 12-13: SMPU_RADDR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/W)	BADDR	Region n Base Address 20 Most Significant Bits. The SMPU_RADDR[n].BADDR bit field defines the base address for a memory region to be protected.

Region n Control Register

The SMPU_RCTL[n] register is used to define the level of protection for a region of memory. The protection of a region is controlled and defined by this register and the SMPU_RADDR[n], SMPU_RIDA[n], SMPU_RIDB[n], SMPU_RIDMSKA[n], and SMPU_RIDMSKB[n] registers.

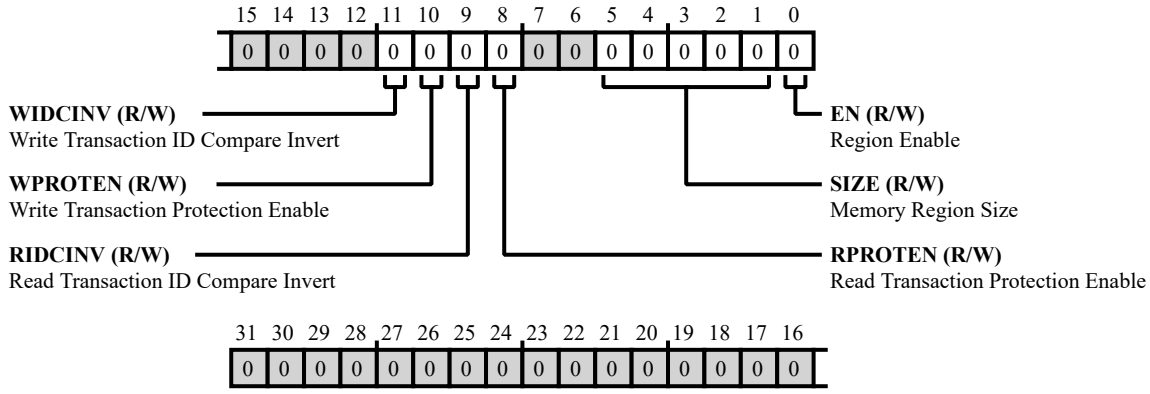


Figure 12-10: SMPU_RCTL[n] Register Diagram

Table 12-14: SMPU_RCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	WIDCINV	Write Transaction ID Compare Invert. The SMPU_RCTL[n].WIDCINV bit inverts the write ID match result.	
		0	Write transaction ID comparison result not inverted
		1	Write transaction ID comparison result inverted
10 (R/W)	WPROTEN	Write Transaction Protection Enable. The SMPU_RCTL[n].WPROTEN bit enables protection against ID-based write transactions for the memory region.	
		0	Write transaction ID-based protection disabled
		1	Write transaction ID-based protection enabled
9 (R/W)	RIDCINV	Read Transaction ID Compare Invert. When the SMPU_RCTL[n].RIDCINV bit is set, the read ID match result is inverted.	
		0	Read transaction ID comparison result not inverted
		1	Read transaction ID comparison result inverted

Table 12-14: SMPU_RCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	RPROTEN	Read Transaction Protection Enable. The `SMPU_RCTL[n].RPROTEN` bit enable bit to turn on protection against ID-based read transactions for the memory region.	
		0	Read transaction ID-based protection disabled
		1	Read transaction ID-based protection enabled
5:1 (R/W)	SIZE	Memory Region Size. The `SMPU_RCTL[n].SIZE` bit defines the size of the memory region to be protected.	
		0	4 KB
		1	8 KB
		2	16 KB
		3	32 KB
		4	64 KB
		5	128 KB
		6	256 KB
		7	512 KB
		8	1 MB
		9	2 MB
		10	4 MB
		11	8 MB
		12	16 MB
		13	32 MB
		14	64 MB
		15	128 MB
		16	256 MB
		17	512 MB
		18	1 GB
		19	2 GB
		20	4 GB
		21-31	Reserved

Table 12-14: SMPU_RCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Region Enable. The `SMPU_RCTL[n].EN` bit enables the protection of a region.	
		0	Disabled
		1	Enabled

SMPU Revision ID Register

The SMPU_REVID register provides the major and minor revision numbers of this module.

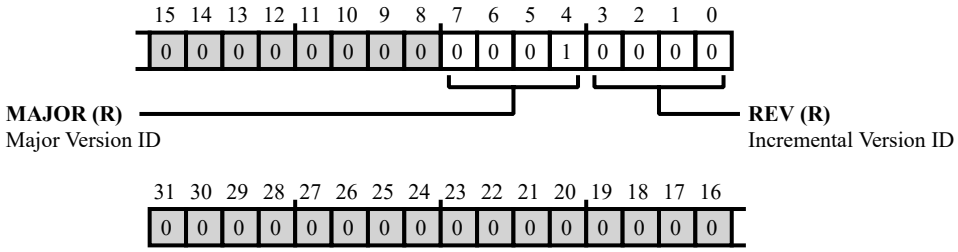


Figure 12-11: SMPU_REVID Register Diagram

Table 12-15: SMPU_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version ID.
3:0 (R/NW)	REV	Incremental Version ID.

Region n ID A Register

The SMPU_RIDA[n] register is used for ID comparison 'A'. This comparison is performed after a mask is applied to both the transaction ID (from either the read or write IDs) and the register value. An ID match means that the ID is the exception to the rule and the read or write is allowed even if the region is read or write-protected. For more detail, refer to the ID Comparison section.

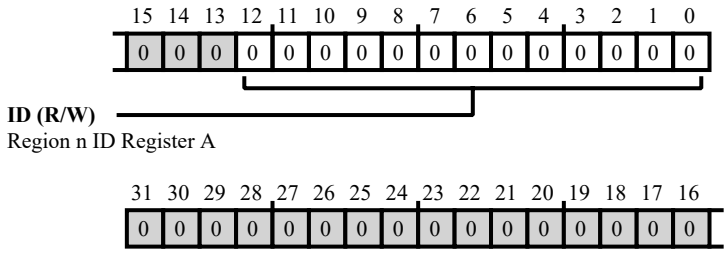


Figure 12-12: SMPU_RIDA[n] Register Diagram

Table 12-16: SMPU_RIDA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	ID	Region n ID Register A. The SMPU_RIDA[n].ID bit field, combined with the mask provides the means to bypass the configured memory protection for a region.

Region n ID B Register

The SMPU_RIDB[n] register is used for ID comparison 'B'. This comparison is performed after a mask is applied to both the transaction ID (from either the read or write IDs) and the register value. An ID match means that the ID is the exception to the rule and the read or write is allowed even if the region is read or write-protected. For more details, refer to the ID Comparison section.

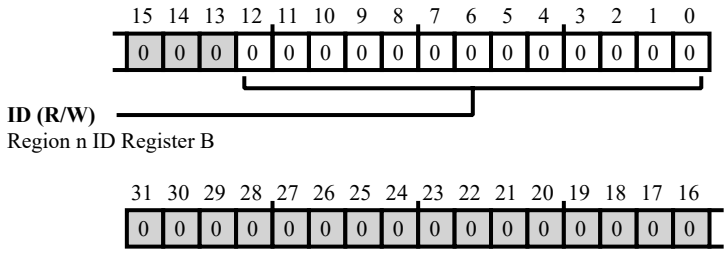


Figure 12-13: SMPU_RIDB[n] Register Diagram

Table 12-17: SMPU_RIDB[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	ID	Region n ID Register B. The SMPU_RIDB[n].ID bit field, combined with the mask provides the means to bypass the configured memory protection for a region.

Region n ID Mask A Register

The SMPU_RIDMSKA[n] register is used for ID comparison 'A'. The mask allows or disallows certain IDs from affecting the final result of the ID match. For more details, refer to the ID Comparison section.

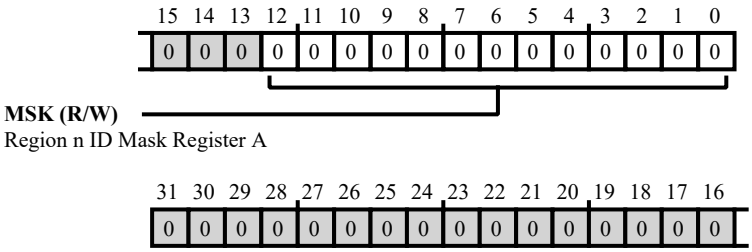


Figure 12-14: SMPU_RIDMSKA[n] Register Diagram

Table 12-18: SMPU_RIDMSKA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	MSK	Region n ID Mask Register A. The SMPU_RIDMSKA[n].MSK bit field, combined with the incoming transaction, provides the means to bypass the configured memory protection for a region.

Region n ID Mask B Register

The SMPU_RIDMSKB[n] register is used for ID comparison 'B'. The mask allows or disallows certain IDs from affecting the final result of the ID match. For more details, refer to the ID Comparison section.

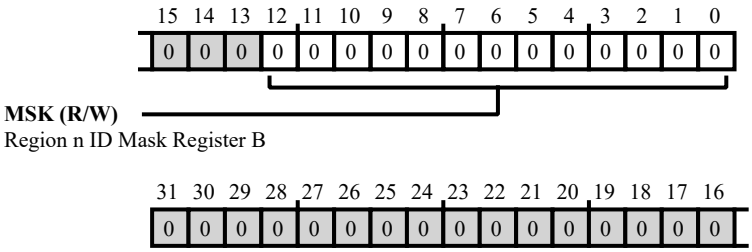


Figure 12-15: SMPU_RIDMSKB[n] Register Diagram

Table 12-19: SMPU_RIDMSKB[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	MSK	Region n ID Mask Register B. The SMPU_RIDMSKB[n].MSK bit field, combined with the incoming transaction provides the means to bypass the configured memory protection for a region.

SMPU Control Secure Accesses Register

The SMPU_SECURECTL register provides the bits required to set up the security settings for the processor. These settings includes error generation and read/write security.

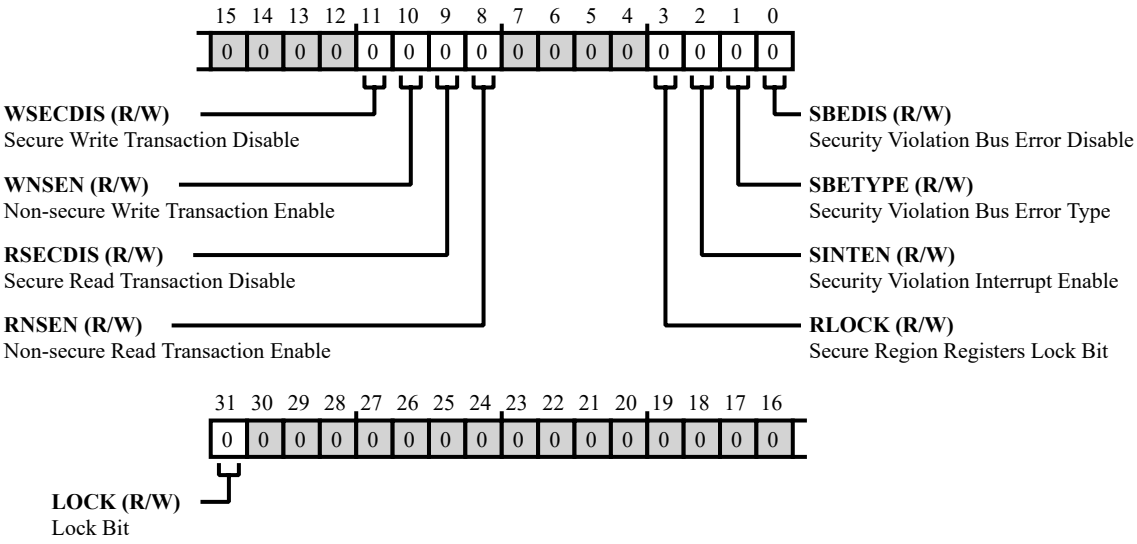


Figure 12-16: SMPU_SECURECTL Register Diagram

Table 12-20: SMPU_SECURECTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit. When the SMPU_SECURECTL.LOCK bit is set and the global lock signal is asserted from the SPU, the SMPU_SECURECTL register is write-protected. Write-protection is disabled only when the global lock signal becomes deasserted again.	
		0	SMPU_SECURECTL is not write-protected
		1	SMPU_SECURECTL is write-protected
11 (R/W)	WSECDIS	Secure Write Transaction Disable. The SMPU_SECURECTL.WSECDIS bit disables secure write transactions.	
		0	Enable secure write transactions
		1	Disable secure write transactions
10 (R/W)	WNSEN	Non-secure Write Transaction Enable. The SMPU_SECURECTL.WNSEN bit enables non-secure write transactions.	
		0	Disable non-secure writes
		1	Enable non-secure writes

Table 12-20: SMPU_SECURECTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	RSECDIS	Secure Read Transaction Disable. The SMPU_SECURECTL.RSECDIS bit disables secure read transactions.	
		0	Enable secure read transactions
		1	Disable secure read transactions
8 (R/W)	RNSEN	Non-secure Read Transaction Enable. The SMPU_SECURECTL.RNSEN bit enables non-secure read transactions.	
		0	Disable non-secure read transactions
		1	Enable non-secure read transactions
3 (R/W)	RLOCK	Secure Region Registers Lock Bit. When the SMPU_SECURECTL.RLOCK bit is set, the secure region control registers, SMPU_SECURERCTL[n], are write-protected when the global lock signal is active from the SPU. When the global lock signal is deasserted, write access is allowed again.	
		0	Disable write-protection on secure region registers
		1	Enable write-protection on secure region registers
2 (R/W)	SINTEN	Security Violation Interrupt Enable. The SMPU_SECURECTL.SINTEN bit enables interrupt generation when a security violation occurs.	
		0	Disable security settings violation interrupt
		1	Enable security settings violation interrupt
1 (R/W)	SBETYPE	Security Violation Bus Error Type. The SMPU_SECURECTL.SBETYPE bit controls whether a decode error or a completer error is returned when a security violation occurs.	
		0	Decode error
		1	Completer error
0 (R/W)	SBEDIS	Security Violation Bus Error Disable. The SMPU_SECURECTL.SBEDIS bit controls whether or not a bus error is caused when a security violation occurs.	
		0	Enable bus error
		1	Disable bus error

Region n Control Secure Accesses Register

The SMPU_SECURERCTL[n] register contains bits that configure read/write security for a specific region.

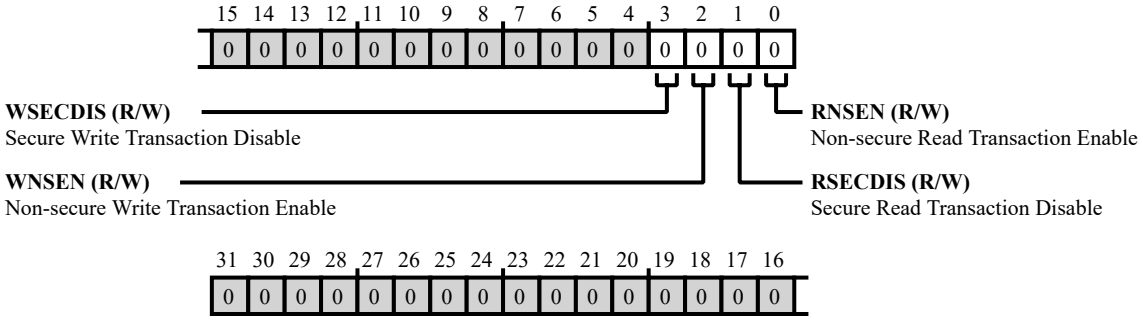


Figure 12-17: SMPU_SECURERCTL[n] Register Diagram

Table 12-21: SMPU_SECURERCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	WSECDIS	Secure Write Transaction Disable. The SMPU_SECURERCTL[n].WSECDIS bit disables secure write transactions for the memory region.	
		0	Enable secure write transactions to this region
		1	Disable secure write transactions to this region
2 (R/W)	WNSEN	Non-secure Write Transaction Enable. This SMPU_SECURERCTL[n].WNSEN bit enables non-secure write transactions for the memory region.	
		0	Disable non-secure write transactions to this region
		1	Enable non-secure write transactions to this region
1 (R/W)	RSECDIS	Secure Read Transaction Disable. The SMPU_SECURERCTL[n].RSECDIS bit disables secure read transactions for the memory region.	
		0	Enable secure read transactions to this region
		1	Disable secure read transactions to this region
0 (R/W)	RNSEN	Non-secure Read Transaction Enable. The SMPU_SECURERCTL[n].RNSEN bit enables non-secure read transactions for the memory region.	
		0	Disable non-secure read transactions to this region
		1	Enable non-secure read transactions to this region

SMPU Status Register

The SMPU_STAT register provides the state of the SMPU and indicates various errors. All bits in this register are write 1 to clear.

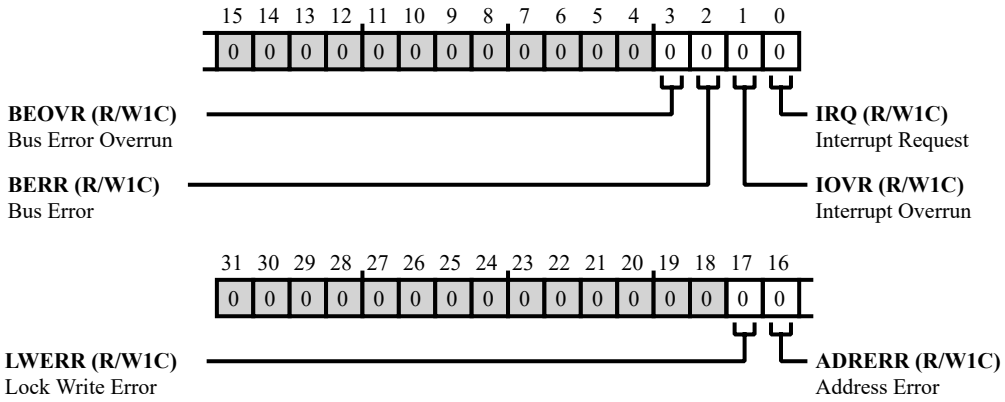


Figure 12-18: SMPU_STAT Register Diagram

Table 12-22: SMPU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	LWERR	Lock Write Error. The SMPU_STAT.LWERR bit is set when SMPU_CTL.LOCK bit =1, the global lock signal is asserted from the SPU and a read or write attempt was made to the SMPU_CTL MMR.	
		0	No Lock Write Error
		1	Lock Write Error
16 (R/W1C)	ADRERR	Address Error. The SMPU_STAT.ADRERR bit is set when the SMPU MMR is accessed as an unaligned address, or when a read-only MMR is written to.	
		0	No Address Error
		1	Address Error
3 (R/W1C)	BEOVR	Bus Error Overrun. The SMPU_STAT.BEOVR bit indicates that another bus error had occurred. Any new information about the most recent violation which caused the bus error is not captured.	
		0	No Bus Error overrun
		1	Bus Error overrun has occurred

Table 12-22: SMPU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	BERR	Bus Error. This `SMPU_STAT.BERR` bit indicates if a bus error was generated.	
		0	No Bus Error since this bit has been cleared
		1	Bus Error has been generated
1 (R/W1C)	IOVR	Interrupt Overrun. The `SMPU_STAT.IOVR` bit indicates if another violation occurred while the previous violation interrupt was not finished being serviced. Information about the most recent violation is then not captured.	
		0	No Interrupt overrun
		1	Interrupt overrun has occurred
0 (R/W1C)	IRQ	Interrupt Request. The `SMPU_STAT.IRQ` bit provides an indication that an interrupt has been generated.	
		0	No Interrupt since this bit has been cleared
		1	Interrupt has been generated

13 General-Purpose Ports (PORT)

This section describes general-purpose ports, pin multiplexing, general-purpose input/output (GPIO) functionality, and pin interrupts. The general-purpose ports provide the following three functions:

- Pin multiplexing scheme
- GPIO functionality
- Pin interrupt requests

NOTE: In this chapter, the naming convention for registers and bits omits the alphabetic group enumeration to refer to any and all of the ports. For example, `PORT_FER` represents registers `PORTA_FER`, `PORTB_FER`, and so on. Likewise `PORT_FER.PX1` represents bits PA1, PB1, and so on.

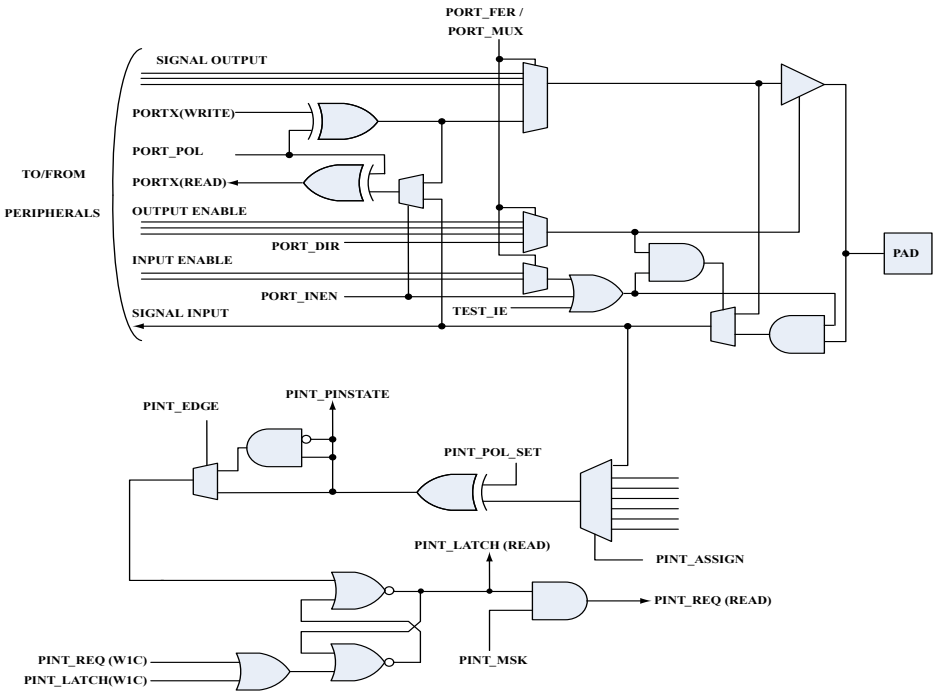


Figure 13-1: Simplified GPIO and Pin Interrupt Signal Flow

PORT Features

The PORTs include the following features:

- Input mode, output mode, and open-drain mode of GPIO operation
- Port multiplexing controlled on a pin-by-pin basis
- No external glue hardware required for unused pins
- All port pins provide interrupt request functionality
- Byte-wide pin-to-interrupt request assignment

PORT Functional Description

The number of ports and their composition is defined in the processor data sheet. Each port has a dedicated set of MMR registers that control pin functions and operates in general-purpose I/O (GPIO) mode by default, as controlled by the port-specific PORT_FER register. Each bit in this register, as well as the other PORT MMRs, represents a specific GPIO pin on the specified port.

Input Mode, Output Mode, and Open-Drain Mode of GPIO Operation

At reset, every GPIO pin defaults to input mode with the input drivers disabled. To enable any GPIO input driver, set the bits corresponding to the individual pins in the appropriate input enable register (PORT_INEN).

The GPIO output drivers are enabled by setting the corresponding bits in the direction registers (PORT_DIR).

The PORT can use every GPIO in open-drain mode by clearing the respective bit in the PORT_DATA register or setting the respective bit in the PORT_DATA_CLR register. Then, set the corresponding bit in the PORT_INEN register. Read from the PORT_DATA register to obtain the status from the pin.

Port Multiplexing Controlled on Pin-by-Pin Basis

Each port has two dedicated MMRs that control the port multiplexing, the 16-bit function enable (PORT_FER) registers and the 32-bit port multiplexing (PORT_MUX) registers.

All Port Pins Provide Interrupt Functionality

Pin interrupts are completely decoupled from GPIO functionality. Pins are connected directly via the PINTx modules, each of which is configurable in terms of which port pins are sensed for interrupt generation.

ADSP-2184x PORT Register List

The PORT module (PORT) regulates the use of the multiplexable processor pins. Every port pin can operate in general-purpose I/O (GPIO) mode or as an alternate function. This GPIO operation is the default after processor reset and is controlled by a set of registers that control GPIO functionality. Every bit in these registers represents a certain GPIO pin of a specific port. For more information on PORT functionality, see the PORT register descriptions.

Table 13-1: ADSP-2184x PORT Register List

Name	Description
PORT_DATA	Port x GPIO Data Register
PORT_DATA_CLR	Port x GPIO Data Clear Register
PORT_DATA_SET	Port x GPIO Data Set Register
PORT_DATA_TGL	Port x GPIO Output Toggle Register
PORT_DIR	Port x GPIO Direction Register
PORT_DIR_CLR	Port x GPIO Direction Clear Register
PORT_DIR_SET	Port x GPIO Direction Set Register
PORT_FER	Port x Function Enable Register
PORT_FER_CLR	Port x Function Enable Clear Register
PORT_FER_SET	Port x Function Enable Set Register
PORT_INEN	Port x GPIO Input Enable Register
PORT_INEN_CLR	Port x GPIO Input Enable Clear Register
PORT_INEN_SET	Port x GPIO Input Enable Set Register
PORT_LOCK	Port x GPIO Lock Register
PORT_MUX	Port x Multiplexer Control Register
PORT_POL	Port x GPIO Polarity Invert Register
PORT_POL_CLR	Port x GPIO Polarity Invert Clear Register
PORT_POL_SET	Port x GPIO Polarity Invert Set Register
PORT_TRIG_TGL	Port x GPIO Trigger Toggle Register

ADSP-2184x PORT Trigger List

Table 13-2: ADSP-2184x PORT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
None			

Table 13-3: ADSP-2184x PORT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
136	PORTA_TOGGLE	Port Toggle Trigger	Pulse
137	PORTB_TOGGLE	Port Toggle Trigger	Pulse
138	PORTC_TOGGLE	Port Toggle Trigger	Pulse
139	PORTD_TOGGLE	Port Toggle Trigger	Pulse
140	PORTE_TOGGLE	Port Toggle Trigger	Pulse
141	PORTF_TOGGLE	Port Toggle Trigger	Pulse
142	PORTG_TOGGLE	Port Toggle Trigger	Pulse
143	PORTH_TOGGLE	Port Toggle Trigger	Pulse

ADSP-2184x PINT Register List

The Pin Interrupt module (PINT) controls the pin-to-interrupt assignment in a byte-wide manner. The pin-interrupt assignment registers do not consist of 32 individual bits. They consist of four control bytes, each functioning as a multiplexer control. For more information, see the PINT register descriptions.

All PINT registers are 32 bits wide and can be accessed by 32-bit load/store instructions. They also support 16-bit operation where the upper 16 bits are ignored and the application uses the lower 16 bits only. Consequently, all PINT registers support 32-bit accesses as well as 16-bit accesses for the lower half words. Applications may use faster 16-bit accesses as long as they do not require functionality of upper register halves.

Table 13-4: ADSP-2184x PINT Register List

Name	Description
PINT_ASSIGN	PINT Assign Register
PINT_EDGE_CLR	PINT Edge Clear Register
PINT_EDGE_SET	PINT Edge Set Register
PINT_INV_CLR	PINT Invert Clear Register
PINT_INV_SET	PINT Invert Set Register
PINT_LATCH	PINT Latch Register
PINT_MSK_CLR	PINT Mask Clear Register
PINT_MSK_SET	PINT Mask Set Register
PINT_PINSTATE	PINT Pin State Register
PINT_REQ	PINT Request Register

ADSP-2184x PINT Interrupt List

Table 13-5: ADSP-2184x PINT Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
217	PINT0_BLOCK	PINT0 Pin Interrupt Block 0	Level	
218	PINT1_BLOCK	PINT1 Pin Interrupt Block 1	Level	
219	PINT2_BLOCK	PINT2 Pin Interrupt Block 2	Level	
220	PINT3_BLOCK	PINT3 Pin Interrupt Block 3	Level	
221	PINT4_BLOCK	PINT4 Pin Interrupt Block 4	Level	
222	PINT5_BLOCK	PINT5 Pin Interrupt Block 5	Level	
223	PINT6_BLOCK	PINT6 Pin Interrupt Block 6	Level	
224	PINT7_BLOCK	PINT7 Pin Interrupt Block 7	Level	

ADSP-2184x PINT Trigger List

Table 13-6: ADSP-2184x PINT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
108	PINT0_BLOCK	PINT0 Pin Interrupt Block	Level
109	PINT1_BLOCK	PINT1 Pin Interrupt Block	Level
110	PINT2_BLOCK	PINT2 Pin Interrupt Block	Level
111	PINT3_BLOCK	PINT3 Pin Interrupt Block	Level
112	PINT4_BLOCK	PINT4 Pin Interrupt Block	Level
113	PINT5_BLOCK	PINT5 Pin Interrupt Block	Level
114	PINT6_BLOCK	PINT6 Pin Interrupt Block	Level
115	PINT7_BLOCK	PINT7 Pin Interrupt Block	Level

Table 13-7: ADSP-2184x PINT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

ADSP-2184x PADS Register List

The PADS controls signal hysteresis and other system interface signal features for a number of module interfaces.

Table 13-8: ADSP-2184x PADS Register List

Name	Description
PADS_DAI0_0_DS	DAI0 Pins 1 to 8 Drive Strength Control Register

Table 13-8: ADSP-2184x PADS Register List (Continued)

Name	Description
PADS_DAI0_1_DS	DAI0 Pins 9 to 16 Drive Strength Control Register
PADS_DAI0_2_DS	DAI0 Pins 17 to 20 Drive Strength Control Register
PADS_DAI0_IE	DAI0 Port Input Enable Control Register
PADS_DAI0_PE	DAI0 Pull Enable Register
PADS_DAI0_PS	DAI0 Pull Selector Register
PADS_DAI0_SL	DAI0 Slew Rate Control Register
PADS_DAI0_SPU	DAI0 Strong Pull Up Control Register
PADS_DAI1_0_DS	DAI1 Pins 1 to 8 Drive Strength Control Register
PADS_DAI1_1_DS	DAI1 Pins 9 to 16 Drive Strength Control Register
PADS_DAI1_2_DS	DAI1 Pins 17 to 20 Drive Strength Control Register
PADS_DAI1_IE	DAI1 Port Input Enable Control Register
PADS_DAI1_PE	DAI1 Pull Enable Register
PADS_DAI1_PS	DAI1 Pull Selector Register
PADS_DAI1_SL	DAI1 Slew Rate Control Register
PADS_DAI1_SPU	DAI1 Strong Pull up Control Register
PADS_NONPORTS_CTL	Non-GPIO Control Register
PADS_NONPORTS_DS	Non-GPIO Drive Strength Register
PADS_NONPORTS_DS_1	EMAC Non-GPIO Drive Strength Register
PADS_OTPC_BOOT_STAT	OTPC Boot Status Register
PADS_PCFG0	Peripheral PAD Configuration 0 Register
PADS_PCFG1	Peripheral Configuration 1 Register
PADS_PORTA0_DS	PORTA Pins 0 to 7 Drive Strength Control Register
PADS_PORTA1_DS	PORTA Pins 8 to 15 Drive Strength Control Register
PADS_PORTB0_DS	PORTB Pins 0 to 7 Drive Strength Control Register
PADS_PORTB1_DS	PORTB Pins 8 to 15 Drive Strength Control Register
PADS_PORTC0_DS	PORTC Pins 0 to 7 Drive Strength Control Register
PADS_PORTC1_DS	PORTC Pins 8 to 15 Drive Strength Control Register
PADS_PORTD0_DS	PORTD Pins 0 to 7 Drive Strength Control Register
PADS_PORTD1_DS	PORTD Pins 8 to 15 Drive Strength Control Register
PADS_PORTE0_DS	PORTE Pins 0 to 7 Drive Strength Control Register
PADS_PORTE1_DS	PORTE Pins 8 to 15 Drive Strength Control Register

Table 13-8: ADSP-2184x PADS Register List (Continued)

Name	Description
PADS_PORTF0_DS	PORTF Pins 0 to 7 Drive Strength Control Register
PADS_PORTF1_DS	PORTF Pins 8 to 15 Drive Strength Control Register
PADS_PORTG0_DS	PORTG Pins 0 to 7 Drive Strength Control Register
PADS_PORTG1_DS	PORTG Pins 8 to 15 Drive Strength Control Register
PADS_PORTH0_DS	PORTH Pins 0 to 7 Drive Strength Control Register
PADS_PORTH1_DS	PORTH Pins 8 and 9 Drive Strength Control Register
PADS_PORT_AB_PE	PORTA and PORTB Pull Enable Register
PADS_PORT_AB_PS	PORTA and PORTB Pull Selector Register
PADS_PORT_AB_SL	PORTA and PORTB Slew Rate Control Register
PADS_PORT_AB_SPU	PORTA and PORTB Strong Pull Up Control Register
PADS_PORT_CD_PE	PORTC and PORTD Pull Enable Register
PADS_PORT_CD_PS	PORTC and PORTD Pull Selector Register
PADS_PORT_CD_SL	PORTC and PORTD Slew Rate Control Register
PADS_PORT_CD_SPU	PORTC and PORTD Strong Pull Up Control Register
PADS_PORT_EF_PE	PORTE and PORTF Pull Enable Register
PADS_PORT_EF_PS	PORTE and PORTF Pull Selector Register
PADS_PORT_EF_SL	PORTE and PORTF Slew Rate Control Register
PADS_PORT_EF_SPU	PORTE and PORTF Strong Pull Up Control Register
PADS_PORT_GH_PE	PORTG and PORTH Pull Enable Register
PADS_PORT_GH_PS	PORTG and PORTH Pull Selector Register
PADS_PORT_GH_SL	PORTG and PORTH Slew Rate Control Register
PADS_PORT_GH_SPU	PORTG and PORTH Strong Pull Up Control Register
PADS_STAT	Status Register

PORT Architectural Concepts

These sections describe in more detail how the PORT module connects externally to pins and internally to the MMR bus. Ports are named alphabetically beginning with A.

- Internal Interfaces
- External Interfaces
- GPIO Pin Function
- Port Multiplexing Control

Internal Interfaces

All of the pin multiplexing, GPIO, and pin interrupt control block MMRs can be accessed through the MMR bus. There is no DMA support. Each of the pin interrupt (PINTx) modules has its own dedicated interrupt request output signal that connects directly to the system core.

External Interfaces

The pin multiplexing hardware can be seen as a layer between the on-chip peripherals and the silicon pads connecting to the physical pins/balls or the package, as controlled by the PORT unit.

GPIO Pin Function

By default, the PORT sets every GPIO pin to input mode. The input drivers are not enabled, which avoids the need for unnecessary current sinks and external termination resistors on unused pins.

Input Mode

The default mode of every GPIO pin after reset is input mode, but the input drivers are not enabled. To enable GPIO input drivers, set the bits corresponding to the PORT pins in the appropriate input enable register (PORT_INEN). When enabled, a read from the PORT_DATA register returns the logical state of the input pins. However, the input signal does not overwrite the state of the internal flip-flop used for providing output to the same pin. Only software can alter the state. If the input driver is enabled, a write to the PORT_DATA register can alter the state of the flip-flop, but the change cannot be read back.

Output Mode

Any GPIO pin can be configured for output mode. The GPIO output drivers are enabled by setting the bits corresponding to the PORT pins in the appropriate direction register. The PORT implements direction registers as a pair of write-1-to-set (W1S) and write-1-to-clear (W1C) MMRs called PORT_DIR_SET and PORT_DIR_CLR, respectively. As such, software can alter the direction of the signal flow on individual GPIO pins without impacting other GPIOs on the same port.

Both the PORT_DIR_SET and PORT_DIR_CLR registers return the same value when read, and a logical 1 indicates an enabled output. The PORT_DATA registers control the state of output pins. A logical 0 drives the output low while a logical 1 drives the output high.

While writes to the PORT_DATA register can alter all of the GPIOs on a specific port at once, there are also a pair of W1S and W1C MMRs called PORT_DATA_SET and PORT_DATA_CLR, respectively. These registers enable the manipulation of individual GPIO outputs. The state of the outputs can be obtained by reading the PORT_DATA registers. Because the state of the GPIO output can be controlled before the output driver is enabled, set or clear the internal flip-flop first by programming this register to avoid volatile levels on the output pin.

Trigger Toggle Mode

Any GPIO pin that has been configured for output mode can be toggled using a trigger input from the Trigger Routing Unit (TRU). To enable this functionality for a particular GPIO, set the appropriate bit in the PORT_TRIG_TGL register. Any subsequent trigger for the designated port causes all GPIO outputs set in the PORT_TRIG_TGL register to toggle.

To avoid unpredictable behavior, do not set, clear, or toggle the PORT_DATA, PORT_DATA_SET, PORT_DATA_CLR, or PORT_DATA_TGL registers when the GPIO output has the corresponding PORT_TRIG_TGL bit set. To change the state of the GPIO output using one of these registers, first clear the corresponding bit in the PORT_TRIG_TGL register.

Open-Drain Mode

Every GPIO can also be used in open-drain mode. First, either clear the respective bit in the PORT_DATA register or set the respective bit in the PORT_DATA_CLR register. Then, set the appropriate bit in the PORT_INEN register. Read from the PORT_DATA register to return the status from the pin rather than the state of the internal flip-flop.

By toggling the output driver through the PORT_DIR_SET and PORT_DIR_CLR register pair, the output signal can be pulled low or three-stated, as required. The polarity of the driven signal can be inverted when the internal flip-flop is set. When using a GPIO port in open-drain mode, take care to not exceed the V_{IH} operating condition associated with the respective pins.

Port Multiplexing Control

To configure pins properly, consult the processor data sheet to determine which bits in the PORT_FER and PORT_MUX register map to the pin of interest, and then set these registers appropriately for the desired function.

After reset, all port pins default to GPIO input mode with their output and input drivers disabled. As a result, all unused port pins can be left unconnected. Disabled pins appear as a high impedance to external circuits.

Each port has two dedicated MMRs that control the port multiplexing, the 16-bit function enable (PORT_FER) registers and the 32-bit port multiplexing (PORT_MUX) registers.

The function enable register specifies whether the pin is used as a GPIO pin or allocated for use by a specific peripheral, but it does not specify what the peripheral function is. Each bit in the 16-bit PORT_FER register corresponds to an individual port pin. For example, if bit 1 (PA1) of the PORTA_FER register is cleared, the PA_01 pin is configured as a GPIO. When set, one of the available peripheral functions becomes active on the PA_01 pin instead.

Pairs of bits in the PORT_MUX register control the multiplexing between the peripheral functions available to an individual pin, as some PORT pins provide up to four possible peripheral functions.

Refer to the Signal Muxing table in the data sheet for the specific PORT_MUX settings.

PORT Event Control

The following sections describe event generation in the PORT module.

PORT Interrupt Signals

The pin interrupts are decoupled from GPIO functionality, providing the following advantages.

- Flexible mapping scheme enables pins from up to four different ports to be grouped into one common interrupt scheme.

- Interrupt requests work on input and output pins regardless of whether the pin is functioning as a GPIO or a peripheral.

The processor has a number of interrupt channels dedicated to pin interrupts, managed by a set of pin interrupt (PINTx) blocks. Each PINTx block can sense up to 32 GPIO pins, as described in the following list and figure.

The processor supports both 32-bit and 16-bit peripheral bus accesses to PINTx registers.

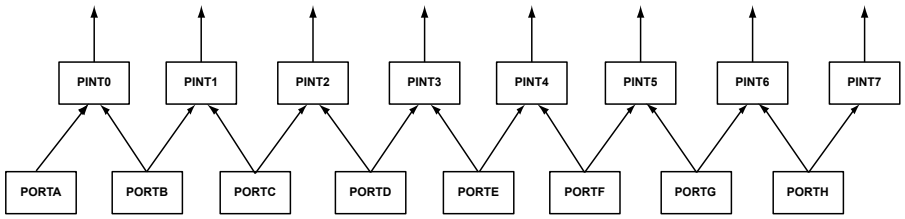


Figure 13-2: GPIO to PINTx Assignment

Pins connect to the PINTx module, as shown in the ***PINTx Block Diagram***.

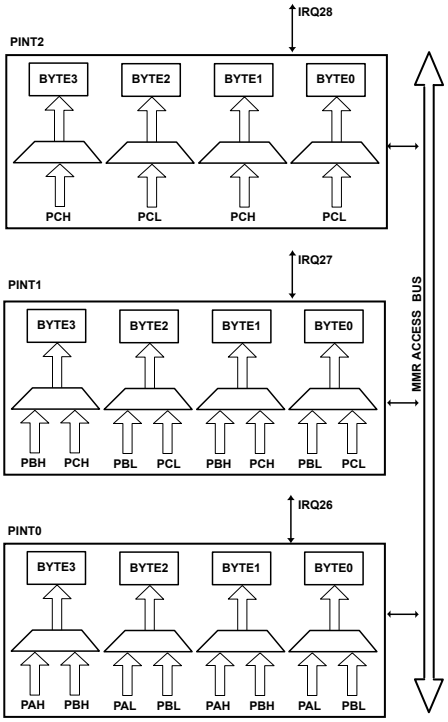


Figure 13-3: PINTx Block Diagram

As shown in the ***PINTx Block Diagram***, each port is subdivided into two 8-pin half ports, upper (`PxH`) and lower (`PxL`). The `PINT_ASSIGN` registers control the 8-bit multiplexers associated with these half ports, where the lower half units (eight pins) can be forwarded to either byte 0 or byte 2 of the PINTx blocks, and the upper half units (eight pins) can be forwarded to either byte 1 or byte 3 of the PINTx blocks.

When a half port is assigned to a byte in any PINTx block, the state of the eight pins appears in the PINT_PINSTATE register, regardless of whether the pin is enabled for GPIO or peripheral functions (input or output). When neither the input nor output drivers of the pin are enabled, the pin state is read as zero. The PINT_PINSTATE register reports the inverted state of the pin when the PINT_INV_SET register activates the signal inverter. The inverter can be enabled on an individual bit-by-bit basis. Each bit in the PINT_INV_SET/ PINT_INV_CLR register pair represents a pin signal.

By default, PORT interrupt request generation is level-sensitive, and an interrupt request occurs when the enabled pin is sensed as active high. Use the PINT_EDGE_SET register to change the interrupt request generation scheme to instead be edge-sensitive (rising edge generates the interrupt request). Use the PINT_INV_SET register to invert the polarity such that the PINTx block generates the interrupt request on active-low signals or falling edges.

The PINTx modules also assist when both signal edges must generate unique interrupt requests. If two different interrupt requests are required, the PINT_ASSIGN registers can route a single input signal to two different PINTx blocks, where one block inverts the signal in the PINT_INV_SET register and the other one does not. In this fashion, a unique software routine is associated with the hardware PINTx block that is generating the unique interrupt request for each signal edge. When both signal edges can be serviced by the same interrupt request, each half port can be routed to two separate bytes within a single PINTx block using the PINT_ASSIGN register, and then one of the half ports needs to have the inversion enabled in the PINT_INV_SET register. The servicing software routine can then detect from the PINT_LATCH register whether a falling, rising, or both edges have occurred.

Regardless of whether level-sensitive or edge-sensitive mode is used, the hardware always latches an interrupt request. Latched signals can be read from the PINT_LATCH registers. Only a software or hardware reset clears the latches. To clear the latch, a W1C operation must be performed to the PINT_REQ or PINT_LATCH register. When in level-sensitive mode, the interrupt request remains asserted if the pin state does not change by the time the interrupt service routine exits.

Because every PINTx block groups up to 32 pin signals, the PINT_MSK_SET/ PINT_MSK_CLR register pair can control which of the signals can request an interrupt at the system level. Software can interrogate the PINT_REQ register for signaling pins. The PINT_REQ bits represent a logical AND between the mask and the latch. When any of these bits is set, the block output interrupt request is asserted.

PORT Programming Model

The ***GPIO Programming Model Flow (Part 1)***, ***GPIO Programming Model Flow (Part 2)***, and ***GPIO Programming Model Flow (Part 3)*** figures show the programming model for the general-purpose ports. This programming includes the GPIO input and output operation, open-drain mode, and the pin interrupt PINTx modules.

NOTE: These process flow diagrams connect where call-out letters appear. For example, call-out A in the ***GPIO Programming Model Flow (Part 1)*** diagram connects to call-out A in the ***GPIO Programming Model Flow (Part 2)*** diagram.

The following flowcharts describe the processes for setting up pins for various functions. Begin the process from the start label in the ***GPIO Programming Model Flow (Part 1)*** figure. The first decision (GPIO or peripheral) determines how to program the PORT_FER register. Set the bit(s) corresponding to the pin(s) to 1 to enable peripheral functionality or to 0 to enable GPIO mode. For more information on setting up for peripheral functions, refer to Port Multiplexing Control.

If the pin is to be a GPIO pin, a subsequent series of decisions must be made that will impact how the PORT_DATA, PORT_POL, PORT_DIR, and PORT_INEN configuration registers must be programmed. Depending on the type of GPIO pin desired, some configurations do not apply and can have different meanings. The following paragraphs briefly describe the function of the different settings for each of the pin functions in the input, output, and open-drain GPIO modes. It is a best practice to use the SET or CLR versions of the PORT registers, where applicable, to effect changes on a pin-by-pin basis rather than on the full port.

For output mode, first clear the PORT_DATA register to set all the pins low. Then write the PORT_DIR register to define the direction of each pin (set the bits associated with the desired output pins to 1). In output mode, the other registers are not significant. The ***GPIO Programming Model Flow (Part 1)*** chart shows this flow starting at label 2.

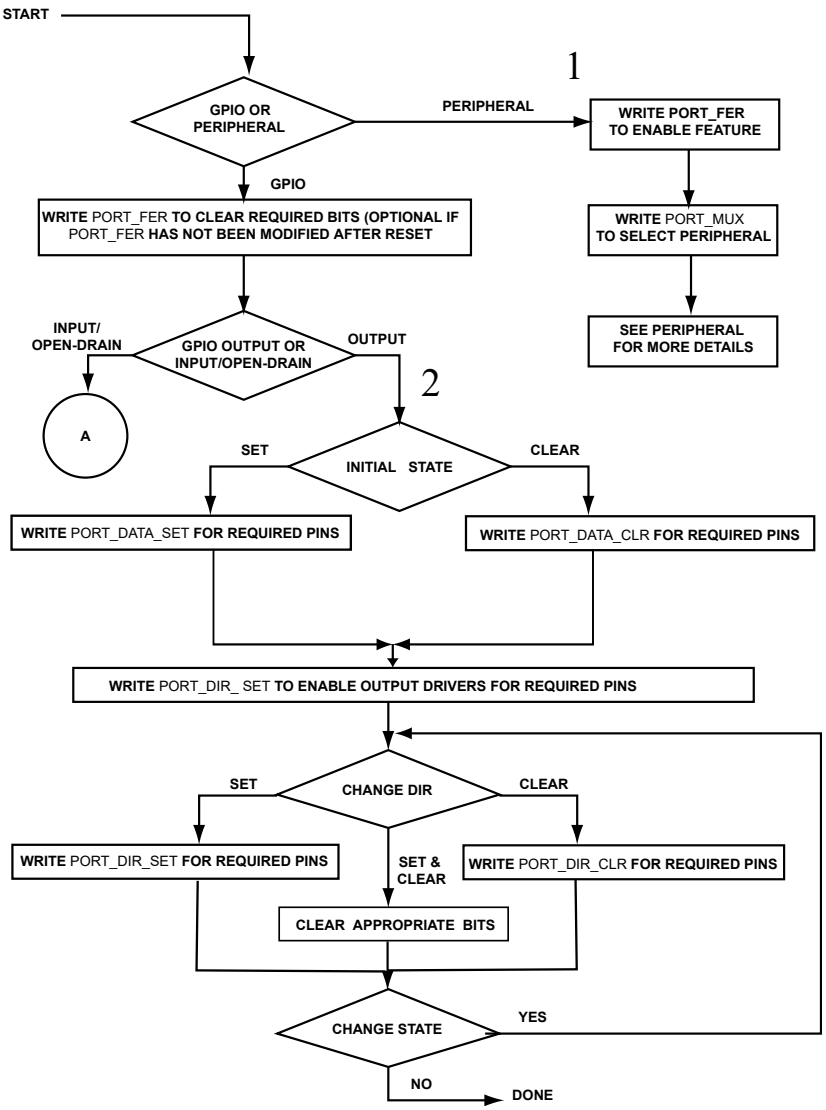


Figure 13-4: GPIO Programming Model Flow (Part 1)

For input mode, first decide the polarity for each pin using the PORT_POL register. Program the PORT_DIR register to define the appropriate pins as inputs (write a 0 to the bit location associated with the pin). If interrupt requests are desired, configure the PINT module as shown in the ***GPIO Programming Model Flow (Part 3)*** figure starting at label B. Finally, write the PORT_INEN register to enable the associated input drivers. The ***GPIO Programming Model Flow (Part 2)*** chart shows this entire flow starting at label 3.

For open-drain mode, set all pins low by clearing the PORT_DATA register. Then, use the PORT_INEN register to enable the appropriate input drivers. Set the PORT_DIR register in this mode to indicate whether the pin is in an active state or not (active being 0). The ***GPIO Programming Model Flow (Part 2)*** chart shows this flow starting at label 4.

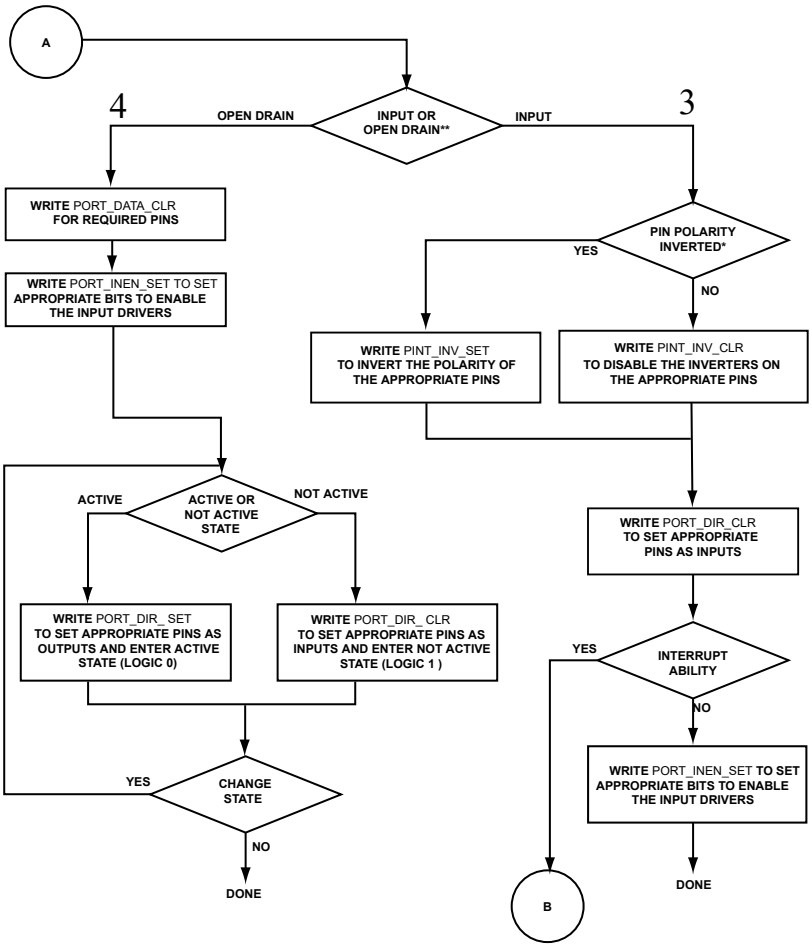


Figure 13-5: GPIO Programming Model Flow (Part 2)

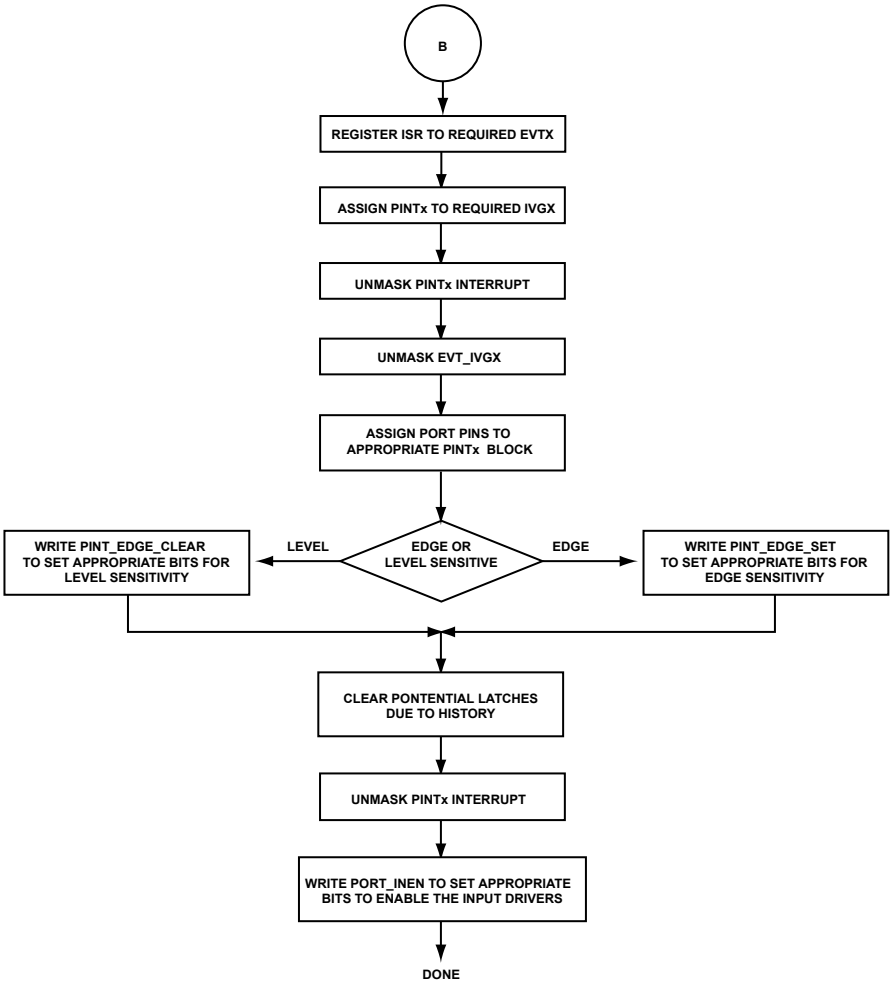


Figure 13-6: GPIO Programming Model Flow (Part 3)

Programmable Pull Enable and Pull Select Resistors for PORT and DAI

There are programmable weak pull-up resistors for DAI and PORT pins. These resistors can be enabled or disabled by configuring the PADS_PORT_AB_PE, PADS_PORT_CD_PE, PADS_PORT_EF_PE, PADS_PORT_GH_PE; PADS_PORT_AB_PS, PADS_PORT_CD_PS, PADS_PORT_EF_PS, PADS_PORT_GH_PS; PADS_DAI0_PE, PADS_DAI1_PE; and PADS_DAI0_PS, PADS_DAI1_PS registers.

Pull Enable (PE) and Pull Select (PS) Control

The GPIO pad provides optional internal pull resistors to define a stable logic level when the pin is not actively driven. The following controls are configurable:

- Pull Enable (PE): Determines whether a pull resistor is connected. When PE = 0, no pull is applied. When PE = 1, a pull resistor is enabled

- Pull Select (PS): Defines the type of pull resistor when PE is enabled. Setting PS = 0 applies a pull-down, while PS = 1 applies a pull-up.
- Strong Pull-Up (SPU): In input mode, enabling SPU forces the PAD strongly high, overriding weak pull configurations

NOTE: For unused PORT/DAI pins, ensure that any unused pin does not go into a floating state, by clearing (=0) the PE bit of that pin.

ADSP-2184x PORT Register Descriptions

The General-Purpose Input/Output Port (PORT) contains the following registers.

Table 13-9: ADSP-2184x PORT Register List

Name	Description
PORT_DATA	Port x GPIO Data Register
PORT_DATA_CLR	Port x GPIO Data Clear Register
PORT_DATA_SET	Port x GPIO Data Set Register
PORT_DATA_TGL	Port x GPIO Output Toggle Register
PORT_DIR	Port x GPIO Direction Register
PORT_DIR_CLR	Port x GPIO Direction Clear Register
PORT_DIR_SET	Port x GPIO Direction Set Register
PORT_FER	Port x Function Enable Register
PORT_FER_CLR	Port x Function Enable Clear Register
PORT_FER_SET	Port x Function Enable Set Register
PORT_INEN	Port x GPIO Input Enable Register
PORT_INEN_CLR	Port x GPIO Input Enable Clear Register
PORT_INEN_SET	Port x GPIO Input Enable Set Register
PORT_LOCK	Port x GPIO Lock Register
PORT_MUX	Port x Multiplexer Control Register
PORT_POL	Port x GPIO Polarity Invert Register
PORT_POL_CLR	Port x GPIO Polarity Invert Clear Register
PORT_POL_SET	Port x GPIO Polarity Invert Set Register
PORT_TRIG_TGL	Port x GPIO Trigger Toggle Register

Port x GPIO Data Register

The operation of the PORT_DATA register depends on whether the bit/pin is in output mode or input mode. In both modes, a set bit in the PORT_DATA register corresponds to a signal high on a GPIO pin. A cleared bit in the PORT_DATA register corresponds to a signal low on a GPIO pin.

The PORT_DATA, PORT_DATA_SET, and PORT_DATA_CLR registers control the state of GPIO pins in output mode. To enable output mode (and output drivers), use the PORT_DIR_SET and PORT_DIR_CLR registers.

Writes to the PORT_DATA register affect the state of all pins of the port that are in output mode. To set or clear specific pins without impacting other pins of the port, use the PORT_DATA_SET and PORT_DATA_CLR registers.

When the GPIO pins are in input mode (input driver is enabled with the PORT_INEN register), reads from the PORT_DATA, PORT_DATA_SET, and PORT_DATA_CLR registers return the state of the respective GPIO pins.

Note that when the input driver is not enabled, reads from the PORT_DATA, PORT_DATA_SET, and PORT_DATA_CLR registers return the value previously written to the registers.

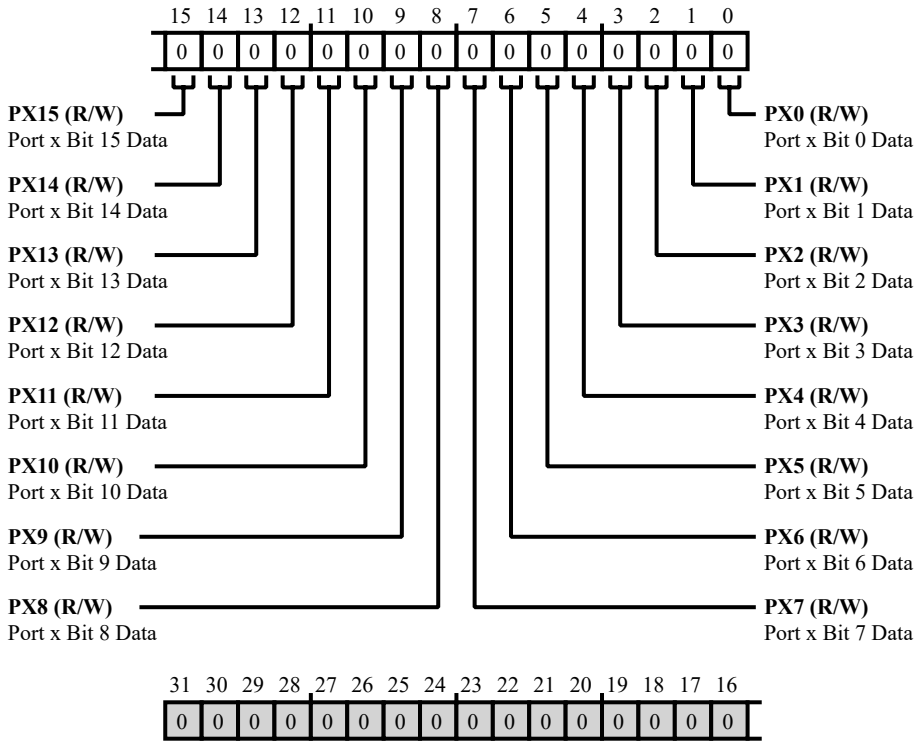


Figure 13-7: PORT_DATA Register Diagram

Table 13-10: PORT_DATA Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Data. The PORT_DATA.PX15 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
14 (R/W)	PX14	Port x Bit 14 Data. The PORT_DATA.PX14 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
13 (R/W)	PX13	Port x Bit 13 Data. The PORT_DATA.PX13 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
12 (R/W)	PX12	Port x Bit 12 Data. The PORT_DATA.PX12 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
11 (R/W)	PX11	Port x Bit 11 Data. The PORT_DATA.PX11 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
10 (R/W)	PX10	Port x Bit 10 Data. The PORT_DATA.PX10 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
9 (R/W)	PX9	Port x Bit 9 Data. The PORT_DATA.PX9 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High

Table 13-10: PORT_DATA Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	PX8	Port x Bit 8 Data. The PORT_DATA.PX8 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
7 (R/W)	PX7	Port x Bit 7 Data. The PORT_DATA.PX7 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
6 (R/W)	PX6	Port x Bit 6 Data. The PORT_DATA.PX6 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
5 (R/W)	PX5	Port x Bit 5 Data. The PORT_DATA.PX5 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
4 (R/W)	PX4	Port x Bit 4 Data. The PORT_DATA.PX4 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
3 (R/W)	PX3	Port x Bit 3 Data. The PORT_DATA.PX3 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
2 (R/W)	PX2	Port x Bit 2 Data. The PORT_DATA.PX2 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High

Table 13-10: PORT_DATA Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PX1	Port x Bit 1 Data. The `PORT_DATA.PX1` bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
0 (R/W)	PX0	Port x Bit 0 Data. The `PORT_DATA.PX0` bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High

Port x GPIO Data Clear Register

The PORT_DATA_CLR register operates differently for port bits/pins, depending on whether the bit/pin is output mode or input mode. For more information, see the PORT_DATA register description.

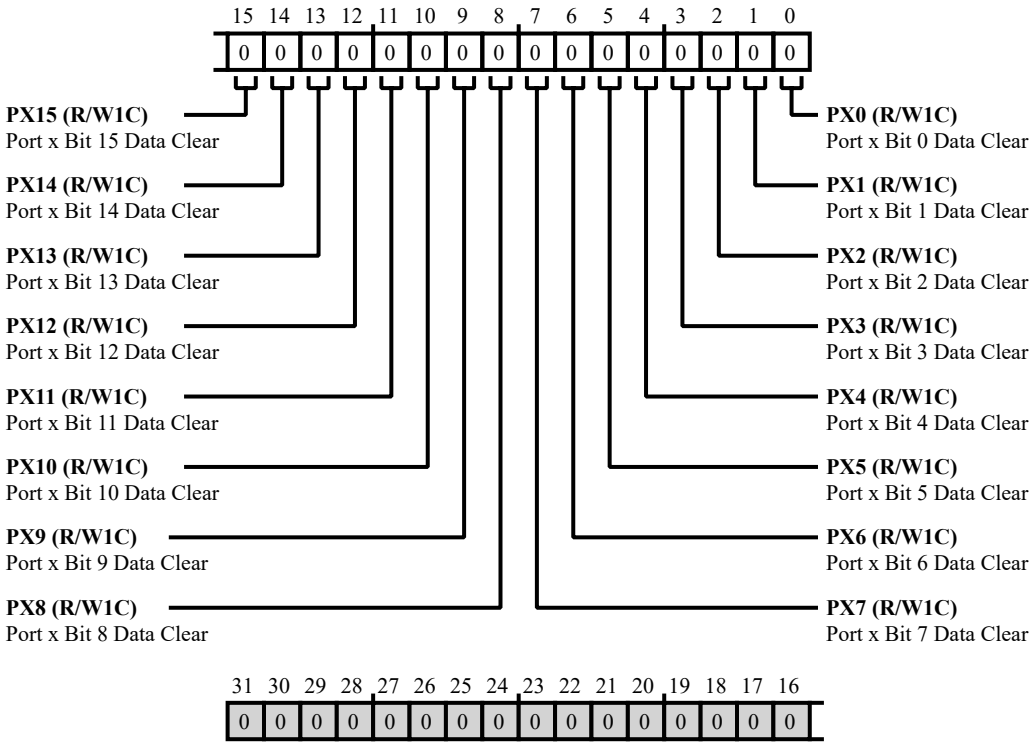


Figure 13-8: PORT_DATA_CLR Register Diagram

Table 13-11: PORT_DATA_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Data Clear. The PORT_DATA_CLR.PX15 bit clears the pin without impacting other pins of the port.	
		0	No Effect
		1	Clear Bit. Write 1 for signal low in output mode.
14 (R/W1C)	PX14	Port x Bit 14 Data Clear. The PORT_DATA_CLR.PX14 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.

Table 13-11: PORT_DATA_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1C)	PX13	Port x Bit 13 Data Clear. The PORT_DATA_CLR.PX13 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
12 (R/W1C)	PX12	Port x Bit 12 Data Clear. The PORT_DATA_CLR.PX12 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
11 (R/W1C)	PX11	Port x Bit 11 Data Clear. The PORT_DATA_CLR.PX11 bit clears the pin without impacting other pins of the port.	
		0	No Effect
		1	Clear Bit. Write 1 for signal low in output mode.
10 (R/W1C)	PX10	Port x Bit 10 Data Clear. The PORT_DATA_CLR.PX10 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
9 (R/W1C)	PX9	Port x Bit 9 Data Clear. The PORT_DATA_CLR.PX9 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
8 (R/W1C)	PX8	Port x Bit 8 Data Clear. The PORT_DATA_CLR.PX8 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.

Table 13-11: PORT_DATA_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	PX7	Port x Bit 7 Data Clear. The PORT_DATA_CLR.PX7 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
6 (R/W1C)	PX6	Port x Bit 6 Data Clear. The PORT_DATA_CLR.PX6 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
5 (R/W1C)	PX5	Port x Bit 5 Data Clear. The PORT_DATA_CLR.PX5 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
4 (R/W1C)	PX4	Port x Bit 4 Data Clear. The PORT_DATA_CLR.PX4 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
3 (R/W1C)	PX3	Port x Bit 3 Data Clear. The PORT_DATA_CLR.PX3 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
2 (R/W1C)	PX2	Port x Bit 2 Data Clear. The PORT_DATA_CLR.PX2 bit clears the pin without impacting other pins of the port.	
		0	No Effect Write 0 has no effect in output mode.
		1	Clear Bit Write 1 for signal low in output mode.

Table 13-11: PORT_DATA_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	PX1	Port x Bit 1 Data Clear. The `PORT_DATA_CLR.PX1` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
0 (R/W1C)	PX0	Port x Bit 0 Data Clear. The `PORT_DATA_CLR.PX0` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.

Port x GPIO Data Set Register

The PORT_DATA_SET register operates differently for port bits/pins, depending on whether the bit/pin is output mode or input mode. For more information, see the PORT_DATA register description.

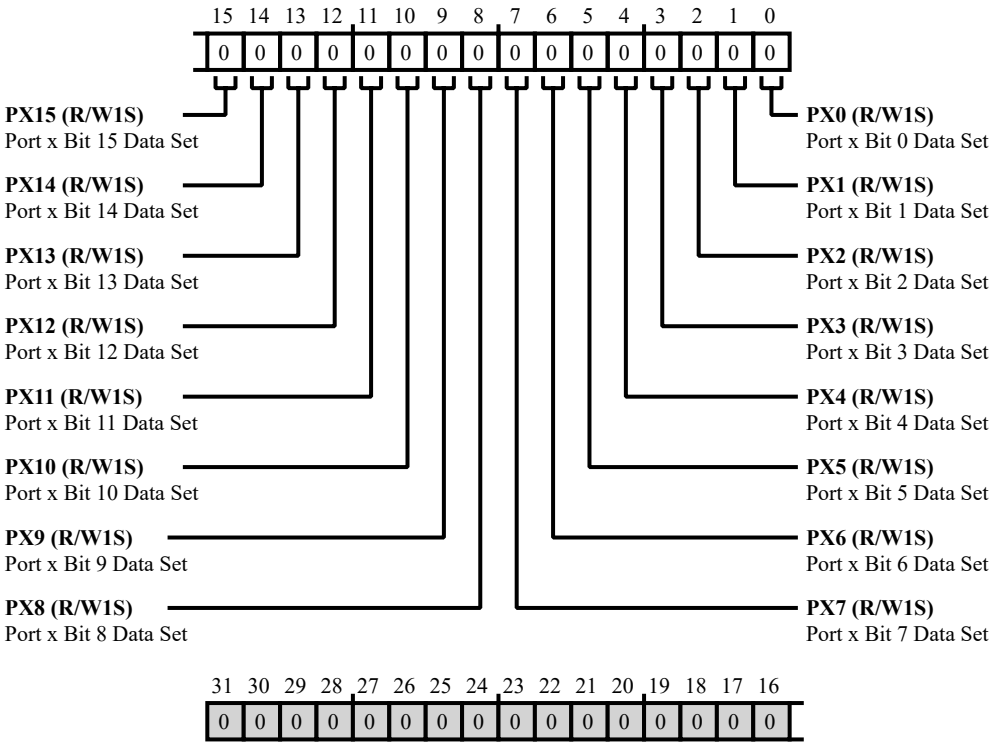


Figure 13-9: PORT_DATA_SET Register Diagram

Table 13-12: PORT_DATA_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
14 (R/W1S)	PX14	Port x Bit 14 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
13 (R/W1S)	PX13	Port x Bit 13 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.

Table 13-12: PORT_DATA_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1S)	PX12	Port x Bit 12 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
11 (R/W1S)	PX11	Port x Bit 11 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit.
10 (R/W1S)	PX10	Port x Bit 10 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
9 (R/W1S)	PX9	Port x Bit 9 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
8 (R/W1S)	PX8	Port x Bit 8 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
7 (R/W1S)	PX7	Port x Bit 7 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
6 (R/W1S)	PX6	Port x Bit 6 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
5 (R/W1S)	PX5	Port x Bit 5 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
4 (R/W1S)	PX4	Port x Bit 4 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
3 (R/W1S)	PX3	Port x Bit 3 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.

Table 13-12: PORT_DATA_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1S)	PX2	Port x Bit 2 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
1 (R/W1S)	PX1	Port x Bit 1 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
0 (R/W1S)	PX0	Port x Bit 0 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.

Port x GPIO Output Toggle Register

The PORT_DATA_TGL register permits toggling the state of output GPIO pins. Setting bits in the PORT_DATA_TGL register affects the state of specific pins without impacting other pins of the port.

Reading the PORT_DATA_TGL returns the state of the PORT_DATA register output pin state, but does not return the input pin/signal state.

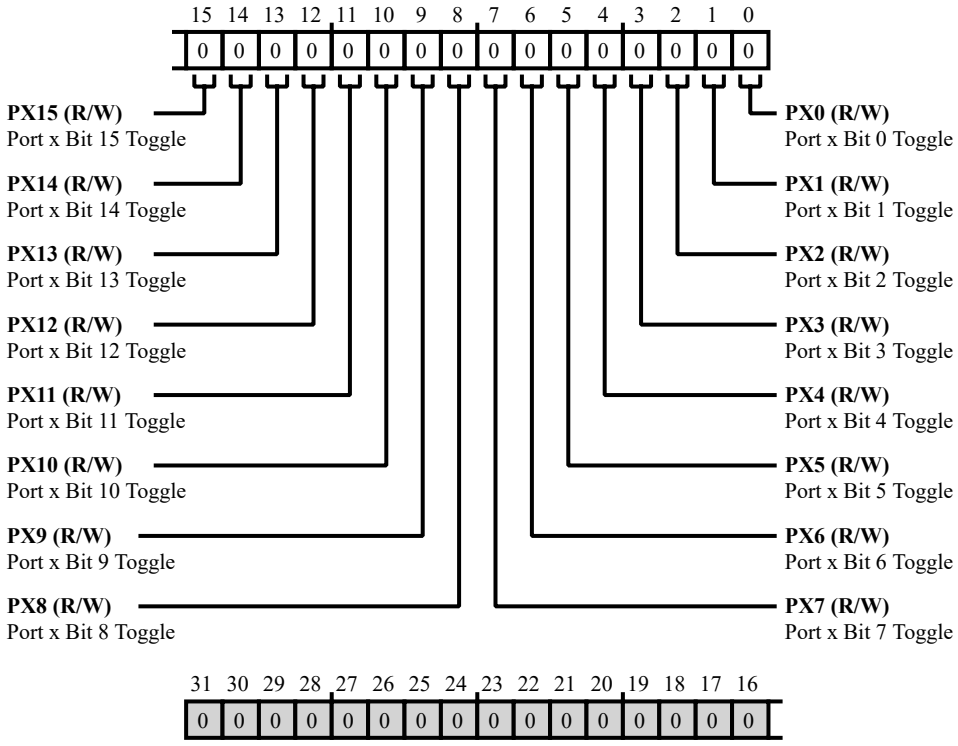


Figure 13-10: PORT_DATA_TGL Register Diagram

Table 13-13: PORT_DATA_TGL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Toggle. The PORT_DATA_TGL.PX15 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit.
14 (R/W)	PX14	Port x Bit 14 Toggle. The PORT_DATA_TGL.PX14 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit

Table 13-13: PORT_DATA_TGL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	PX13	Port x Bit 13 Toggle. The PORT_DATA_TGL.PX13 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
12 (R/W)	PX12	Port x Bit 12 Toggle. The PORT_DATA_TGL.PX12 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
11 (R/W)	PX11	Port x Bit 11 Toggle. The PORT_DATA_TGL.PX11 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
10 (R/W)	PX10	Port x Bit 10 Toggle. The PORT_DATA_TGL.PX10 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
9 (R/W)	PX9	Port x Bit 9 Toggle. The PORT_DATA_TGL.PX9 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
8 (R/W)	PX8	Port x Bit 8 Toggle. The PORT_DATA_TGL.PX8 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
7 (R/W)	PX7	Port x Bit 7 Toggle. The PORT_DATA_TGL.PX7 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit

Table 13-13: PORT_DATA_TGL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	PX6	Port x Bit 6 Toggle. The PORT_DATA_TGL.PX6 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
5 (R/W)	PX5	Port x Bit 5 Toggle. The PORT_DATA_TGL.PX5 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
4 (R/W)	PX4	Port x Bit 4 Toggle. The PORT_DATA_TGL.PX4 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
3 (R/W)	PX3	Port x Bit 3 Toggle. The PORT_DATA_TGL.PX3 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
2 (R/W)	PX2	Port x Bit 2 Toggle. The PORT_DATA_TGL.PX2 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
1 (R/W)	PX1	Port x Bit 1 Toggle. The PORT_DATA_TGL.PX1 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
0 (R/W)	PX0	Port x Bit 0 Toggle. The PORT_DATA_TGL.PX0 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit

Port x GPIO Direction Register

The PORT_DIR, PORT_DIR_SET, and PORT_DIR_CLR registers select output or input mode for GPIO pins and enable output drivers. Use the PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers to enable or disable input drivers.

Writes to the PORT_DIR register affect the state of all pins of the port. To select a direction for specific pins without impacting other pins of the port, use the PORT_DIR_SET and PORT_DIR_CLR registers.

Setting a bit in the PORT_DIR register enables output mode on the corresponding a GPIO pin. Clearing a bit in the PORT_DIR register disables output mode on the corresponding GPIO pin.

Input Mode - The default mode of every GPIO pin after reset is the input mode, but the input drivers are not enabled. To enable GPIO input drivers, set the corresponding bits in the PORT_INEN register. When enabled, a read from the PORT_DATA register returns the logical state of the input pin. The input signal does not overwrite the state of the bit used for the output case. That state can only be altered by software. If the input driver is enabled, a write to the PORT_DATA register can alter the state of the bit, but the change cannot be read back.

Output Mode - Any GPIO pin can be configured for output mode. The GPIO output drivers are enabled by setting the corresponding bits in the PORT_DIR, PORT_DIR_SET, or PORT_DIR_CLR registers. By using the PORT_DIR_SET and PORT_DIR_CLR registers, the direction of the signal flow of individual GPIO pins can be altered by separate software threads without mutually impacting other GPIOs on the same port. Both registers return the same value when read. Because the state of the GPIO output can already be controlled before the output driver is enabled, it is recommended to first set or clear the bit (using the PORT_DATA, PORT_DATA_SET, or PORT_DATA_CLR registers) to avoid any volatile levels on the output.

Open-Drain Mode - Every GPIO can also be used in open-drain mode. To accomplish this, first, clear the respective bit in the PORT_DATA or PORT_DATA_CLR register. Then, set the one bit in the PORT_INEN register. Reads from the PORT_DATA register then return the status from the pin and do not return the state of the internal flip-flop. By toggling the output driver through the PORT_DIR_SET and PORT_DIR_CLR register pair, the output signal can be pulled low or three-stated as required. Note that the polarity of the driven signal can be inverted when the internal flip-flop is set instead. When a GPIO port is used in open-drain mode, take care to not exceed the V_{IH} operating condition associated with the respective pin.

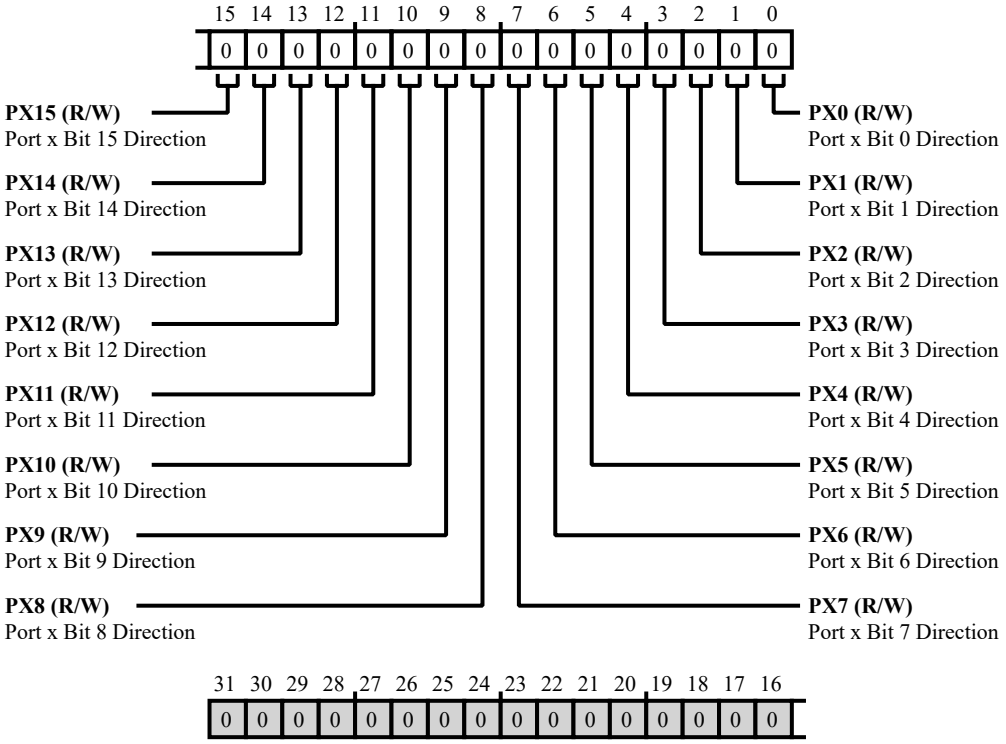


Figure 13-11: PORT_DIR Register Diagram

Table 13-14: PORT_DIR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
14 (R/W)	PX14	Port x Bit 14 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
13 (R/W)	PX13	Port x Bit 13 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
12 (R/W)	PX12	Port x Bit 12 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.

Table 13-14: PORT_DIR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	PX11	Port x Bit 11 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
10 (R/W)	PX10	Port x Bit 10 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
9 (R/W)	PX9	Port x Bit 9 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
8 (R/W)	PX8	Port x Bit 8 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
7 (R/W)	PX7	Port x Bit 7 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
6 (R/W)	PX6	Port x Bit 6 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
5 (R/W)	PX5	Port x Bit 5 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
4 (R/W)	PX4	Port x Bit 4 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
3 (R/W)	PX3	Port x Bit 3 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
2 (R/W)	PX2	Port x Bit 2 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.

Table 13-14: PORT_DIR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PX1	Port x Bit 1 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
0 (R/W)	PX0	Port x Bit 0 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.

Port x GPIO Direction Clear Register

The PORT_DIR_CLR register disables output mode and disables the output drivers for GPIO pins. For more information, see the PORT_DIR register description.

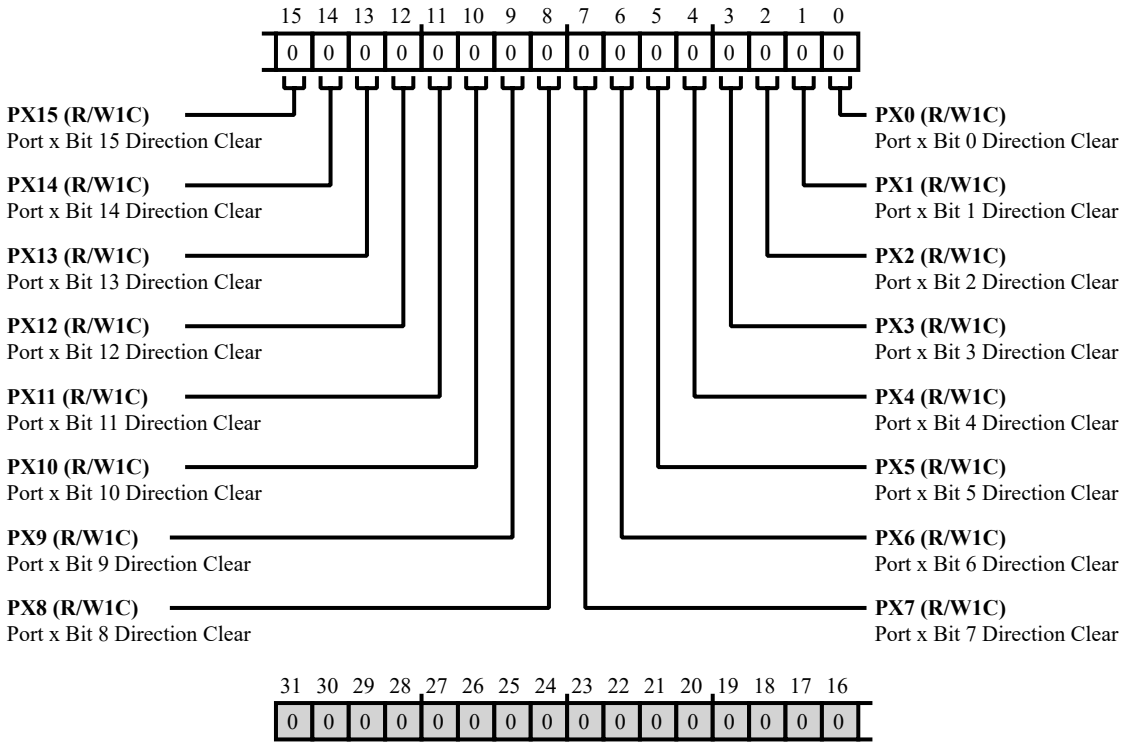


Figure 13-12: PORT_DIR_CLR Register Diagram

Table 13-15: PORT_DIR_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Direction Clear. The PORT_DIR_CLR.PX15 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
14 (R/W1C)	PX14	Port x Bit 14 Direction Clear. The PORT_DIR_CLR.PX14 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Table 13-15: PORT_DIR_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1C)	PX13	Port x Bit 13 Direction Clear. The PORT_DIR_CLR.PX13 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
12 (R/W1C)	PX12	Port x Bit 12 Direction Clear. The PORT_DIR_CLR.PX12 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
11 (R/W1C)	PX11	Port x Bit 11 Direction Clear. The PORT_DIR_CLR.PX11 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
10 (R/W1C)	PX10	Port x Bit 10 Direction Clear. The PORT_DIR_CLR.PX10 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
9 (R/W1C)	PX9	Port x Bit 9 Direction Clear. The PORT_DIR_CLR.PX9 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
8 (R/W1C)	PX8	Port x Bit 8 Direction Clear. The PORT_DIR_CLR.PX8 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Table 13-15: PORT_DIR_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	PX7	Port x Bit 7 Direction Clear. The PORT_DIR_CLR.PX7 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
6 (R/W1C)	PX6	Port x Bit 6 Direction Clear. The PORT_DIR_CLR.PX6 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
5 (R/W1C)	PX5	Port x Bit 5 Direction Clear. The PORT_DIR_CLR.PX5 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
4 (R/W1C)	PX4	Port x Bit 4 Direction Clear. The PORT_DIR_CLR.PX4 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
3 (R/W1C)	PX3	Port x Bit 3 Direction Clear. The PORT_DIR_CLR.PX3 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
2 (R/W1C)	PX2	Port x Bit 2 Direction Clear. The PORT_DIR_CLR.PX2 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Table 13-15: PORT_DIR_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	PX1	Port x Bit 1 Direction Clear. The `PORT_DIR_CLR.PX1` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
0 (R/W1C)	PX0	Port x Bit 0 Direction Clear. The `PORT_DIR_CLR.PX0` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Port x GPIO Direction Set Register

The PORT_DIR_SET register enables output mode and output drivers for GPIO pins. For more information, see the PORT_DIR register description.

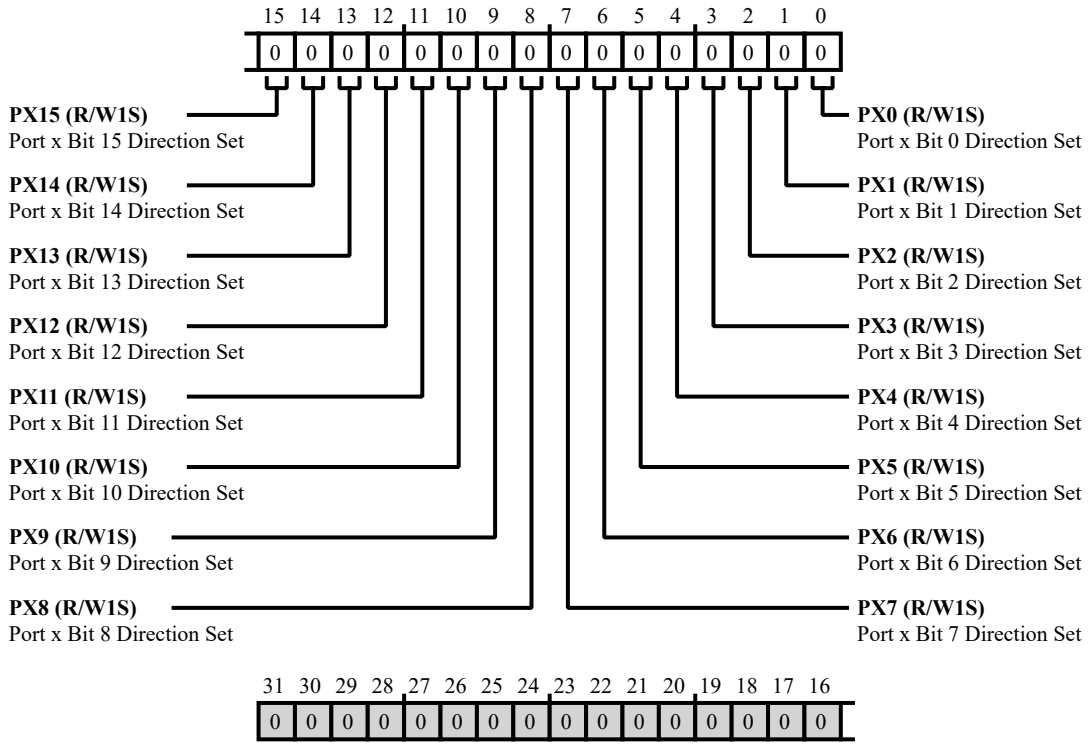


Figure 13-13: PORT_DIR_SET Register Diagram

Table 13-16: PORT_DIR_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Direction Set. The PORT_DIR_SET.PX15 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
14 (R/W1S)	PX14	Port x Bit 14 Direction Set. The PORT_DIR_SET.PX14 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver

Table 13-16: PORT_DIR_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1S)	PX13	Port x Bit 13 Direction Set. The PORT_DIR_SET.PX13 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
12 (R/W1S)	PX12	Port x Bit 12 Direction Set. The PORT_DIR_SET.PX12 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
11 (R/W1S)	PX11	Port x Bit 11 Direction Set. The PORT_DIR_SET.PX11 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
10 (R/W1S)	PX10	Port x Bit 10 Direction Set. The PORT_DIR_SET.PX10 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
9 (R/W1S)	PX9	Port x Bit 9 Direction Set. The PORT_DIR_SET.PX9 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
8 (R/W1S)	PX8	Port x Bit 8 Direction Set. The PORT_DIR_SET.PX8 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
7 (R/W1S)	PX7	Port x Bit 7 Direction Set. The PORT_DIR_SET.PX7 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver

Table 13-16: PORT_DIR_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	PX6	Port x Bit 6 Direction Set. The PORT_DIR_SET.PX6 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
5 (R/W1S)	PX5	Port x Bit 5 Direction Set. The PORT_DIR_SET.PX5 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
4 (R/W1S)	PX4	Port x Bit 4 Direction Set. The PORT_DIR_SET.PX4 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
3 (R/W1S)	PX3	Port x Bit 3 Direction Set. The PORT_DIR_SET.PX3 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
2 (R/W1S)	PX2	Port x Bit 2 Direction Set. The PORT_DIR_SET.PX2 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
1 (R/W1S)	PX1	Port x Bit 1 Direction Set. The PORT_DIR_SET.PX1 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
0 (R/W1S)	PX0	Port x Bit 0 Direction Set. The PORT_DIR_SET.PX0 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver

Port x Function Enable Register

The PORT_FER register bits indicate each port bit's operating mode: general purpose I/O mode or peripheral mode. After reset, all pins default to GPIO mode. Setting a bit in the PORT_FER registers enables a peripheral module to take ownership of the pin. The function enable bits impact output control only. Regardless of the setting of the function enable bits, both GPIO and peripherals can still sense the pin input. After a function is enabled, it is up to the PORT_MUX registers as to which peripheral takes control.

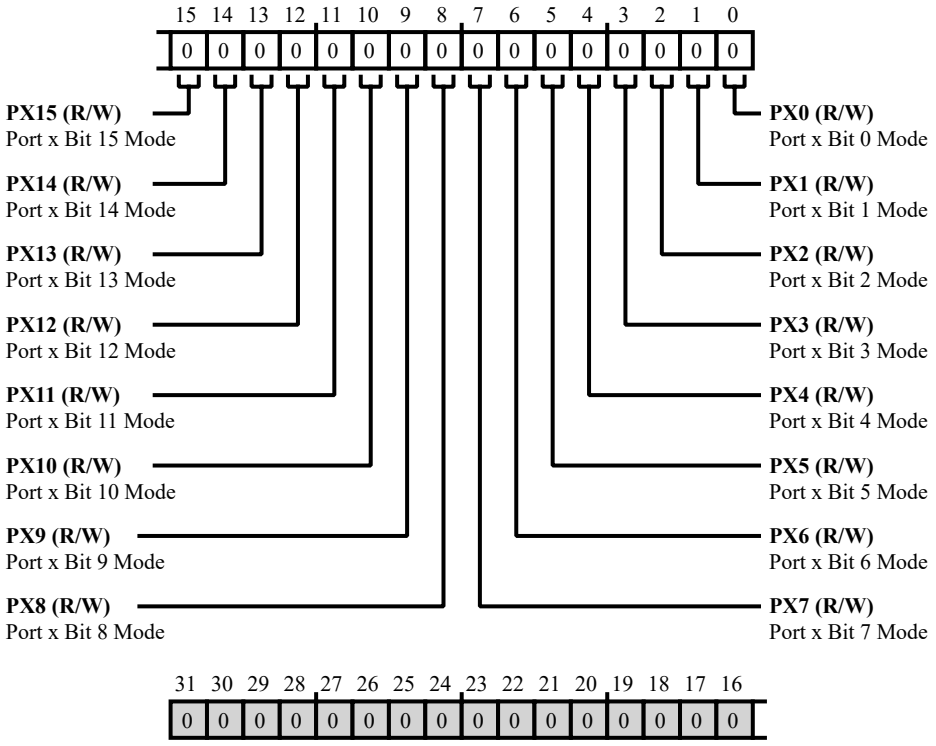


Figure 13-14: PORT_FER Register Diagram

Table 13-17: PORT_FER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Mode. The PORT_FER.PX15 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
14 (R/W)	PX14	Port x Bit 14 Mode. The PORT_FER.PX14 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode

Table 13-17: PORT_FER Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	PX13	Port x Bit 13 Mode. The PORT_FER.PX13 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
12 (R/W)	PX12	Port x Bit 12 Mode. The PORT_FER.PX12 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
11 (R/W)	PX11	Port x Bit 11 Mode. The PORT_FER.PX11 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
10 (R/W)	PX10	Port x Bit 10 Mode. The PORT_FER.PX10 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
9 (R/W)	PX9	Port x Bit 9 Mode. The PORT_FER.PX9 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
8 (R/W)	PX8	Port x Bit 8 Mode. The PORT_FER.PX8 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
7 (R/W)	PX7	Port x Bit 7 Mode. The PORT_FER.PX7 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode

Table 13-17: PORT_FER Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	PX6	Port x Bit 6 Mode. The PORT_FER.PX6 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
5 (R/W)	PX5	Port x Bit 5 Mode. The PORT_FER.PX5 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
4 (R/W)	PX4	Port x Bit 4 Mode. The PORT_FER.PX4 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
3 (R/W)	PX3	Port x Bit 3 Mode. The PORT_FER.PX3 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
2 (R/W)	PX2	Port x Bit 2 Mode. The PORT_FER.PX2 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
1 (R/W)	PX1	Port x Bit 1 Mode. The PORT_FER.PX1 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
0 (R/W)	PX0	Port x Bit 0 Mode. The PORT_FER.PX0 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode

Port x Function Enable Clear Register

The PORT_FER_CLR register permits enabling GPIO mode for each bit and corresponding GPIO pin. Writing 1 to a bit in PORT_FER_CLR enables GPIO mode for the corresponding pin.

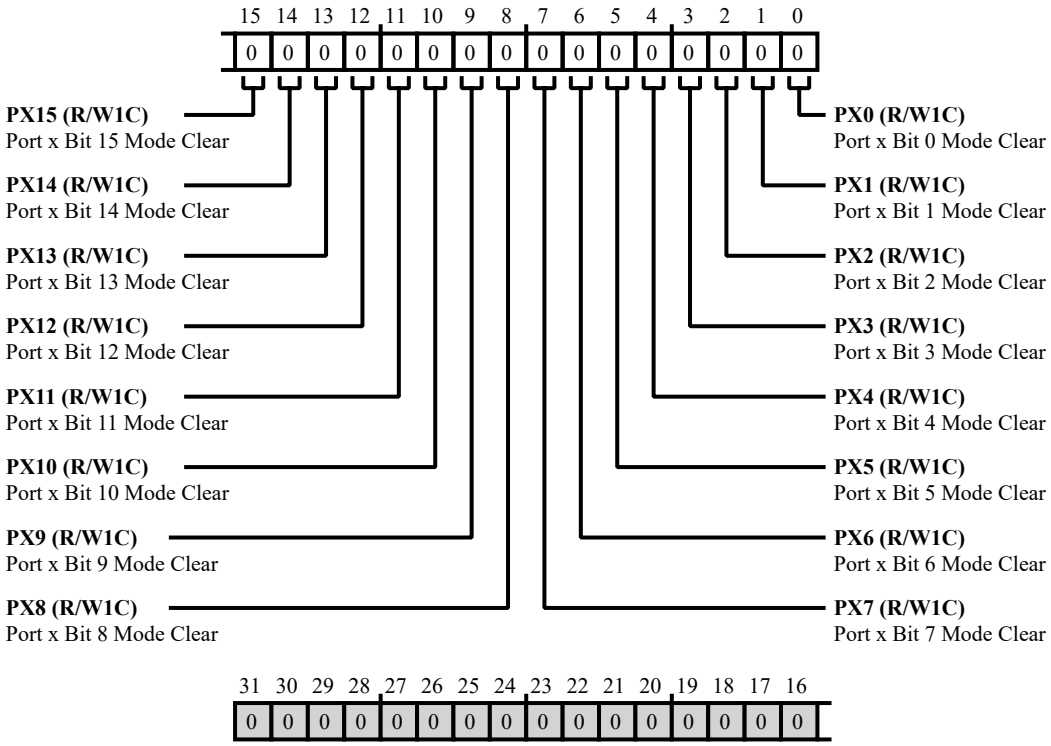


Figure 13-15: PORT_FER_CLR Register Diagram

Table 13-18: PORT_FER_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Mode Clear. The PORT_FER_CLR.PX15 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
14 (R/W1C)	PX14	Port x Bit 14 Mode Clear. The PORT_FER_CLR.PX14 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode

Table 13-18: PORT_FER_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1C)	PX13	Port x Bit 13 Mode Clear. The PORT_FER_CLR.PX13 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
12 (R/W1C)	PX12	Port x Bit 12 Mode Clear. The PORT_FER_CLR.PX12 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
11 (R/W1C)	PX11	Port x Bit 11 Mode Clear. The PORT_FER_CLR.PX11 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
10 (R/W1C)	PX10	Port x Bit 10 Mode Clear. The PORT_FER_CLR.PX10 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
9 (R/W1C)	PX9	Port x Bit 9 Mode Clear. The PORT_FER_CLR.PX9 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
8 (R/W1C)	PX8	Port x Bit 8 Mode Clear. The PORT_FER_CLR.PX8 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
7 (R/W1C)	PX7	Port x Bit 7 Mode Clear. The PORT_FER_CLR.PX7 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode

Table 13-18: PORT_FER_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	PX6	Port x Bit 6 Mode Clear. The PORT_FER_CLR.PX6 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
5 (R/W1C)	PX5	Port x Bit 5 Mode Clear. The PORT_FER_CLR.PX5 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
4 (R/W1C)	PX4	Port x Bit 4 Mode Clear. The PORT_FER_CLR.PX4 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
3 (R/W1C)	PX3	Port x Bit 3 Mode Clear. The PORT_FER_CLR.PX3 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
2 (R/W1C)	PX2	Port x Bit 2 Mode Clear. The PORT_FER_CLR.PX2 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
1 (R/W1C)	PX1	Port x Bit 1 Mode Clear. The PORT_FER_CLR.PX1 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
0 (R/W1C)	PX0	Port x Bit 0 Mode Clear. The PORT_FER_CLR.PX0 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode

Port x Function Enable Set Register

The PORT_FER_SET register permits enabling peripheral mode for each bit and corresponding GPIO pin. Writing 1 to a bit in PORT_FER_SET enables peripheral mode for the corresponding pin.

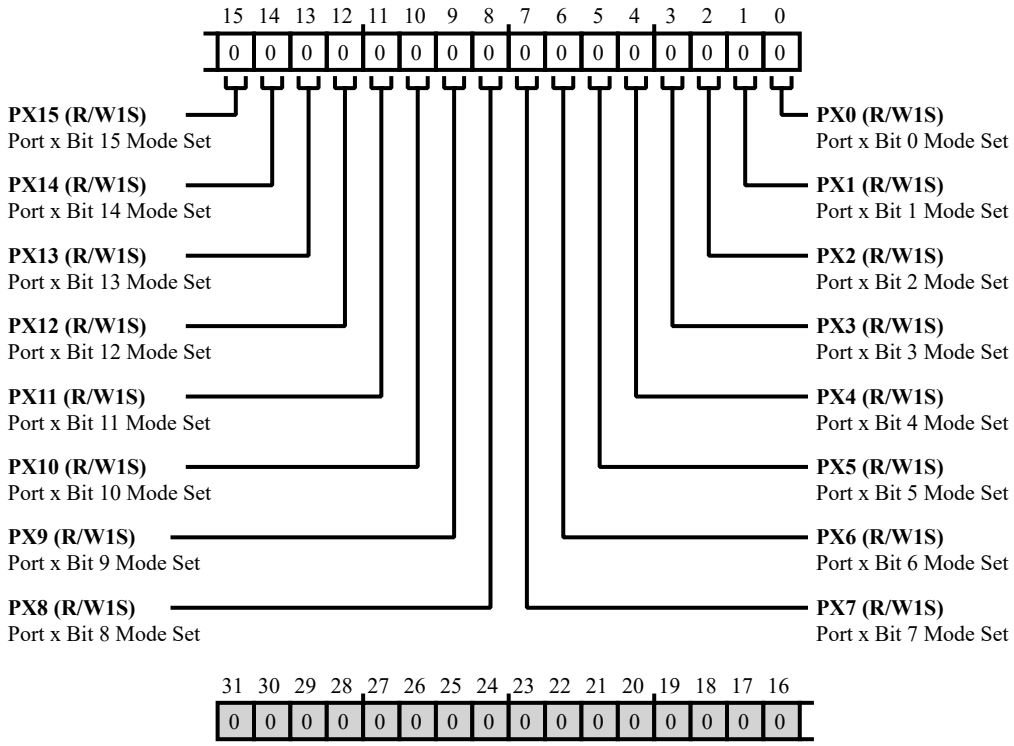


Figure 13-16: PORT_FER_SET Register Diagram

Table 13-19: PORT_FER_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Mode Set. The PORT_FER_SET.PX15 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
14 (R/W1S)	PX14	Port x Bit 14 Mode Set. The PORT_FER_SET.PX14 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode

Table 13-19: PORT_FER_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1S)	PX13	Port x Bit 13 Mode Set. The PORT_FER_SET.PX13 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
12 (R/W1S)	PX12	Port x Bit 12 Mode Set. The PORT_FER_SET.PX12 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
11 (R/W1S)	PX11	Port x Bit 11 Mode Set. The PORT_FER_SET.PX11 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
10 (R/W1S)	PX10	Port x Bit 10 Mode Set. The PORT_FER_SET.PX10 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
9 (R/W1S)	PX9	Port x Bit 9 Mode Set. The PORT_FER_SET.PX9 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
8 (R/W1S)	PX8	Port x Bit 8 Mode Set. The PORT_FER_SET.PX8 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
7 (R/W1S)	PX7	Port x Bit 7 Mode Set. The PORT_FER_SET.PX7 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode

Table 13-19: PORT_FER_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	PX6	Port x Bit 6 Mode Set. The PORT_FER_SET.PX6 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
5 (R/W1S)	PX5	Port x Bit 5 Mode Set. The PORT_FER_SET.PX5 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
4 (R/W1S)	PX4	Port x Bit 4 Mode Set. The PORT_FER_SET.PX4 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
3 (R/W1S)	PX3	Port x Bit 3 Mode Set. The PORT_FER_SET.PX3 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
2 (R/W1S)	PX2	Port x Bit 2 Mode Set. The PORT_FER_SET.PX2 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
1 (R/W1S)	PX1	Port x Bit 1 Mode Set. The PORT_FER_SET.PX1 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
0 (R/W1S)	PX0	Port x Bit 0 Mode Set. The PORT_FER_SET.PX0 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode

Port x GPIO Input Enable Register

The PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers enable or disable input drivers, which are required for using a GPIO pin in input mode.

Writes to the PORT_INEN register affect the input drivers for all pins of the port. To set or clear specific pin drivers without impacting other pin drivers of the port, use the PORT_INEN_SET and PORT_INEN_CLR registers.

If the input is enabled, reads from the PORT_DATA, PORT_DATA_SET, or PORT_DATA_CLR registers return the state of the pins. However, the state of the output is not overwritten by the input. It is altered by software writes only. Input and output drivers can be enabled at the same time. In this case, a read of the data register returns the true value of the data register and not the pin state.

For more information, see the PORT_DATA register description and the PORT_DIR register description.

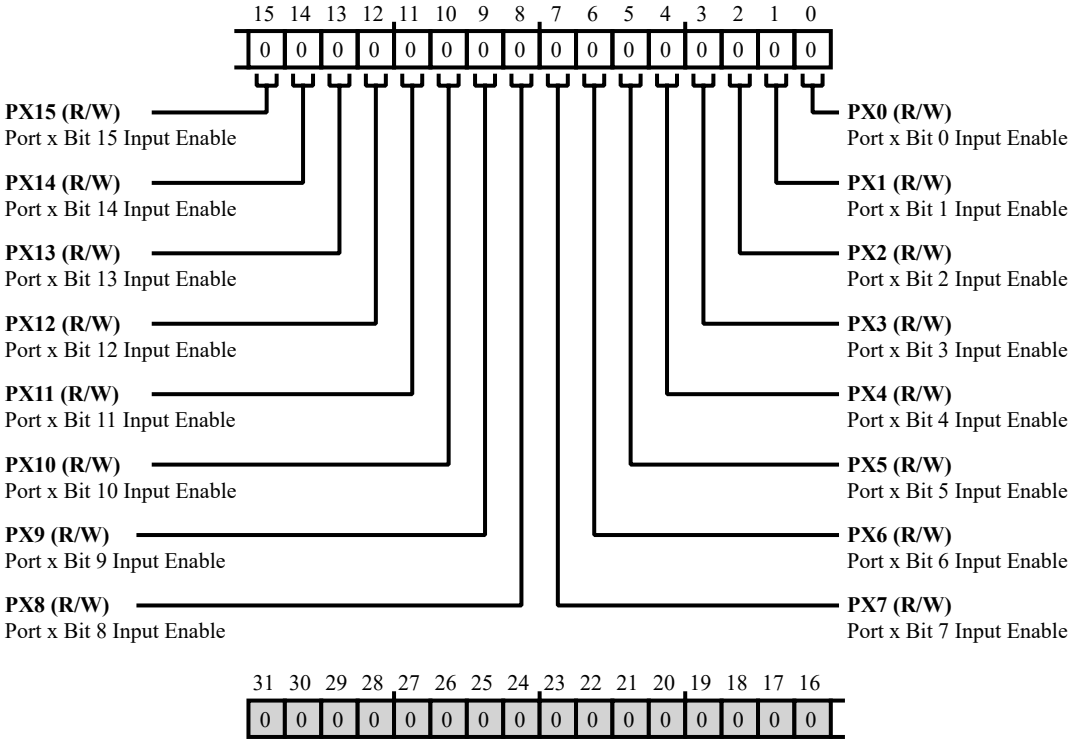


Figure 13-17: PORT_INEN Register Diagram

Table 13-20: PORT_INEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver

Table 13-20: PORT_INEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	PX14	Port x Bit 14 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
13 (R/W)	PX13	Port x Bit 13 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
12 (R/W)	PX12	Port x Bit 12 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
11 (R/W)	PX11	Port x Bit 11 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
10 (R/W)	PX10	Port x Bit 10 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
9 (R/W)	PX9	Port x Bit 9 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
8 (R/W)	PX8	Port x Bit 8 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
7 (R/W)	PX7	Port x Bit 7 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
6 (R/W)	PX6	Port x Bit 6 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
5 (R/W)	PX5	Port x Bit 5 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver

Table 13-20: PORT_INEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	PX4	Port x Bit 4 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
3 (R/W)	PX3	Port x Bit 3 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
2 (R/W)	PX2	Port x Bit 2 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
1 (R/W)	PX1	Port x Bit 1 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
0 (R/W)	PX0	Port x Bit 0 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver

Port x GPIO Input Enable Clear Register

The PORT_INEN_CLR register disables the input drivers for GPIO pins. For more information, see the PORT_INEN register description.

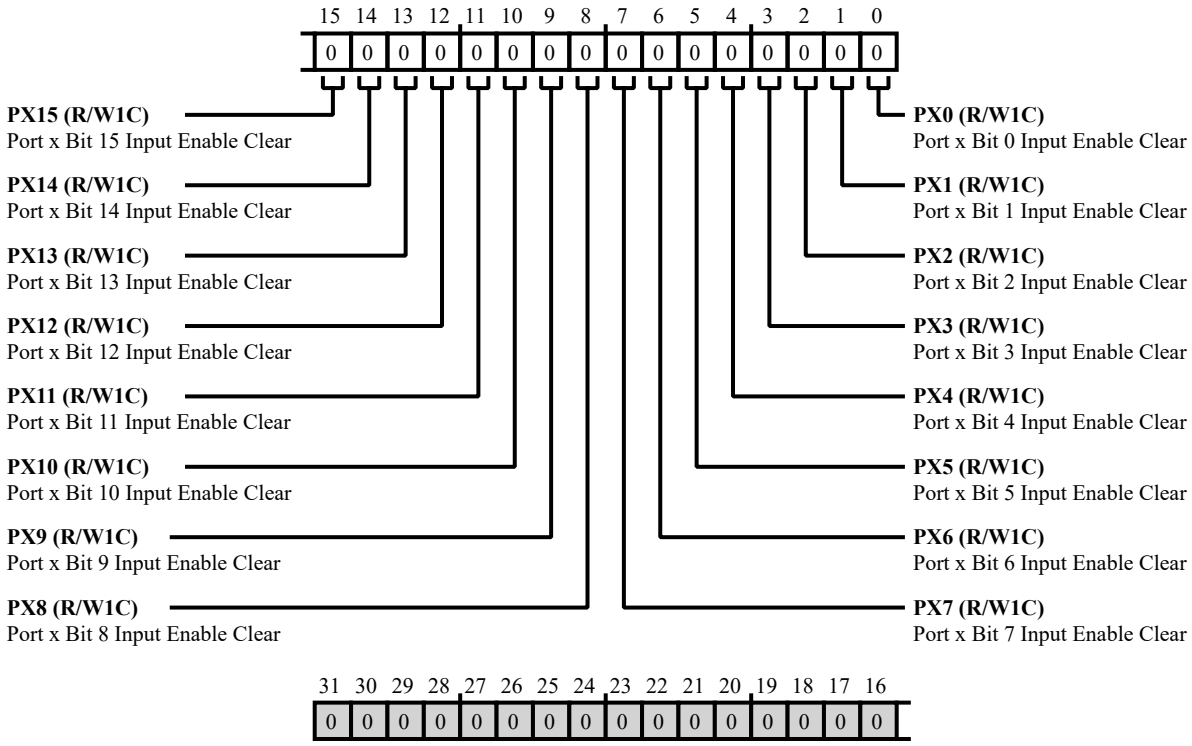


Figure 13-18: PORT_INEN_CLR Register Diagram

Table 13-21: PORT_INEN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
14 (R/W1C)	PX14	Port x Bit 14 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
13 (R/W1C)	PX13	Port x Bit 13 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.

Table 13-21: PORT_INEN_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1C)	PX12	Port x Bit 12 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
11 (R/W1C)	PX11	Port x Bit 11 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
10 (R/W1C)	PX10	Port x Bit 10 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
9 (R/W1C)	PX9	Port x Bit 9 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
8 (R/W1C)	PX8	Port x Bit 8 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
7 (R/W1C)	PX7	Port x Bit 7 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
6 (R/W1C)	PX6	Port x Bit 6 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
5 (R/W1C)	PX5	Port x Bit 5 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
4 (R/W1C)	PX4	Port x Bit 4 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
3 (R/W1C)	PX3	Port x Bit 3 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.

Table 13-21: PORT_INEN_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	PX2	Port x Bit 2 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
1 (R/W1C)	PX1	Port x Bit 1 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
0 (R/W1C)	PX0	Port x Bit 0 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.

Port x GPIO Input Enable Set Register

The PORT_INEN_SET register enables input drivers for GPIO pins. For more information, see the PORT_INEN register description.

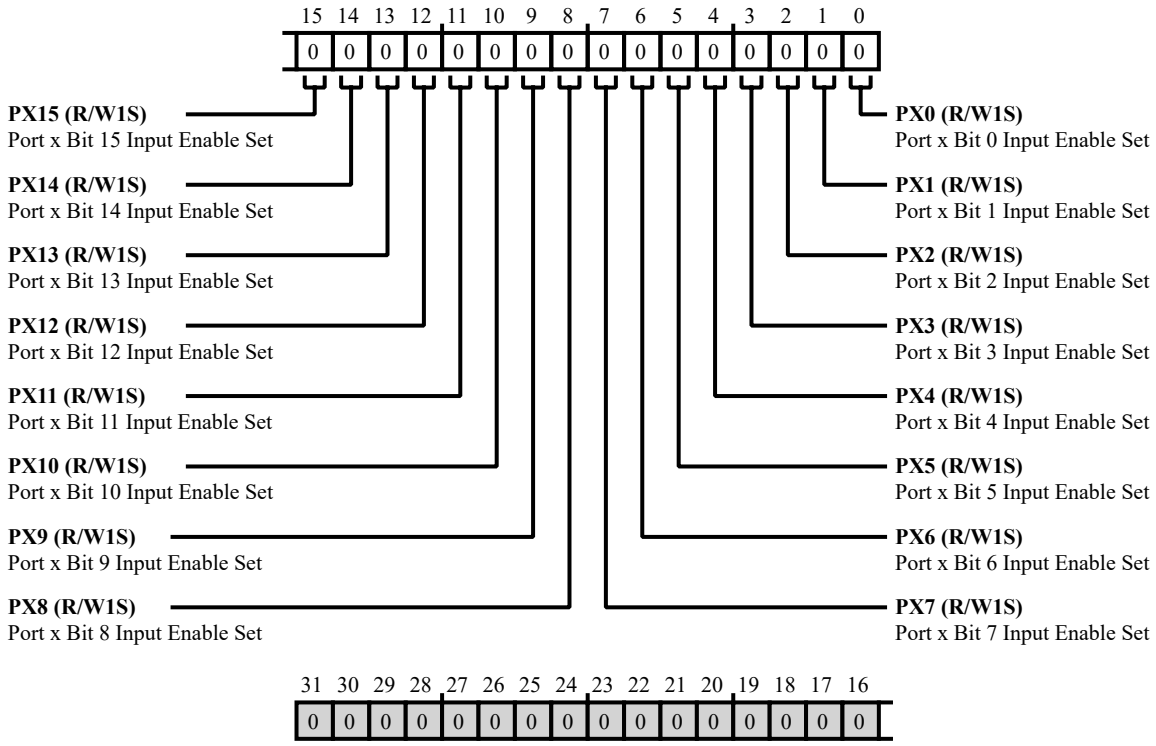


Figure 13-19: PORT_INEN_SET Register Diagram

Table 13-22: PORT_INEN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
14 (R/W1S)	PX14	Port x Bit 14 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
13 (R/W1S)	PX13	Port x Bit 13 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.

Table 13-22: PORT_INEN_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1S)	PX12	Port x Bit 12 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
11 (R/W1S)	PX11	Port x Bit 11 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
10 (R/W1S)	PX10	Port x Bit 10 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
9 (R/W1S)	PX9	Port x Bit 9 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
8 (R/W1S)	PX8	Port x Bit 8 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
7 (R/W1S)	PX7	Port x Bit 7 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
6 (R/W1S)	PX6	Port x Bit 6 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
5 (R/W1S)	PX5	Port x Bit 5 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
4 (R/W1S)	PX4	Port x Bit 4 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
3 (R/W1S)	PX3	Port x Bit 3 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.

Table 13-22: PORT_INEN_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1S)	PX2	Port x Bit 2 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
1 (R/W1S)	PX1	Port x Bit 1 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
0 (R/W1S)	PX0	Port x Bit 0 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.

Port x GPIO Lock Register

The PORT_LOCK register enables (unlocks) or disables (locks) write access selectively for the PORT control registers.

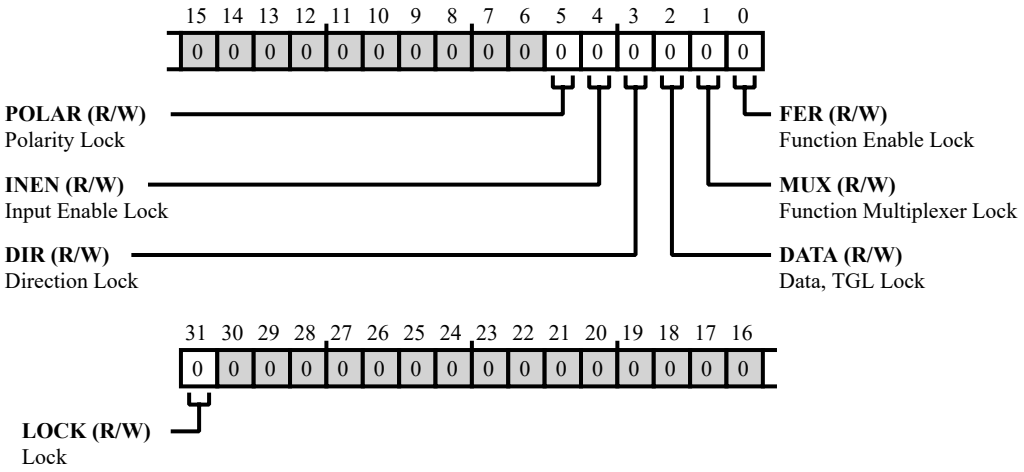


Figure 13-20: PORT_LOCK Register Diagram

Table 13-23: PORT_LOCK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the PORT_LOCK.LOCK bit is set, the PORT_LOCK register is read only (locked).	
		0	Unlock
		1	Lock
5 (R/W)	POLAR	Polarity Lock. The PORT_LOCK.POLAR disables write access to the PORT_POL, PORT_POL_SET, and PORT_POL_CLR registers.	
		0	Unlock POL
		1	Lock POL
4 (R/W)	INEN	Input Enable Lock. The PORT_LOCK.INEN disables write access to the PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers.	
		0	Unlock INEN
		1	Lock INEN

Table 13-23: PORT_LOCK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DIR	Direction Lock. The PORT_LOCK.DIR disables write access to the PORT_DIR, PORT_DIR_SET, PORT_DIR_CLR registers.	
		0	Unlock DIR
		1	Lock DIR
2 (R/W)	DATA	Data, TGL Lock. The PORT_LOCK.DATA disables write access to the PORT_DATA, PORT_DATA_SET, PORT_DATA_CLR, and PORT_DATA_TGL registers.	
		0	Unlock DATA
		1	Lock DATA
1 (R/W)	MUX	Function Multiplexer Lock. The PORT_LOCK.MUX disables write accesses to the PORT_MUX register.	
		0	Unlock MUX
		1	Lock MUX
0 (R/W)	FER	Function Enable Lock. The PORT_LOCK.FER disables write access to the PORT_FER, PORT_FER_SET, and PORT_FER_CLR registers.	
		0	Unlock FER
		1	Lock FER

Port x Multiplexer Control Register

When a pin is in peripheral mode (not GPIO mode), the PORT_MUX register controls which peripheral takes ownership of a pin. Ports may have multiple, different peripheral functions. Two bits are required to describe every multiplexer on an individual pin-by-pin scheme. For example, bit 0 and bit 1 of the PORT_MUX register control the multiplexer of pin 0, bit 2 and bit 3 of PORT_MUX control the multiplexer of pin 1, and so on. The value of any PORT_MUX bit has no effect on the port pins when the associated bit in the PORT_FER register is 0 (selects GPIO mode). Even if a port has only one function, the PORT_MUX register is still present. For single function ports (no multiplexing is needed), leave the PORT_MUX bits at 0 (default). For all PORT_MUX bit fields: 00 = default/reset peripheral option, 01 = first alternate peripheral option, 10 = second alternate peripheral option, and 11 = third alternate peripheral option.

See the processor data sheet for details regarding the peripheral options associated with each port.

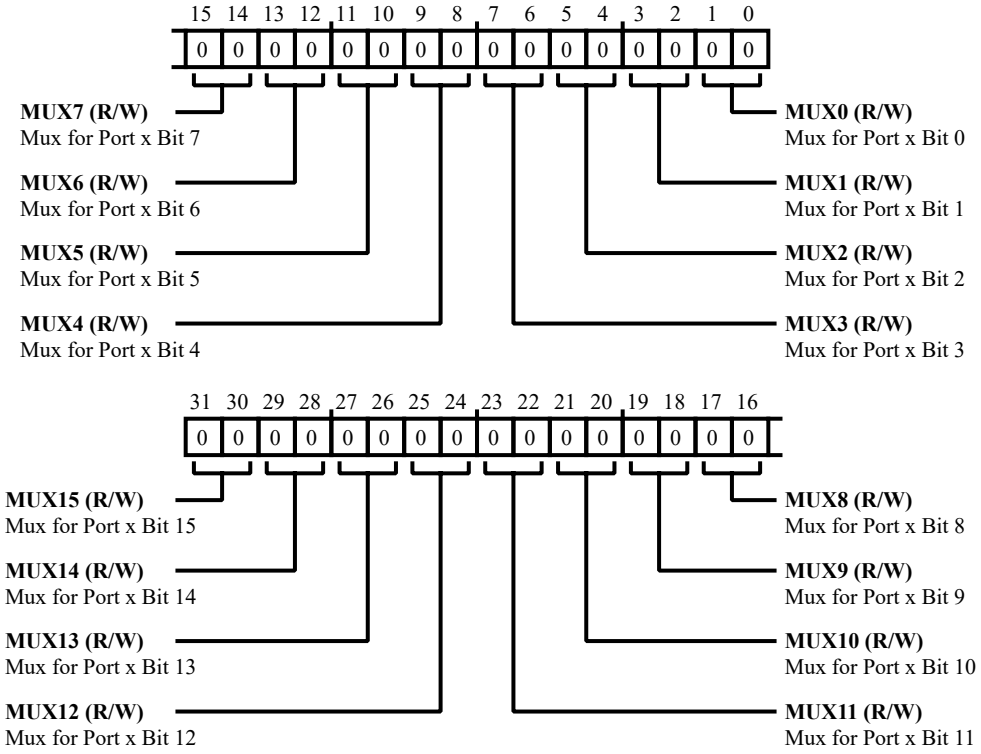


Figure 13-21: PORT_MUX Register Diagram

Table 13-24: PORT_MUX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	MUX15	Mux for Port x Bit 15. The PORT_MUX.MUX15 bit provides multiplexer control for port x bit 15.

Table 13-24: PORT_MUX Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:28 (R/W)	MUX14	Mux for Port x Bit 14. The PORT_MUX.MUX14 bit provides multiplexer control for port x bit 14.
27:26 (R/W)	MUX13	Mux for Port x Bit 13. The PORT_MUX.MUX13 bit provides multiplexer control for port x bit 13.
25:24 (R/W)	MUX12	Mux for Port x Bit 12. The PORT_MUX.MUX12 bit provides multiplexer control for port x bit 12.
23:22 (R/W)	MUX11	Mux for Port x Bit 11. The PORT_MUX.MUX11 bit provides multiplexer control for port x bit 11.
21:20 (R/W)	MUX10	Mux for Port x Bit 10. The PORT_MUX.MUX10 bit provides multiplexer control for port x bit 10.
19:18 (R/W)	MUX9	Mux for Port x Bit 9. The PORT_MUX.MUX9 bit provides multiplexer control for port x bit 9.
17:16 (R/W)	MUX8	Mux for Port x Bit 8. The PORT_MUX.MUX8 bit provides multiplexer control for port x bit 8.
15:14 (R/W)	MUX7	Mux for Port x Bit 7. The PORT_MUX.MUX7 bit provides multiplexer control for port x bit 7.
13:12 (R/W)	MUX6	Mux for Port x Bit 6. The PORT_MUX.MUX6 bit provides multiplexer control for port x bit 6.
11:10 (R/W)	MUX5	Mux for Port x Bit 5. The PORT_MUX.MUX5 bit provides multiplexer control for port x bit 5.
9:8 (R/W)	MUX4	Mux for Port x Bit 4. The PORT_MUX.MUX4 bit provides multiplexer control for port x bit 4.
7:6 (R/W)	MUX3	Mux for Port x Bit 3. The PORT_MUX.MUX3 bit provides multiplexer control for port x bit 3.
5:4 (R/W)	MUX2	Mux for Port x Bit 2. The PORT_MUX.MUX2 bit provides multiplexer control for port x bit 2.
3:2 (R/W)	MUX1	Mux for Port x Bit 1. The PORT_MUX.MUX1 bit provides multiplexer control for port x bit 1.
1:0 (R/W)	MUX0	Mux for Port x Bit 0. The PORT_MUX.MUX0 bit provides multiplexer control for port x bit 0.

Port x GPIO Polarity Invert Register

The PORT_POL, PORT_POL_SET, and PORT_POL_CLR registers enable or disable inverting polarity of GPIO signals. To invert polarity of peripheral signals, use the inversion selection programming in the signal's corresponding module.

Writes to the PORT_POL register affect the polarity inversion selection of all pins of the port. To enable or disable polarity inversion for specific pins without impacting other pins of the port, use the PORT_POL_SET and PORT_POL_CLR registers.

Setting a bit in the PORT_POL register enables polarity inversion on the corresponding inversion GPIO pin, making the pin active-low or falling-edge sensitive. Clearing a bit in the PORT_POL register disables polarity (default state) on the corresponding GPIO pin, making it active-high or rising-edge sensitive.

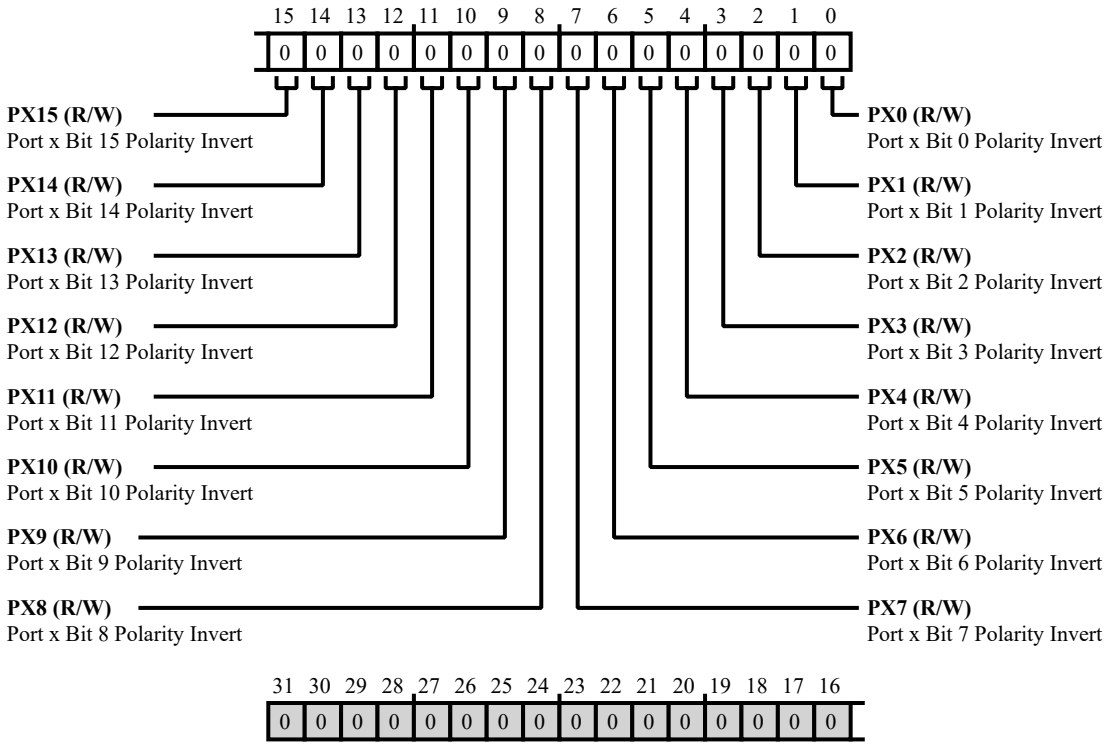


Figure 13-22: PORT_POL Register Diagram

Table 13-25: PORT_POL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Polarity Invert. The PORT_POL.PX15 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Table 13-25: PORT_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	PX14	Port x Bit 14 Polarity Invert. The PORT_POL.PX14 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
13 (R/W)	PX13	Port x Bit 13 Polarity Invert. The PORT_POL.PX13 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
12 (R/W)	PX12	Port x Bit 12 Polarity Invert. The PORT_POL.PX12 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
11 (R/W)	PX11	Port x Bit 11 Polarity Invert. The PORT_POL.PX11 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
10 (R/W)	PX10	Port x Bit 10 Polarity Invert. The PORT_POL.PX10 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
9 (R/W)	PX9	Port x Bit 9 Polarity Invert. The PORT_POL.PX9 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
8 (R/W)	PX8	Port x Bit 8 Polarity Invert. The PORT_POL.PX8 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Table 13-25: PORT_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	PX7	Port x Bit 7 Polarity Invert. The PORT_POL.PX7 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
6 (R/W)	PX6	Port x Bit 6 Polarity Invert. The PORT_POL.PX6 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
5 (R/W)	PX5	Port x Bit 5 Polarity Invert. The PORT_POL.PX5 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
4 (R/W)	PX4	Port x Bit 4 Polarity Invert. The PORT_POL.PX4 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
3 (R/W)	PX3	Port x Bit 3 Polarity Invert. The PORT_POL.PX3 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
2 (R/W)	PX2	Port x Bit 2 Polarity Invert. The PORT_POL.PX2 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
1 (R/W)	PX1	Port x Bit 1 Polarity Invert. The PORT_POL.PX1 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Table 13-25: PORT_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PX0	Port x Bit 0 Polarity Invert. The `PORT_POL.PX0` bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Port x GPIO Polarity Invert Clear Register

The PORT_POL_CLR register disables polarity inversion for GPIO pins. For more information, see the PORT_POL register description.

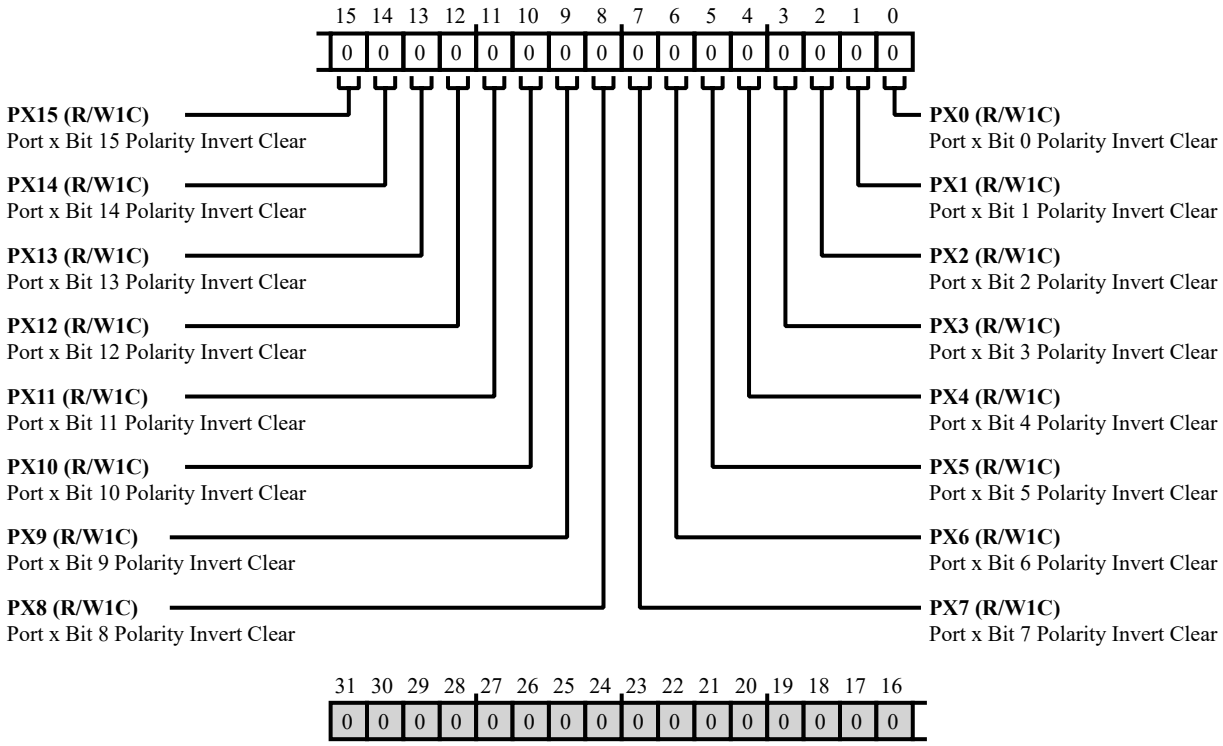


Figure 13-23: PORT_POL_CLR Register Diagram

Table 13-26: PORT_POL_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
14 (R/W1C)	PX14	Port x Bit 14 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
13 (R/W1C)	PX13	Port x Bit 13 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.

Table 13-26: PORT_POL_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1C)	PX12	Port x Bit 12 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
11 (R/W1C)	PX11	Port x Bit 11 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
10 (R/W1C)	PX10	Port x Bit 10 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
9 (R/W1C)	PX9	Port x Bit 9 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
8 (R/W1C)	PX8	Port x Bit 8 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
7 (R/W1C)	PX7	Port x Bit 7 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
6 (R/W1C)	PX6	Port x Bit 6 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
5 (R/W1C)	PX5	Port x Bit 5 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
4 (R/W1C)	PX4	Port x Bit 4 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
3 (R/W1C)	PX3	Port x Bit 3 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.

Table 13-26: PORT_POL_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	PX2	Port x Bit 2 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
1 (R/W1C)	PX1	Port x Bit 1 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
0 (R/W1C)	PX0	Port x Bit 0 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.

Port x GPIO Polarity Invert Set Register

The PORT_POL_SET register enables polarity inversion for GPIO pins. For more information, see the PORT_POL register description.

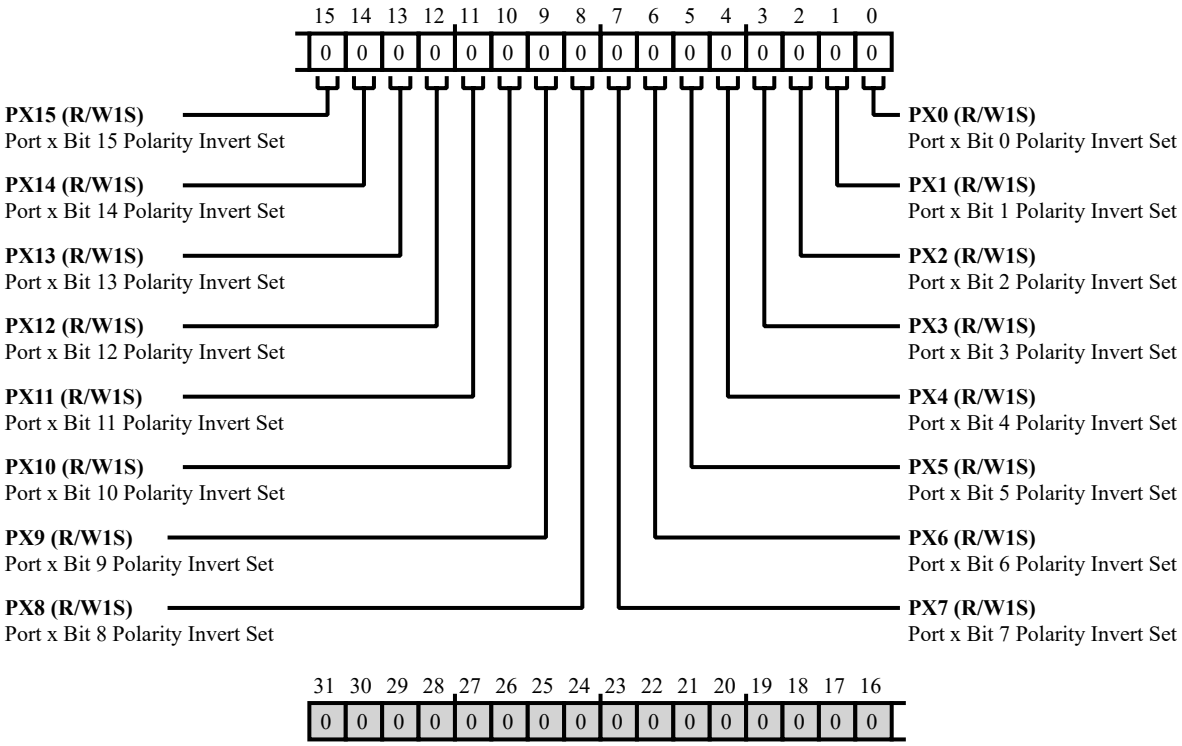


Figure 13-24: PORT_POL_SET Register Diagram

Table 13-27: PORT_POL_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Polarity Invert Set. The PORT_POL_SET.PX15 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
14 (R/W1S)	PX14	Port x Bit 14 Polarity Invert Set. The PORT_POL_SET.PX14 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.

Table 13-27: PORT_POL_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1S)	PX13	Port x Bit 13 Polarity Invert Set. The PORT_POL_SET.PX13 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
12 (R/W1S)	PX12	Port x Bit 12 Polarity Invert Set. The PORT_POL_SET.PX12 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
11 (R/W1S)	PX11	Port x Bit 11 Polarity Invert Set. The PORT_POL_SET.PX11 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
10 (R/W1S)	PX10	Port x Bit 10 Polarity Invert Set. The PORT_POL_SET.PX10 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
9 (R/W1S)	PX9	Port x Bit 9 Polarity Invert Set. The PORT_POL_SET.PX9 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
8 (R/W1S)	PX8	Port x Bit 8 Polarity Invert Set. The PORT_POL_SET.PX8 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
7 (R/W1S)	PX7	Port x Bit 7 Polarity Invert Set. The PORT_POL_SET.PX7 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.

Table 13-27: PORT_POL_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	PX6	Port x Bit 6 Polarity Invert Set. The PORT_POL_SET.PX6 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
5 (R/W1S)	PX5	Port x Bit 5 Polarity Invert Set. The PORT_POL_SET.PX5 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
4 (R/W1S)	PX4	Port x Bit 4 Polarity Invert Set. The PORT_POL_SET.PX4 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
3 (R/W1S)	PX3	Port x Bit 3 Polarity Invert Set. The PORT_POL_SET.PX3 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
2 (R/W1S)	PX2	Port x Bit 2 Polarity Invert Set. The PORT_POL_SET.PX2 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
1 (R/W1S)	PX1	Port x Bit 1 Polarity Invert Set. The PORT_POL_SET.PX1 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
0 (R/W1S)	PX0	Port x Bit 0 Polarity Invert Set. The PORT_POL_SET.PX0 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.

Port x GPIO Trigger Toggle Register

The PORT_TRIG_TGL register permits toggling the state of output GPIO pins in response to a trigger from the TRU for the corresponding port. Setting bits in the PORT_TRIG_TGL register enables triggers to toggle the state of those specific pins without impacting other pins of the port.

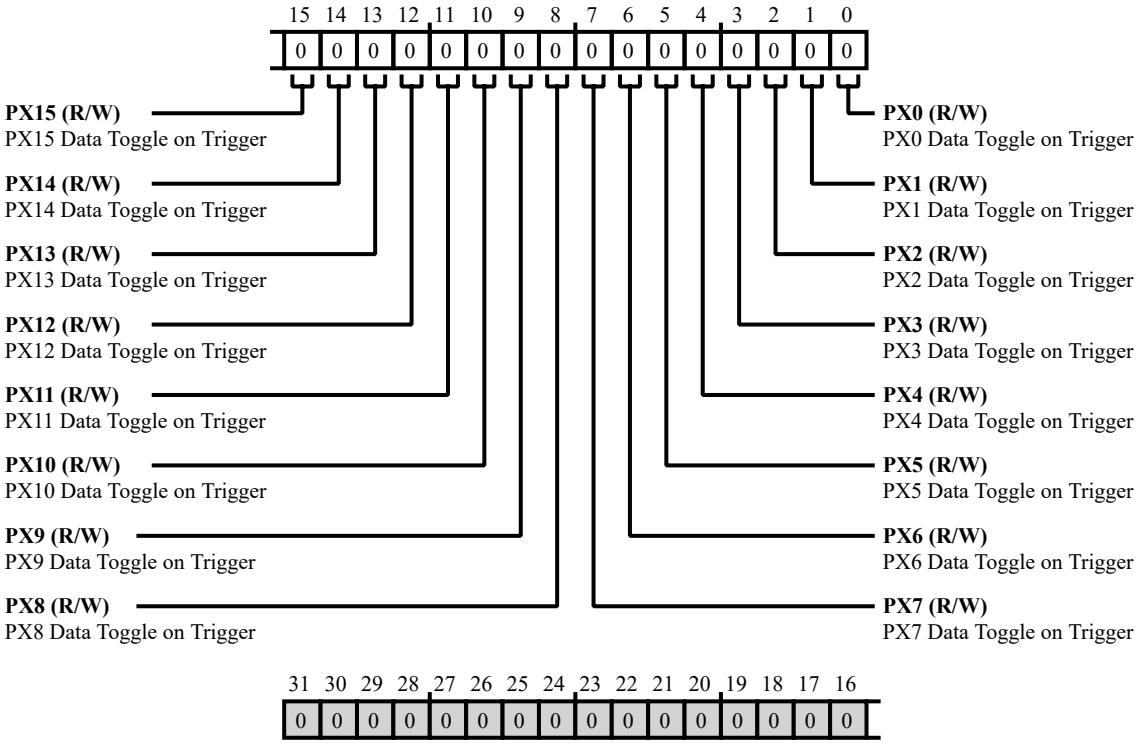


Figure 13-25: PORT_TRIG_TGL Register Diagram

Table 13-28: PORT_TRIG_TGL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PX15	PX15 Data Toggle on Trigger. The PORT_TRIG_TGL.PX15 bit enables triggers to toggle the state of the pin.
14 (R/W)	PX14	PX14 Data Toggle on Trigger. The PORT_TRIG_TGL.PX14 bit enables triggers to toggle the state of the pin.
13 (R/W)	PX13	PX13 Data Toggle on Trigger. The PORT_TRIG_TGL.PX13 bit enables triggers to toggle the state of the pin.
12 (R/W)	PX12	PX12 Data Toggle on Trigger. The PORT_TRIG_TGL.PX12 bit enables triggers to toggle the state of the pin.
11 (R/W)	PX11	PX11 Data Toggle on Trigger. The PORT_TRIG_TGL.PX11 bit enables triggers to toggle the state of the pin.

Table 13-28: PORT_TRIG_TGL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	PX10	PX10 Data Toggle on Trigger. The PORT_TRIG_TGL.PX10 bit enables triggers to toggle the state of the pin.
9 (R/W)	PX9	PX9 Data Toggle on Trigger. The PORT_TRIG_TGL.PX9 bit enables triggers to toggle the state of the pin.
8 (R/W)	PX8	PX8 Data Toggle on Trigger. The PORT_TRIG_TGL.PX8 bit enables triggers to toggle the state of the pin.
7 (R/W)	PX7	PX7 Data Toggle on Trigger. The PORT_TRIG_TGL.PX7 bit enables triggers to toggle the state of the pin.
6 (R/W)	PX6	PX6 Data Toggle on Trigger. The PORT_TRIG_TGL.PX6 bit enables triggers to toggle the state of the pin.
5 (R/W)	PX5	PX5 Data Toggle on Trigger. The PORT_TRIG_TGL.PX5 bit enables triggers to toggle the state of the pin.
4 (R/W)	PX4	PX4 Data Toggle on Trigger. The PORT_TRIG_TGL.PX4 bit enables triggers to toggle the state of the pin.
3 (R/W)	PX3	PX3 Data Toggle on Trigger. The PORT_TRIG_TGL.PX3 bit enables triggers to toggle the state of the pin.
2 (R/W)	PX2	PX2 Data Toggle on Trigger. The PORT_TRIG_TGL.PX2 bit enables triggers to toggle the state of the pin.
1 (R/W)	PX1	PX1 Data Toggle on Trigger. The PORT_TRIG_TGL.PX1 bit enables triggers to toggle the state of the pin.
0 (R/W)	PX0	PX0 Data Toggle on Trigger. The PORT_TRIG_TGL.PX0 bit enables triggers to toggle the state of the pin.

ADSP-2184x PINT Register Descriptions

The Pin Interrupt module (PINT) contains the following registers.

Table 13-29: ADSP-2184x PINT Register List

Name	Description
PINT_ASSIGN	PINT Assign Register
PINT_EDGE_CLR	PINT Edge Clear Register
PINT_EDGE_SET	PINT Edge Set Register
PINT_INV_CLR	PINT Invert Clear Register

Table 13-29: ADSP-2184x PINT Register List (Continued)

Name	Description
PINT_INV_SET	PINT Invert Set Register
PINT_LATCH	PINT Latch Register
PINT_MSK_CLR	PINT Mask Clear Register
PINT_MSK_SET	PINT Mask Set Register
PINT_PINSTATE	PINT Pin State Register
PINT_REQ	PINT Request Register

PINT Assign Register

The PINT_ASSIGN register controls the pin-to-interrupt request assignment in a byte-wide manner. This register consists of four control bytes that each function as a multiplexer control.

The PINT ports are subdivided into 8-bit half ports, resulting in lower and upper half 8-bit units. Using the multiplexers controlled by the PINT_ASSIGN register, the lower half units of eight pins can be forwarded to either byte 0 or byte 2 of either associated PINT block. The upper half units can be forwarded to either byte 1 or byte 3 of the PINT block, without further restrictions.

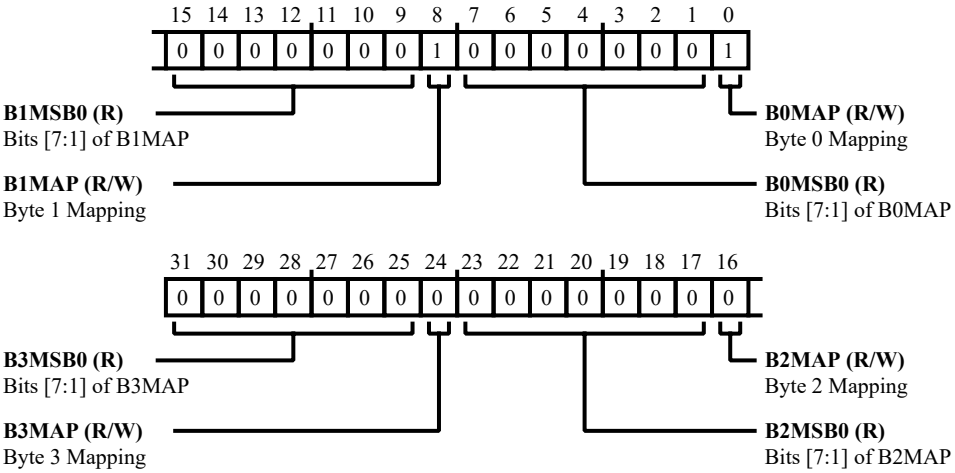


Figure 13-26: PINT_ASSIGN Register Diagram

Table 13-30: PINT_ASSIGN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:25 (R/NW)	B3MSB0	Bits [7:1] of B3MAP.	
24 (R/W)	B3MAP	Byte 3 Mapping.	
		0	B3MAP_PAH. Byte 3 = PA.H
		1	B3MAP_PBH. Byte 3 = PB.H
23:17 (R/NW)	B2MSB0	Bits [7:1] of B2MAP.	
16 (R/W)	B2MAP	Byte 2 Mapping.	
		0	B2MAP_PAL. Byte 2 = PA.L
		1	B2MAP_PBL. Byte 2 = PB.L
15:9 (R/NW)	B1MSB0	Bits [7:1] of B1MAP.	

Table 13-30: PINT_ASSIGN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	B1MAP	Byte 1 Mapping.	
		0	B1MAP_PAH. Byte 1 = PA.H
		1	B1MAP_PBH. Byte 1 = PB.H
7:1 (R/NW)	B0MSB0	Bits [7:1] of B0MAP.	
0 (R/W)	B0MAP	Byte 0 Mapping.	
		0	B0MAP_PAL. Byte 0 = PA.L
		1	B0MAP_PBL. Byte 0 = PB.L

PINT Edge Clear Register

The PINT_EDGE_CLR register permits selecting level-sensitive interrupts. Writing 1 to a bit in PINT_EDGE_CLR enables level sensitivity for the corresponding pin interrupt.

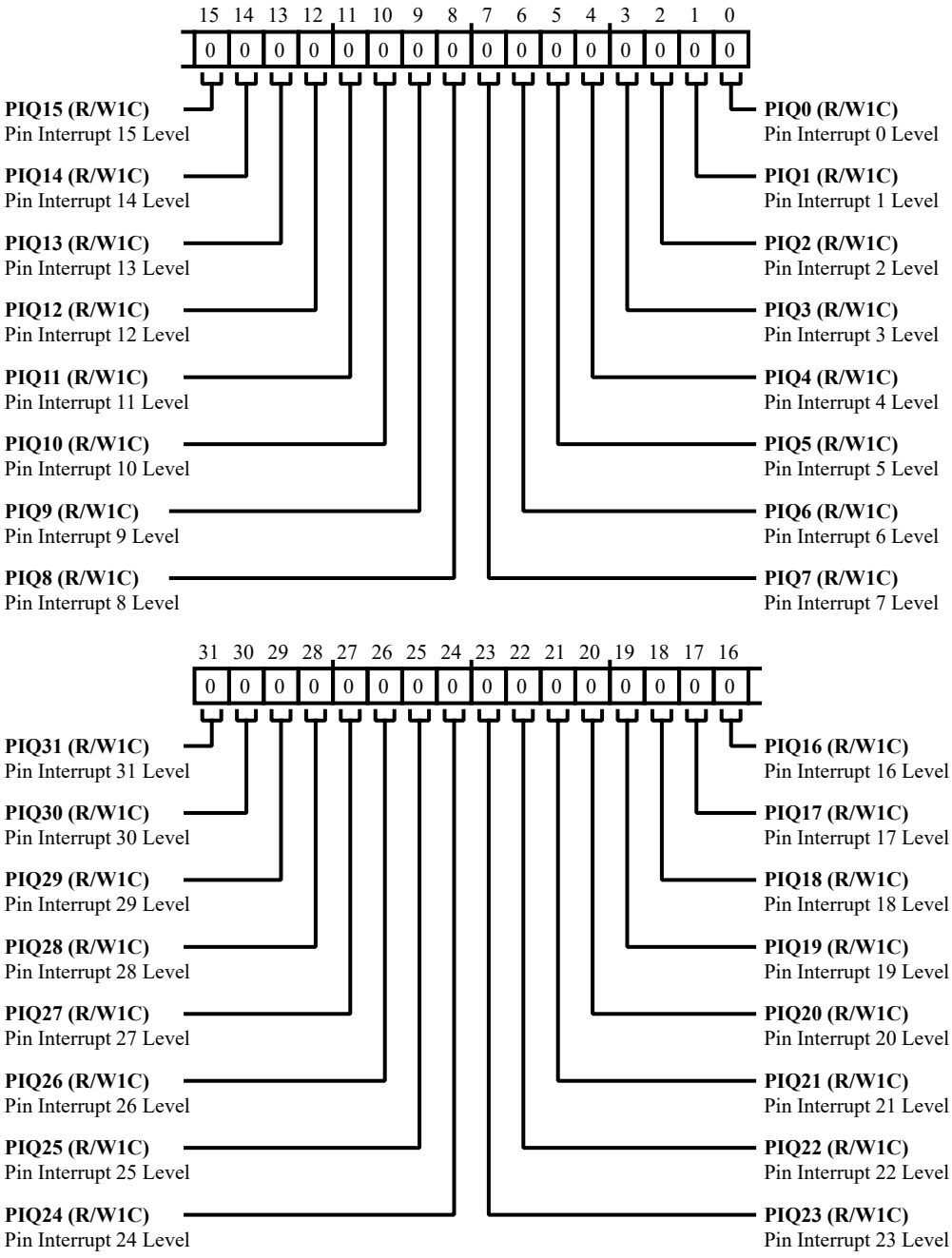


Figure 13-27: PINT_EDGE_CLR Register Diagram

Table 13-31: PINT_EDGE_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Level. Set the PINT_EDGE_CLR.PIQ31 bit to enable level sensitivity.
30 (R/W1C)	PIQ30	Pin Interrupt 30 Level. Set the PINT_EDGE_CLR.PIQ30 bit to enable level sensitivity.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Level. Set the PINT_EDGE_CLR.PIQ29 bit to enable level sensitivity.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Level. Set the PINT_EDGE_CLR.PIQ28 bit to enable level sensitivity.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Level. Set the PINT_EDGE_CLR.PIQ27 bit to enable level sensitivity.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Level. Set the PINT_EDGE_CLR.PIQ26 bit to enable level sensitivity.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Level. Set the PINT_EDGE_CLR.PIQ25 bit to enable level sensitivity.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Level. Set the PINT_EDGE_CLR.PIQ24 bit to enable level sensitivity.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Level. Set the PINT_EDGE_CLR.PIQ23 bit to enable level sensitivity.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Level. Set the PINT_EDGE_CLR.PIQ22 bit to enable level sensitivity.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Level. Set the PINT_EDGE_CLR.PIQ21 bit to enable level sensitivity.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Level. Set the PINT_EDGE_CLR.PIQ20 bit to enable level sensitivity.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Level. Set the PINT_EDGE_CLR.PIQ19 bit to enable level sensitivity.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Level. Set the PINT_EDGE_CLR.PIQ18 bit to enable level sensitivity.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Level. Set the PINT_EDGE_CLR.PIQ17 bit to enable level sensitivity.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Level. Set the PINT_EDGE_CLR.PIQ16 bit to enable level sensitivity.

Table 13-31: PINT_EDGE_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1C)	PIQ15	Pin Interrupt 15 Level. Set the PINT_EDGE_CLR.PIQ15 bit to enable level sensitivity.
14 (R/W1C)	PIQ14	Pin Interrupt 14 Level. Set the PINT_EDGE_CLR.PIQ14 bit to enable level sensitivity.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Level. Set the PINT_EDGE_CLR.PIQ13 bit to enable level sensitivity.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Level. Set the PINT_EDGE_CLR.PIQ12 bit to enable level sensitivity.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Level. Set the PINT_EDGE_CLR.PIQ11 bit to enable level sensitivity.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Level. Set the PINT_EDGE_CLR.PIQ10 bit to enable level sensitivity.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Level. Set the PINT_EDGE_CLR.PIQ9 bit to enable level sensitivity.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Level. Set the PINT_EDGE_CLR.PIQ8 bit to enable level sensitivity.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Level. Set the PINT_EDGE_CLR.PIQ7 bit to enable level sensitivity.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Level. Set the PINT_EDGE_CLR.PIQ6 bit to enable level sensitivity.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Level. Set the PINT_EDGE_CLR.PIQ5 bit to enable level sensitivity.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Level. Set the PINT_EDGE_CLR.PIQ4 bit to enable level sensitivity.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Level. Set the PINT_EDGE_CLR.PIQ3 bit to enable level sensitivity.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Level. Set the PINT_EDGE_CLR.PIQ2 bit to enable level sensitivity.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Level. Set the PINT_EDGE_CLR.PIQ1 bit to enable level sensitivity.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Level. Set the PINT_EDGE_CLR.PIQ0 bit to enable level sensitivity.

PINT Edge Set Register

The PINT_EDGE_SET register permits selecting edge-sensitive interrupts. Writing 1 to a bit in PINT_EDGE_SET enables edge sensitivity for the corresponding pin interrupt.

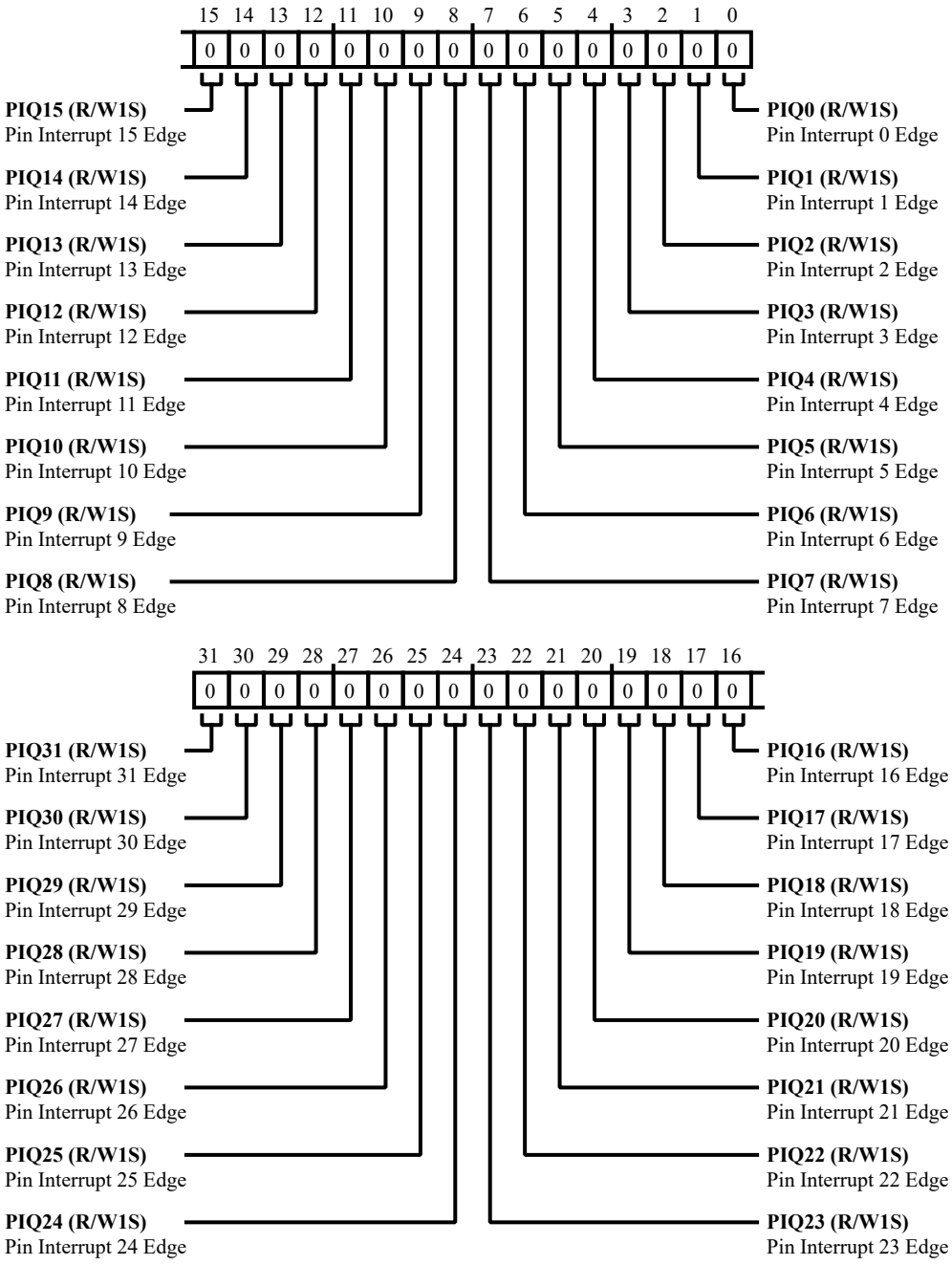


Figure 13-28: PINT_EDGE_SET Register Diagram

Table 13-32: PINT_EDGE_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	PIQ31	Pin Interrupt 31 Edge. Set the PINT_EDGE_SET.PIQ31 bit to enable edge sensitivity.
30 (R/W1S)	PIQ30	Pin Interrupt 30 Edge. Set the PINT_EDGE_SET.PIQ30 bit to enable edge sensitivity.
29 (R/W1S)	PIQ29	Pin Interrupt 29 Edge. Set the PINT_EDGE_SET.PIQ29 bit to enable edge sensitivity.
28 (R/W1S)	PIQ28	Pin Interrupt 28 Edge. Set the PINT_EDGE_SET.PIQ28 bit to enable edge sensitivity.
27 (R/W1S)	PIQ27	Pin Interrupt 27 Edge. Set the PINT_EDGE_SET.PIQ27 bit to enable edge sensitivity.
26 (R/W1S)	PIQ26	Pin Interrupt 26 Edge. Set the PINT_EDGE_SET.PIQ26 bit to enable edge sensitivity.
25 (R/W1S)	PIQ25	Pin Interrupt 25 Edge. Set the PINT_EDGE_SET.PIQ25 bit to enable edge sensitivity.
24 (R/W1S)	PIQ24	Pin Interrupt 24 Edge. Set the PINT_EDGE_SET.PIQ24 bit to enable edge sensitivity.
23 (R/W1S)	PIQ23	Pin Interrupt 23 Edge. Set the PINT_EDGE_SET.PIQ23 bit to enable edge sensitivity.
22 (R/W1S)	PIQ22	Pin Interrupt 22 Edge. Set the PINT_EDGE_SET.PIQ22 bit to enable edge sensitivity.
21 (R/W1S)	PIQ21	Pin Interrupt 21 Edge. Set the PINT_EDGE_SET.PIQ21 bit to enable edge sensitivity.
20 (R/W1S)	PIQ20	Pin Interrupt 20 Edge. Set the PINT_EDGE_SET.PIQ20 bit to enable edge sensitivity.
19 (R/W1S)	PIQ19	Pin Interrupt 19 Edge. Set the PINT_EDGE_SET.PIQ19 bit to enable edge sensitivity.
18 (R/W1S)	PIQ18	Pin Interrupt 18 Edge. Set the PINT_EDGE_SET.PIQ18 bit to enable edge sensitivity.
17 (R/W1S)	PIQ17	Pin Interrupt 17 Edge. Set the PINT_EDGE_SET.PIQ17 bit to enable edge sensitivity.
16 (R/W1S)	PIQ16	Pin Interrupt 16 Edge. Set the PINT_EDGE_SET.PIQ16 bit to enable edge sensitivity.

Table 13-32: PINT_EDGE_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1S)	PIQ15	Pin Interrupt 15 Edge. Set the PINT_EDGE_SET.PIQ15 bit to enable edge sensitivity.
14 (R/W1S)	PIQ14	Pin Interrupt 14 Edge. Set the PINT_EDGE_SET.PIQ14 bit to enable edge sensitivity.
13 (R/W1S)	PIQ13	Pin Interrupt 13 Edge. Set the PINT_EDGE_SET.PIQ13 bit to enable edge sensitivity.
12 (R/W1S)	PIQ12	Pin Interrupt 12 Edge. Set the PINT_EDGE_SET.PIQ12 bit to enable edge sensitivity.
11 (R/W1S)	PIQ11	Pin Interrupt 11 Edge. Set the PINT_EDGE_SET.PIQ11 bit to enable edge sensitivity.
10 (R/W1S)	PIQ10	Pin Interrupt 10 Edge. Set the PINT_EDGE_SET.PIQ10 bit to enable edge sensitivity.
9 (R/W1S)	PIQ9	Pin Interrupt 9 Edge. Set the PINT_EDGE_SET.PIQ9 bit to enable edge sensitivity.
8 (R/W1S)	PIQ8	Pin Interrupt 8 Edge. Set the PINT_EDGE_SET.PIQ8 bit to enable edge sensitivity.
7 (R/W1S)	PIQ7	Pin Interrupt 7 Edge. Set the PINT_EDGE_SET.PIQ7 bit to enable edge sensitivity.
6 (R/W1S)	PIQ6	Pin Interrupt 6 Edge. Set the PINT_EDGE_SET.PIQ6 bit to enable edge sensitivity.
5 (R/W1S)	PIQ5	Pin Interrupt 5 Edge. Set the PINT_EDGE_SET.PIQ5 bit to enable edge sensitivity.
4 (R/W1S)	PIQ4	Pin Interrupt 4 Edge. Set the PINT_EDGE_SET.PIQ4 bit to enable edge sensitivity.
3 (R/W1S)	PIQ3	Pin Interrupt 3 Edge. Set the PINT_EDGE_SET.PIQ3 bit to enable edge sensitivity.
2 (R/W1S)	PIQ2	Pin Interrupt 2 Edge. Set the PINT_EDGE_SET.PIQ2 bit to enable edge sensitivity.
1 (R/W1S)	PIQ1	Pin Interrupt 1 Edge. Set the PINT_EDGE_SET.PIQ1 bit to enable edge sensitivity.
0 (R/W1S)	PIQ0	Pin Interrupt 0 Edge. Set the PINT_EDGE_SET.PIQ0 bit to enable edge sensitivity.

PINT Invert Clear Register

The PINT_INV_CLR register disables inverting input polarity. Writing 1 to a bit in PINT_INV_CLR disables an inverter for input on the corresponding pin.

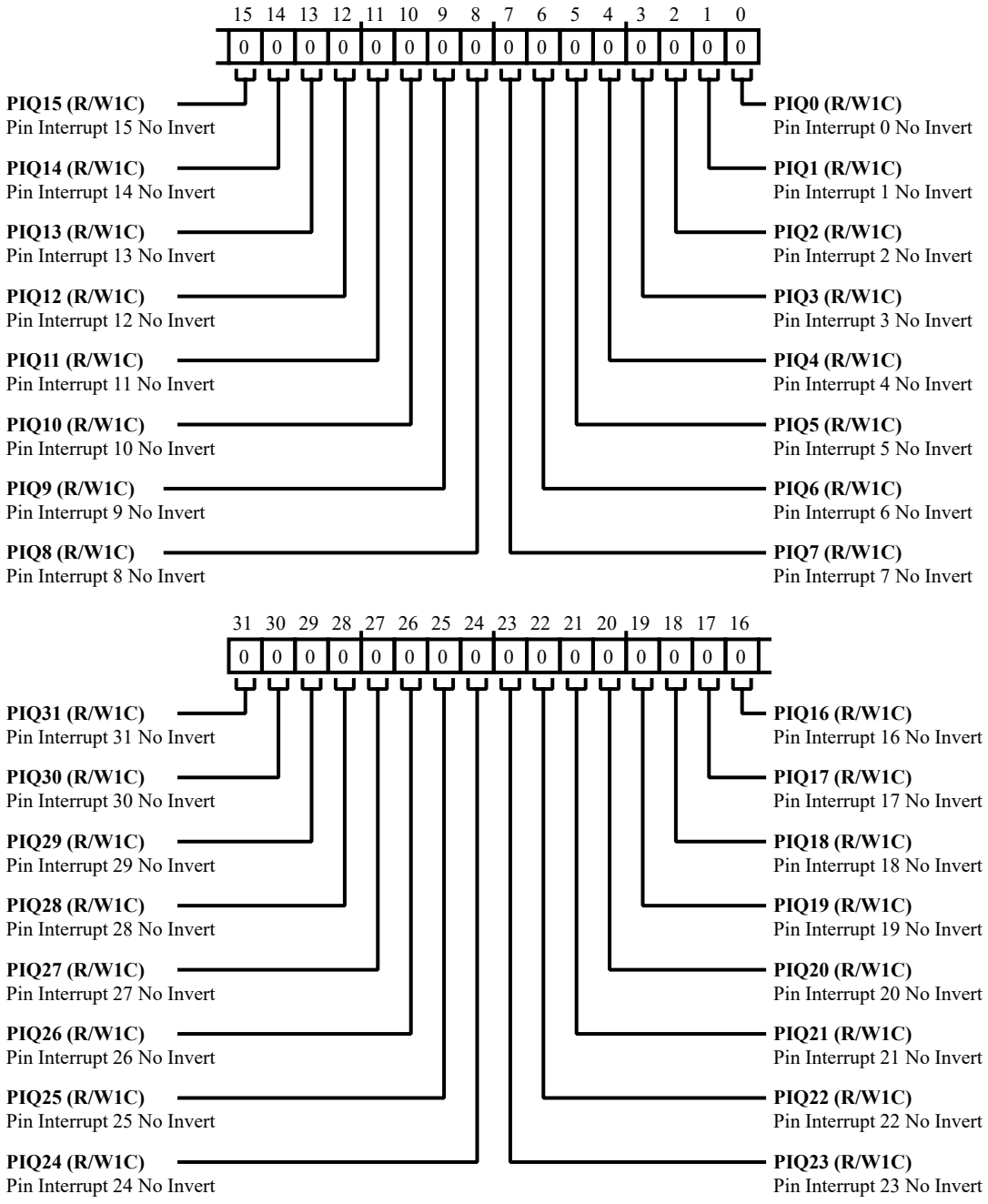


Figure 13-29: PINT_INV_CLR Register Diagram

Table 13-33: PINT_INV_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 No Invert. Set the PINT_INV_CLR.PIQ31 bit to disable inverted input.
30 (R/W1C)	PIQ30	Pin Interrupt 30 No Invert. Set the PINT_INV_CLR.PIQ30 bit to disable inverted input.
29 (R/W1C)	PIQ29	Pin Interrupt 29 No Invert. Set the PINT_INV_CLR.PIQ29 bit to disable inverted input.
28 (R/W1C)	PIQ28	Pin Interrupt 28 No Invert. Set the PINT_INV_CLR.PIQ28 bit to disable inverted input.
27 (R/W1C)	PIQ27	Pin Interrupt 27 No Invert. Set the PINT_INV_CLR.PIQ27 bit to disable inverted input.
26 (R/W1C)	PIQ26	Pin Interrupt 26 No Invert. Set the PINT_INV_CLR.PIQ26 bit to disable inverted input.
25 (R/W1C)	PIQ25	Pin Interrupt 25 No Invert. Set the PINT_INV_CLR.PIQ25 bit to disable inverted input.
24 (R/W1C)	PIQ24	Pin Interrupt 24 No Invert. Set the PINT_INV_CLR.PIQ24 bit to disable inverted input.
23 (R/W1C)	PIQ23	Pin Interrupt 23 No Invert. Set the PINT_INV_CLR.PIQ23 bit to disable inverted input.
22 (R/W1C)	PIQ22	Pin Interrupt 22 No Invert. Set the PINT_INV_CLR.PIQ22 bit to disable inverted input.
21 (R/W1C)	PIQ21	Pin Interrupt 21 No Invert. Set the PINT_INV_CLR.PIQ21 bit to disable inverted input.
20 (R/W1C)	PIQ20	Pin Interrupt 20 No Invert. Set the PINT_INV_CLR.PIQ20 bit to disable inverted input.
19 (R/W1C)	PIQ19	Pin Interrupt 19 No Invert. Set the PINT_INV_CLR.PIQ19 bit to disable inverted input.
18 (R/W1C)	PIQ18	Pin Interrupt 18 No Invert. Set the PINT_INV_CLR.PIQ18 bit to disable inverted input.
17 (R/W1C)	PIQ17	Pin Interrupt 17 No Invert. Set the PINT_INV_CLR.PIQ17 bit to disable inverted input.
16 (R/W1C)	PIQ16	Pin Interrupt 16 No Invert. Set the PINT_INV_CLR.PIQ16 bit to disable inverted input.

Table 13-33: PINT_INV_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1C)	PIQ15	Pin Interrupt 15 No Invert. Set the PINT_INV_CLR.PIQ15 bit to disable inverted input.
14 (R/W1C)	PIQ14	Pin Interrupt 14 No Invert. Set the PINT_INV_CLR.PIQ14 bit to disable inverted input.
13 (R/W1C)	PIQ13	Pin Interrupt 13 No Invert. Set the PINT_INV_CLR.PIQ13 bit to disable inverted input.
12 (R/W1C)	PIQ12	Pin Interrupt 12 No Invert. Set the PINT_INV_CLR.PIQ12 bit to disable inverted input.
11 (R/W1C)	PIQ11	Pin Interrupt 11 No Invert. Set the PINT_INV_CLR.PIQ11 bit to disable inverted input.
10 (R/W1C)	PIQ10	Pin Interrupt 10 No Invert. Set the PINT_INV_CLR.PIQ10 bit to disable inverted input.
9 (R/W1C)	PIQ9	Pin Interrupt 9 No Invert. Set the PINT_INV_CLR.PIQ9 bit to disable inverted input.
8 (R/W1C)	PIQ8	Pin Interrupt 8 No Invert. Set the PINT_INV_CLR.PIQ8 bit to disable inverted input.
7 (R/W1C)	PIQ7	Pin Interrupt 7 No Invert. Set the PINT_INV_CLR.PIQ7 bit to disable inverted input.
6 (R/W1C)	PIQ6	Pin Interrupt 6 No Invert. Set the PINT_INV_CLR.PIQ6 bit to disable inverted input.
5 (R/W1C)	PIQ5	Pin Interrupt 5 No Invert. Set the PINT_INV_CLR.PIQ5 bit to disable inverted input.
4 (R/W1C)	PIQ4	Pin Interrupt 4 No Invert. Set the PINT_INV_CLR.PIQ4 bit to disable inverted input.
3 (R/W1C)	PIQ3	Pin Interrupt 3 No Invert. Set the PINT_INV_CLR.PIQ3 bit to disable inverted input.
2 (R/W1C)	PIQ2	Pin Interrupt 2 No Invert. Set the PINT_INV_CLR.PIQ2 bit to disable inverted input.
1 (R/W1C)	PIQ1	Pin Interrupt 1 No Invert. Set the PINT_INV_CLR.PIQ1 bit to disable inverted input.
0 (R/W1C)	PIQ0	Pin Interrupt 0 No Invert. Set the PINT_INV_CLR.PIQ0 bit to disable inverted input.

PINT Invert Set Register

The PINT_INV_SET register enables inverting input polarity. Writing 1 to a bit in PINT_INV_SET enables an inverter for input on the corresponding pin.

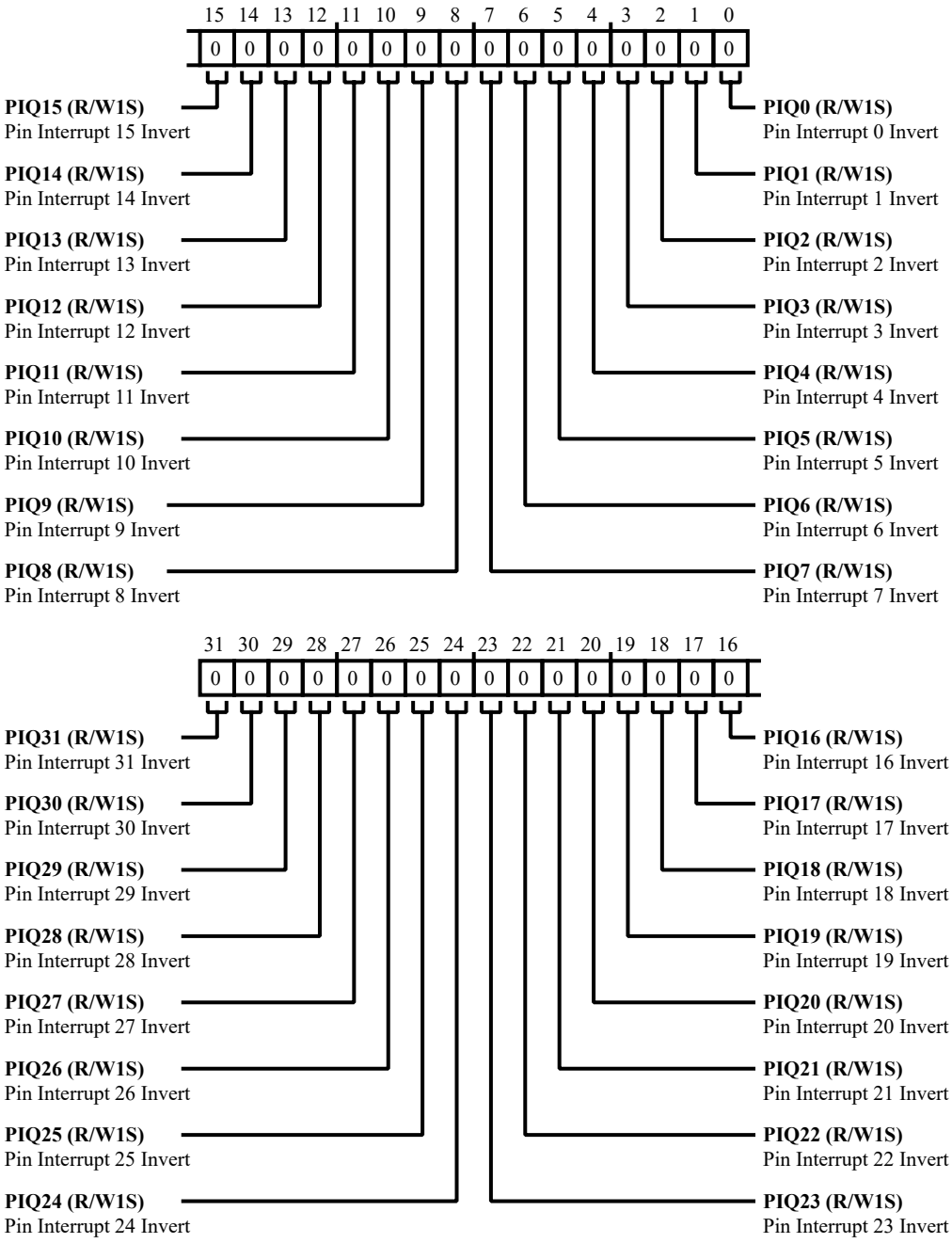


Figure 13-30: PINT_INV_SET Register Diagram

Table 13-34: PINT_INV_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	PIQ31	Pin Interrupt 31 Invert. Set the PINT_INV_SET.PIQ31 bit to enable inverted input.
30 (R/W1S)	PIQ30	Pin Interrupt 30 Invert. Set the PINT_INV_SET.PIQ30 bit to enable inverted input.
29 (R/W1S)	PIQ29	Pin Interrupt 29 Invert. Set the PINT_INV_SET.PIQ29 bit to enable inverted input.
28 (R/W1S)	PIQ28	Pin Interrupt 28 Invert. Set the PINT_INV_SET.PIQ28 bit to enable inverted input.
27 (R/W1S)	PIQ27	Pin Interrupt 27 Invert. Set the PINT_INV_SET.PIQ27 bit to enable inverted input.
26 (R/W1S)	PIQ26	Pin Interrupt 26 Invert. Set the PINT_INV_SET.PIQ26 bit to enable inverted input.
25 (R/W1S)	PIQ25	Pin Interrupt 25 Invert. Set the PINT_INV_SET.PIQ25 bit to enable inverted input.
24 (R/W1S)	PIQ24	Pin Interrupt 24 Invert. Set the PINT_INV_SET.PIQ24 bit to enable inverted input.
23 (R/W1S)	PIQ23	Pin Interrupt 23 Invert. Set the PINT_INV_SET.PIQ23 bit to enable inverted input.
22 (R/W1S)	PIQ22	Pin Interrupt 22 Invert. Set the PINT_INV_SET.PIQ22 bit to enable inverted input.
21 (R/W1S)	PIQ21	Pin Interrupt 21 Invert. Set the PINT_INV_SET.PIQ21 bit to enable inverted input.
20 (R/W1S)	PIQ20	Pin Interrupt 20 Invert. Set the PINT_INV_SET.PIQ20 bit to enable inverted input.
19 (R/W1S)	PIQ19	Pin Interrupt 19 Invert. Set the PINT_INV_SET.PIQ19 bit to enable inverted input.
18 (R/W1S)	PIQ18	Pin Interrupt 18 Invert. Set the PINT_INV_SET.PIQ18 bit to enable inverted input.
17 (R/W1S)	PIQ17	Pin Interrupt 17 Invert. Set the PINT_INV_SET.PIQ17 bit to enable inverted input.
16 (R/W1S)	PIQ16	Pin Interrupt 16 Invert. Set the PINT_INV_SET.PIQ16 bit to enable inverted input.

Table 13-34: PINT_INV_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1S)	PIQ15	Pin Interrupt 15 Invert. Set the PINT_INV_SET.PIQ15 bit to enable inverted input.
14 (R/W1S)	PIQ14	Pin Interrupt 14 Invert. Set the PINT_INV_SET.PIQ14 bit to enable inverted input.
13 (R/W1S)	PIQ13	Pin Interrupt 13 Invert. Set the PINT_INV_SET.PIQ13 bit to enable inverted input.
12 (R/W1S)	PIQ12	Pin Interrupt 12 Invert. Set the PINT_INV_SET.PIQ12 bit to enable inverted input.
11 (R/W1S)	PIQ11	Pin Interrupt 11 Invert. Set the PINT_INV_SET.PIQ11 bit to enable inverted input.
10 (R/W1S)	PIQ10	Pin Interrupt 10 Invert. Set the PINT_INV_SET.PIQ10 bit to enable inverted input.
9 (R/W1S)	PIQ9	Pin Interrupt 9 Invert. Set the PINT_INV_SET.PIQ9 bit to enable inverted input.
8 (R/W1S)	PIQ8	Pin Interrupt 8 Invert. Set the PINT_INV_SET.PIQ8 bit to enable inverted input.
7 (R/W1S)	PIQ7	Pin Interrupt 7 Invert. Set the PINT_INV_SET.PIQ7 bit to enable inverted input.
6 (R/W1S)	PIQ6	Pin Interrupt 6 Invert. Set the PINT_INV_SET.PIQ6 bit to enable inverted input.
5 (R/W1S)	PIQ5	Pin Interrupt 5 Invert. Set the PINT_INV_SET.PIQ5 bit to enable inverted input.
4 (R/W1S)	PIQ4	Pin Interrupt 4 Invert. Set the PINT_INV_SET.PIQ4 bit to enable inverted input.
3 (R/W1S)	PIQ3	Pin Interrupt 3 Invert. Set the PINT_INV_SET.PIQ3 bit to enable inverted input.
2 (R/W1S)	PIQ2	Pin Interrupt 2 Invert. Set the PINT_INV_SET.PIQ2 bit to enable inverted input.
1 (R/W1S)	PIQ1	Pin Interrupt 1 Invert. Set the PINT_INV_SET.PIQ1 bit to enable inverted input.
0 (R/W1S)	PIQ0	Pin Interrupt 0 Invert. Set the PINT_INV_SET.PIQ0 bit to enable inverted input.

PINT Latch Register

The PINT_LATCH register indicates the interrupt latch status for pin interrupts. When set, an interrupt request is latched. When cleared, there is no interrupt request latched.

Both the PINT_REQ and PINT_LATCH registers indicate whether an interrupt request is latched on the respective pin. The PINT_LATCH register is a latch that operates regardless of the interrupt masks. Bits of the PINT_REQ register depend on the mask register. The PINT_REQ register is a logical AND of the PINT_LATCH register and the interrupt mask.

Having two separate registers here enables the user to interrogate certain pins in polling mode while others work in interrupt mode. The PINT_LATCH registers can be used for edge detection or pin activity detection.

Both registers have W1C behavior. Writing a 1 to either register clears the respective bits in both registers. For interrupt operation, the user may prefer to W1C the PINT_REQ register (address still loaded in Px pointer). In polling mode, it might be cleaner to W1C the PINT_LATCH register.

Whether in edge-sensitive mode or level-sensitive mode, PINT_LATCH bits are never cleared by hardware except at system reset. Even in level-sensitive mode, the PINT_LATCH register functions as latch.

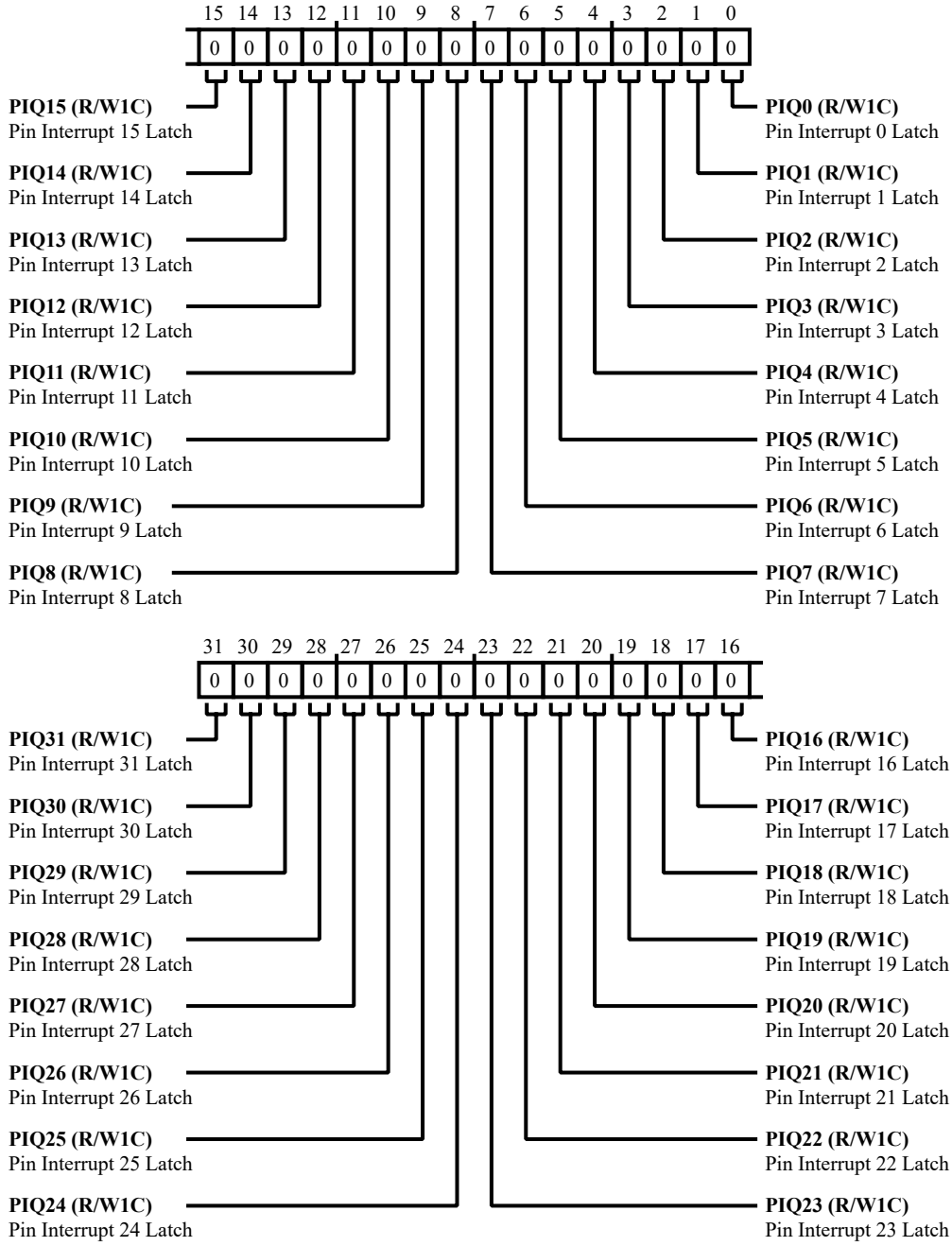


Figure 13-31: PINT_LATCH Register Diagram

Table 13-35: PINT_LATCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Latch. If the `PINT_LATCH.PIQ31` bit is set, the request is latched.

Table 13-35: PINT_LATCH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W1C)	PIQ30	Pin Interrupt 30 Latch. If the `PINT_LATCH.PIQ30` bit is set, the request is latched.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Latch. If the `PINT_LATCH.PIQ29` bit is set, the request is latched.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Latch. If the `PINT_LATCH.PIQ28` bit is set, the request is latched.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Latch. If the `PINT_LATCH.PIQ27` bit is set, the request is latched.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Latch. If the `PINT_LATCH.PIQ26` bit is set, the request is latched.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Latch. If the `PINT_LATCH.PIQ25` bit is set, the request is latched.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Latch. If the `PINT_LATCH.PIQ24` bit is set, the request is latched.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Latch. If the `PINT_LATCH.PIQ23` bit is set, the request is latched.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Latch. If the `PINT_LATCH.PIQ22` bit is set, the request is latched.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Latch. If the `PINT_LATCH.PIQ21` bit is set, the request is latched.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Latch. If the `PINT_LATCH.PIQ20` bit is set, the request is latched.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Latch. If the `PINT_LATCH.PIQ19` bit is set, the request is latched.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Latch. If the `PINT_LATCH.PIQ18` bit is set, the request is latched.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Latch. If the `PINT_LATCH.PIQ17` bit is set, the request is latched.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Latch. If the `PINT_LATCH.PIQ16` bit is set, the request is latched.
15 (R/W1C)	PIQ15	Pin Interrupt 15 Latch. If the `PINT_LATCH.PIQ15` bit is set, the request is latched.

Table 13-35: PINT_LATCH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	PIQ14	Pin Interrupt 14 Latch. If the `PINT_LATCH.PIQ14` bit is set, the request is latched.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Latch. If the `PINT_LATCH.PIQ13` bit is set, the request is latched.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Latch. If the `PINT_LATCH.PIQ12` bit is set, the request is latched.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Latch. If the `PINT_LATCH.PIQ11` bit is set, the request is latched.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Latch. If the `PINT_LATCH.PIQ10` bit is set, the request is latched.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Latch. If the `PINT_LATCH.PIQ9` bit is set, the request is latched.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Latch. If the `PINT_LATCH.PIQ8` bit is set, the request is latched.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Latch. If the `PINT_LATCH.PIQ7` bit is set, the request is latched.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Latch. If the `PINT_LATCH.PIQ6` bit is set, the request is latched.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Latch. If the `PINT_LATCH.PIQ5` bit is set, the request is latched.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Latch. If the `PINT_LATCH.PIQ4` bit is set, the request is latched.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Latch. If the `PINT_LATCH.PIQ3` bit is set, the request is latched.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Latch. If the `PINT_LATCH.PIQ2` bit is set, the request is latched.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Latch. If the `PINT_LATCH.PIQ1` bit is set, the request is latched.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Latch. If the `PINT_LATCH.PIQ0` bit is set, the request is latched.

PINT Mask Clear Register

The PINT_MSK_CLR register permits masking (disabling) of interrupt requests. Writing 1 to a bit in PINT_MSK_CLR masks the corresponding pin interrupt.

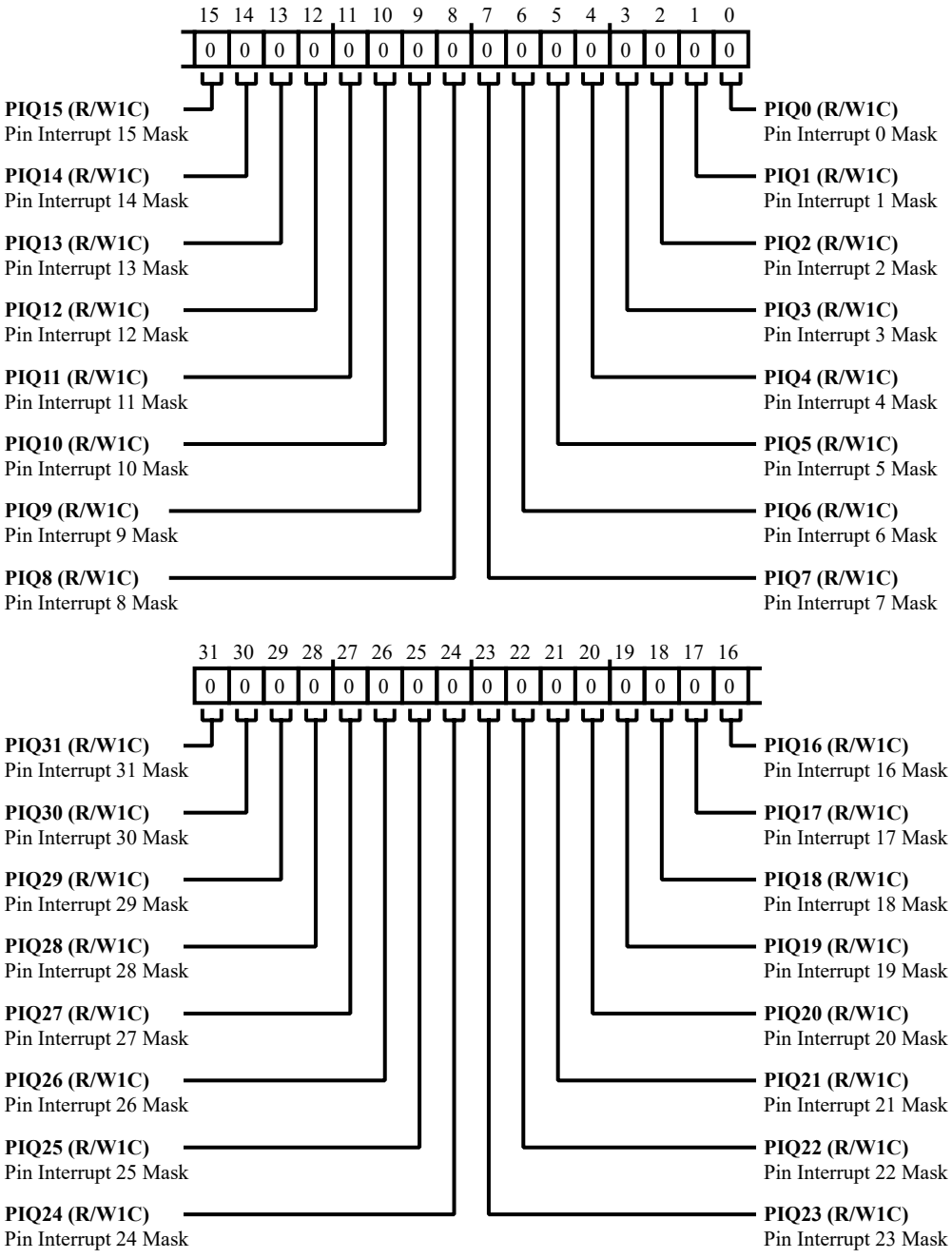


Figure 13-32: PINT_MSK_CLR Register Diagram

Table 13-36: PINT_MSK_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Mask. Set the PINT_MSK_CLR.PIQ31 bit to disable the interrupt.
30 (R/W1C)	PIQ30	Pin Interrupt 30 Mask. Set the PINT_MSK_CLR.PIQ30 bit to disable the interrupt.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Mask. Set the PINT_MSK_CLR.PIQ29 bit to disable the interrupt.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Mask. Set the PINT_MSK_CLR.PIQ28 bit to disable the interrupt.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Mask. Set the PINT_MSK_CLR.PIQ27 bit to disable the interrupt.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Mask. Set the PINT_MSK_CLR.PIQ26 bit to disable the interrupt.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Mask. Set the PINT_MSK_CLR.PIQ25 bit to disable the interrupt.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Mask. Set the PINT_MSK_CLR.PIQ24 bit to disable the interrupt.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Mask. Set the PINT_MSK_CLR.PIQ23 bit to disable the interrupt.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Mask. Set the PINT_MSK_CLR.PIQ22 bit to disable the interrupt.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Mask. Set the PINT_MSK_CLR.PIQ21 bit to disable the interrupt.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Mask. Set the PINT_MSK_CLR.PIQ20 bit to disable the interrupt.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Mask. Set the PINT_MSK_CLR.PIQ19 bit to disable the interrupt.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Mask. Set the PINT_MSK_CLR.PIQ18 bit to disable the interrupt.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Mask. Set the PINT_MSK_CLR.PIQ17 bit to disable the interrupt.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Mask. Set the PINT_MSK_CLR.PIQ16 bit to disable the interrupt.

Table 13-36: PINT_MSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1C)	PIQ15	Pin Interrupt 15 Mask. Set the PINT_MSK_CLR.PIQ15 bit to disable the interrupt.
14 (R/W1C)	PIQ14	Pin Interrupt 14 Mask. Set the PINT_MSK_CLR.PIQ14 bit to disable the interrupt.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Mask. Set the PINT_MSK_CLR.PIQ13 bit to disable the interrupt.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Mask. Set the PINT_MSK_CLR.PIQ12 bit to disable the interrupt.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Mask. Set the PINT_MSK_CLR.PIQ11 bit to disable the interrupt.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Mask. Set the PINT_MSK_CLR.PIQ10 bit to disable the interrupt.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Mask. Set the PINT_MSK_CLR.PIQ9 bit to disable the interrupt.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Mask. Set the PINT_MSK_CLR.PIQ8 bit to disable the interrupt.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Mask. Set the PINT_MSK_CLR.PIQ7 bit to disable the interrupt.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Mask. Set the PINT_MSK_CLR.PIQ6 bit to disable the interrupt.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Mask. Set the PINT_MSK_CLR.PIQ5 bit to disable the interrupt.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Mask. Set the PINT_MSK_CLR.PIQ4 bit to disable the interrupt.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Mask. Set the PINT_MSK_CLR.PIQ3 bit to disable the interrupt.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Mask. Set the PINT_MSK_CLR.PIQ2 bit to disable the interrupt.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Mask. Set the PINT_MSK_CLR.PIQ1 bit to disable the interrupt.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Mask. Set the PINT_MSK_CLR.PIQ0 bit to disable the interrupt.

PINT Mask Set Register

The PINT_MSK_SET register permits unmasking (enabling) of interrupt requests. Writing 1 to a bit in PINT_MSK_SET unmasks the corresponding pin interrupt.

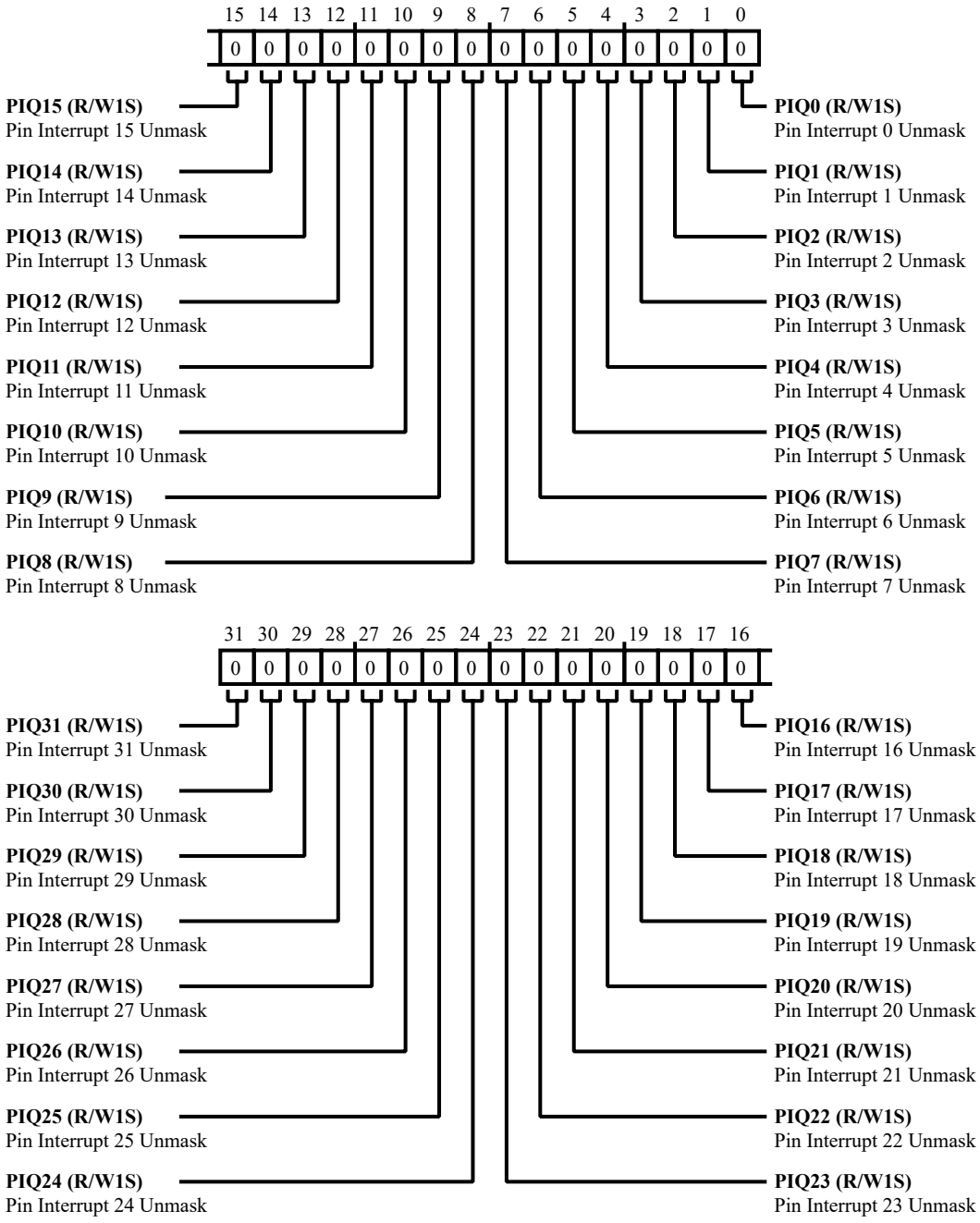


Figure 13-33: PINT_MSK_SET Register Diagram

Table 13-37: PINT_MSK_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	PIQ31	Pin Interrupt 31 Unmask. Set the PINT_MSK_SET.PIQ31 bit to enable the interrupt.
30 (R/W1S)	PIQ30	Pin Interrupt 30 Unmask. Set the PINT_MSK_SET.PIQ30 bit to enable the interrupt.
29 (R/W1S)	PIQ29	Pin Interrupt 29 Unmask. Set the PINT_MSK_SET.PIQ29 bit to enable the interrupt.
28 (R/W1S)	PIQ28	Pin Interrupt 28 Unmask. Set the PINT_MSK_SET.PIQ28 bit to enable the interrupt.
27 (R/W1S)	PIQ27	Pin Interrupt 27 Unmask. Set the PINT_MSK_SET.PIQ27 bit to enable the interrupt.
26 (R/W1S)	PIQ26	Pin Interrupt 26 Unmask. Set the PINT_MSK_SET.PIQ26 bit to enable the interrupt.
25 (R/W1S)	PIQ25	Pin Interrupt 25 Unmask. Set the PINT_MSK_SET.PIQ25 bit to enable the interrupt.
24 (R/W1S)	PIQ24	Pin Interrupt 24 Unmask. Set the PINT_MSK_SET.PIQ24 bit to enable the interrupt.
23 (R/W1S)	PIQ23	Pin Interrupt 23 Unmask. Set the PINT_MSK_SET.PIQ23 bit to enable the interrupt.
22 (R/W1S)	PIQ22	Pin Interrupt 22 Unmask. Set the PINT_MSK_SET.PIQ22 bit to enable the interrupt.
21 (R/W1S)	PIQ21	Pin Interrupt 21 Unmask. Set the PINT_MSK_SET.PIQ21 bit to enable the interrupt.
20 (R/W1S)	PIQ20	Pin Interrupt 20 Unmask. Set the PINT_MSK_SET.PIQ20 bit to enable the interrupt.
19 (R/W1S)	PIQ19	Pin Interrupt 19 Unmask. Set the PINT_MSK_SET.PIQ19 bit to enable the interrupt.
18 (R/W1S)	PIQ18	Pin Interrupt 18 Unmask. Set the PINT_MSK_SET.PIQ18 bit to enable the interrupt.
17 (R/W1S)	PIQ17	Pin Interrupt 17 Unmask. Set the PINT_MSK_SET.PIQ17 bit to enable the interrupt.
16 (R/W1S)	PIQ16	Pin Interrupt 16 Unmask. Set the PINT_MSK_SET.PIQ16 bit to enable the interrupt.

Table 13-37: PINT_MSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1S)	PIQ15	Pin Interrupt 15 Unmask. Set the `PINT_MSK_SET.PIQ15` bit to enable the interrupt.
14 (R/W1S)	PIQ14	Pin Interrupt 14 Unmask. Set the `PINT_MSK_SET.PIQ14` bit to enable the interrupt.
13 (R/W1S)	PIQ13	Pin Interrupt 13 Unmask. Set the `PINT_MSK_SET.PIQ13` bit to enable the interrupt.
12 (R/W1S)	PIQ12	Pin Interrupt 12 Unmask. Set the `PINT_MSK_SET.PIQ12` bit to enable the interrupt.
11 (R/W1S)	PIQ11	Pin Interrupt 11 Unmask. Set the `PINT_MSK_SET.PIQ11` bit to enable the interrupt.
10 (R/W1S)	PIQ10	Pin Interrupt 10 Unmask. Set the `PINT_MSK_SET.PIQ10` bit to enable the interrupt.
9 (R/W1S)	PIQ9	Pin Interrupt 9 Unmask. Set the `PINT_MSK_SET.PIQ9` bit to enable the interrupt.
8 (R/W1S)	PIQ8	Pin Interrupt 8 Unmask. Set the `PINT_MSK_SET.PIQ8` bit to enable the interrupt.
7 (R/W1S)	PIQ7	Pin Interrupt 7 Unmask. Set the `PINT_MSK_SET.PIQ7` bit to enable the interrupt.
6 (R/W1S)	PIQ6	Pin Interrupt 6 Unmask. Set the `PINT_MSK_SET.PIQ6` bit to enable the interrupt.
5 (R/W1S)	PIQ5	Pin Interrupt 5 Unmask. Set the `PINT_MSK_SET.PIQ5` bit to enable the interrupt.
4 (R/W1S)	PIQ4	Pin Interrupt 4 Unmask. Set the `PINT_MSK_SET.PIQ4` bit to enable the interrupt.
3 (R/W1S)	PIQ3	Pin Interrupt 3 Unmask. Set the `PINT_MSK_SET.PIQ3` bit to enable the interrupt.
2 (R/W1S)	PIQ2	Pin Interrupt 2 Unmask. Set the `PINT_MSK_SET.PIQ2` bit to enable the interrupt.
1 (R/W1S)	PIQ1	Pin Interrupt 1 Unmask. Set the `PINT_MSK_SET.PIQ1` bit to enable the interrupt.
0 (R/W1S)	PIQ0	Pin Interrupt 0 Unmask. Set the `PINT_MSK_SET.PIQ0` bit to enable the interrupt.

PINT Pin State Register

When a half port is assigned to a byte in any PINT block, the state of the eight pins (regardless of GPIO or function, input or output) can be seen in the PINT_PINSTATE register. While neither input nor output drivers of the pin are enabled, reads of the pin state in PINT_PINSTATE return zero. The PINT_PINSTATE register reports the inverted state of the pin if the signal inverter is activated by the PINT_INV_SET register. The inverter can be enabled on an individual bit-by-bit basis. Every bit in the PINT_INV_SET and PINT_INV_CLR register pair represents a pin signal.

The pin interrupt pin state registers enable the service routine to read the current state of the pin without reading from GPIO space. If there was an edge-sensitive interrupt, the service routine can check whether the state of the pin is still high or turned low.

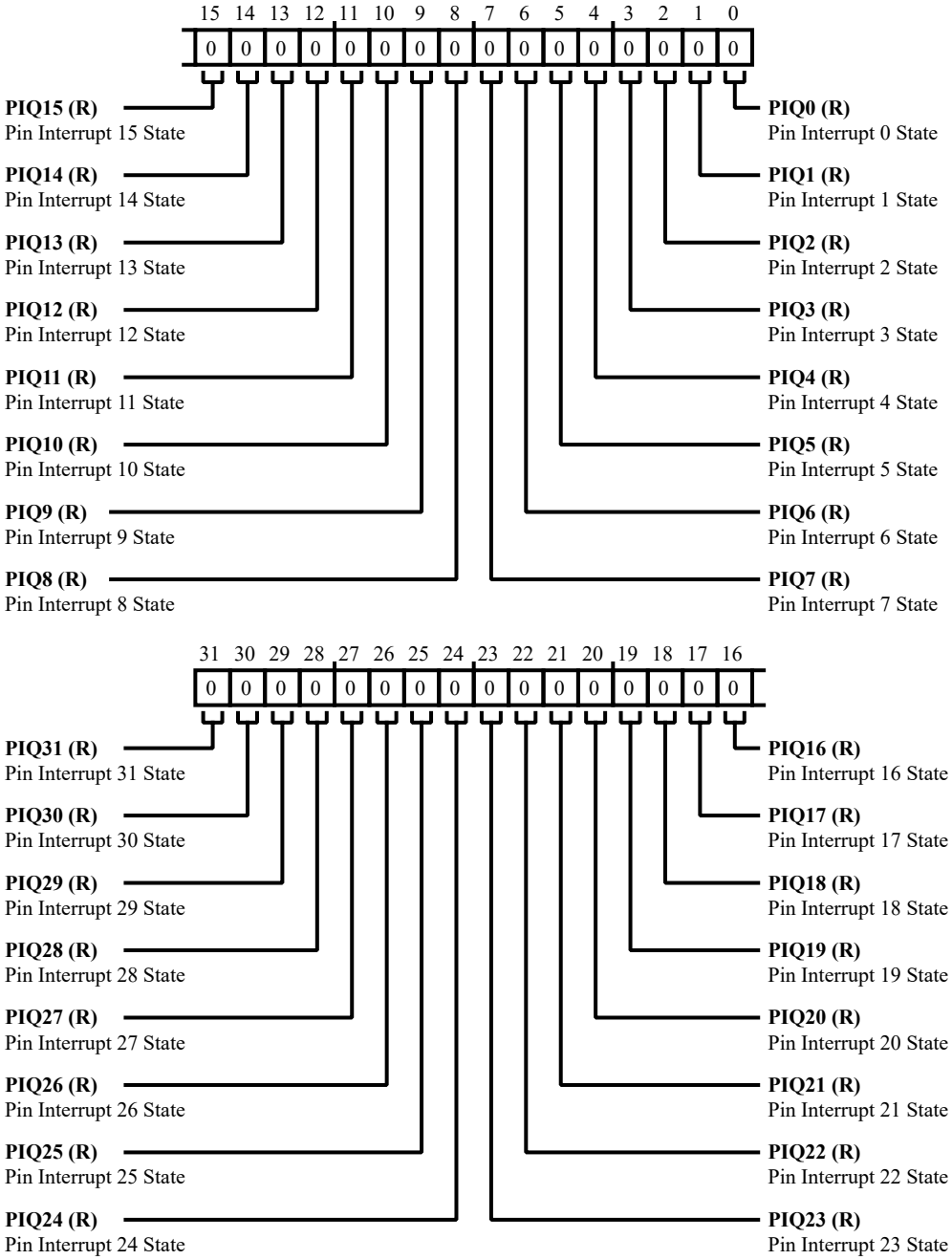


Figure 13-34: PINT_PINSTATE Register Diagram

Table 13-38: PINT_PINSTATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	PIQ31	Pin Interrupt 31 State. A read of the `PINT_PINSTATE.PIQ31` bit returns the pin state.

Table 13-38: PINT_PINSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/NW)	PIQ30	Pin Interrupt 30 State. A read of the PINT_PINSTATE.PIQ30 bit returns the pin state.
29 (R/NW)	PIQ29	Pin Interrupt 29 State. A read of the PINT_PINSTATE.PIQ29 bit returns the pin state.
28 (R/NW)	PIQ28	Pin Interrupt 28 State. A read of the PINT_PINSTATE.PIQ28 bit returns the pin state.
27 (R/NW)	PIQ27	Pin Interrupt 27 State. A read of the PINT_PINSTATE.PIQ27 bit returns the pin state.
26 (R/NW)	PIQ26	Pin Interrupt 26 State. A read of the PINT_PINSTATE.PIQ26 bit returns the pin state.
25 (R/NW)	PIQ25	Pin Interrupt 25 State. A read of the PINT_PINSTATE.PIQ25 bit returns the pin state.
24 (R/NW)	PIQ24	Pin Interrupt 24 State. A read of the PINT_PINSTATE.PIQ24 bit returns the pin state.
23 (R/NW)	PIQ23	Pin Interrupt 23 State. A read of the PINT_PINSTATE.PIQ23 bit returns the pin state.
22 (R/NW)	PIQ22	Pin Interrupt 22 State. A read of the PINT_PINSTATE.PIQ22 bit returns the pin state.
21 (R/NW)	PIQ21	Pin Interrupt 21 State. A read of the PINT_PINSTATE.PIQ21 bit returns the pin state.
20 (R/NW)	PIQ20	Pin Interrupt 20 State. A read of the PINT_PINSTATE.PIQ20 bit returns the pin state.
19 (R/NW)	PIQ19	Pin Interrupt 19 State. A read of the PINT_PINSTATE.PIQ19 bit returns the pin state.
18 (R/NW)	PIQ18	Pin Interrupt 18 State. A read of the PINT_PINSTATE.PIQ18 bit returns the pin state.
17 (R/NW)	PIQ17	Pin Interrupt 17 State. A read of the PINT_PINSTATE.PIQ17 bit returns the pin state.
16 (R/NW)	PIQ16	Pin Interrupt 16 State. A read of the PINT_PINSTATE.PIQ16 bit returns the pin state.
15 (R/NW)	PIQ15	Pin Interrupt 15 State. A read of the PINT_PINSTATE.PIQ15 bit returns the pin state.

Table 13-38: PINT_PINSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	PIQ14	Pin Interrupt 14 State. A read of the PINT_PINSTATE.PIQ14 bit returns the pin state.
13 (R/NW)	PIQ13	Pin Interrupt 13 State. A read of the PINT_PINSTATE.PIQ13 bit returns the pin state.
12 (R/NW)	PIQ12	Pin Interrupt 12 State. A read of the PINT_PINSTATE.PIQ12 bit returns the pin state.
11 (R/NW)	PIQ11	Pin Interrupt 11 State. A read of the PINT_PINSTATE.PIQ11 bit returns the pin state.
10 (R/NW)	PIQ10	Pin Interrupt 10 State. A read of the PINT_PINSTATE.PIQ10 bit returns the pin state.
9 (R/NW)	PIQ9	Pin Interrupt 9 State. A read of the PINT_PINSTATE.PIQ9 bit returns the pin state.
8 (R/NW)	PIQ8	Pin Interrupt 8 State. A read of the PINT_PINSTATE.PIQ8 bit returns the pin state.
7 (R/NW)	PIQ7	Pin Interrupt 7 State. A read of the PINT_PINSTATE.PIQ7 bit returns the pin state.
6 (R/NW)	PIQ6	Pin Interrupt 6 State. A read of the PINT_PINSTATE.PIQ6 bit returns the pin state.
5 (R/NW)	PIQ5	Pin Interrupt 5 State. A read of the PINT_PINSTATE.PIQ5 bit returns the pin state.
4 (R/NW)	PIQ4	Pin Interrupt 4 State. A read of the PINT_PINSTATE.PIQ4 bit returns the pin state.
3 (R/NW)	PIQ3	Pin Interrupt 3 State. A read of the PINT_PINSTATE.PIQ3 bit returns the pin state.
2 (R/NW)	PIQ2	Pin Interrupt 2 State. A read of the PINT_PINSTATE.PIQ2 bit returns the pin state.
1 (R/NW)	PIQ1	Pin Interrupt 1 State. A read of the PINT_PINSTATE.PIQ1 bit returns the pin state.
0 (R/NW)	PIQ0	Pin Interrupt 0 State. A read of the PINT_PINSTATE.PIQ0 bit returns the pin state.

PINT Request Register

The PINT_REQ register indicates the interrupt request status for pin interrupts. When set, an interrupt request is pending. When cleared, there is no interrupt request pending.

Both the PINT_REQ and PINT_LATCH registers indicate whether an interrupt request is latched on the respective pin. The PINT_LATCH register is a latch that operates regardless of the interrupt masks. Bits of the PINT_REQ register depend on the mask register. The PINT_REQ register is a logical AND of the PINT_LATCH register and the interrupt mask.

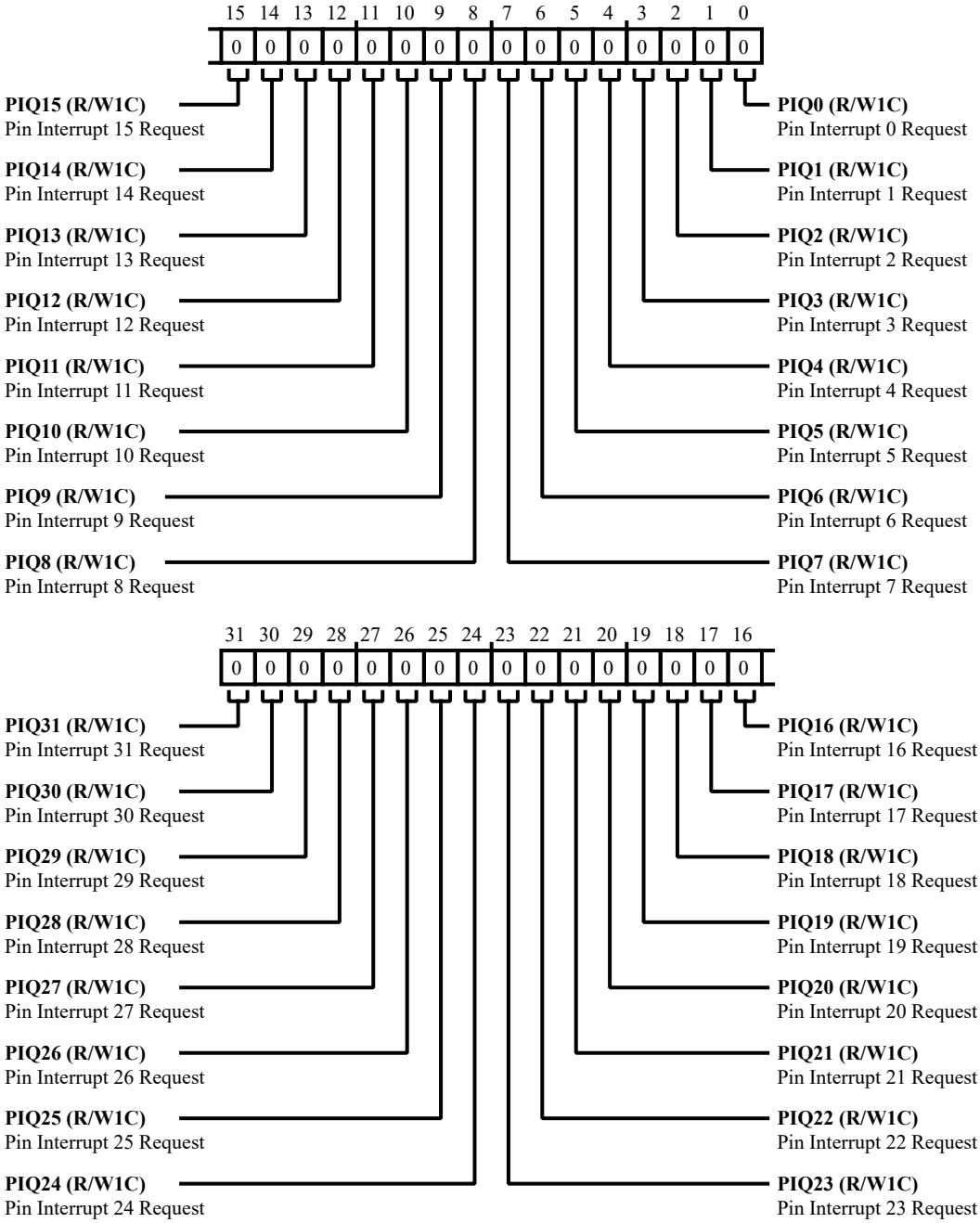


Figure 13-35: PINT_REQ Register Diagram

Table 13-39: PINT_REQ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Request. If the `PINT_REQ.PIQ31` bit is set, a request is pending.

Table 13-39: PINT_REQ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W1C)	PIQ30	Pin Interrupt 30 Request. If the PINT_REQ.PIQ30 bit is set, a request is pending.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Request. If the PINT_REQ.PIQ29 bit is set, a request is pending.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Request. If the PINT_REQ.PIQ28 bit is set, a request is pending.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Request. If the PINT_REQ.PIQ27 bit is set, a request is pending.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Request. If the PINT_REQ.PIQ26 bit is set, a request is pending.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Request. If the PINT_REQ.PIQ25 bit is set, a request is pending.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Request. If the PINT_REQ.PIQ24 bit is set, a request is pending.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Request. If the PINT_REQ.PIQ23 bit is set, a request is pending.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Request. If the PINT_REQ.PIQ22 bit is set, a request is pending.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Request. If the PINT_REQ.PIQ21 bit is set, a request is pending.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Request. If the PINT_REQ.PIQ20 bit is set, a request is pending.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Request. If the PINT_REQ.PIQ19 bit is set, a request is pending.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Request. If the PINT_REQ.PIQ18 bit is set, a request is pending.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Request. If the PINT_REQ.PIQ17 bit is set, a request is pending.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Request. If the PINT_REQ.PIQ16 bit is set, a request is pending.
15 (R/W1C)	PIQ15	Pin Interrupt 15 Request. If the PINT_REQ.PIQ15 bit is set, a request is pending.

Table 13-39: PINT_REQ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	PIQ14	Pin Interrupt 14 Request. If the `PINT_REQ.PIQ14` bit is set, a request is pending.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Request. If the `PINT_REQ.PIQ13` bit is set, a request is pending.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Request. If the `PINT_REQ.PIQ12` bit is set, a request is pending.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Request. If the `PINT_REQ.PIQ11` bit is set, a request is pending.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Request. If the `PINT_REQ.PIQ10` bit is set, a request is pending.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Request. If the `PINT_REQ.PIQ9` bit is set, a request is pending.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Request. If the `PINT_REQ.PIQ8` bit is set, a request is pending.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Request. If the `PINT_REQ.PIQ7` bit is set, a request is pending.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Request. If the `PINT_REQ.PIQ6` bit is set, a request is pending.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Request. If the `PINT_REQ.PIQ5` bit is set, a request is pending.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Request. If the `PINT_REQ.PIQ4` bit is set, a request is pending.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Request. If the `PINT_REQ.PIQ3` bit is set, a request is pending.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Request. If the `PINT_REQ.PIQ2` bit is set, a request is pending.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Request. If the `PINT_REQ.PIQ1` bit is set, a request is pending.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Request. If the `PINT_REQ.PIQ0` bit is set, a request is pending.

ADSP-2184x PADS Register Descriptions

Pads Controller (PADS) contains the following registers.

Table 13-40: ADSP-2184x PADS Register List

Name	Description
PADS_DAI0_0_DS	DAI0 Pins 1 to 8 Drive Strength Control Register
PADS_DAI0_1_DS	DAI0 Pins 9 to 16 Drive Strength Control Register
PADS_DAI0_2_DS	DAI0 Pins 17 to 20 Drive Strength Control Register
PADS_DAI0_IE	DAI0 Port Input Enable Control Register
PADS_DAI0_PE	DAI0 Pull Enable Register
PADS_DAI0_PS	DAI0 Pull Selector Register
PADS_DAI0_SL	DAI0 Slew Rate Control Register
PADS_DAI0_SPU	DAI0 Strong Pull Up Control Register
PADS_DAI1_0_DS	DAI1 Pins 1 to 8 Drive Strength Control Register
PADS_DAI1_1_DS	DAI1 Pins 9 to 16 Drive Strength Control Register
PADS_DAI1_2_DS	DAI1 Pins 17 to 20 Drive Strength Control Register
PADS_DAI1_IE	DAI1 Port Input Enable Control Register
PADS_DAI1_PE	DAI1 Pull Enable Register
PADS_DAI1_PS	DAI1 Pull Selector Register
PADS_DAI1_SL	DAI1 Slew Rate Control Register
PADS_DAI1_SPU	DAI1 Strong Pull up Control Register
PADS_NONPORTS_CTL	Non-GPIO Control Register
PADS_NONPORTS_DS	Non-GPIO Drive Strength Register
PADS_NONPORTS_DS_1	EMAC Non-GPIO Drive Strength Register
PADS_OTPC_BOOT_STAT	OTPC Boot Status Register
PADS_PCFG0	Peripheral PAD Configuration 0 Register
PADS_PCFG1	Peripheral Configuration 1 Register
PADS_PORTA0_DS	PORTA Pins 0 to 7 Drive Strength Control Register
PADS_PORTA1_DS	PORTA Pins 8 to 15 Drive Strength Control Register
PADS_PORTB0_DS	PORTB Pins 0 to 7 Drive Strength Control Register
PADS_PORTB1_DS	PORTB Pins 8 to 15 Drive Strength Control Register
PADS_PORTC0_DS	PORTC Pins 0 to 7 Drive Strength Control Register
PADS_PORTC1_DS	PORTC Pins 8 to 15 Drive Strength Control Register
PADS_PORTD0_DS	PORTD Pins 0 to 7 Drive Strength Control Register

Table 13-40: ADSP-2184x PADS Register List (Continued)

Name	Description
PADS_PORTD1_DS	PORTD Pins 8 to 15 Drive Strength Control Register
PADS_PORTE0_DS	PORTE Pins 0 to 7 Drive Strength Control Register
PADS_PORTE1_DS	PORTE Pins 8 to 15 Drive Strength Control Register
PADS_PORTF0_DS	PORTF Pins 0 to 7 Drive Strength Control Register
PADS_PORTF1_DS	PORTF Pins 8 to 15 Drive Strength Control Register
PADS_PORTG0_DS	PORTG Pins 0 to 7 Drive Strength Control Register
PADS_PORTG1_DS	PORTG Pins 8 to 15 Drive Strength Control Register
PADS_PORTH0_DS	PORTH Pins 0 to 7 Drive Strength Control Register
PADS_PORTH1_DS	PORTH Pins 8 and 9 Drive Strength Control Register
PADS_PORT_AB_PE	PORTA and PORTB Pull Enable Register
PADS_PORT_AB_PS	PORTA and PORTB Pull Selector Register
PADS_PORT_AB_SL	PORTA and PORTB Slew Rate Control Register
PADS_PORT_AB_SPU	PORTA and PORTB Strong Pull Up Control Register
PADS_PORT_CD_PE	PORTC and PORTD Pull Enable Register
PADS_PORT_CD_PS	PORTC and PORTD Pull Selector Register
PADS_PORT_CD_SL	PORTC and PORTD Slew Rate Control Register
PADS_PORT_CD_SPU	PORTC and PORTD Strong Pull Up Control Register
PADS_PORT_EF_PE	PORTE and PORTF Pull Enable Register
PADS_PORT_EF_PS	PORTE and PORTF Pull Selector Register
PADS_PORT_EF_SL	PORTE and PORTF Slew Rate Control Register
PADS_PORT_EF_SPU	PORTE and PORTF Strong Pull Up Control Register
PADS_PORT_GH_PE	PORTG and PORTH Pull Enable Register
PADS_PORT_GH_PS	PORTG and PORTH Pull Selector Register
PADS_PORT_GH_SL	PORTG and PORTH Slew Rate Control Register
PADS_PORT_GH_SPU	PORTG and PORTH Strong Pull Up Control Register
PADS_STAT	Status Register

DAI0 Pins 1 to 8 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

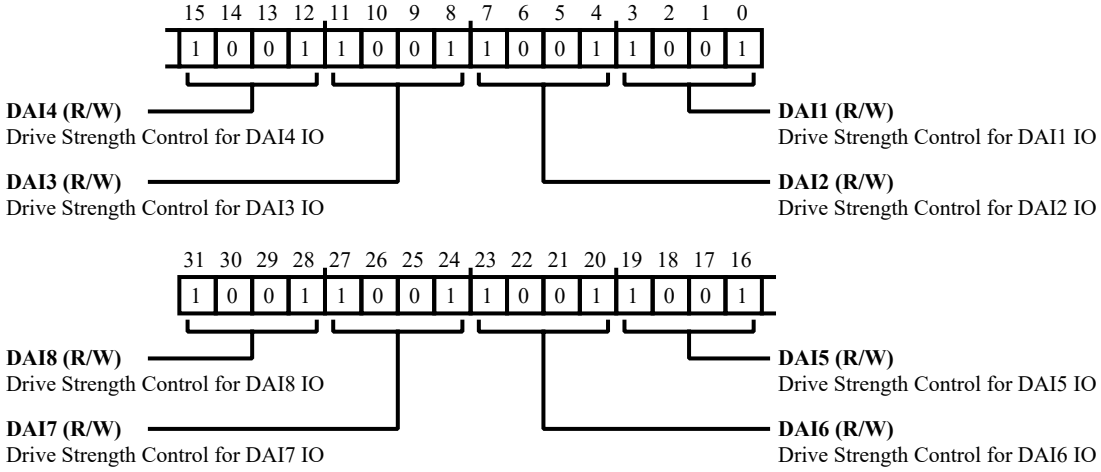


Figure 13-36: PADS_DAI0_0_DS Register Diagram

Table 13-41: PADS_DAI0_0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	DAI8	Drive Strength Control for DAI8 IO.
27:24 (R/W)	DAI7	Drive Strength Control for DAI7 IO.
23:20 (R/W)	DAI6	Drive Strength Control for DAI6 IO.
19:16 (R/W)	DAI5	Drive Strength Control for DAI5 IO.
15:12 (R/W)	DAI4	Drive Strength Control for DAI4 IO.
11:8 (R/W)	DAI3	Drive Strength Control for DAI3 IO.
7:4 (R/W)	DAI2	Drive Strength Control for DAI2 IO.
3:0 (R/W)	DAI1	Drive Strength Control for DAI1 IO.

DAI0 Pins 9 to 16 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

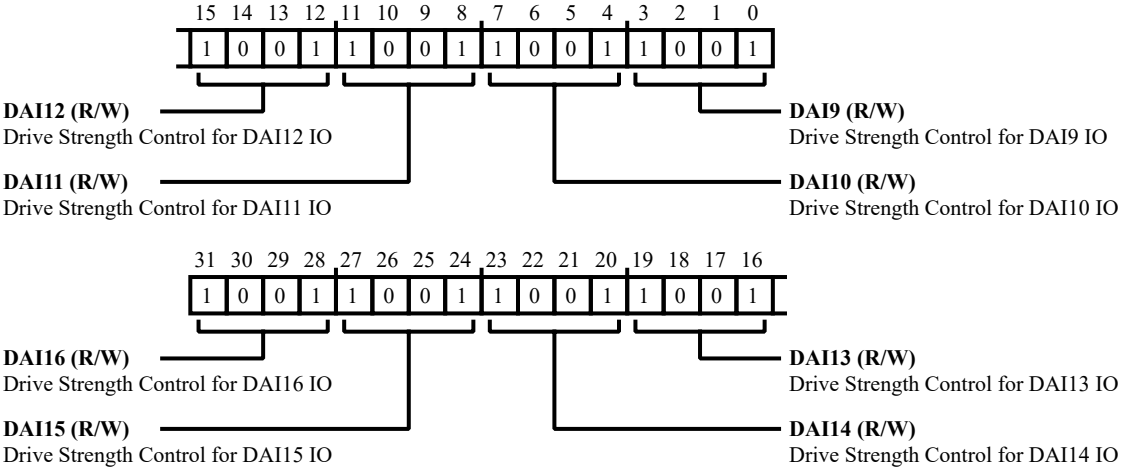


Figure 13-37: PADS_DAI0_1_DS Register Diagram

Table 13-42: PADS_DAI0_1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	DAI16	Drive Strength Control for DAI16 IO.
27:24 (R/W)	DAI15	Drive Strength Control for DAI15 IO.
23:20 (R/W)	DAI14	Drive Strength Control for DAI14 IO.
19:16 (R/W)	DAI13	Drive Strength Control for DAI13 IO.
15:12 (R/W)	DAI12	Drive Strength Control for DAI12 IO.
11:8 (R/W)	DAI11	Drive Strength Control for DAI11 IO.
7:4 (R/W)	DAI10	Drive Strength Control for DAI10 IO.
3:0 (R/W)	DAI9	Drive Strength Control for DAI9 IO.

DAI0 Pins 17 to 20 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

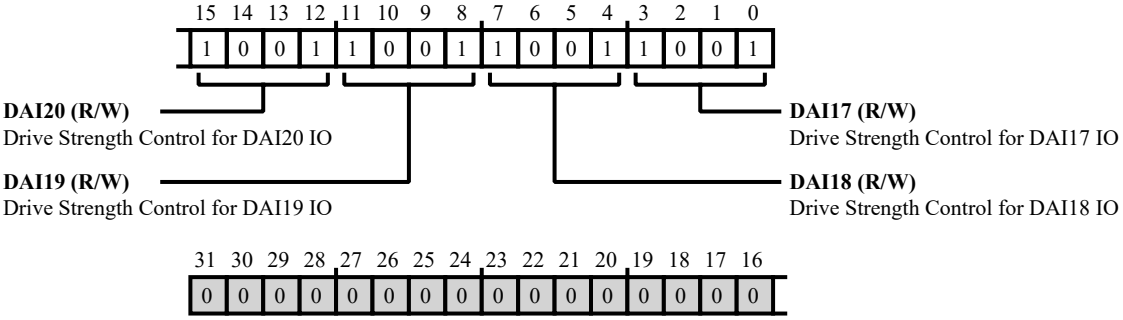


Figure 13-38: PADS_DAI0_2_DS Register Diagram

Table 13-43: PADS_DAI0_2_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:12 (R/W)	DAI20	Drive Strength Control for DAI20 IO.
11:8 (R/W)	DAI19	Drive Strength Control for DAI19 IO.
7:4 (R/W)	DAI18	Drive Strength Control for DAI18 IO.
3:0 (R/W)	DAI17	Drive Strength Control for DAI17 IO.

DAI0 Port Input Enable Control Register

The PADS_DAI0_IE register configures input enable control of the DAI0 (20 pins) pads. If =0 implies input buffer disable and if =1 implies enable.

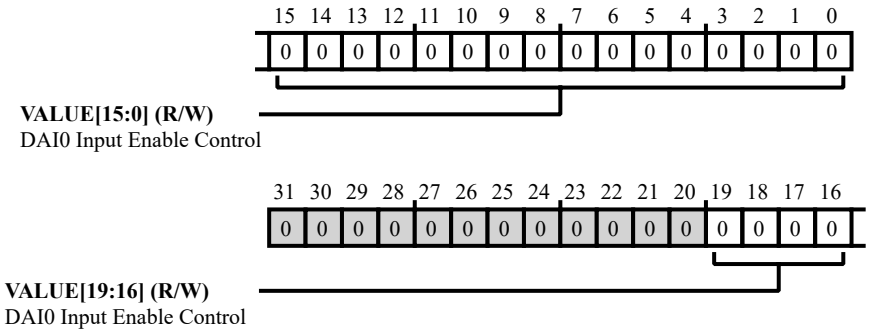


Figure 13-39: PADS_DAI0_IE Register Diagram

Table 13-44: PADS_DAI0_IE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	VALUE	DAI0 Input Enable Control.

DAI0 Pull Enable Register

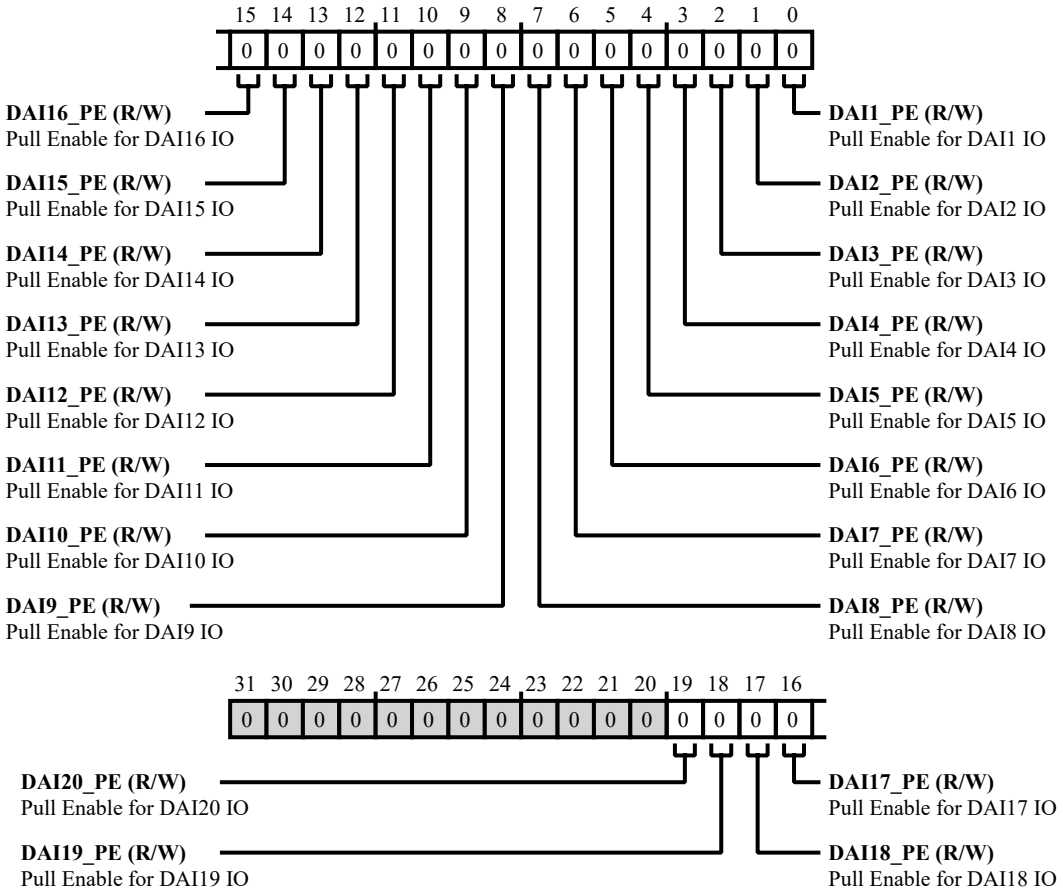


Figure 13-40: PADS_DAI0_PE Register Diagram

Table 13-45: PADS_DAI0_PE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_PE	Pull Enable for DAI20 IO.
18 (R/W)	DAI19_PE	Pull Enable for DAI19 IO.
17 (R/W)	DAI18_PE	Pull Enable for DAI18 IO.
16 (R/W)	DAI17_PE	Pull Enable for DAI17 IO.

Table 13-45: PADS_DAI0_PE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_PE	Pull Enable for DAI16 IO.
14 (R/W)	DAI15_PE	Pull Enable for DAI15 IO.
13 (R/W)	DAI14_PE	Pull Enable for DAI14 IO.
12 (R/W)	DAI13_PE	Pull Enable for DAI13 IO.
11 (R/W)	DAI12_PE	Pull Enable for DAI12 IO.
10 (R/W)	DAI11_PE	Pull Enable for DAI11 IO.
9 (R/W)	DAI10_PE	Pull Enable for DAI10 IO.
8 (R/W)	DAI9_PE	Pull Enable for DAI9 IO.
7 (R/W)	DAI8_PE	Pull Enable for DAI8 IO.
6 (R/W)	DAI7_PE	Pull Enable for DAI7 IO.
5 (R/W)	DAI6_PE	Pull Enable for DAI6 IO.
4 (R/W)	DAI5_PE	Pull Enable for DAI5 IO.
3 (R/W)	DAI4_PE	Pull Enable for DAI4 IO.
2 (R/W)	DAI3_PE	Pull Enable for DAI3 IO.
1 (R/W)	DAI2_PE	Pull Enable for DAI2 IO.
0 (R/W)	DAI1_PE	Pull Enable for DAI1 IO.

DAI0 Pull Selector Register

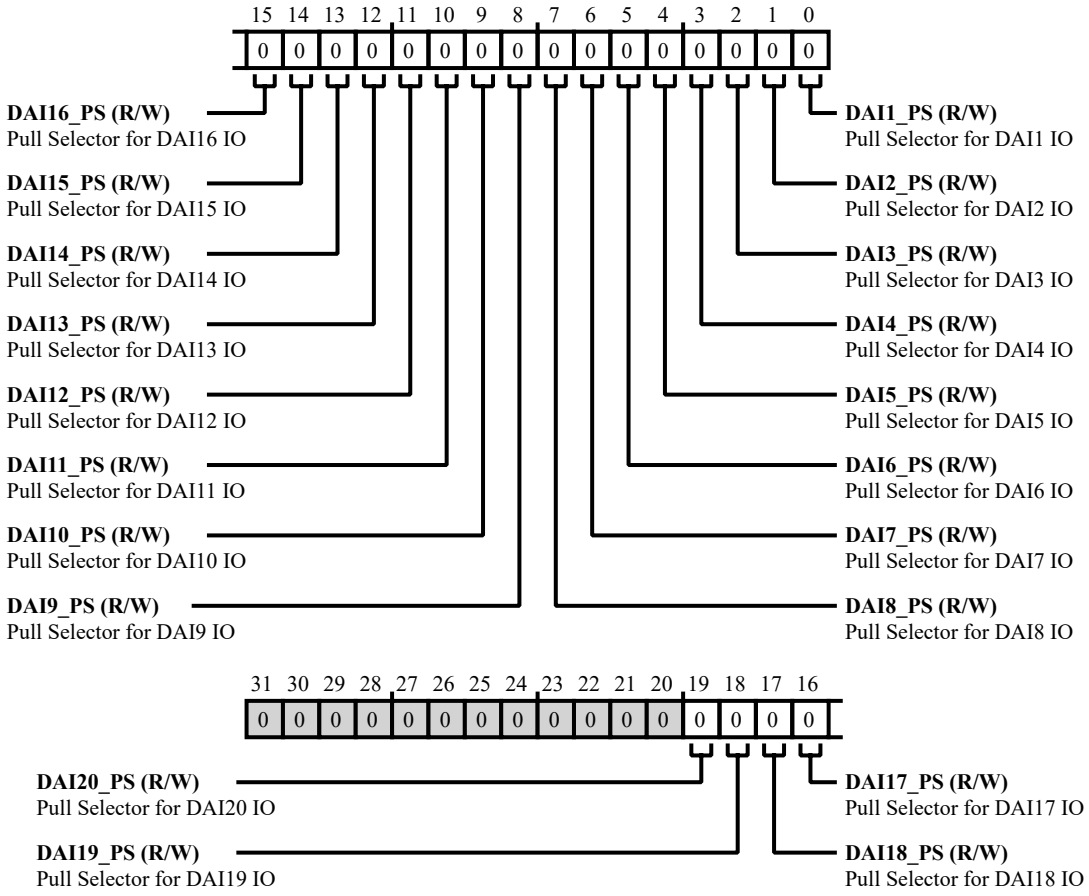


Figure 13-41: PADS_DAI0_PS Register Diagram

Table 13-46: PADS_DAI0_PS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_PS	Pull Selector for DAI20 IO.
18 (R/W)	DAI19_PS	Pull Selector for DAI19 IO.
17 (R/W)	DAI18_PS	Pull Selector for DAI18 IO.
16 (R/W)	DAI17_PS	Pull Selector for DAI17 IO.

Table 13-46: PADS_DAI0_PS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_PS	Pull Selector for DAI16 IO.
14 (R/W)	DAI15_PS	Pull Selector for DAI15 IO.
13 (R/W)	DAI14_PS	Pull Selector for DAI14 IO.
12 (R/W)	DAI13_PS	Pull Selector for DAI13 IO.
11 (R/W)	DAI12_PS	Pull Selector for DAI12 IO.
10 (R/W)	DAI11_PS	Pull Selector for DAI11 IO.
9 (R/W)	DAI10_PS	Pull Selector for DAI10 IO.
8 (R/W)	DAI9_PS	Pull Selector for DAI9 IO.
7 (R/W)	DAI8_PS	Pull Selector for DAI8 IO.
6 (R/W)	DAI7_PS	Pull Selector for DAI7 IO.
5 (R/W)	DAI6_PS	Pull Selector for DAI6 IO.
4 (R/W)	DAI5_PS	Pull Selector for DAI5 IO.
3 (R/W)	DAI4_PS	Pull Selector for DAI4 IO.
2 (R/W)	DAI3_PS	Pull Selector for DAI3 IO.
1 (R/W)	DAI2_PS	Pull Selector for DAI2 IO.
0 (R/W)	DAI1_PS	Pull Selector for DAI1 IO.

DAI0 Slew Rate Control Register

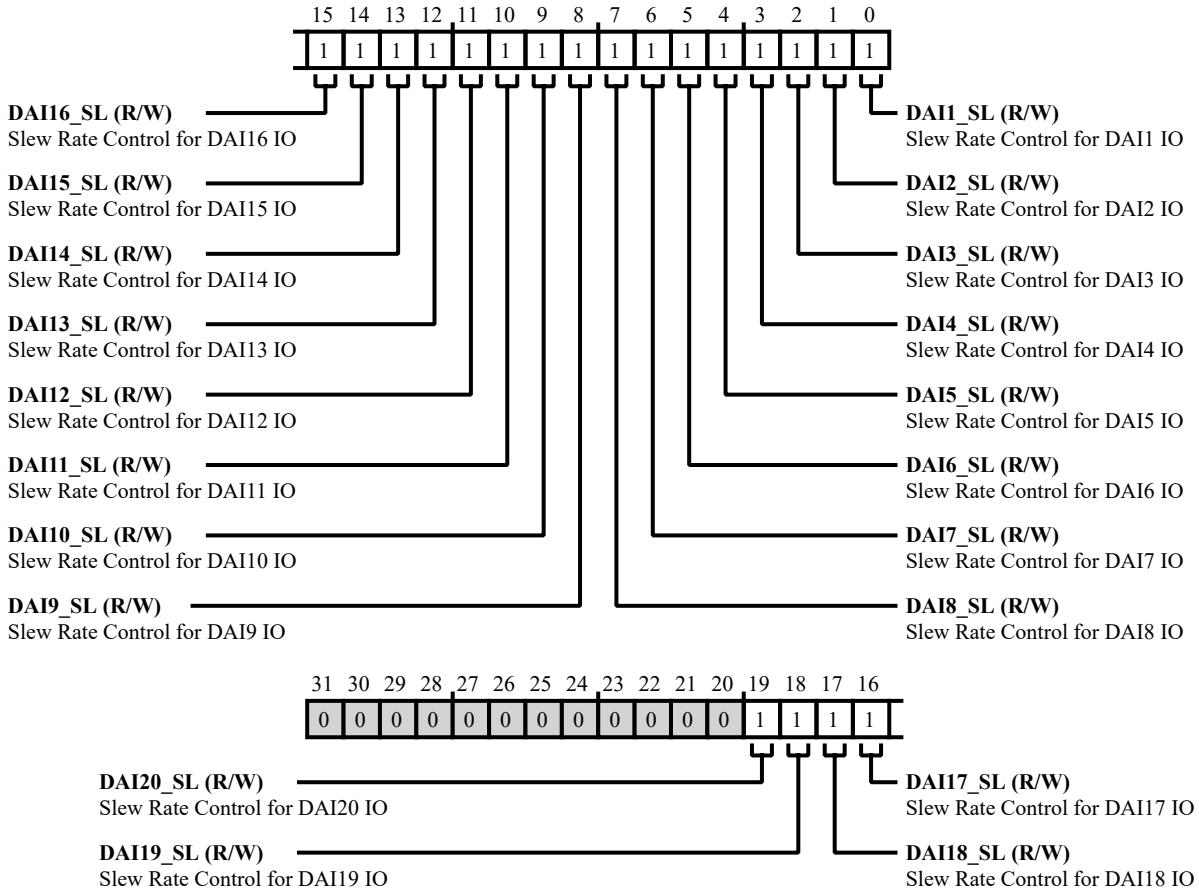


Figure 13-42: PADS_DAI0_SL Register Diagram

Table 13-47: PADS_DAI0_SL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_SL	Slew Rate Control for DAI20 IO.
18 (R/W)	DAI19_SL	Slew Rate Control for DAI19 IO.
17 (R/W)	DAI18_SL	Slew Rate Control for DAI18 IO.
16 (R/W)	DAI17_SL	Slew Rate Control for DAI17 IO.

Table 13-47: PADS_DAI0_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_SL	Slew Rate Control for DAI16 IO.
14 (R/W)	DAI15_SL	Slew Rate Control for DAI15 IO.
13 (R/W)	DAI14_SL	Slew Rate Control for DAI14 IO.
12 (R/W)	DAI13_SL	Slew Rate Control for DAI13 IO.
11 (R/W)	DAI12_SL	Slew Rate Control for DAI12 IO.
10 (R/W)	DAI11_SL	Slew Rate Control for DAI11 IO.
9 (R/W)	DAI10_SL	Slew Rate Control for DAI10 IO.
8 (R/W)	DAI9_SL	Slew Rate Control for DAI9 IO.
7 (R/W)	DAI8_SL	Slew Rate Control for DAI8 IO.
6 (R/W)	DAI7_SL	Slew Rate Control for DAI7 IO.
5 (R/W)	DAI6_SL	Slew Rate Control for DAI6 IO.
4 (R/W)	DAI5_SL	Slew Rate Control for DAI5 IO.
3 (R/W)	DAI4_SL	Slew Rate Control for DAI4 IO.
2 (R/W)	DAI3_SL	Slew Rate Control for DAI3 IO.
1 (R/W)	DAI2_SL	Slew Rate Control for DAI2 IO.
0 (R/W)	DAI1_SL	Slew Rate Control for DAI1 IO.

DAI0 Strong Pull Up Control Register

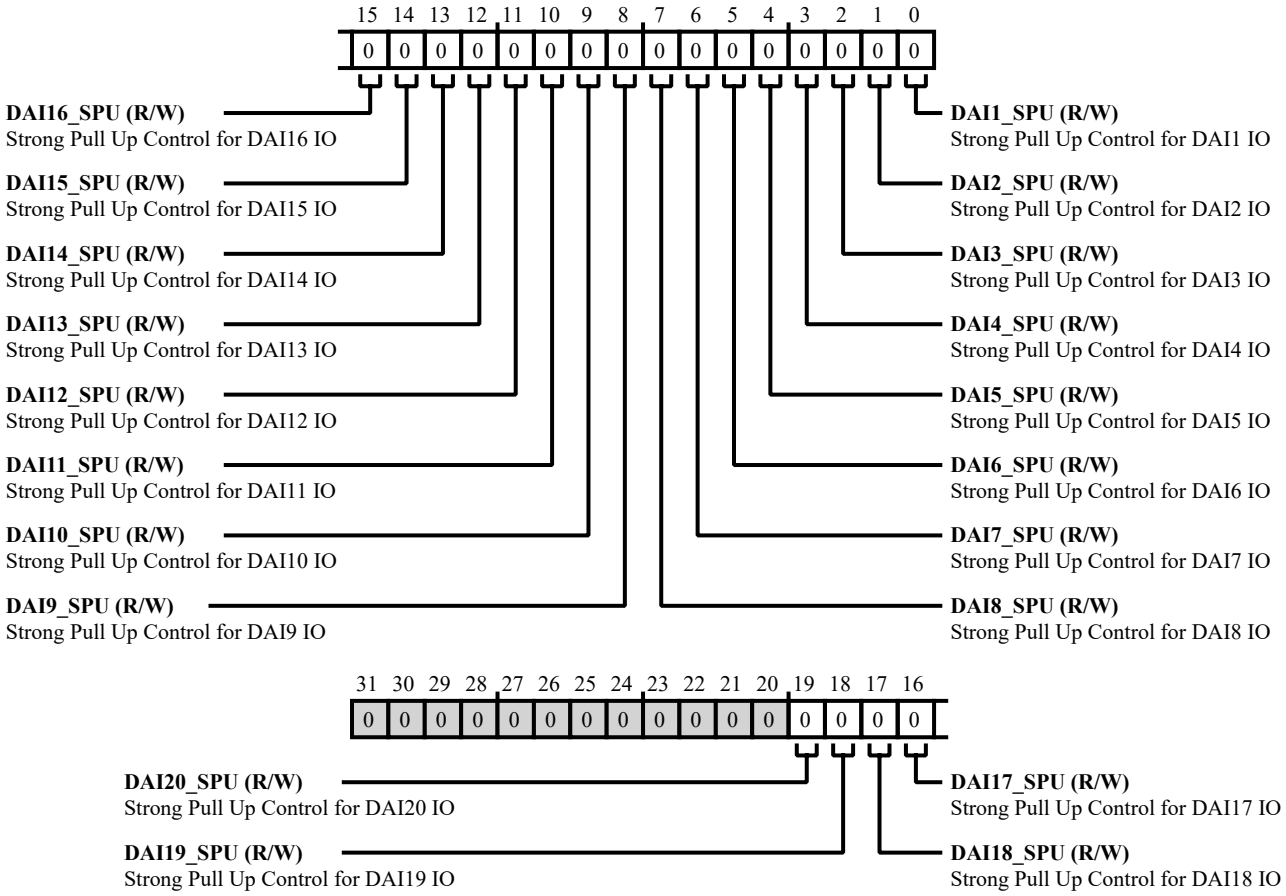


Figure 13-43: PADS_DAI0_SPU Register Diagram

Table 13-48: PADS_DAI0_SPU Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_SPU	Strong Pull Up Control for DAI20 IO.
18 (R/W)	DAI19_SPU	Strong Pull Up Control for DAI19 IO.
17 (R/W)	DAI18_SPU	Strong Pull Up Control for DAI18 IO.
16 (R/W)	DAI17_SPU	Strong Pull Up Control for DAI17 IO.

Table 13-48: PADS_DAI0_SPU Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_SPU	Strong Pull Up Control for DAI16 IO.
14 (R/W)	DAI15_SPU	Strong Pull Up Control for DAI15 IO.
13 (R/W)	DAI14_SPU	Strong Pull Up Control for DAI14 IO.
12 (R/W)	DAI13_SPU	Strong Pull Up Control for DAI13 IO.
11 (R/W)	DAI12_SPU	Strong Pull Up Control for DAI12 IO.
10 (R/W)	DAI11_SPU	Strong Pull Up Control for DAI11 IO.
9 (R/W)	DAI10_SPU	Strong Pull Up Control for DAI10 IO.
8 (R/W)	DAI9_SPU	Strong Pull Up Control for DAI9 IO.
7 (R/W)	DAI8_SPU	Strong Pull Up Control for DAI8 IO.
6 (R/W)	DAI7_SPU	Strong Pull Up Control for DAI7 IO.
5 (R/W)	DAI6_SPU	Strong Pull Up Control for DAI6 IO.
4 (R/W)	DAI5_SPU	Strong Pull Up Control for DAI5 IO.
3 (R/W)	DAI4_SPU	Strong Pull Up Control for DAI4 IO.
2 (R/W)	DAI3_SPU	Strong Pull Up Control for DAI3 IO.
1 (R/W)	DAI2_SPU	Strong Pull Up Control for DAI2 IO.
0 (R/W)	DAI1_SPU	Strong Pull Up Control for DAI1 IO.

DAI1 Pins 1 to 8 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

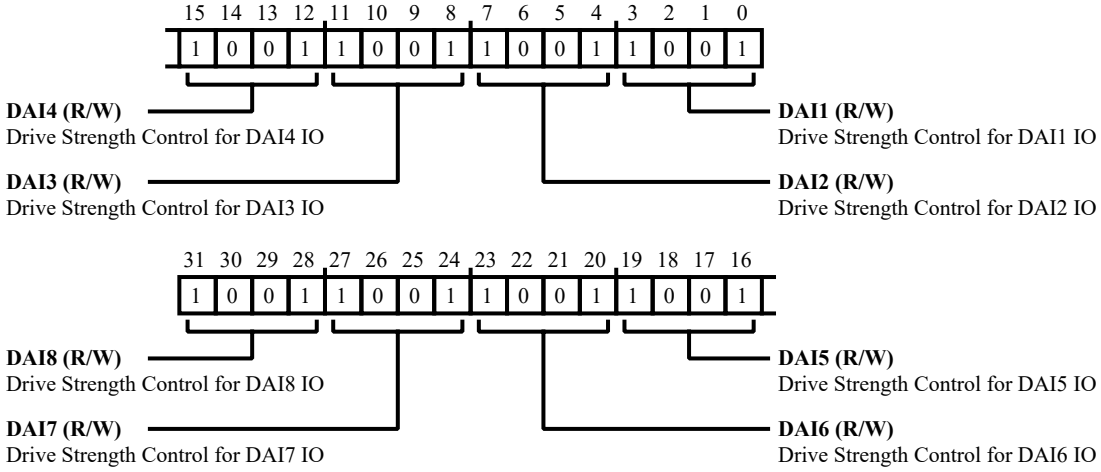


Figure 13-44: PADS_DAI1_0_DS Register Diagram

Table 13-49: PADS_DAI1_0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	DAI8	Drive Strength Control for DAI8 IO.
27:24 (R/W)	DAI7	Drive Strength Control for DAI7 IO.
23:20 (R/W)	DAI6	Drive Strength Control for DAI6 IO.
19:16 (R/W)	DAI5	Drive Strength Control for DAI5 IO.
15:12 (R/W)	DAI4	Drive Strength Control for DAI4 IO.
11:8 (R/W)	DAI3	Drive Strength Control for DAI3 IO.
7:4 (R/W)	DAI2	Drive Strength Control for DAI2 IO.
3:0 (R/W)	DAI1	Drive Strength Control for DAI1 IO.

DAI1 Pins 9 to 16 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

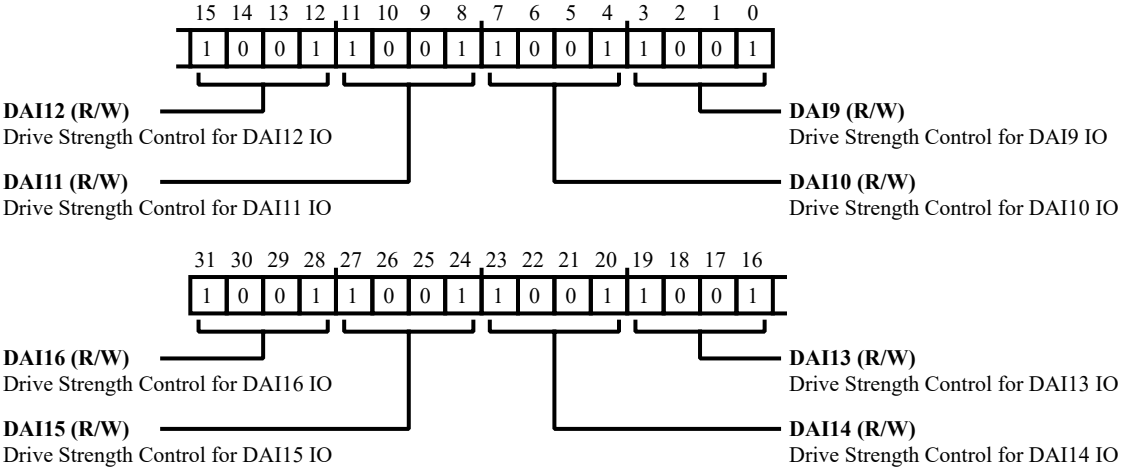


Figure 13-45: PADS_DAI1_1_DS Register Diagram

Table 13-50: PADS_DAI1_1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	DAI16	Drive Strength Control for DAI16 IO.
27:24 (R/W)	DAI15	Drive Strength Control for DAI15 IO.
23:20 (R/W)	DAI14	Drive Strength Control for DAI14 IO.
19:16 (R/W)	DAI13	Drive Strength Control for DAI13 IO.
15:12 (R/W)	DAI12	Drive Strength Control for DAI12 IO.
11:8 (R/W)	DAI11	Drive Strength Control for DAI11 IO.
7:4 (R/W)	DAI10	Drive Strength Control for DAI10 IO.
3:0 (R/W)	DAI9	Drive Strength Control for DAI9 IO.

DAI1 Pins 17 to 20 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

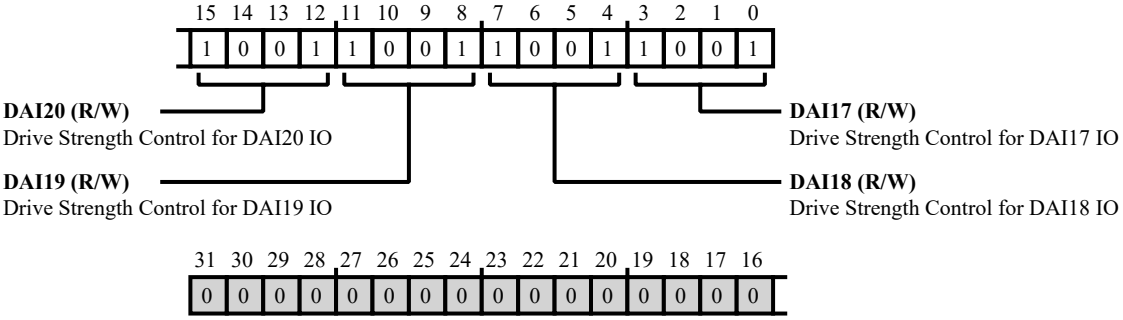


Figure 13-46: PADS_DAI1_2_DS Register Diagram

Table 13-51: PADS_DAI1_2_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:12 (R/W)	DAI20	Drive Strength Control for DAI20 IO.
11:8 (R/W)	DAI19	Drive Strength Control for DAI19 IO.
7:4 (R/W)	DAI18	Drive Strength Control for DAI18 IO.
3:0 (R/W)	DAI17	Drive Strength Control for DAI17 IO.

DAI1 Port Input Enable Control Register

The PADS_DAI1_IE register configures input enable control of the DAI1 (20 pins) pads. If =0 implies input buffer disable and if =1 implies enable.

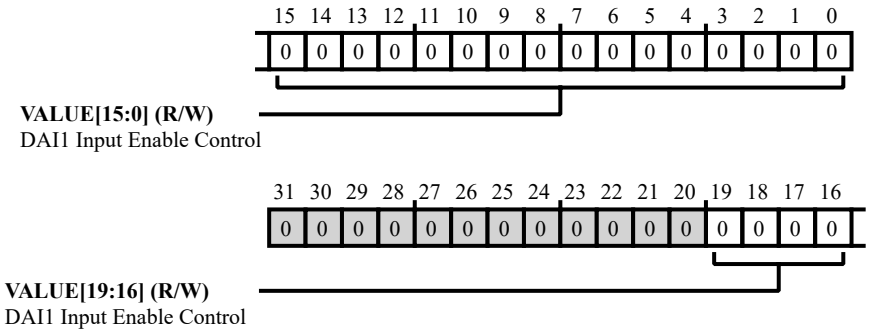


Figure 13-47: PADS_DAI1_IE Register Diagram

Table 13-52: PADS_DAI1_IE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	VALUE	DAI1 Input Enable Control.

DAI1 Pull Enable Register

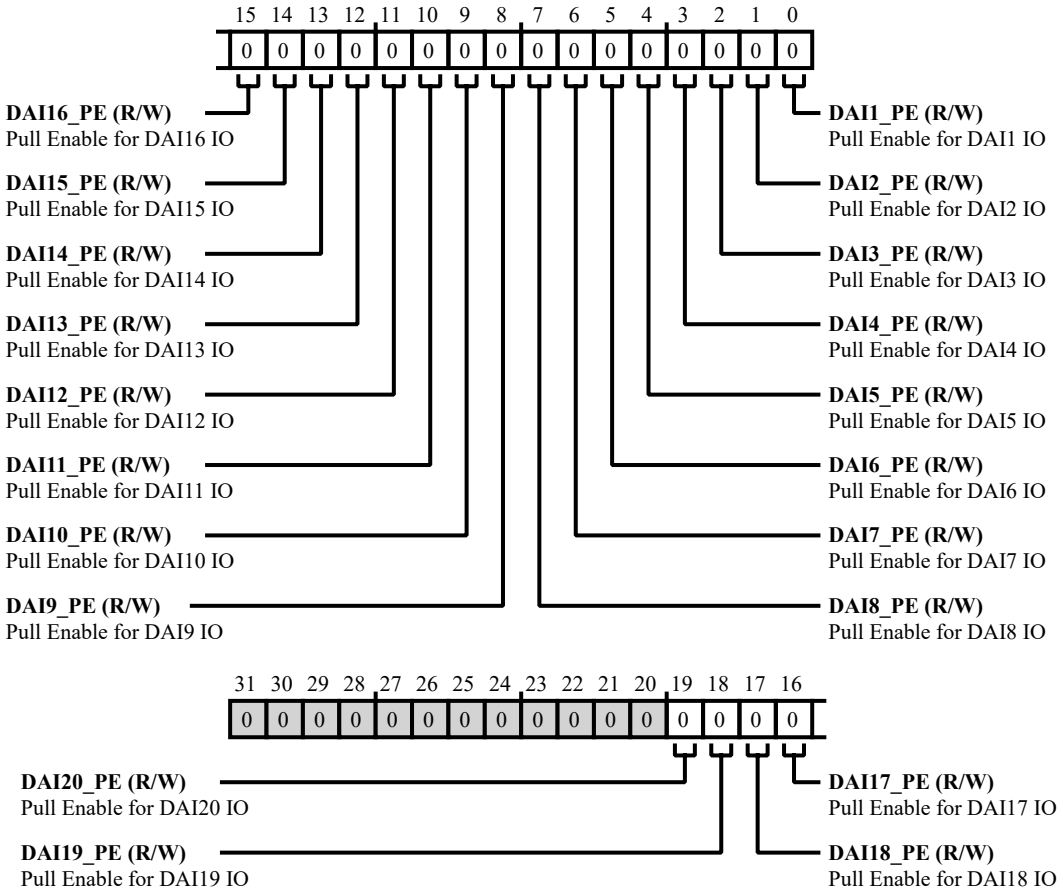


Figure 13-48: PADS_DAI1_PE Register Diagram

Table 13-53: PADS_DAI1_PE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_PE	Pull Enable for DAI20 IO.
18 (R/W)	DAI19_PE	Pull Enable for DAI19 IO.
17 (R/W)	DAI18_PE	Pull Enable for DAI18 IO.
16 (R/W)	DAI17_PE	Pull Enable for DAI17 IO.

Table 13-53: PADS_DAI1_PE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_PE	Pull Enable for DAI16 IO.
14 (R/W)	DAI15_PE	Pull Enable for DAI15 IO.
13 (R/W)	DAI14_PE	Pull Enable for DAI14 IO.
12 (R/W)	DAI13_PE	Pull Enable for DAI13 IO.
11 (R/W)	DAI12_PE	Pull Enable for DAI12 IO.
10 (R/W)	DAI11_PE	Pull Enable for DAI11 IO.
9 (R/W)	DAI10_PE	Pull Enable for DAI10 IO.
8 (R/W)	DAI9_PE	Pull Enable for DAI9 IO.
7 (R/W)	DAI8_PE	Pull Enable for DAI8 IO.
6 (R/W)	DAI7_PE	Pull Enable for DAI7 IO.
5 (R/W)	DAI6_PE	Pull Enable for DAI6 IO.
4 (R/W)	DAI5_PE	Pull Enable for DAI5 IO.
3 (R/W)	DAI4_PE	Pull Enable for DAI4 IO.
2 (R/W)	DAI3_PE	Pull Enable for DAI3 IO.
1 (R/W)	DAI2_PE	Pull Enable for DAI2 IO.
0 (R/W)	DAI1_PE	Pull Enable for DAI1 IO.

DAI1 Pull Selector Register

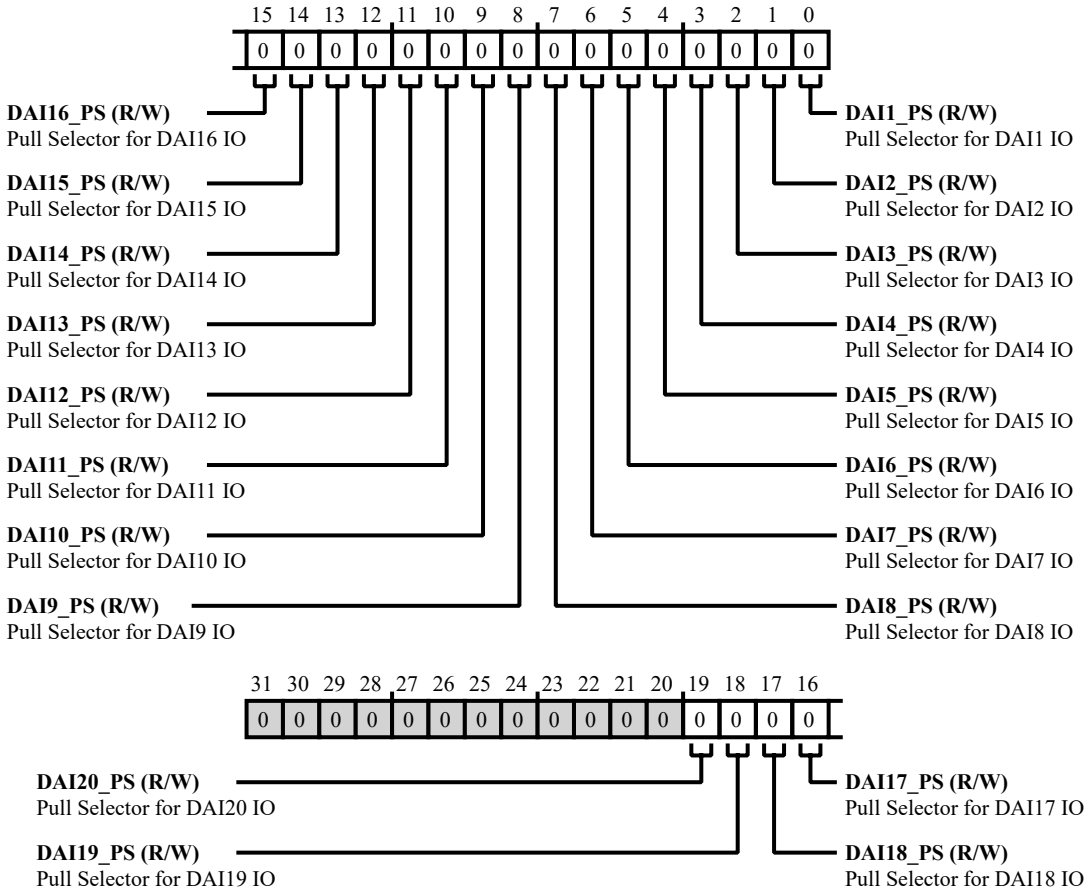


Figure 13-49: PADS_DAI1_PS Register Diagram

Table 13-54: PADS_DAI1_PS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_PS	Pull Selector for DAI20 IO.
18 (R/W)	DAI19_PS	Pull Selector for DAI19 IO.
17 (R/W)	DAI18_PS	Pull Selector for DAI18 IO.
16 (R/W)	DAI17_PS	Pull Selector for DAI17 IO.

Table 13-54: PADS_DAI1_PS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_PS	Pull Selector for DAI16 IO.
14 (R/W)	DAI15_PS	Pull Selector for DAI15 IO.
13 (R/W)	DAI14_PS	Pull Selector for DAI14 IO.
12 (R/W)	DAI13_PS	Pull Selector for DAI13 IO.
11 (R/W)	DAI12_PS	Pull Selector for DAI12 IO.
10 (R/W)	DAI11_PS	Pull Selector for DAI11 IO.
9 (R/W)	DAI10_PS	Pull Selector for DAI10 IO.
8 (R/W)	DAI9_PS	Pull Selector for DAI9 IO.
7 (R/W)	DAI8_PS	Pull Selector for DAI8 IO.
6 (R/W)	DAI7_PS	Pull Selector for DAI7 IO.
5 (R/W)	DAI6_PS	Pull Selector for DAI6 IO.
4 (R/W)	DAI5_PS	Pull Selector for DAI5 IO.
3 (R/W)	DAI4_PS	Pull Selector for DAI4 IO.
2 (R/W)	DAI3_PS	Pull Selector for DAI3 IO.
1 (R/W)	DAI2_PS	Pull Selector for DAI2 IO.
0 (R/W)	DAI1_PS	Pull Selector for DAI1 IO.

DAI1 Slew Rate Control Register

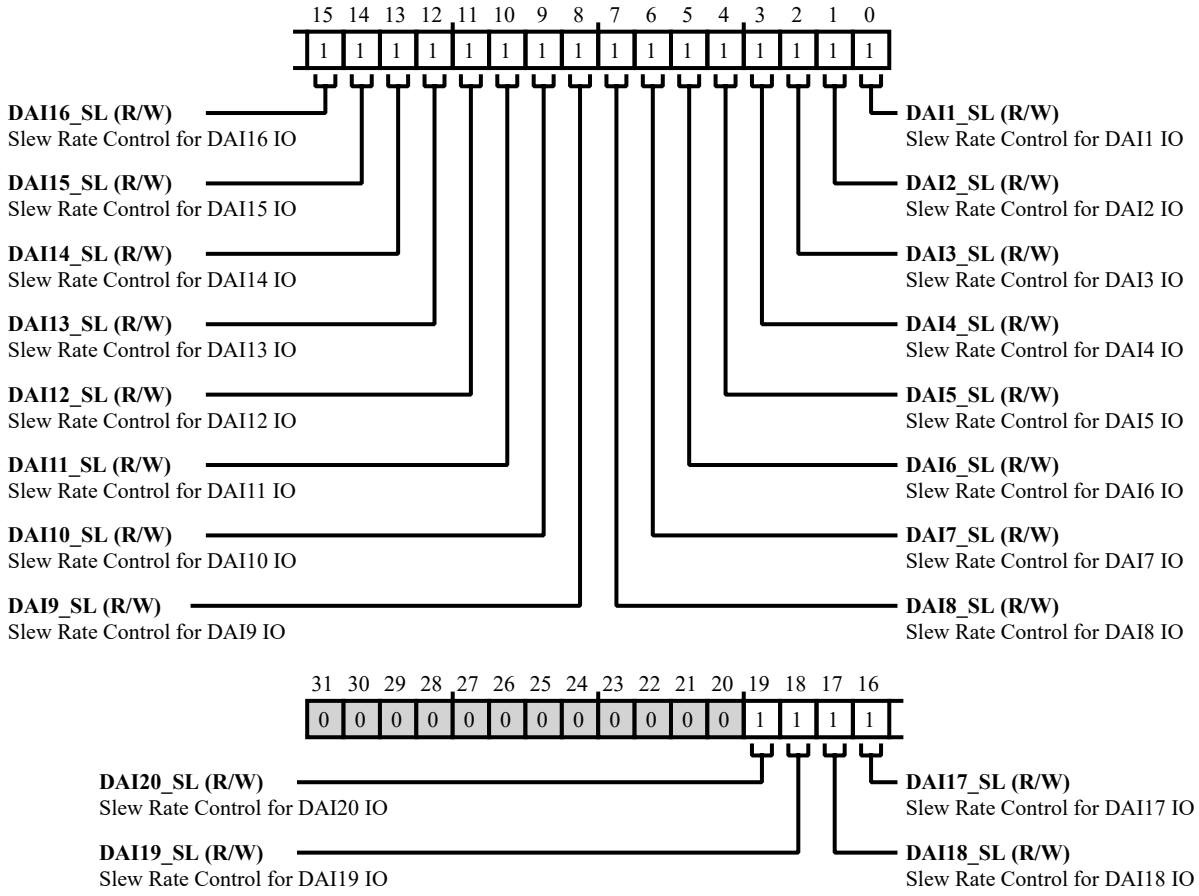


Figure 13-50: PADS_DAI1_SL Register Diagram

Table 13-55: PADS_DAI1_SL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_SL	Slew Rate Control for DAI20 IO.
18 (R/W)	DAI19_SL	Slew Rate Control for DAI19 IO.
17 (R/W)	DAI18_SL	Slew Rate Control for DAI18 IO.
16 (R/W)	DAI17_SL	Slew Rate Control for DAI17 IO.

Table 13-55: PADS_DAI1_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_SL	Slew Rate Control for DAI16 IO.
14 (R/W)	DAI15_SL	Slew Rate Control for DAI15 IO.
13 (R/W)	DAI14_SL	Slew Rate Control for DAI14 IO.
12 (R/W)	DAI13_SL	Slew Rate Control for DAI13 IO.
11 (R/W)	DAI12_SL	Slew Rate Control for DAI12 IO.
10 (R/W)	DAI11_SL	Slew Rate Control for DAI11 IO.
9 (R/W)	DAI10_SL	Slew Rate Control for DAI10 IO.
8 (R/W)	DAI9_SL	Slew Rate Control for DAI9 IO.
7 (R/W)	DAI8_SL	Slew Rate Control for DAI8 IO.
6 (R/W)	DAI7_SL	Slew Rate Control for DAI7 IO.
5 (R/W)	DAI6_SL	Slew Rate Control for DAI6 IO.
4 (R/W)	DAI5_SL	Slew Rate Control for DAI5 IO.
3 (R/W)	DAI4_SL	Slew Rate Control for DAI4 IO.
2 (R/W)	DAI3_SL	Slew Rate Control for DAI3 IO.
1 (R/W)	DAI2_SL	Slew Rate Control for DAI2 IO.
0 (R/W)	DAI1_SL	Slew Rate Control for DAI1 IO.

DAI1 Strong Pull up Control Register

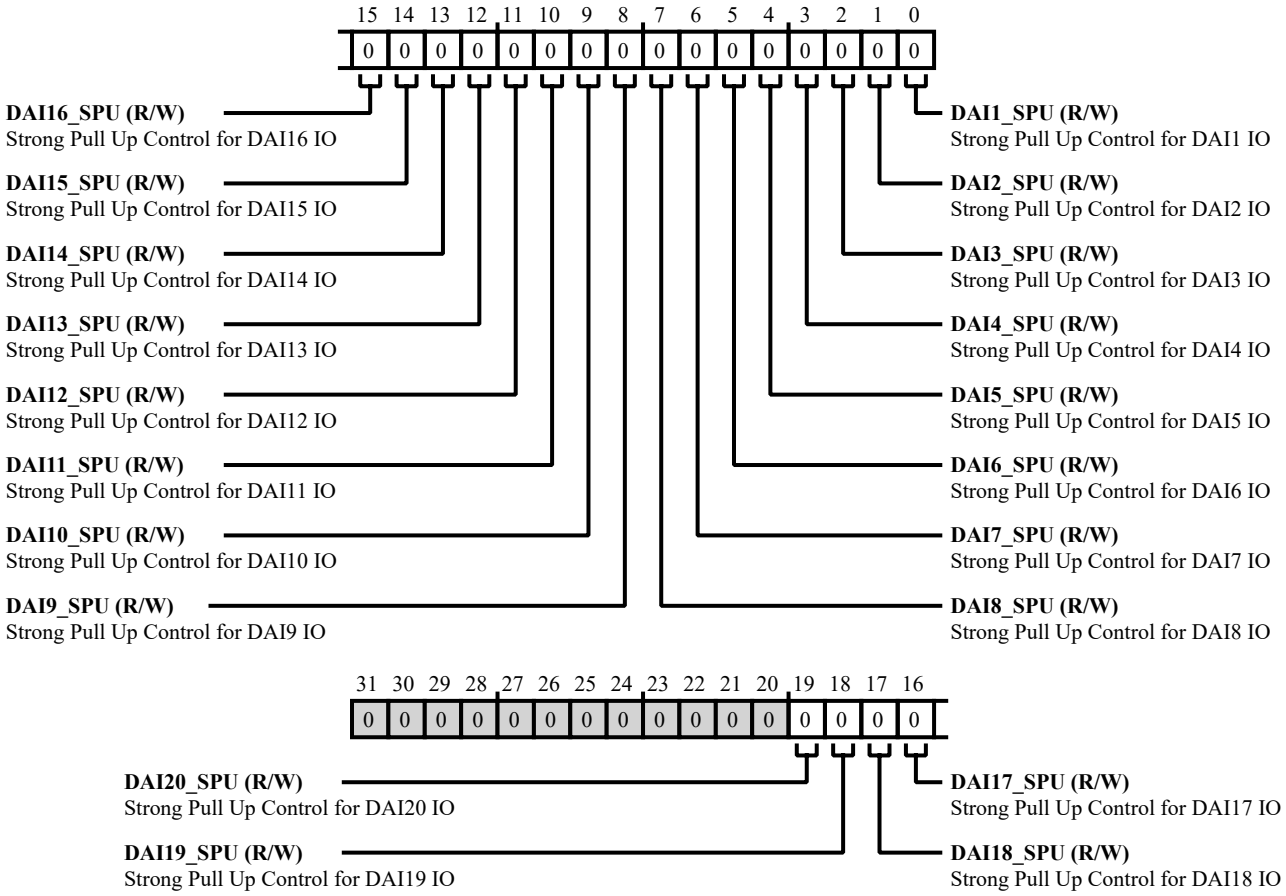


Figure 13-51: PADS_DAI1_SPU Register Diagram

Table 13-56: PADS_DAI1_SPU Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	DAI20_SPU	Strong Pull Up Control for DAI20 IO.
18 (R/W)	DAI19_SPU	Strong Pull Up Control for DAI19 IO.
17 (R/W)	DAI18_SPU	Strong Pull Up Control for DAI18 IO.
16 (R/W)	DAI17_SPU	Strong Pull Up Control for DAI17 IO.

Table 13-56: PADS_DAI1_SPU Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	DAI16_SPU	Strong Pull Up Control for DAI16 IO.
14 (R/W)	DAI15_SPU	Strong Pull Up Control for DAI15 IO.
13 (R/W)	DAI14_SPU	Strong Pull Up Control for DAI14 IO.
12 (R/W)	DAI13_SPU	Strong Pull Up Control for DAI13 IO.
11 (R/W)	DAI12_SPU	Strong Pull Up Control for DAI12 IO.
10 (R/W)	DAI11_SPU	Strong Pull Up Control for DAI11 IO.
9 (R/W)	DAI10_SPU	Strong Pull Up Control for DAI10 IO.
8 (R/W)	DAI9_SPU	Strong Pull Up Control for DAI9 IO.
7 (R/W)	DAI8_SPU	Strong Pull Up Control for DAI8 IO.
6 (R/W)	DAI7_SPU	Strong Pull Up Control for DAI7 IO.
5 (R/W)	DAI6_SPU	Strong Pull Up Control for DAI6 IO.
4 (R/W)	DAI5_SPU	Strong Pull Up Control for DAI5 IO.
3 (R/W)	DAI4_SPU	Strong Pull Up Control for DAI4 IO.
2 (R/W)	DAI3_SPU	Strong Pull Up Control for DAI3 IO.
1 (R/W)	DAI2_SPU	Strong Pull Up Control for DAI2 IO.
0 (R/W)	DAI1_SPU	Strong Pull Up Control for DAI1 IO.

Non-GPIO Control Register

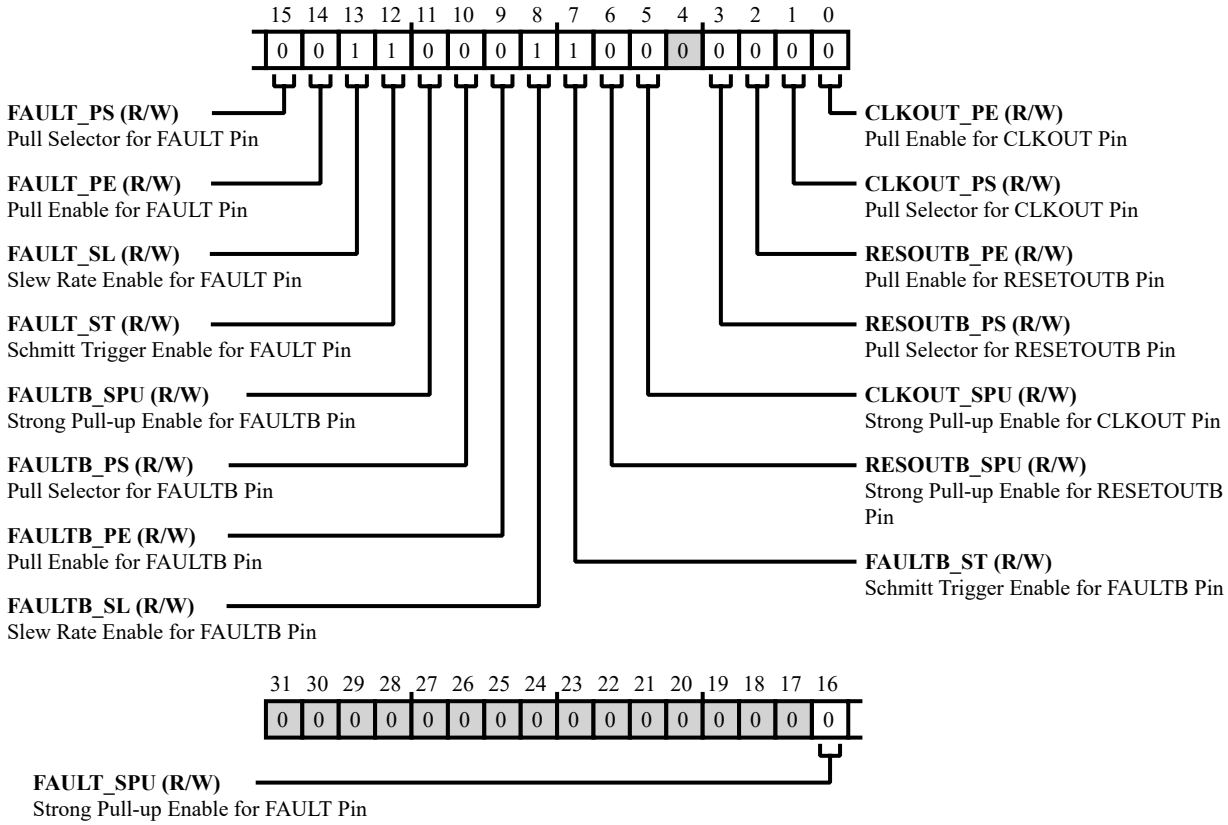


Figure 13-52: PADS_NONPORTS_CTL Register Diagram

Table 13-57: PADS_NONPORTS_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	FAULT_SPU	Strong Pull-up Enable for FAULT Pin.
15 (R/W)	FAULT_PS	Pull Selector for FAULT Pin.
14 (R/W)	FAULT_PE	Pull Enable for FAULT Pin.
13 (R/W)	FAULT_SL	Slew Rate Enable for FAULT Pin.
12 (R/W)	FAULT_ST	Schmitt Trigger Enable for FAULT Pin.

Table 13-57: PADS_NONPORTS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W)	FAULTB_SPU	Strong Pull-up Enable for FAULTB Pin.
10 (R/W)	FAULTB_PS	Pull Selector for FAULTB Pin.
9 (R/W)	FAULTB_PE	Pull Enable for FAULTB Pin.
8 (R/W)	FAULTB_SL	Slew Rate Enable for FAULTB Pin.
7 (R/W)	FAULTB_ST	Schmitt Trigger Enable for FAULTB Pin.
6 (R/W)	RESOUTB_SPU	Strong Pull-up Enable for RESETOUTB Pin.
5 (R/W)	CLKOUT_SPU	Strong Pull-up Enable for CLKOUT Pin.
3 (R/W)	RESOUTB_PS	Pull Selector for RESETOUTB Pin.
2 (R/W)	RESOUTB_PE	Pull Enable for RESETOUTB Pin.
1 (R/W)	CLKOUT_PS	Pull Selector for CLKOUT Pin.
0 (R/W)	CLKOUT_PE	Pull Enable for CLKOUT Pin.

Non-GPIO Drive Strength Register

The PADS_NONPORTS_DS register sets the drive strength and tolerance for the non-GPIO pins. The drive strength is only controlled for pin groups classified by the interface type.

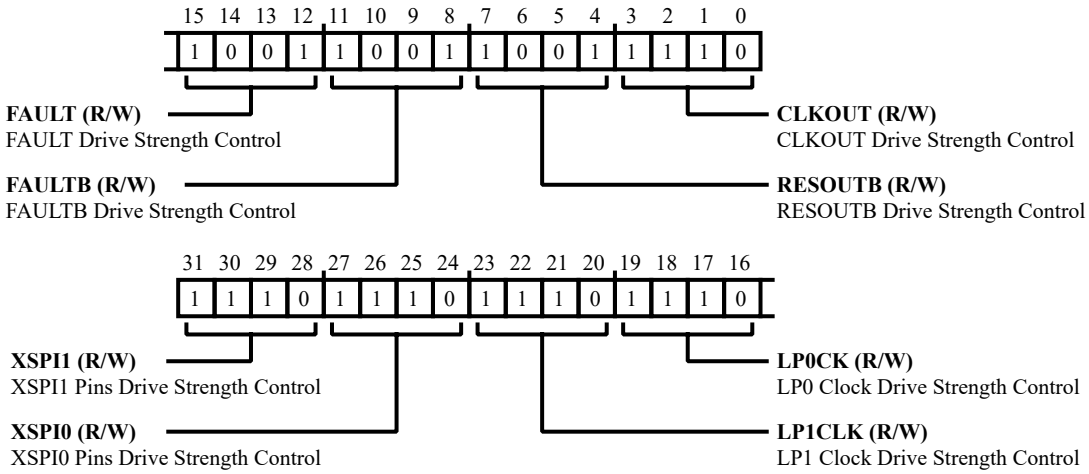


Figure 13-53: PADS_NONPORTS_DS Register Diagram

Table 13-58: PADS_NONPORTS_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	XSPI1	XSPI1 Pins Drive Strength Control.
27:24 (R/W)	XSPI0	XSPI0 Pins Drive Strength Control.
23:20 (R/W)	LP1CLK	LP1 Clock Drive Strength Control.
19:16 (R/W)	LP0CK	LP0 Clock Drive Strength Control.
15:12 (R/W)	FAULT	FAULT Drive Strength Control.
11:8 (R/W)	FAULTB	FAULTB Drive Strength Control.
7:4 (R/W)	RESOUTB	RESOUTB Drive Strength Control.
3:0 (R/W)	CLKOUT	CLKOUT Drive Strength Control.

EMAC Non-GPIO Drive Strength Register

The PADS_NONPORTS_DS_1 register sets the drive strength and tolerance for the non-GPIO pins. The drive strength is only controlled for pin groups classified by the interface type.

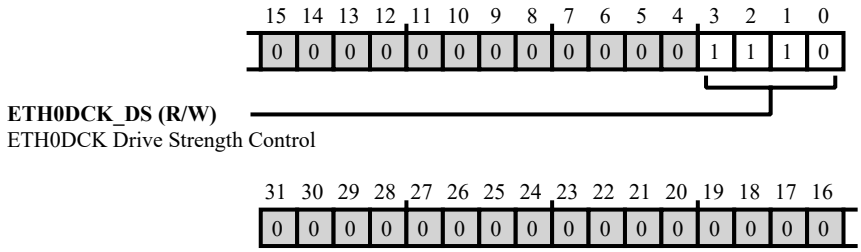


Figure 13-54: PADS_NONPORTS_DS_1 Register Diagram

Table 13-59: PADS_NONPORTS_DS_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	ETH0DCK_DS	ETH0DCK Drive Strength Control. The default drive strength is 1001. Do not change this value.

OTPC Boot Status Register

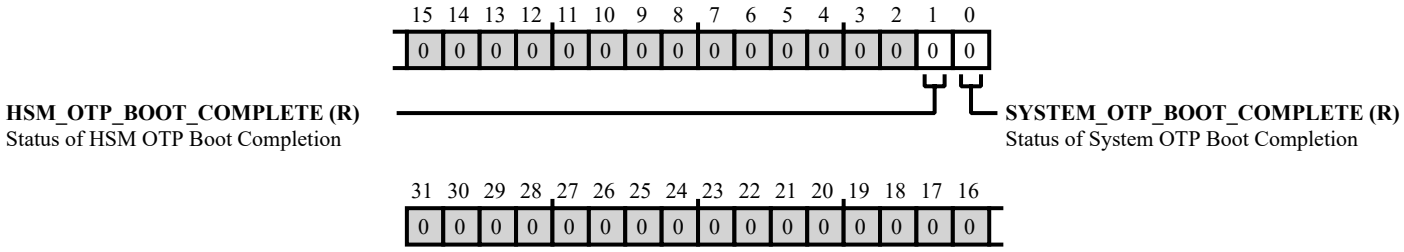


Figure 13-55: PADS_OTPC_BOOT_STAT Register Diagram

Table 13-60: PADS_OTPC_BOOT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	HSM_OTP_BOOT_COMPLETE	Status of HSM OTP Boot Completion. 1> Success 0> Failure
0 (R/NW)	SYSTEM_OTP_BOOT_COMPLETE	Status of System OTP Boot Completion. 1> Success 0> Failure

Peripheral PAD Configuration 0 Register

The PADS_PCFG0 register provides several configuration options for the pads and multiplexing for peripherals.

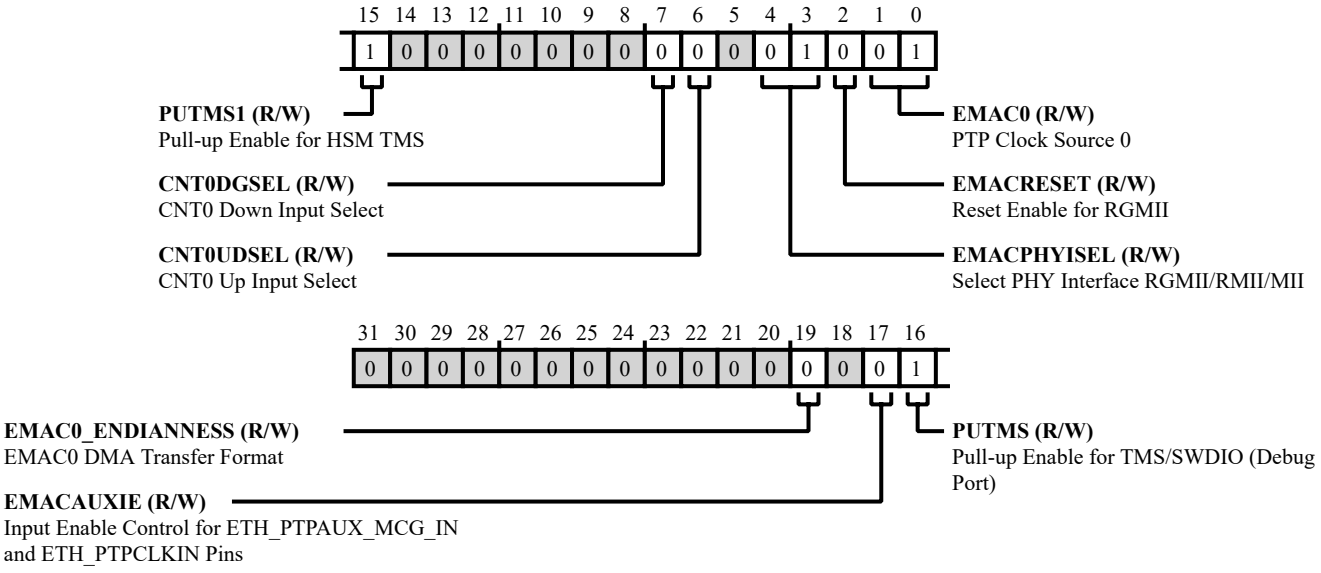


Figure 13-56: PADS_PCFG0 Register Diagram

Table 13-61: PADS_PCFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	EMAC0_ENDIANNESS	EMAC0 DMA Transfer Format. Configure the PADS_PCFG0.EMAC0_ENDIANNESS bit field to set the EMAC0 DMA transfers to use big-endian or little-endian format.	
		0	Little-Endian
		1	Big-Endian
17 (R/W)	EMACAUXIE	Input Enable Control for ETH_PTPAUX_MCG_IN and ETH_PTPCLKIN Pins.	
		0	Disable input
		1	Enable input
16 (R/W)	PUTMS	Pull-up Enable for TMS/SWDIO (Debug Port).	
		0	Disable pull-up
		1	Enable pull-up
15 (R/W)	PUTMS1	Pull-up Enable for HSM TMS.	
		0	Disable pull-up
		1	Enable pull-up

Table 13-61: PADS_PCFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	CNT0DGSEL	CNT0 Down Input Select. The PADS_PCFG0 bit selects the input for the CNT0 counter down and gate.	
		0	GPIO port pins
		1	Trigger connection
6 (R/W)	CNT0UDSEL	CNT0 Up Input Select. The PADS_PCFG0 bit selects the input for the CNT0 counter up and direction.	
		0	GPIO port pins
		1	Trigger connection
4:3 (R/W)	EMACPHYISEL	Select PHY Interface RGMII/RMII/MII.	
		0	MII Interface Selects PHY Interface MII
		1	RGMII Interface Selects PHY Interface RGMII
		2	RMII Interface Selects PHY Interface RMII
		3	Reserved
2 (R/W)	EMACRESET	Reset Enable for RGMII. The PADS_PCFG0.EMACRESET bit asserts the reset on the RGMII interface. To select the PHY interface (RGMII or RMII), set the EMACPHYISEL bit as required and then set PADS_PCFG0.EMACRESET.	
		0	RGMII reset is asserted
		1	RGMII reset reset is deasserted
1:0 (R/W)	EMAC0	PTP Clock Source 0. The PADS_PCFG0.EMAC0 selects the clock source for the PTP Block in EMAC0.	
		0	REFCLK (MII or RMII) or CLKO7 (RGMII)
		1	SCLK0
		2	External PTP clock
		3	SCLK0

Peripheral Configuration 1 Register

The PADS_PCFG1 register provides bits to configure PWM secondary inputs to timers and enable inputs from the TRU.

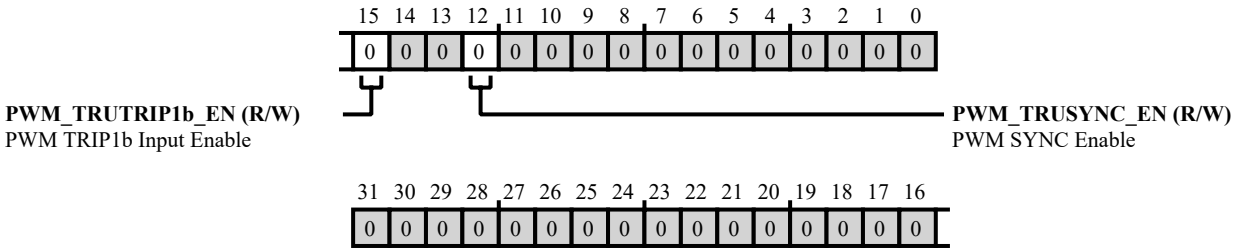


Figure 13-57: PADS_PCFG1 Register Diagram

Table 13-62: PADS_PCFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PWM_TRUTRIP1B_EN	PWM TRIP1b Input Enable. The PADS_PCFG1.PWM_TRUTRIP1B_EN bit enables the PWM TRIP1b input from the TRU. 0> TRU Trip1b input to PWM disabled 1>TRU Trip1b input to PWM enabled
12 (R/W)	PWM_TRUSYNC_EN	PWM SYNC Enable. The PADS_PCFG1.PWM_TRUSYNC_EN bit enables the PWM SYNC input from the TRU. 0> Disable, PWM sync input is from external pin 1> Enable, PWM sync input is from the TRU

PORTA Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

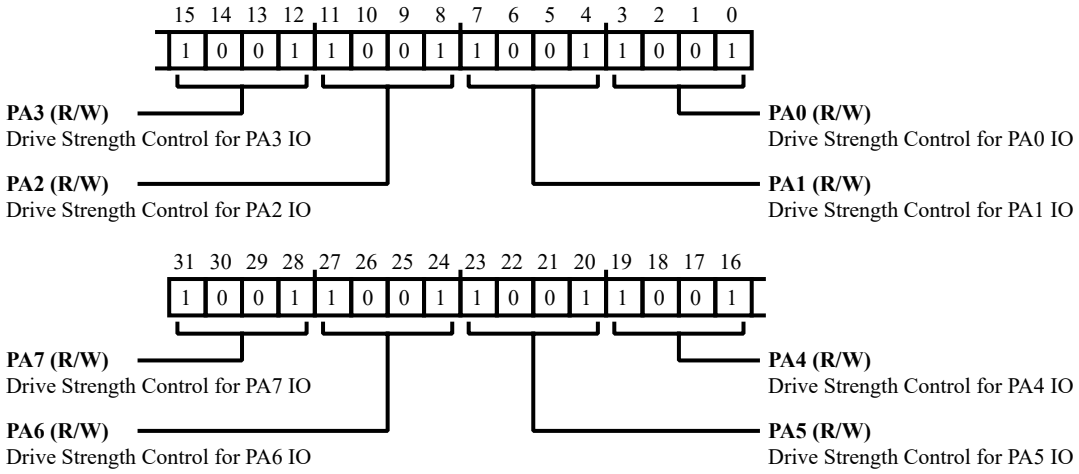


Figure 13-58: PADS_PORTA0_DS Register Diagram

Table 13-63: PADS_PORTA0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PA7	Drive Strength Control for PA7 IO.
27:24 (R/W)	PA6	Drive Strength Control for PA6 IO.
23:20 (R/W)	PA5	Drive Strength Control for PA5 IO.
19:16 (R/W)	PA4	Drive Strength Control for PA4 IO.
15:12 (R/W)	PA3	Drive Strength Control for PA3 IO.
11:8 (R/W)	PA2	Drive Strength Control for PA2 IO.
7:4 (R/W)	PA1	Drive Strength Control for PA1 IO.
3:0 (R/W)	PA0	Drive Strength Control for PA0 IO.

PORTA Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

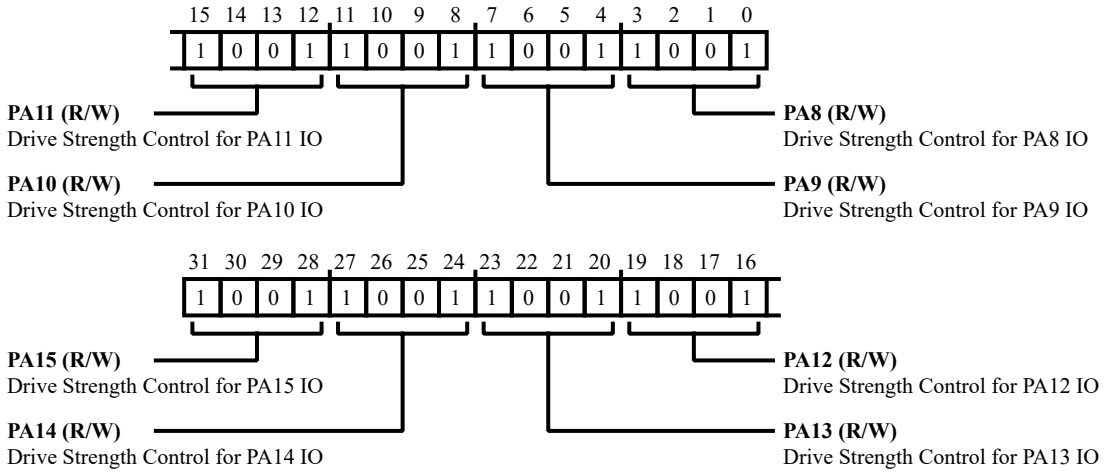


Figure 13-59: PADS_PORTA1_DS Register Diagram

Table 13-64: PADS_PORTA1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PA15	Drive Strength Control for PA15 IO.
27:24 (R/W)	PA14	Drive Strength Control for PA14 IO.
23:20 (R/W)	PA13	Drive Strength Control for PA13 IO.
19:16 (R/W)	PA12	Drive Strength Control for PA12 IO.
15:12 (R/W)	PA11	Drive Strength Control for PA11 IO.
11:8 (R/W)	PA10	Drive Strength Control for PA10 IO.
7:4 (R/W)	PA9	Drive Strength Control for PA9 IO.
3:0 (R/W)	PA8	Drive Strength Control for PA8 IO.

PORTB Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

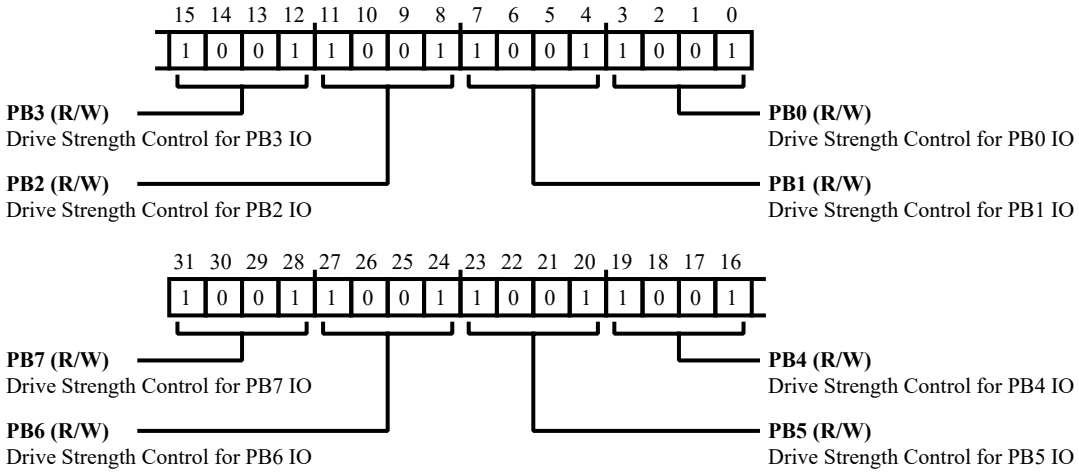


Figure 13-60: PADS_PORTB0_DS Register Diagram

Table 13-65: PADS_PORTB0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PB7	Drive Strength Control for PB7 IO.
27:24 (R/W)	PB6	Drive Strength Control for PB6 IO.
23:20 (R/W)	PB5	Drive Strength Control for PB5 IO.
19:16 (R/W)	PB4	Drive Strength Control for PB4 IO.
15:12 (R/W)	PB3	Drive Strength Control for PB3 IO.
11:8 (R/W)	PB2	Drive Strength Control for PB2 IO.
7:4 (R/W)	PB1	Drive Strength Control for PB1 IO.
3:0 (R/W)	PB0	Drive Strength Control for PB0 IO.

PORTB Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

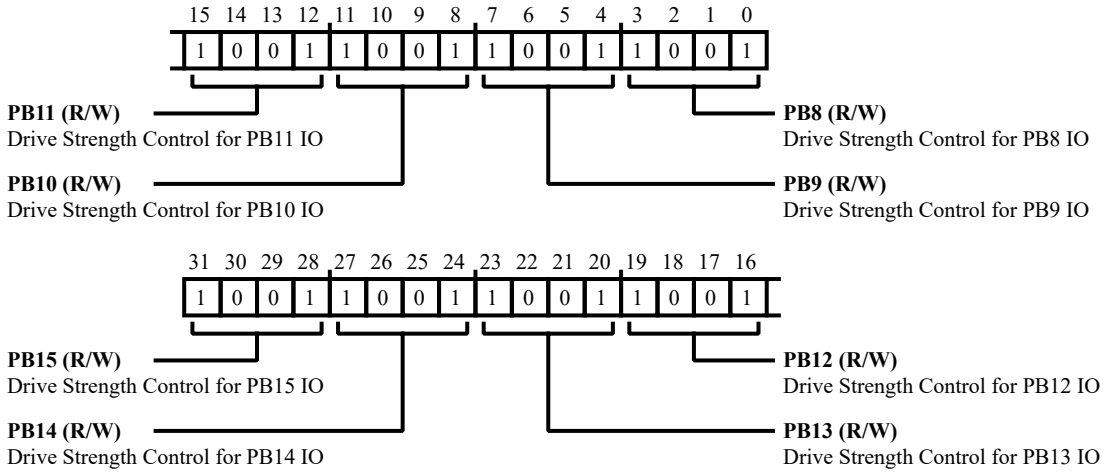


Figure 13-61: PADS_PORTB1_DS Register Diagram

Table 13-66: PADS_PORTB1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PB15	Drive Strength Control for PB15 IO.
27:24 (R/W)	PB14	Drive Strength Control for PB14 IO.
23:20 (R/W)	PB13	Drive Strength Control for PB13 IO.
19:16 (R/W)	PB12	Drive Strength Control for PB12 IO.
15:12 (R/W)	PB11	Drive Strength Control for PB11 IO.
11:8 (R/W)	PB10	Drive Strength Control for PB10 IO.
7:4 (R/W)	PB9	Drive Strength Control for PB9 IO.
3:0 (R/W)	PB8	Drive Strength Control for PB8 IO.

PORTC Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

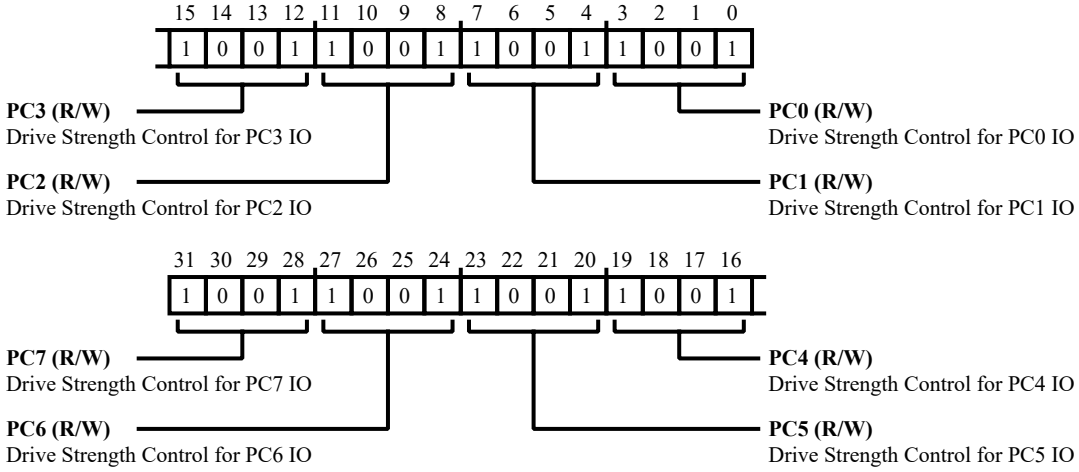


Figure 13-62: PADS_PORTC0_DS Register Diagram

Table 13-67: PADS_PORTC0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PC7	Drive Strength Control for PC7 IO.
27:24 (R/W)	PC6	Drive Strength Control for PC6 IO.
23:20 (R/W)	PC5	Drive Strength Control for PC5 IO.
19:16 (R/W)	PC4	Drive Strength Control for PC4 IO.
15:12 (R/W)	PC3	Drive Strength Control for PC3 IO.
11:8 (R/W)	PC2	Drive Strength Control for PC2 IO.
7:4 (R/W)	PC1	Drive Strength Control for PC1 IO.
3:0 (R/W)	PC0	Drive Strength Control for PC0 IO.

PORTC Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

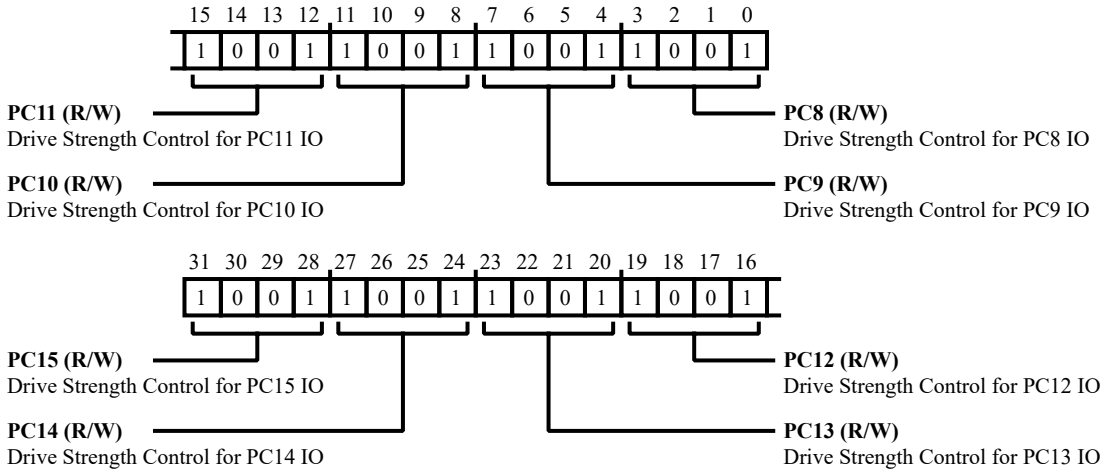


Figure 13-63: PADS_PORTC1_DS Register Diagram

Table 13-68: PADS_PORTC1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PC15	Drive Strength Control for PC15 IO.
27:24 (R/W)	PC14	Drive Strength Control for PC14 IO.
23:20 (R/W)	PC13	Drive Strength Control for PC13 IO.
19:16 (R/W)	PC12	Drive Strength Control for PC12 IO.
15:12 (R/W)	PC11	Drive Strength Control for PC11 IO.
11:8 (R/W)	PC10	Drive Strength Control for PC10 IO.
7:4 (R/W)	PC9	Drive Strength Control for PC9 IO.
3:0 (R/W)	PC8	Drive Strength Control for PC8 IO.

PORTD Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

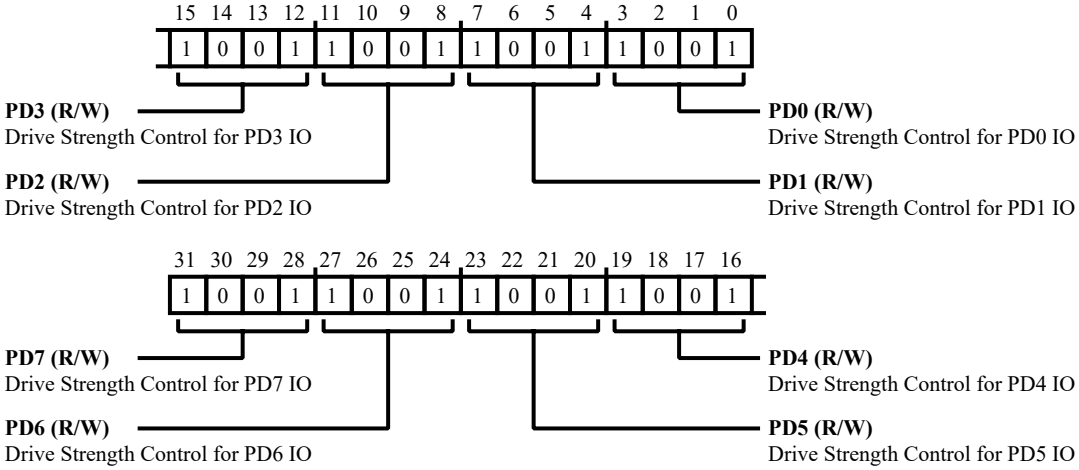


Figure 13-64: PADS_PORTD0_DS Register Diagram

Table 13-69: PADS_PORTD0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PD7	Drive Strength Control for PD7 IO.
27:24 (R/W)	PD6	Drive Strength Control for PD6 IO.
23:20 (R/W)	PD5	Drive Strength Control for PD5 IO.
19:16 (R/W)	PD4	Drive Strength Control for PD4 IO.
15:12 (R/W)	PD3	Drive Strength Control for PD3 IO.
11:8 (R/W)	PD2	Drive Strength Control for PD2 IO.
7:4 (R/W)	PD1	Drive Strength Control for PD1 IO.
3:0 (R/W)	PD0	Drive Strength Control for PD0 IO.

PORTD Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

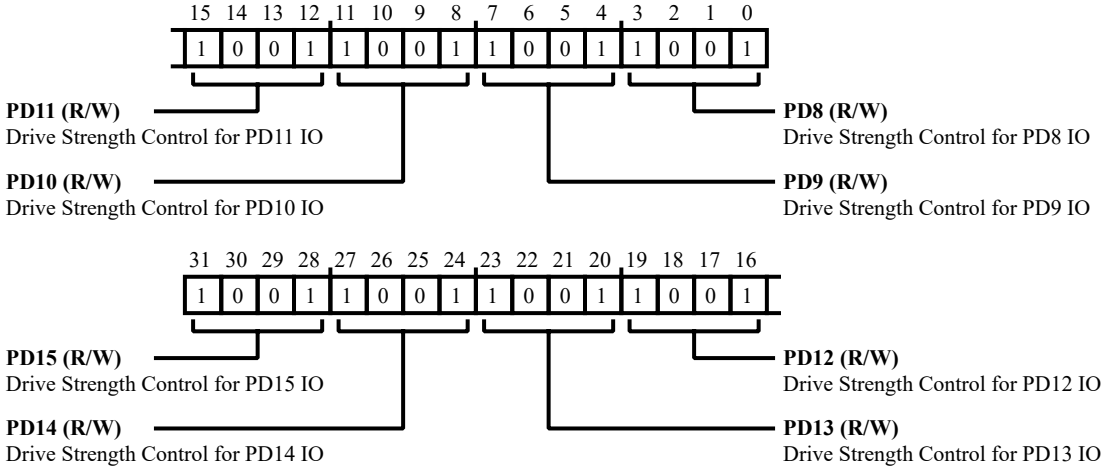


Figure 13-65: PADS_PORTD1_DS Register Diagram

Table 13-70: PADS_PORTD1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PD15	Drive Strength Control for PD15 IO.
27:24 (R/W)	PD14	Drive Strength Control for PD14 IO.
23:20 (R/W)	PD13	Drive Strength Control for PD13 IO.
19:16 (R/W)	PD12	Drive Strength Control for PD12 IO.
15:12 (R/W)	PD11	Drive Strength Control for PD11 IO.
11:8 (R/W)	PD10	Drive Strength Control for PD10 IO.
7:4 (R/W)	PD9	Drive Strength Control for PD9 IO.
3:0 (R/W)	PD8	Drive Strength Control for PD8 IO.

PORTE Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

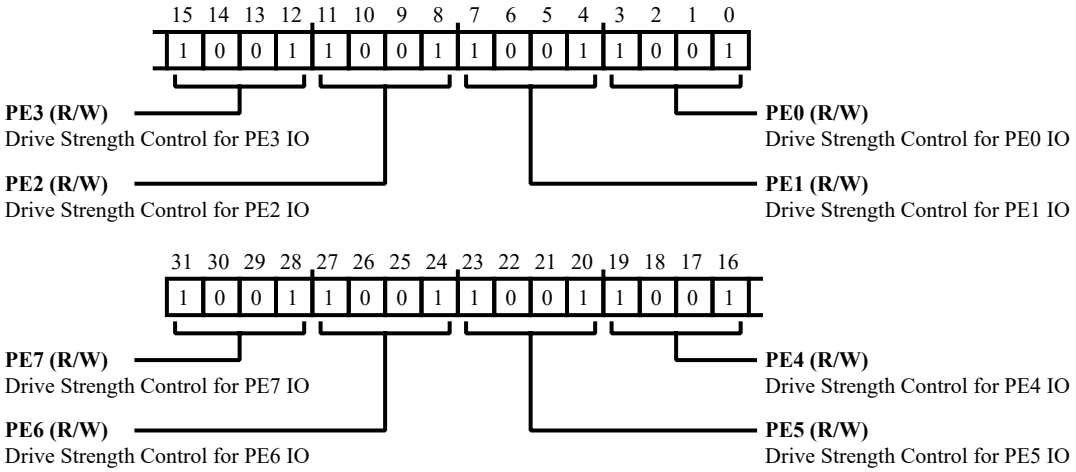


Figure 13-66: PADS_PORTE0_DS Register Diagram

Table 13-71: PADS_PORTE0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PE7	Drive Strength Control for PE7 IO.
27:24 (R/W)	PE6	Drive Strength Control for PE6 IO.
23:20 (R/W)	PE5	Drive Strength Control for PE5 IO.
19:16 (R/W)	PE4	Drive Strength Control for PE4 IO.
15:12 (R/W)	PE3	Drive Strength Control for PE3 IO.
11:8 (R/W)	PE2	Drive Strength Control for PE2 IO.
7:4 (R/W)	PE1	Drive Strength Control for PE1 IO.
3:0 (R/W)	PE0	Drive Strength Control for PE0 IO.

PORTE Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

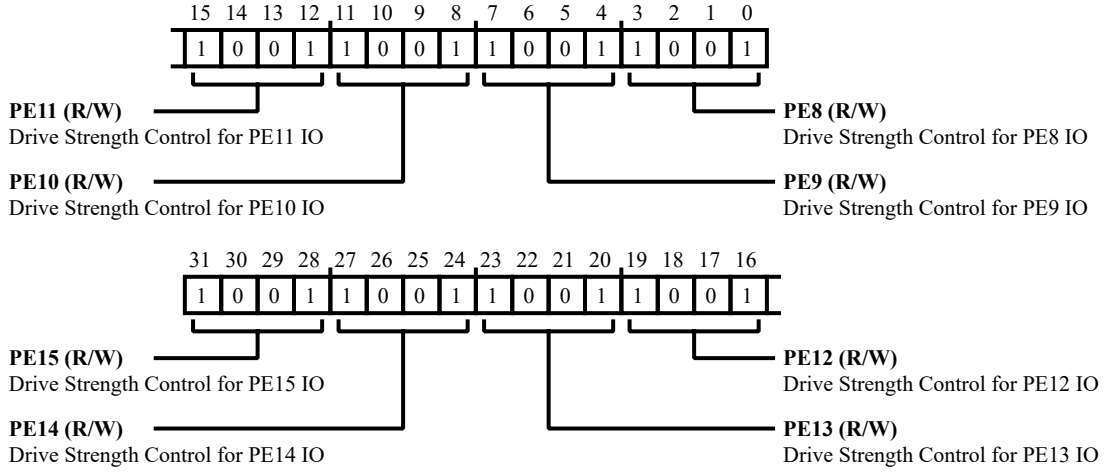


Figure 13-67: PADS_PORTE1_DS Register Diagram

Table 13-72: PADS_PORTE1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PE15	Drive Strength Control for PE15 IO.
27:24 (R/W)	PE14	Drive Strength Control for PE14 IO.
23:20 (R/W)	PE13	Drive Strength Control for PE13 IO.
19:16 (R/W)	PE12	Drive Strength Control for PE12 IO.
15:12 (R/W)	PE11	Drive Strength Control for PE11 IO.
11:8 (R/W)	PE10	Drive Strength Control for PE10 IO.
7:4 (R/W)	PE9	Drive Strength Control for PE9 IO.
3:0 (R/W)	PE8	Drive Strength Control for PE8 IO.

PORTF Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

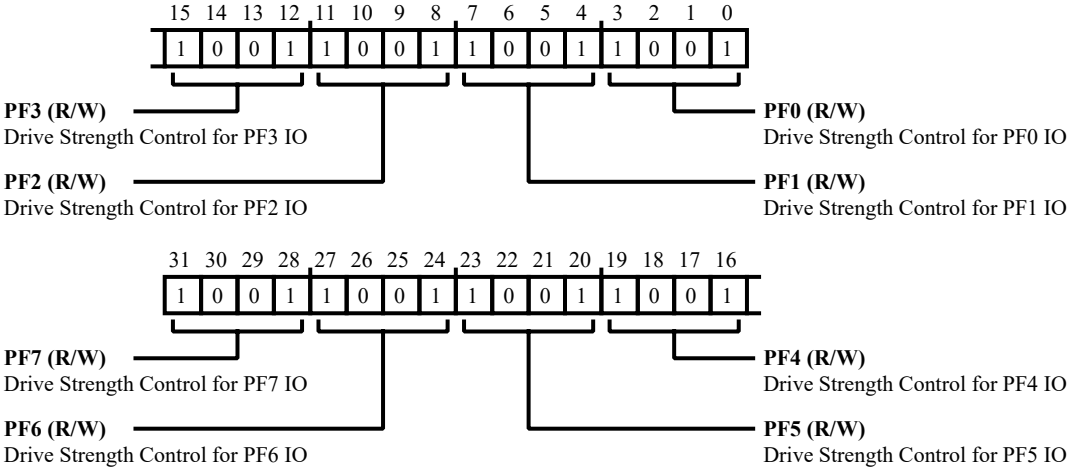


Figure 13-68: PADS_PORTF0_DS Register Diagram

Table 13-73: PADS_PORTF0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PF7	Drive Strength Control for PF7 IO.
27:24 (R/W)	PF6	Drive Strength Control for PF6 IO.
23:20 (R/W)	PF5	Drive Strength Control for PF5 IO.
19:16 (R/W)	PF4	Drive Strength Control for PF4 IO.
15:12 (R/W)	PF3	Drive Strength Control for PF3 IO.
11:8 (R/W)	PF2	Drive Strength Control for PF2 IO.
7:4 (R/W)	PF1	Drive Strength Control for PF1 IO.
3:0 (R/W)	PF0	Drive Strength Control for PF0 IO.

PORTF Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

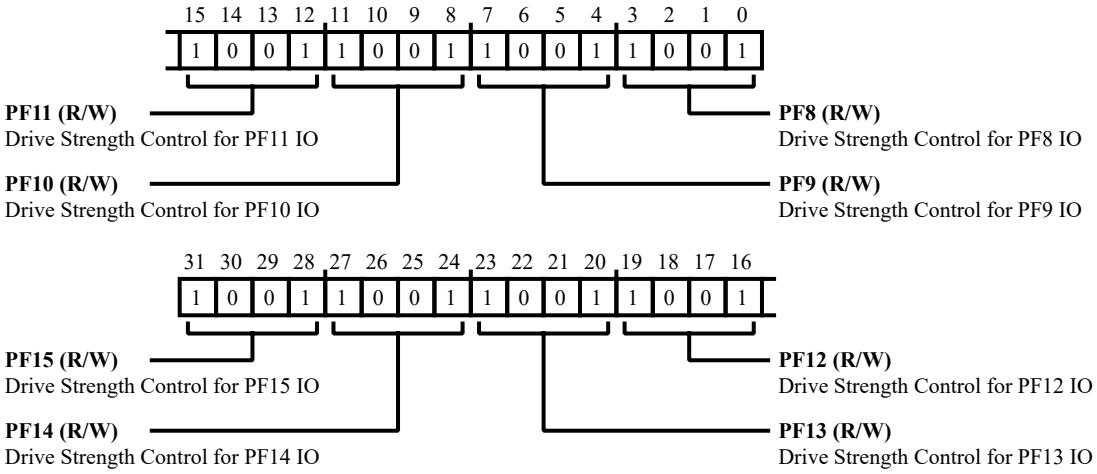


Figure 13-69: PADS_PORTF1_DS Register Diagram

Table 13-74: PADS_PORTF1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PF15	Drive Strength Control for PF15 IO.
27:24 (R/W)	PF14	Drive Strength Control for PF14 IO.
23:20 (R/W)	PF13	Drive Strength Control for PF13 IO.
19:16 (R/W)	PF12	Drive Strength Control for PF12 IO.
15:12 (R/W)	PF11	Drive Strength Control for PF11 IO.
11:8 (R/W)	PF10	Drive Strength Control for PF10 IO.
7:4 (R/W)	PF9	Drive Strength Control for PF9 IO.
3:0 (R/W)	PF8	Drive Strength Control for PF8 IO.

PORTG Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

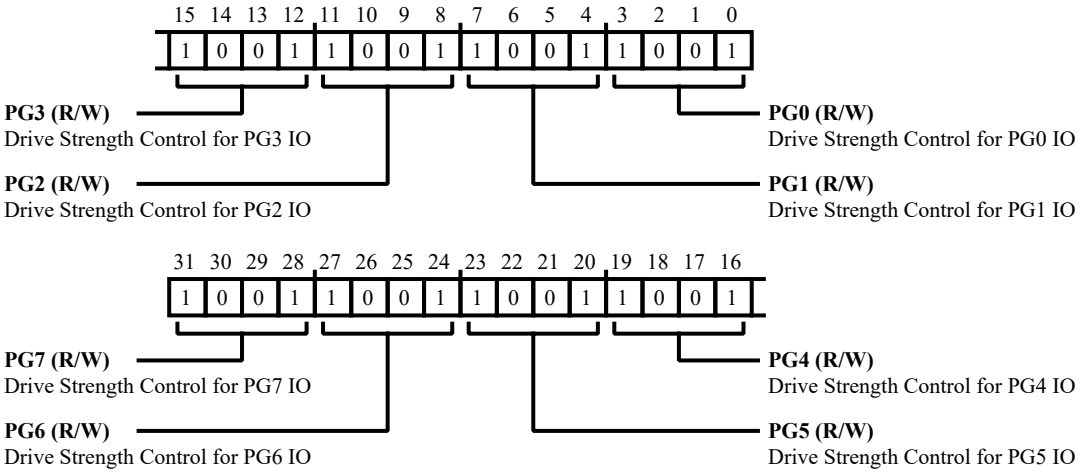


Figure 13-70: PADS_PORTG0_DS Register Diagram

Table 13-75: PADS_PORTG0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PG7	Drive Strength Control for PG7 IO.
27:24 (R/W)	PG6	Drive Strength Control for PG6 IO.
23:20 (R/W)	PG5	Drive Strength Control for PG5 IO.
19:16 (R/W)	PG4	Drive Strength Control for PG4 IO.
15:12 (R/W)	PG3	Drive Strength Control for PG3 IO.
11:8 (R/W)	PG2	Drive Strength Control for PG2 IO.
7:4 (R/W)	PG1	Drive Strength Control for PG1 IO.
3:0 (R/W)	PG0	Drive Strength Control for PG0 IO.

PORTG Pins 8 to 15 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

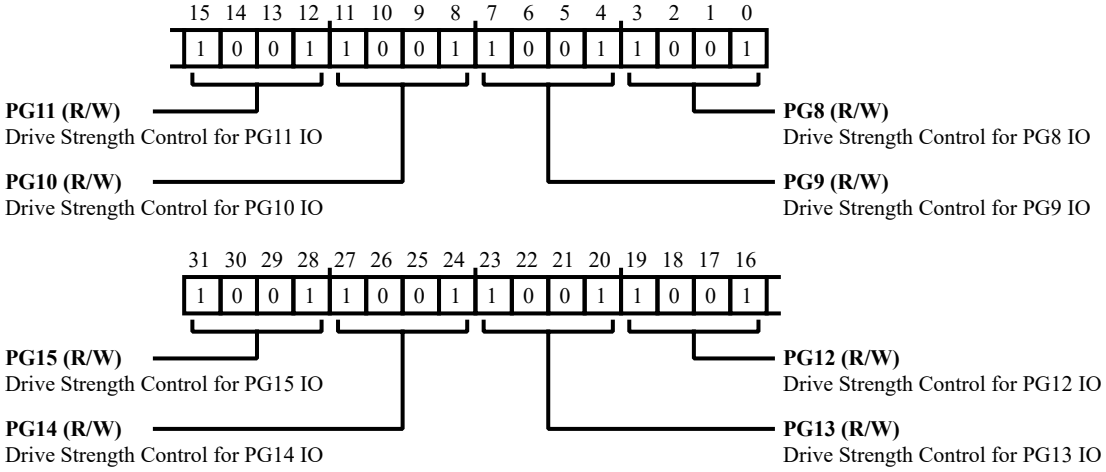


Figure 13-71: PADS_PORTG1_DS Register Diagram

Table 13-76: PADS_PORTG1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PG15	Drive Strength Control for PG15 IO.
27:24 (R/W)	PG14	Drive Strength Control for PG14 IO.
23:20 (R/W)	PG13	Drive Strength Control for PG13 IO.
19:16 (R/W)	PG12	Drive Strength Control for PG12 IO.
15:12 (R/W)	PG11	Drive Strength Control for PG11 IO.
11:8 (R/W)	PG10	Drive Strength Control for PG10 IO.
7:4 (R/W)	PG9	Drive Strength Control for PG9 IO.
3:0 (R/W)	PG8	Drive Strength Control for PG8 IO.

PORTH Pins 0 to 7 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

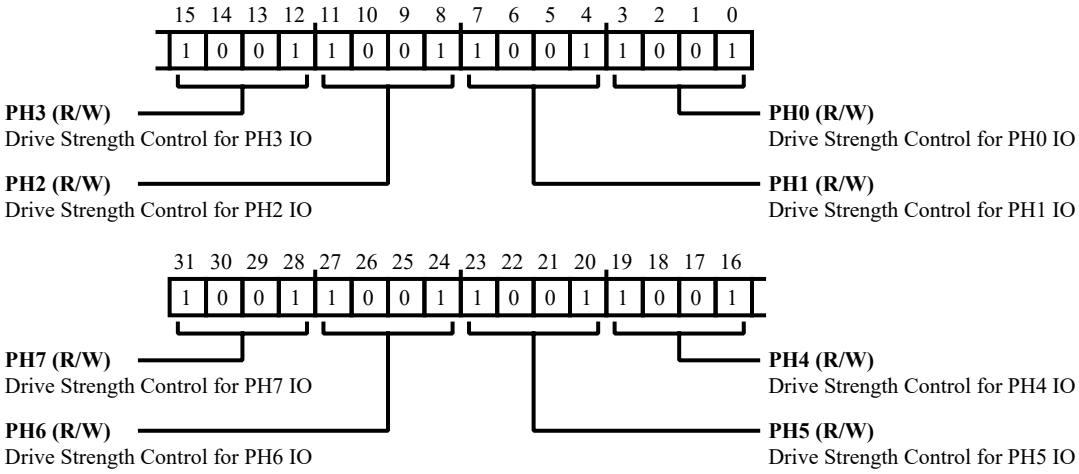


Figure 13-72: PADS_PORTH0_DS Register Diagram

Table 13-77: PADS_PORTH0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PH7	Drive Strength Control for PH7 IO.
27:24 (R/W)	PH6	Drive Strength Control for PH6 IO.
23:20 (R/W)	PH5	Drive Strength Control for PH5 IO.
19:16 (R/W)	PH4	Drive Strength Control for PH4 IO.
15:12 (R/W)	PH3	Drive Strength Control for PH3 IO.
11:8 (R/W)	PH2	Drive Strength Control for PH2 IO.
7:4 (R/W)	PH1	Drive Strength Control for PH1 IO.
3:0 (R/W)	PH0	Drive Strength Control for PH0 IO.

PORTH Pins 8 and 9 Drive Strength Control Register

The default drive strength is 1001. Do not change this value.

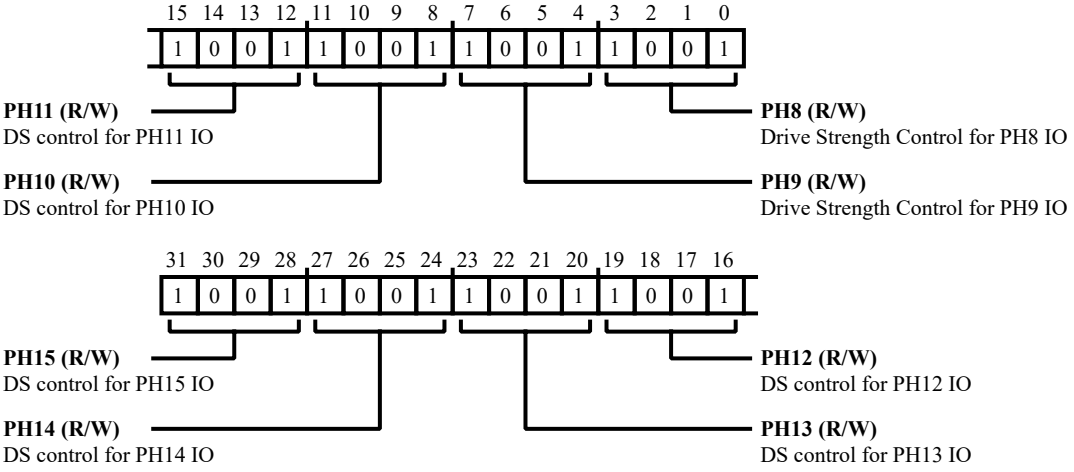


Figure 13-73: PADS_PORTH1_DS Register Diagram

Table 13-78: PADS_PORTH1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	PH15	DS control for PH15 IO.
27:24 (R/W)	PH14	DS control for PH14 IO.
23:20 (R/W)	PH13	DS control for PH13 IO.
19:16 (R/W)	PH12	DS control for PH12 IO.
15:12 (R/W)	PH11	DS control for PH11 IO.
11:8 (R/W)	PH10	DS control for PH10 IO.
7:4 (R/W)	PH9	Drive Strength Control for PH9 IO.
3:0 (R/W)	PH8	Drive Strength Control for PH8 IO.

PORTA and PORTB Pull Enable Register

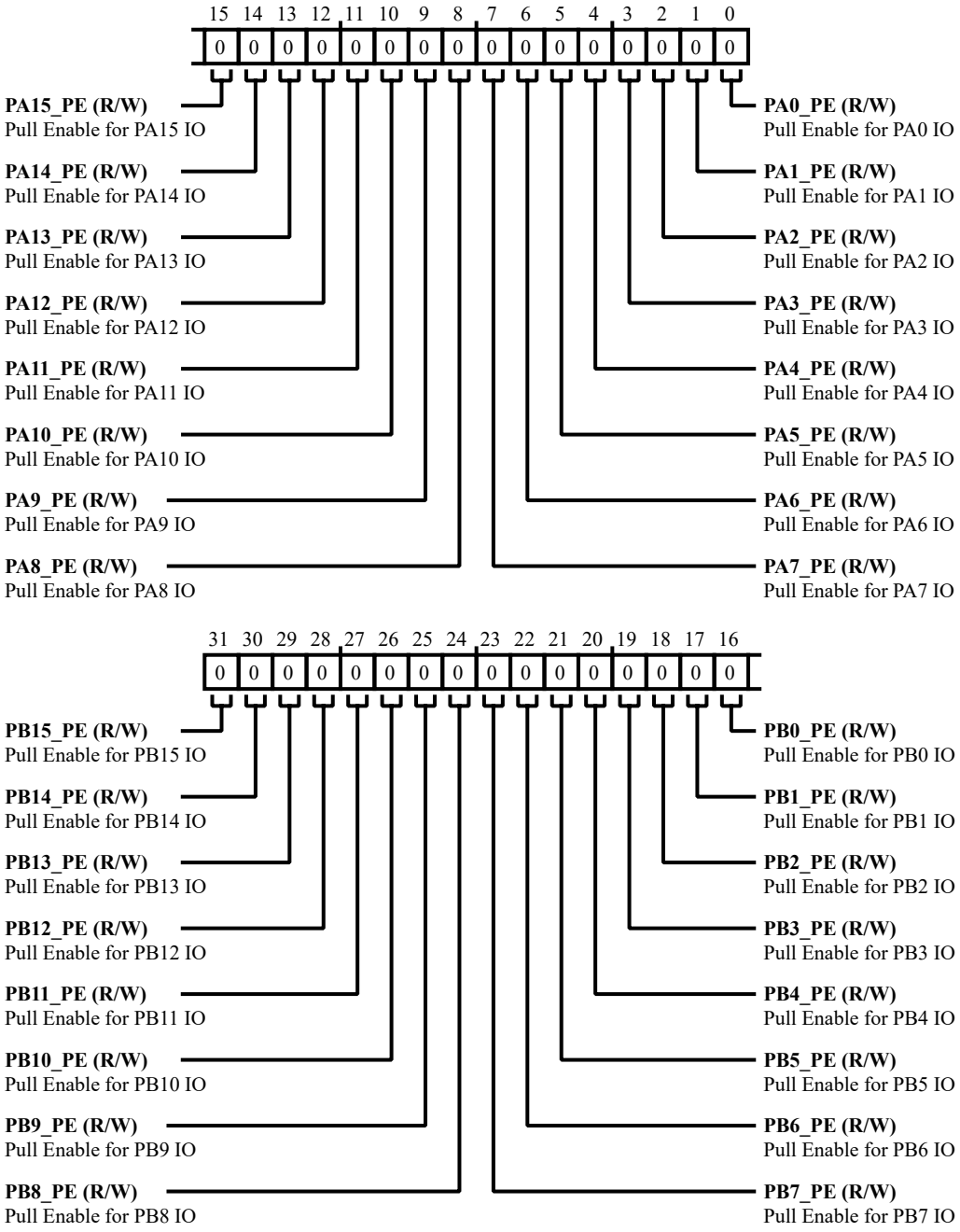


Figure 13-74: PADS_PORT_AB_PE Register Diagram

Table 13-79: PADS_PORT_AB_PE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PB15_PE	Pull Enable for PB15 IO.
30 (R/W)	PB14_PE	Pull Enable for PB14 IO.
29 (R/W)	PB13_PE	Pull Enable for PB13 IO.
28 (R/W)	PB12_PE	Pull Enable for PB12 IO.
27 (R/W)	PB11_PE	Pull Enable for PB11 IO.
26 (R/W)	PB10_PE	Pull Enable for PB10 IO.
25 (R/W)	PB9_PE	Pull Enable for PB9 IO.
24 (R/W)	PB8_PE	Pull Enable for PB8 IO.
23 (R/W)	PB7_PE	Pull Enable for PB7 IO.
22 (R/W)	PB6_PE	Pull Enable for PB6 IO.
21 (R/W)	PB5_PE	Pull Enable for PB5 IO.
20 (R/W)	PB4_PE	Pull Enable for PB4 IO.
19 (R/W)	PB3_PE	Pull Enable for PB3 IO.
18 (R/W)	PB2_PE	Pull Enable for PB2 IO.
17 (R/W)	PB1_PE	Pull Enable for PB1 IO.
16 (R/W)	PB0_PE	Pull Enable for PB0 IO.

Table 13-79: PADS_PORT_AB_PE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PA15_PE	Pull Enable for PA15 IO.
14 (R/W)	PA14_PE	Pull Enable for PA14 IO.
13 (R/W)	PA13_PE	Pull Enable for PA13 IO.
12 (R/W)	PA12_PE	Pull Enable for PA12 IO.
11 (R/W)	PA11_PE	Pull Enable for PA11 IO.
10 (R/W)	PA10_PE	Pull Enable for PA10 IO.
9 (R/W)	PA9_PE	Pull Enable for PA9 IO.
8 (R/W)	PA8_PE	Pull Enable for PA8 IO.
7 (R/W)	PA7_PE	Pull Enable for PA7 IO.
6 (R/W)	PA6_PE	Pull Enable for PA6 IO.
5 (R/W)	PA5_PE	Pull Enable for PA5 IO.
4 (R/W)	PA4_PE	Pull Enable for PA4 IO.
3 (R/W)	PA3_PE	Pull Enable for PA3 IO.
2 (R/W)	PA2_PE	Pull Enable for PA2 IO.
1 (R/W)	PA1_PE	Pull Enable for PA1 IO.
0 (R/W)	PA0_PE	Pull Enable for PA0 IO.

PORTA and PORTB Pull Selector Register

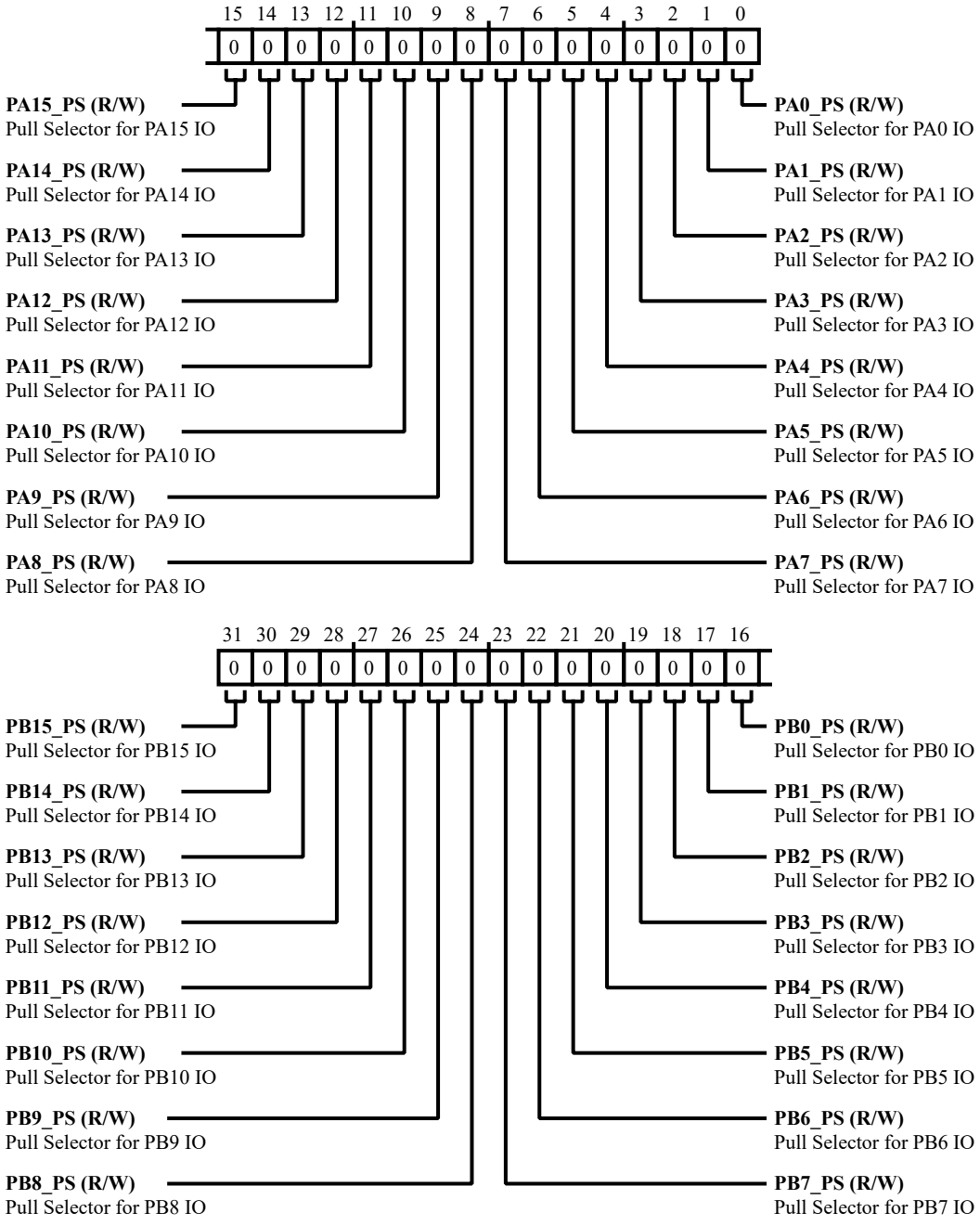


Figure 13-75: PADS_PORT_AB_PS Register Diagram

Table 13-80: PADS_PORT_AB_PS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PB15_PS	Pull Selector for PB15 IO.
30 (R/W)	PB14_PS	Pull Selector for PB14 IO.
29 (R/W)	PB13_PS	Pull Selector for PB13 IO.
28 (R/W)	PB12_PS	Pull Selector for PB12 IO.
27 (R/W)	PB11_PS	Pull Selector for PB11 IO.
26 (R/W)	PB10_PS	Pull Selector for PB10 IO.
25 (R/W)	PB9_PS	Pull Selector for PB9 IO.
24 (R/W)	PB8_PS	Pull Selector for PB8 IO.
23 (R/W)	PB7_PS	Pull Selector for PB7 IO.
22 (R/W)	PB6_PS	Pull Selector for PB6 IO.
21 (R/W)	PB5_PS	Pull Selector for PB5 IO.
20 (R/W)	PB4_PS	Pull Selector for PB4 IO.
19 (R/W)	PB3_PS	Pull Selector for PB3 IO.
18 (R/W)	PB2_PS	Pull Selector for PB2 IO.
17 (R/W)	PB1_PS	Pull Selector for PB1 IO.
16 (R/W)	PB0_PS	Pull Selector for PB0 IO.

Table 13-80: PADS_PORT_AB_PS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PA15_PS	Pull Selector for PA15 IO.
14 (R/W)	PA14_PS	Pull Selector for PA14 IO.
13 (R/W)	PA13_PS	Pull Selector for PA13 IO.
12 (R/W)	PA12_PS	Pull Selector for PA12 IO.
11 (R/W)	PA11_PS	Pull Selector for PA11 IO.
10 (R/W)	PA10_PS	Pull Selector for PA10 IO.
9 (R/W)	PA9_PS	Pull Selector for PA9 IO.
8 (R/W)	PA8_PS	Pull Selector for PA8 IO.
7 (R/W)	PA7_PS	Pull Selector for PA7 IO.
6 (R/W)	PA6_PS	Pull Selector for PA6 IO.
5 (R/W)	PA5_PS	Pull Selector for PA5 IO.
4 (R/W)	PA4_PS	Pull Selector for PA4 IO.
3 (R/W)	PA3_PS	Pull Selector for PA3 IO.
2 (R/W)	PA2_PS	Pull Selector for PA2 IO.
1 (R/W)	PA1_PS	Pull Selector for PA1 IO.
0 (R/W)	PA0_PS	Pull Selector for PA0 IO.

PORTA and PORTB Slew Rate Control Register

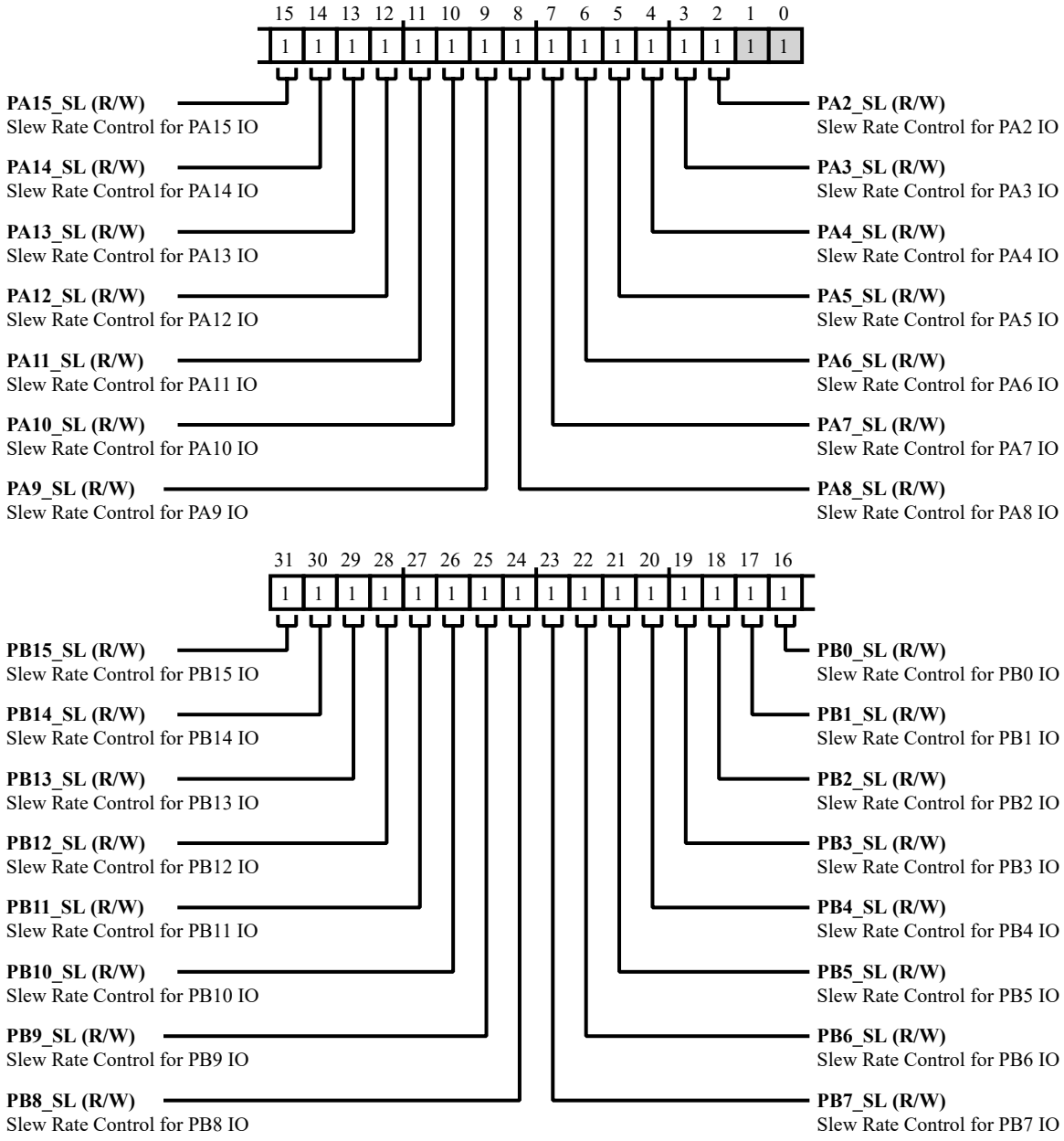


Figure 13-76: PADS_PORT_AB_SL Register Diagram

Table 13-81: PADS_PORT_AB_SL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PB15_SL	Slew Rate Control for PB15 IO.

Table 13-81: PADS_PORT_AB_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	PB14_SL	Slew Rate Control for PB14 IO.
29 (R/W)	PB13_SL	Slew Rate Control for PB13 IO.
28 (R/W)	PB12_SL	Slew Rate Control for PB12 IO.
27 (R/W)	PB11_SL	Slew Rate Control for PB11 IO.
26 (R/W)	PB10_SL	Slew Rate Control for PB10 IO.
25 (R/W)	PB9_SL	Slew Rate Control for PB9 IO.
24 (R/W)	PB8_SL	Slew Rate Control for PB8 IO.
23 (R/W)	PB7_SL	Slew Rate Control for PB7 IO.
22 (R/W)	PB6_SL	Slew Rate Control for PB6 IO.
21 (R/W)	PB5_SL	Slew Rate Control for PB5 IO.
20 (R/W)	PB4_SL	Slew Rate Control for PB4 IO.
19 (R/W)	PB3_SL	Slew Rate Control for PB3 IO.
18 (R/W)	PB2_SL	Slew Rate Control for PB2 IO.
17 (R/W)	PB1_SL	Slew Rate Control for PB1 IO.
16 (R/W)	PB0_SL	Slew Rate Control for PB0 IO.
15 (R/W)	PA15_SL	Slew Rate Control for PA15 IO.

Table 13-81: PADS_PORT_AB_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	PA14_SL	Slew Rate Control for PA14 IO.
13 (R/W)	PA13_SL	Slew Rate Control for PA13 IO.
12 (R/W)	PA12_SL	Slew Rate Control for PA12 IO.
11 (R/W)	PA11_SL	Slew Rate Control for PA11 IO.
10 (R/W)	PA10_SL	Slew Rate Control for PA10 IO.
9 (R/W)	PA9_SL	Slew Rate Control for PA9 IO.
8 (R/W)	PA8_SL	Slew Rate Control for PA8 IO.
7 (R/W)	PA7_SL	Slew Rate Control for PA7 IO.
6 (R/W)	PA6_SL	Slew Rate Control for PA6 IO.
5 (R/W)	PA5_SL	Slew Rate Control for PA5 IO.
4 (R/W)	PA4_SL	Slew Rate Control for PA4 IO.
3 (R/W)	PA3_SL	Slew Rate Control for PA3 IO.
2 (R/W)	PA2_SL	Slew Rate Control for PA2 IO.

PORTA and PORTB Strong Pull Up Control Register

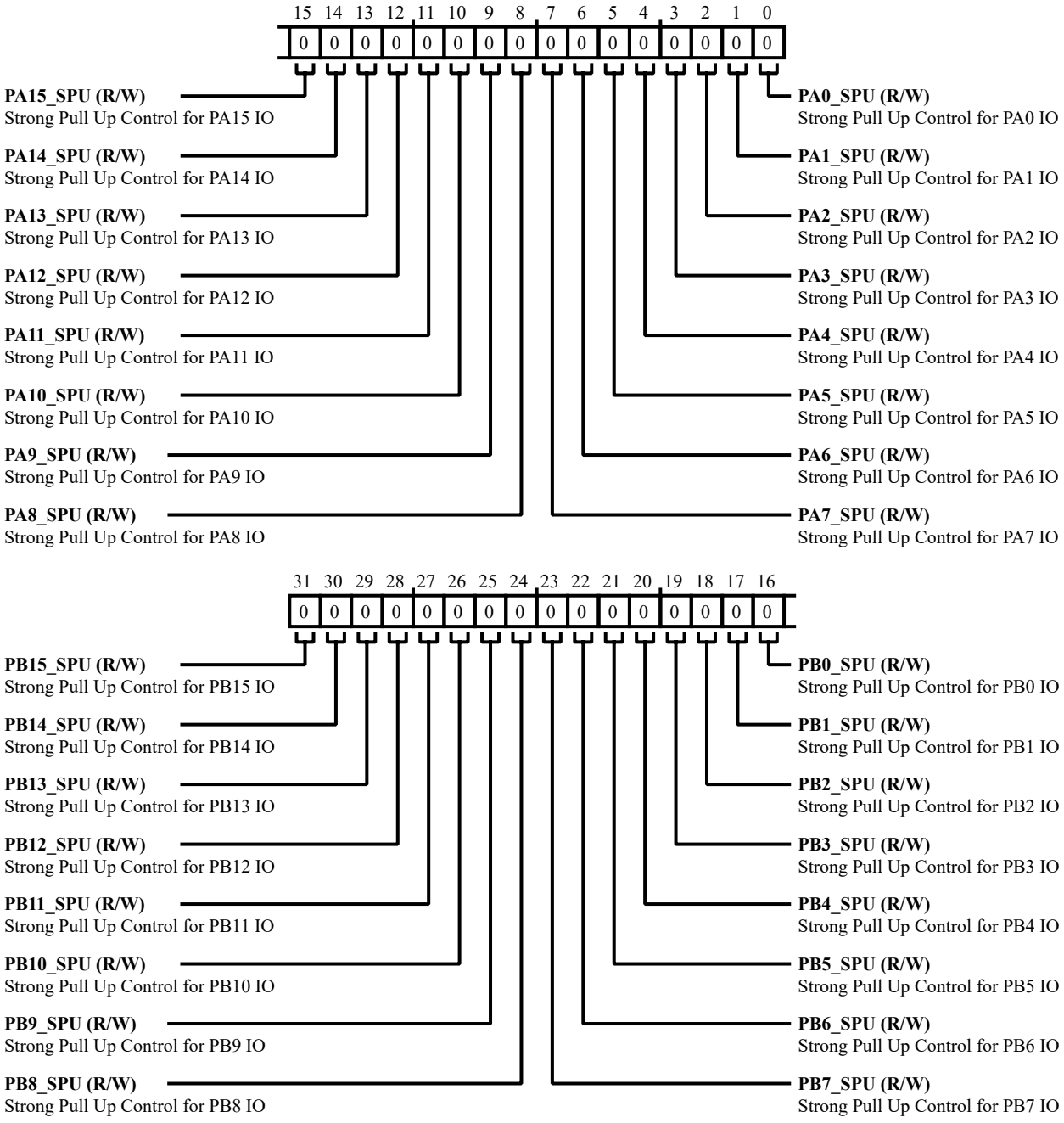


Figure 13-77: PADS_PORT_AB_SPU Register Diagram

Table 13-82: PADS_PORT_AB_SPU Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PB15_SPU	Strong Pull Up Control for PB15 IO.
30 (R/W)	PB14_SPU	Strong Pull Up Control for PB14 IO.
29 (R/W)	PB13_SPU	Strong Pull Up Control for PB13 IO.
28 (R/W)	PB12_SPU	Strong Pull Up Control for PB12 IO.
27 (R/W)	PB11_SPU	Strong Pull Up Control for PB11 IO.
26 (R/W)	PB10_SPU	Strong Pull Up Control for PB10 IO.
25 (R/W)	PB9_SPU	Strong Pull Up Control for PB9 IO.
24 (R/W)	PB8_SPU	Strong Pull Up Control for PB8 IO.
23 (R/W)	PB7_SPU	Strong Pull Up Control for PB7 IO.
22 (R/W)	PB6_SPU	Strong Pull Up Control for PB6 IO.
21 (R/W)	PB5_SPU	Strong Pull Up Control for PB5 IO.
20 (R/W)	PB4_SPU	Strong Pull Up Control for PB4 IO.
19 (R/W)	PB3_SPU	Strong Pull Up Control for PB3 IO.
18 (R/W)	PB2_SPU	Strong Pull Up Control for PB2 IO.
17 (R/W)	PB1_SPU	Strong Pull Up Control for PB1 IO.
16 (R/W)	PB0_SPU	Strong Pull Up Control for PB0 IO.

Table 13-82: PADS_PORT_AB_SPU Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PA15_SPU	Strong Pull Up Control for PA15 IO.
14 (R/W)	PA14_SPU	Strong Pull Up Control for PA14 IO.
13 (R/W)	PA13_SPU	Strong Pull Up Control for PA13 IO.
12 (R/W)	PA12_SPU	Strong Pull Up Control for PA12 IO.
11 (R/W)	PA11_SPU	Strong Pull Up Control for PA11 IO.
10 (R/W)	PA10_SPU	Strong Pull Up Control for PA10 IO.
9 (R/W)	PA9_SPU	Strong Pull Up Control for PA9 IO.
8 (R/W)	PA8_SPU	Strong Pull Up Control for PA8 IO.
7 (R/W)	PA7_SPU	Strong Pull Up Control for PA7 IO.
6 (R/W)	PA6_SPU	Strong Pull Up Control for PA6 IO.
5 (R/W)	PA5_SPU	Strong Pull Up Control for PA5 IO.
4 (R/W)	PA4_SPU	Strong Pull Up Control for PA4 IO.
3 (R/W)	PA3_SPU	Strong Pull Up Control for PA3 IO.
2 (R/W)	PA2_SPU	Strong Pull Up Control for PA2 IO.
1 (R/W)	PA1_SPU	Strong Pull Up Control for PA1 IO.
0 (R/W)	PA0_SPU	Strong Pull Up Control for PA0 IO.

PORTC and PORTD Pull Enable Register

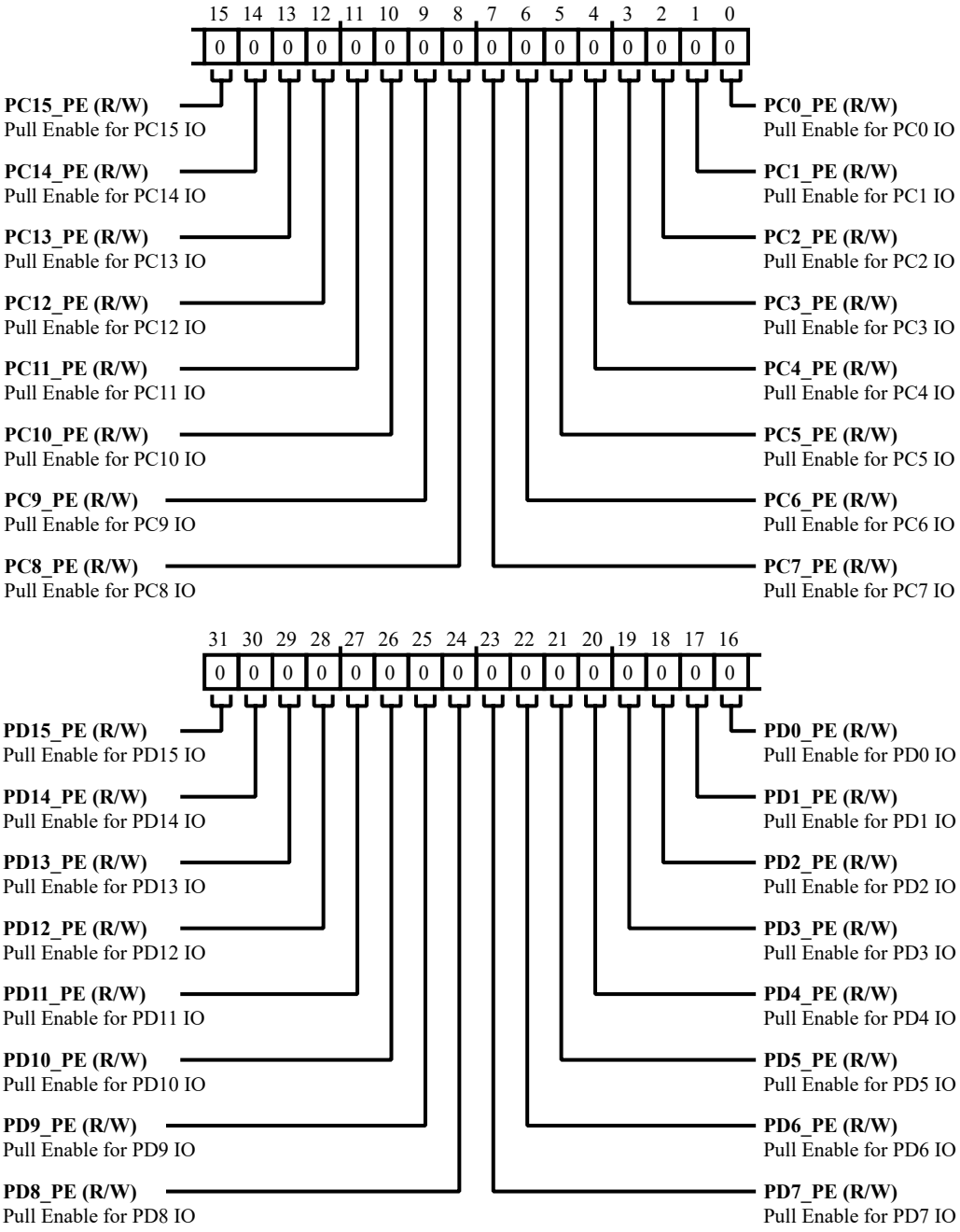


Figure 13-78: PADS_PORT_CD_PE Register Diagram

Table 13-83: PADS_PORT_CD_PE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PD15_PE	Pull Enable for PD15 IO.
30 (R/W)	PD14_PE	Pull Enable for PD14 IO.
29 (R/W)	PD13_PE	Pull Enable for PD13 IO.
28 (R/W)	PD12_PE	Pull Enable for PD12 IO.
27 (R/W)	PD11_PE	Pull Enable for PD11 IO.
26 (R/W)	PD10_PE	Pull Enable for PD10 IO.
25 (R/W)	PD9_PE	Pull Enable for PD9 IO.
24 (R/W)	PD8_PE	Pull Enable for PD8 IO.
23 (R/W)	PD7_PE	Pull Enable for PD7 IO.
22 (R/W)	PD6_PE	Pull Enable for PD6 IO.
21 (R/W)	PD5_PE	Pull Enable for PD5 IO.
20 (R/W)	PD4_PE	Pull Enable for PD4 IO.
19 (R/W)	PD3_PE	Pull Enable for PD3 IO.
18 (R/W)	PD2_PE	Pull Enable for PD2 IO.
17 (R/W)	PD1_PE	Pull Enable for PD1 IO.
16 (R/W)	PD0_PE	Pull Enable for PD0 IO.

Table 13-83: PADS_PORT_CD_PE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PC15_PE	Pull Enable for PC15 IO.
14 (R/W)	PC14_PE	Pull Enable for PC14 IO.
13 (R/W)	PC13_PE	Pull Enable for PC13 IO.
12 (R/W)	PC12_PE	Pull Enable for PC12 IO.
11 (R/W)	PC11_PE	Pull Enable for PC11 IO.
10 (R/W)	PC10_PE	Pull Enable for PC10 IO.
9 (R/W)	PC9_PE	Pull Enable for PC9 IO.
8 (R/W)	PC8_PE	Pull Enable for PC8 IO.
7 (R/W)	PC7_PE	Pull Enable for PC7 IO.
6 (R/W)	PC6_PE	Pull Enable for PC6 IO.
5 (R/W)	PC5_PE	Pull Enable for PC5 IO.
4 (R/W)	PC4_PE	Pull Enable for PC4 IO.
3 (R/W)	PC3_PE	Pull Enable for PC3 IO.
2 (R/W)	PC2_PE	Pull Enable for PC2 IO.
1 (R/W)	PC1_PE	Pull Enable for PC1 IO.
0 (R/W)	PC0_PE	Pull Enable for PC0 IO.

PORTC and PORTD Pull Selector Register

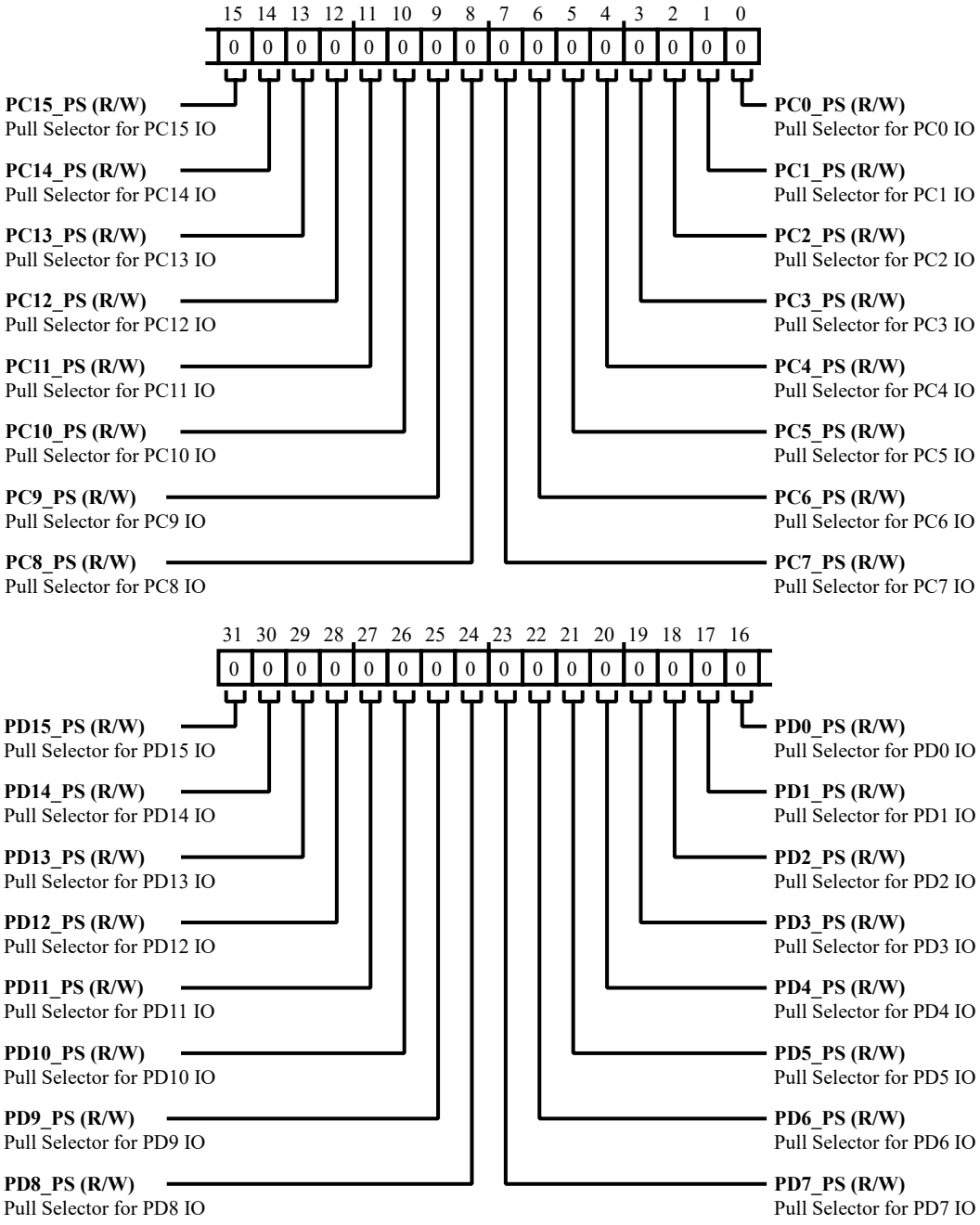


Figure 13-79: PADS_PORT_CD_PS Register Diagram

Table 13-84: PADS_PORT_CD_PS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PD15_PS	Pull Selector for PD15 IO.
30 (R/W)	PD14_PS	Pull Selector for PD14 IO.
29 (R/W)	PD13_PS	Pull Selector for PD13 IO.
28 (R/W)	PD12_PS	Pull Selector for PD12 IO.
27 (R/W)	PD11_PS	Pull Selector for PD11 IO.
26 (R/W)	PD10_PS	Pull Selector for PD10 IO.
25 (R/W)	PD9_PS	Pull Selector for PD9 IO.
24 (R/W)	PD8_PS	Pull Selector for PD8 IO.
23 (R/W)	PD7_PS	Pull Selector for PD7 IO.
22 (R/W)	PD6_PS	Pull Selector for PD6 IO.
21 (R/W)	PD5_PS	Pull Selector for PD5 IO.
20 (R/W)	PD4_PS	Pull Selector for PD4 IO.
19 (R/W)	PD3_PS	Pull Selector for PD3 IO.
18 (R/W)	PD2_PS	Pull Selector for PD2 IO.
17 (R/W)	PD1_PS	Pull Selector for PD1 IO.
16 (R/W)	PD0_PS	Pull Selector for PD0 IO.

Table 13-84: PADS_PORT_CD_PS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PC15_PS	Pull Selector for PC15 IO.
14 (R/W)	PC14_PS	Pull Selector for PC14 IO.
13 (R/W)	PC13_PS	Pull Selector for PC13 IO.
12 (R/W)	PC12_PS	Pull Selector for PC12 IO.
11 (R/W)	PC11_PS	Pull Selector for PC11 IO.
10 (R/W)	PC10_PS	Pull Selector for PC10 IO.
9 (R/W)	PC9_PS	Pull Selector for PC9 IO.
8 (R/W)	PC8_PS	Pull Selector for PC8 IO.
7 (R/W)	PC7_PS	Pull Selector for PC7 IO.
6 (R/W)	PC6_PS	Pull Selector for PC6 IO.
5 (R/W)	PC5_PS	Pull Selector for PC5 IO.
4 (R/W)	PC4_PS	Pull Selector for PC4 IO.
3 (R/W)	PC3_PS	Pull Selector for PC3 IO.
2 (R/W)	PC2_PS	Pull Selector for PC2 IO.
1 (R/W)	PC1_PS	Pull Selector for PC1 IO.
0 (R/W)	PC0_PS	Pull Selector for PC0 IO.

PORTC and PORTD Slew Rate Control Register

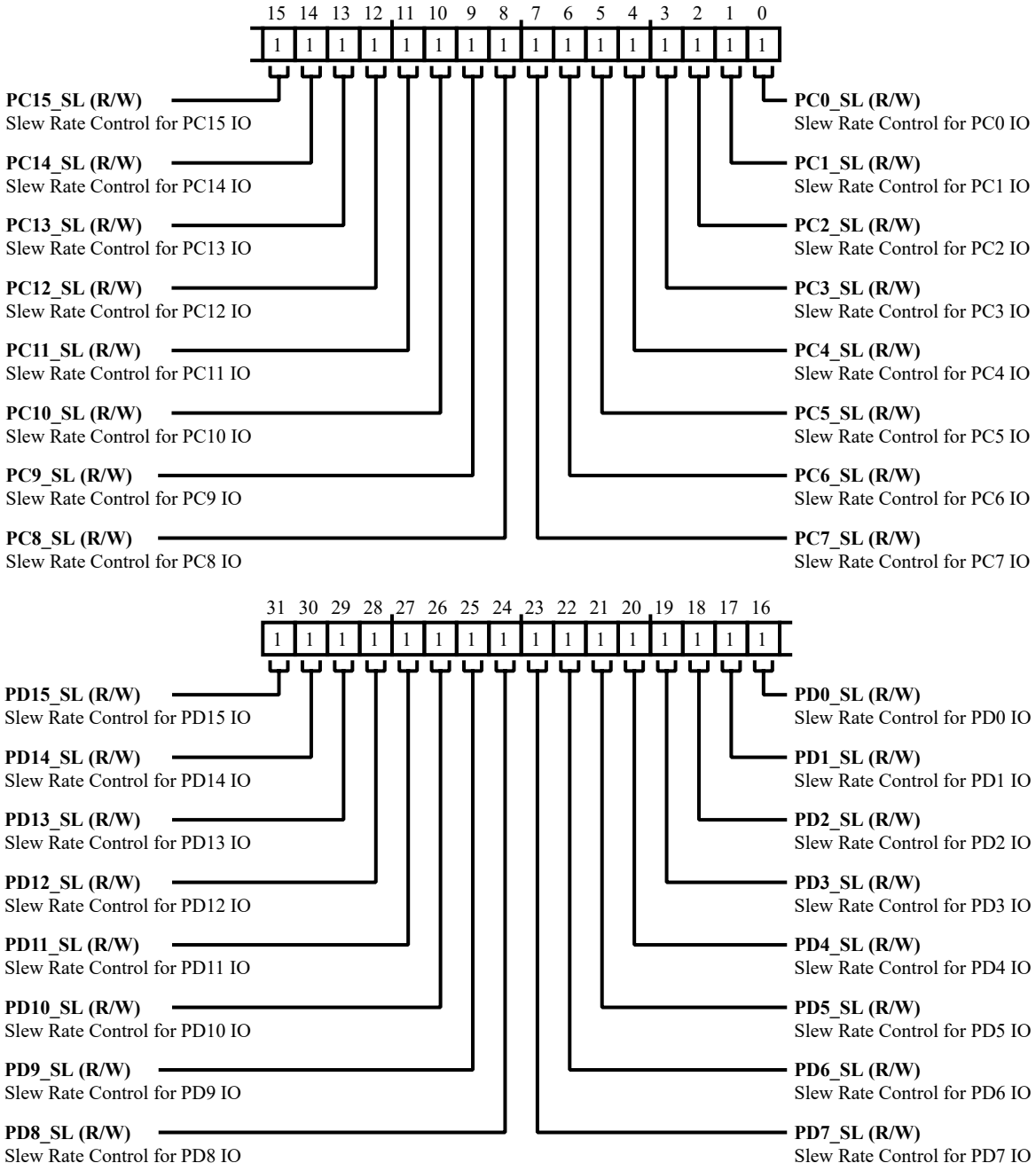


Figure 13-80: PADS_PORT_CD_SL Register Diagram

Table 13-85: PADS_PORT_CD_SL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PD15_SL	Slew Rate Control for PD15 IO.
30 (R/W)	PD14_SL	Slew Rate Control for PD14 IO.
29 (R/W)	PD13_SL	Slew Rate Control for PD13 IO.
28 (R/W)	PD12_SL	Slew Rate Control for PD12 IO.
27 (R/W)	PD11_SL	Slew Rate Control for PD11 IO.
26 (R/W)	PD10_SL	Slew Rate Control for PD10 IO.
25 (R/W)	PD9_SL	Slew Rate Control for PD9 IO.
24 (R/W)	PD8_SL	Slew Rate Control for PD8 IO.
23 (R/W)	PD7_SL	Slew Rate Control for PD7 IO.
22 (R/W)	PD6_SL	Slew Rate Control for PD6 IO.
21 (R/W)	PD5_SL	Slew Rate Control for PD5 IO.
20 (R/W)	PD4_SL	Slew Rate Control for PD4 IO.
19 (R/W)	PD3_SL	Slew Rate Control for PD3 IO.
18 (R/W)	PD2_SL	Slew Rate Control for PD2 IO.
17 (R/W)	PD1_SL	Slew Rate Control for PD1 IO.
16 (R/W)	PD0_SL	Slew Rate Control for PD0 IO.

Table 13-85: PADS_PORT_CD_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PC15_SL	Slew Rate Control for PC15 IO.
14 (R/W)	PC14_SL	Slew Rate Control for PC14 IO.
13 (R/W)	PC13_SL	Slew Rate Control for PC13 IO.
12 (R/W)	PC12_SL	Slew Rate Control for PC12 IO.
11 (R/W)	PC11_SL	Slew Rate Control for PC11 IO.
10 (R/W)	PC10_SL	Slew Rate Control for PC10 IO.
9 (R/W)	PC9_SL	Slew Rate Control for PC9 IO.
8 (R/W)	PC8_SL	Slew Rate Control for PC8 IO.
7 (R/W)	PC7_SL	Slew Rate Control for PC7 IO.
6 (R/W)	PC6_SL	Slew Rate Control for PC6 IO.
5 (R/W)	PC5_SL	Slew Rate Control for PC5 IO.
4 (R/W)	PC4_SL	Slew Rate Control for PC4 IO.
3 (R/W)	PC3_SL	Slew Rate Control for PC3 IO.
2 (R/W)	PC2_SL	Slew Rate Control for PC2 IO.
1 (R/W)	PC1_SL	Slew Rate Control for PC1 IO.
0 (R/W)	PC0_SL	Slew Rate Control for PC0 IO.

PORTC and PORTD Strong Pull Up Control Register

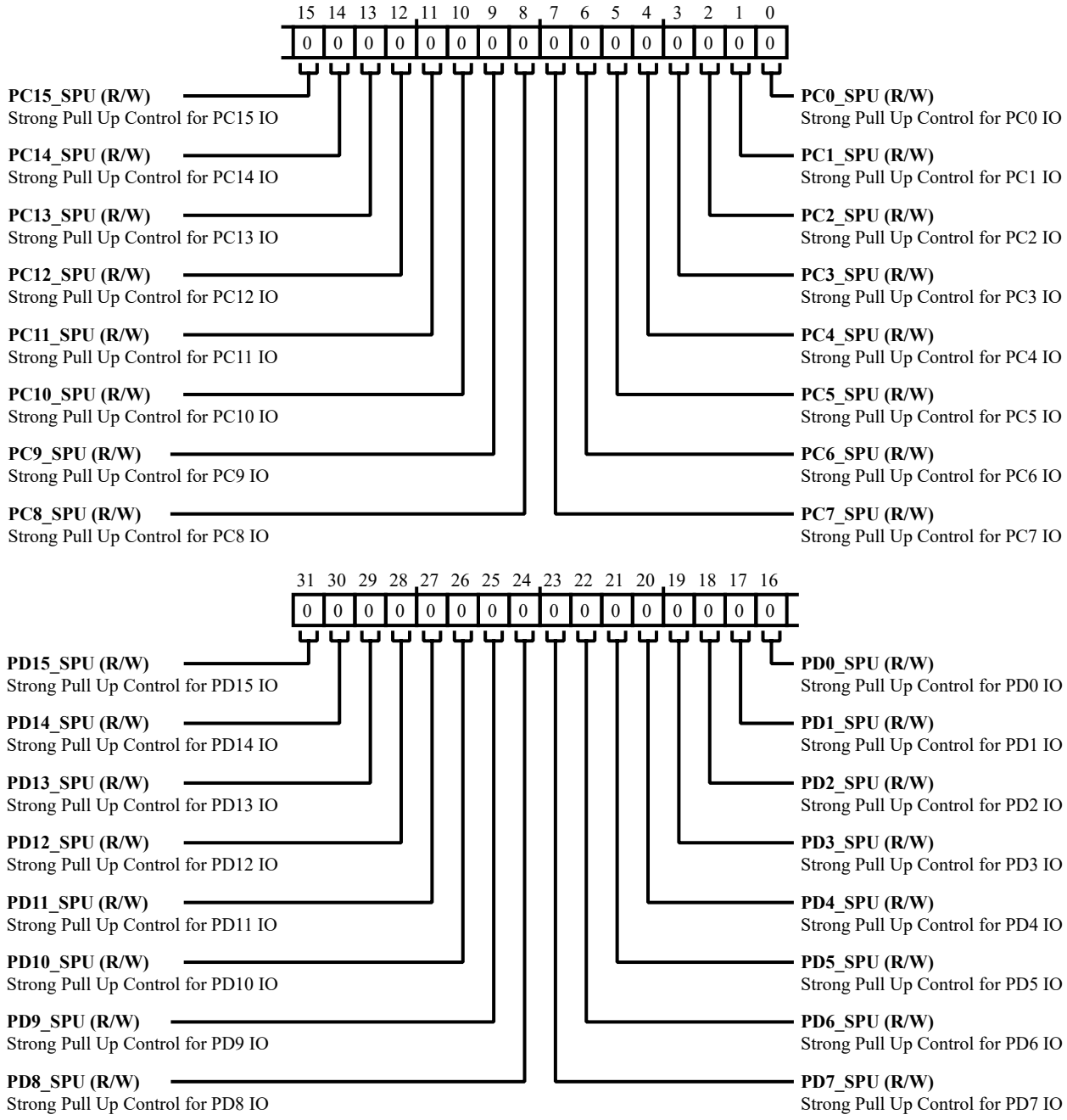


Figure 13-81: PADS_PORT_CD_SPU Register Diagram

Table 13-86: PADS_PORT_CD_SPU Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PD15_SPU	Strong Pull Up Control for PD15 IO.
30 (R/W)	PD14_SPU	Strong Pull Up Control for PD14 IO.
29 (R/W)	PD13_SPU	Strong Pull Up Control for PD13 IO.
28 (R/W)	PD12_SPU	Strong Pull Up Control for PD12 IO.
27 (R/W)	PD11_SPU	Strong Pull Up Control for PD11 IO.
26 (R/W)	PD10_SPU	Strong Pull Up Control for PD10 IO.
25 (R/W)	PD9_SPU	Strong Pull Up Control for PD9 IO.
24 (R/W)	PD8_SPU	Strong Pull Up Control for PD8 IO.
23 (R/W)	PD7_SPU	Strong Pull Up Control for PD7 IO.
22 (R/W)	PD6_SPU	Strong Pull Up Control for PD6 IO.
21 (R/W)	PD5_SPU	Strong Pull Up Control for PD5 IO.
20 (R/W)	PD4_SPU	Strong Pull Up Control for PD4 IO.
19 (R/W)	PD3_SPU	Strong Pull Up Control for PD3 IO.
18 (R/W)	PD2_SPU	Strong Pull Up Control for PD2 IO.
17 (R/W)	PD1_SPU	Strong Pull Up Control for PD1 IO.
16 (R/W)	PD0_SPU	Strong Pull Up Control for PD0 IO.

Table 13-86: PADS_PORT_CD_SPU Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PC15_SPU	Strong Pull Up Control for PC15 IO.
14 (R/W)	PC14_SPU	Strong Pull Up Control for PC14 IO.
13 (R/W)	PC13_SPU	Strong Pull Up Control for PC13 IO.
12 (R/W)	PC12_SPU	Strong Pull Up Control for PC12 IO.
11 (R/W)	PC11_SPU	Strong Pull Up Control for PC11 IO.
10 (R/W)	PC10_SPU	Strong Pull Up Control for PC10 IO.
9 (R/W)	PC9_SPU	Strong Pull Up Control for PC9 IO.
8 (R/W)	PC8_SPU	Strong Pull Up Control for PC8 IO.
7 (R/W)	PC7_SPU	Strong Pull Up Control for PC7 IO.
6 (R/W)	PC6_SPU	Strong Pull Up Control for PC6 IO.
5 (R/W)	PC5_SPU	Strong Pull Up Control for PC5 IO.
4 (R/W)	PC4_SPU	Strong Pull Up Control for PC4 IO.
3 (R/W)	PC3_SPU	Strong Pull Up Control for PC3 IO.
2 (R/W)	PC2_SPU	Strong Pull Up Control for PC2 IO.
1 (R/W)	PC1_SPU	Strong Pull Up Control for PC1 IO.
0 (R/W)	PC0_SPU	Strong Pull Up Control for PC0 IO.

PORTE and PORTF Pull Enable Register

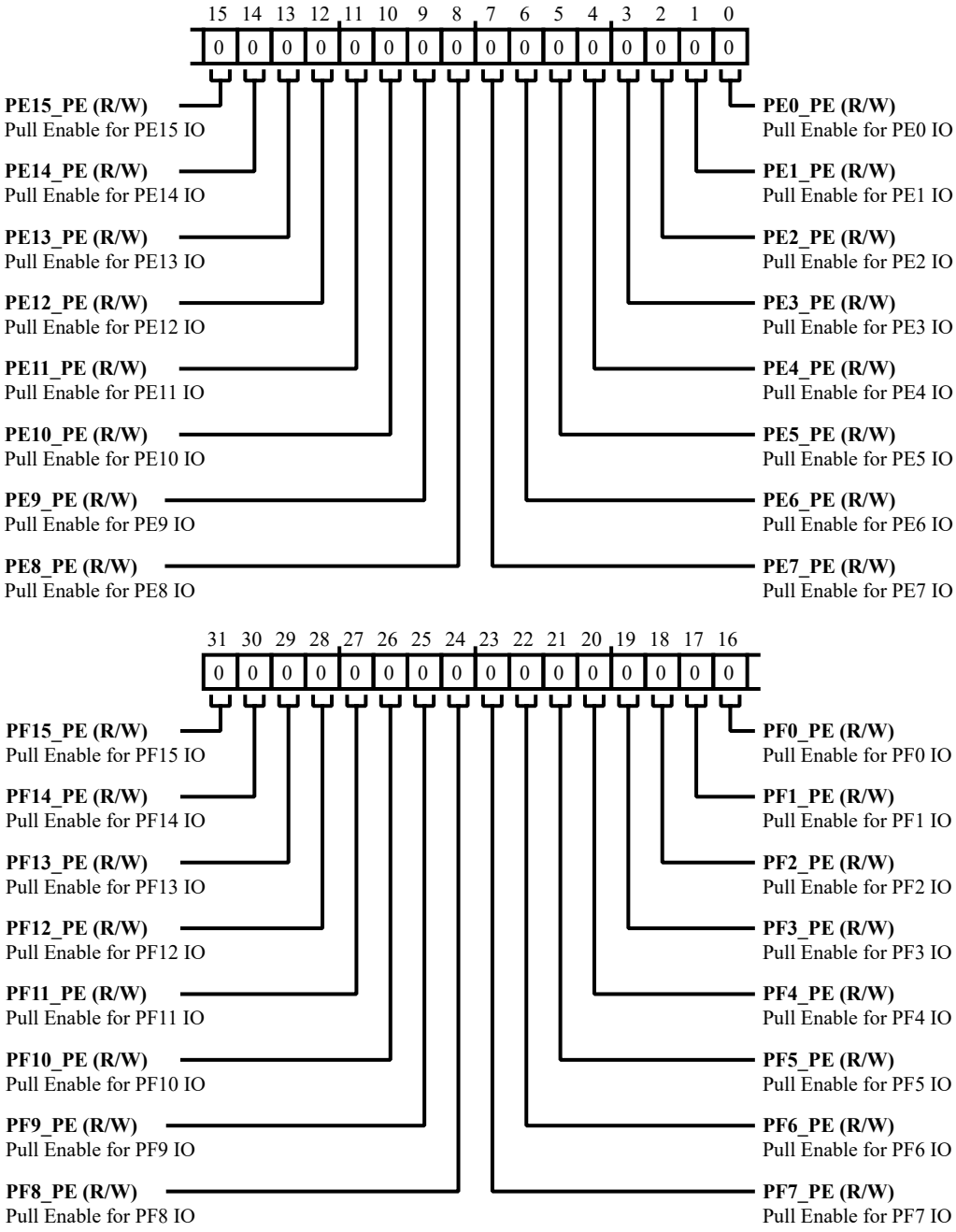


Figure 13-82: PADS_PORT_EF_PE Register Diagram

Table 13-87: PADS_PORT_EF_PE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PF15_PE	Pull Enable for PF15 IO.
30 (R/W)	PF14_PE	Pull Enable for PF14 IO.
29 (R/W)	PF13_PE	Pull Enable for PF13 IO.
28 (R/W)	PF12_PE	Pull Enable for PF12 IO.
27 (R/W)	PF11_PE	Pull Enable for PF11 IO.
26 (R/W)	PF10_PE	Pull Enable for PF10 IO.
25 (R/W)	PF9_PE	Pull Enable for PF9 IO.
24 (R/W)	PF8_PE	Pull Enable for PF8 IO.
23 (R/W)	PF7_PE	Pull Enable for PF7 IO.
22 (R/W)	PF6_PE	Pull Enable for PF6 IO.
21 (R/W)	PF5_PE	Pull Enable for PF5 IO.
20 (R/W)	PF4_PE	Pull Enable for PF4 IO.
19 (R/W)	PF3_PE	Pull Enable for PF3 IO.
18 (R/W)	PF2_PE	Pull Enable for PF2 IO.
17 (R/W)	PF1_PE	Pull Enable for PF1 IO.
16 (R/W)	PF0_PE	Pull Enable for PF0 IO.

Table 13-87: PADS_PORT_EF_PE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PE15_PE	Pull Enable for PE15 IO.
14 (R/W)	PE14_PE	Pull Enable for PE14 IO.
13 (R/W)	PE13_PE	Pull Enable for PE13 IO.
12 (R/W)	PE12_PE	Pull Enable for PE12 IO.
11 (R/W)	PE11_PE	Pull Enable for PE11 IO.
10 (R/W)	PE10_PE	Pull Enable for PE10 IO.
9 (R/W)	PE9_PE	Pull Enable for PE9 IO.
8 (R/W)	PE8_PE	Pull Enable for PE8 IO.
7 (R/W)	PE7_PE	Pull Enable for PE7 IO.
6 (R/W)	PE6_PE	Pull Enable for PE6 IO.
5 (R/W)	PE5_PE	Pull Enable for PE5 IO.
4 (R/W)	PE4_PE	Pull Enable for PE4 IO.
3 (R/W)	PE3_PE	Pull Enable for PE3 IO.
2 (R/W)	PE2_PE	Pull Enable for PE2 IO.
1 (R/W)	PE1_PE	Pull Enable for PE1 IO.
0 (R/W)	PE0_PE	Pull Enable for PE0 IO.

PORTE and PORTF Pull Selector Register

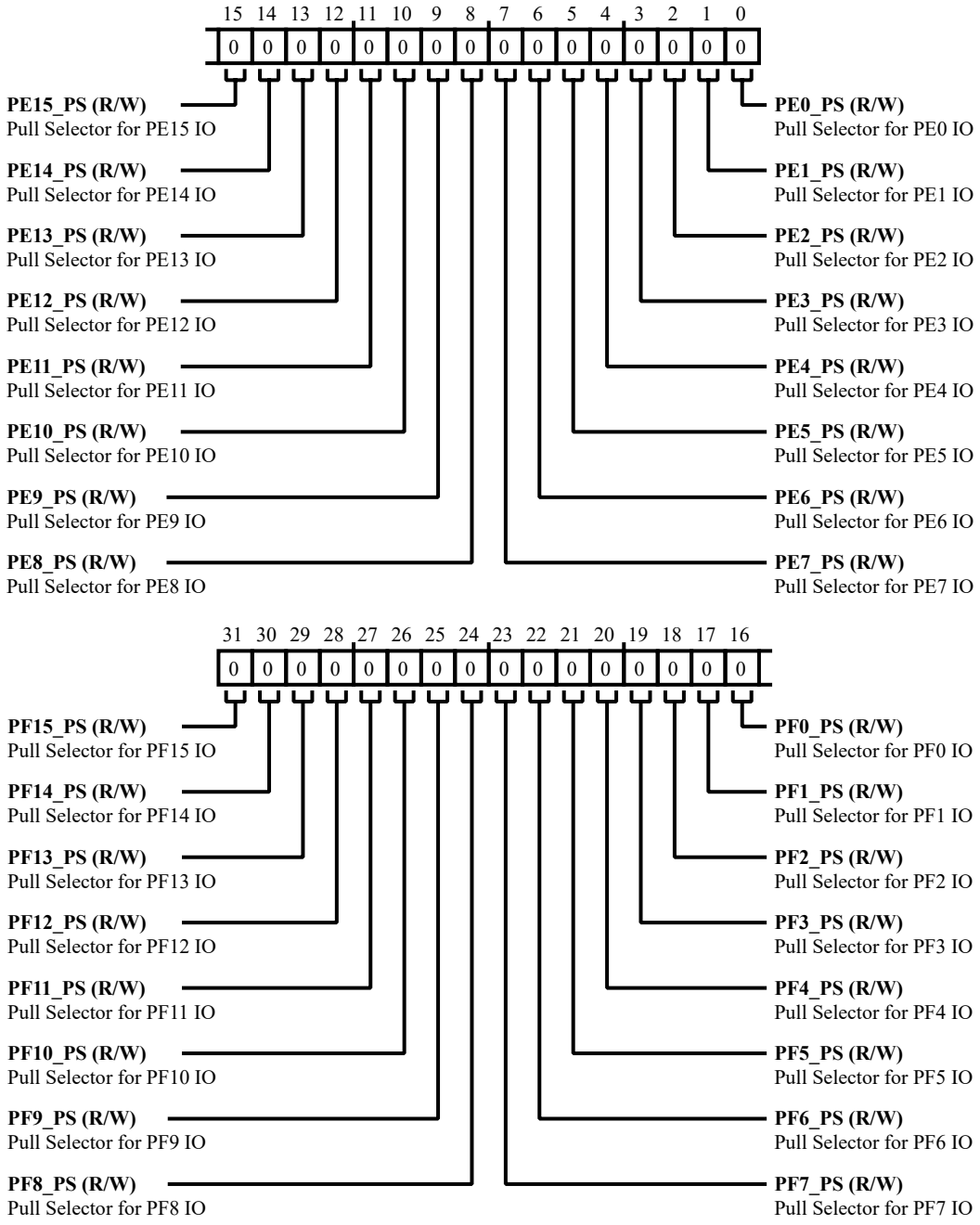


Figure 13-83: PADS_PORT_EF_PS Register Diagram

Table 13-88: PADS_PORT_EF_PS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PF15_PS	Pull Selector for PF15 IO.
30 (R/W)	PF14_PS	Pull Selector for PF14 IO.
29 (R/W)	PF13_PS	Pull Selector for PF13 IO.
28 (R/W)	PF12_PS	Pull Selector for PF12 IO.
27 (R/W)	PF11_PS	Pull Selector for PF11 IO.
26 (R/W)	PF10_PS	Pull Selector for PF10 IO.
25 (R/W)	PF9_PS	Pull Selector for PF9 IO.
24 (R/W)	PF8_PS	Pull Selector for PF8 IO.
23 (R/W)	PF7_PS	Pull Selector for PF7 IO.
22 (R/W)	PF6_PS	Pull Selector for PF6 IO.
21 (R/W)	PF5_PS	Pull Selector for PF5 IO.
20 (R/W)	PF4_PS	Pull Selector for PF4 IO.
19 (R/W)	PF3_PS	Pull Selector for PF3 IO.
18 (R/W)	PF2_PS	Pull Selector for PF2 IO.
17 (R/W)	PF1_PS	Pull Selector for PF1 IO.
16 (R/W)	PF0_PS	Pull Selector for PF0 IO.

Table 13-88: PADS_PORT_EF_PS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PE15_PS	Pull Selector for PE15 IO.
14 (R/W)	PE14_PS	Pull Selector for PE14 IO.
13 (R/W)	PE13_PS	Pull Selector for PE13 IO.
12 (R/W)	PE12_PS	Pull Selector for PE12 IO.
11 (R/W)	PE11_PS	Pull Selector for PE11 IO.
10 (R/W)	PE10_PS	Pull Selector for PE10 IO.
9 (R/W)	PE9_PS	Pull Selector for PE9 IO.
8 (R/W)	PE8_PS	Pull Selector for PE8 IO.
7 (R/W)	PE7_PS	Pull Selector for PE7 IO.
6 (R/W)	PE6_PS	Pull Selector for PE6 IO.
5 (R/W)	PE5_PS	Pull Selector for PE5 IO.
4 (R/W)	PE4_PS	Pull Selector for PE4 IO.
3 (R/W)	PE3_PS	Pull Selector for PE3 IO.
2 (R/W)	PE2_PS	Pull Selector for PE2 IO.
1 (R/W)	PE1_PS	Pull Selector for PE1 IO.
0 (R/W)	PE0_PS	Pull Selector for PE0 IO.

PORTE and PORTF Slew Rate Control Register

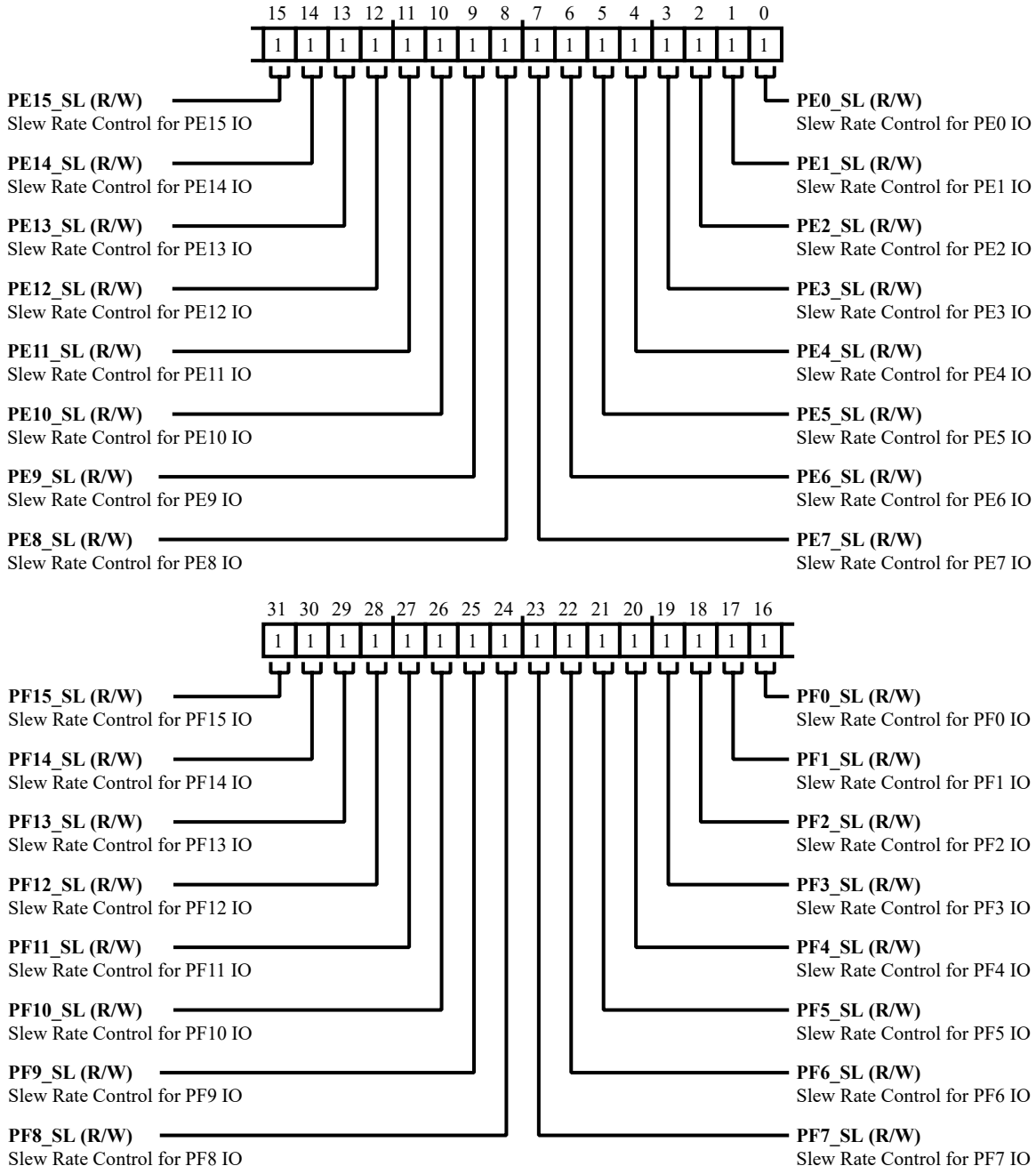


Figure 13-84: PADS_PORT_EF_SL Register Diagram

Table 13-89: PADS_PORT_EF_SL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PF15_SL	Slew Rate Control for PF15 IO.
30 (R/W)	PF14_SL	Slew Rate Control for PF14 IO.
29 (R/W)	PF13_SL	Slew Rate Control for PF13 IO.
28 (R/W)	PF12_SL	Slew Rate Control for PF12 IO.
27 (R/W)	PF11_SL	Slew Rate Control for PF11 IO.
26 (R/W)	PF10_SL	Slew Rate Control for PF10 IO.
25 (R/W)	PF9_SL	Slew Rate Control for PF9 IO.
24 (R/W)	PF8_SL	Slew Rate Control for PF8 IO.
23 (R/W)	PF7_SL	Slew Rate Control for PF7 IO.
22 (R/W)	PF6_SL	Slew Rate Control for PF6 IO.
21 (R/W)	PF5_SL	Slew Rate Control for PF5 IO.
20 (R/W)	PF4_SL	Slew Rate Control for PF4 IO.
19 (R/W)	PF3_SL	Slew Rate Control for PF3 IO.
18 (R/W)	PF2_SL	Slew Rate Control for PF2 IO.
17 (R/W)	PF1_SL	Slew Rate Control for PF1 IO.
16 (R/W)	PF0_SL	Slew Rate Control for PF0 IO.

Table 13-89: PADS_PORT_EF_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PE15_SL	Slew Rate Control for PE15 IO.
14 (R/W)	PE14_SL	Slew Rate Control for PE14 IO.
13 (R/W)	PE13_SL	Slew Rate Control for PE13 IO.
12 (R/W)	PE12_SL	Slew Rate Control for PE12 IO.
11 (R/W)	PE11_SL	Slew Rate Control for PE11 IO.
10 (R/W)	PE10_SL	Slew Rate Control for PE10 IO.
9 (R/W)	PE9_SL	Slew Rate Control for PE9 IO.
8 (R/W)	PE8_SL	Slew Rate Control for PE8 IO.
7 (R/W)	PE7_SL	Slew Rate Control for PE7 IO.
6 (R/W)	PE6_SL	Slew Rate Control for PE6 IO.
5 (R/W)	PE5_SL	Slew Rate Control for PE5 IO.
4 (R/W)	PE4_SL	Slew Rate Control for PE4 IO.
3 (R/W)	PE3_SL	Slew Rate Control for PE3 IO.
2 (R/W)	PE2_SL	Slew Rate Control for PE2 IO.
1 (R/W)	PE1_SL	Slew Rate Control for PE1 IO.
0 (R/W)	PE0_SL	Slew Rate Control for PE0 IO.

PORTE and PORTF Strong Pull Up Control Register

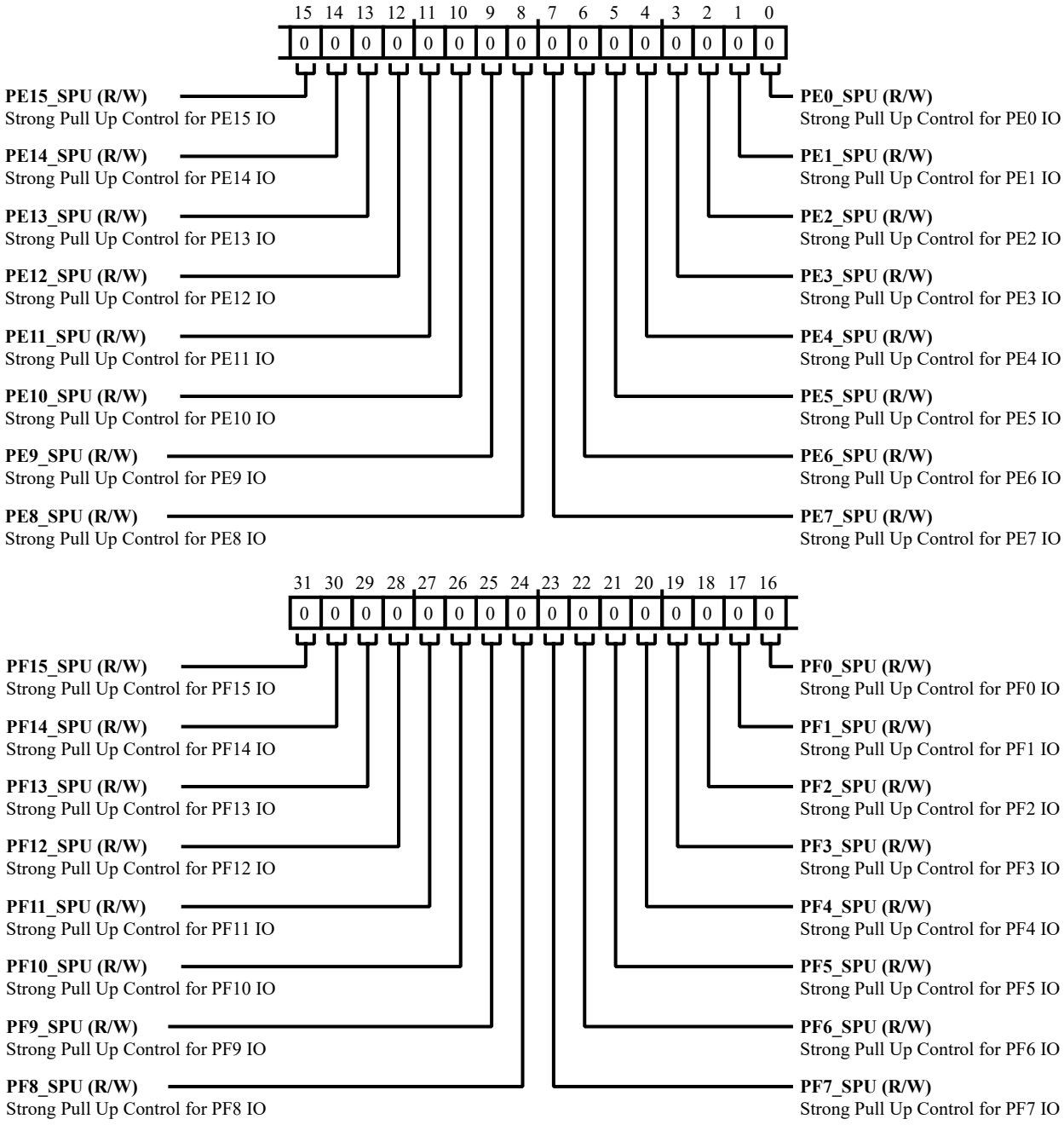


Figure 13-85: PADS_PORT_EF_SPU Register Diagram

Table 13-90: PADS_PORT_EF_SPU Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PF15_SPU	Strong Pull Up Control for PF15 IO.
30 (R/W)	PF14_SPU	Strong Pull Up Control for PF14 IO.
29 (R/W)	PF13_SPU	Strong Pull Up Control for PF13 IO.
28 (R/W)	PF12_SPU	Strong Pull Up Control for PF12 IO.
27 (R/W)	PF11_SPU	Strong Pull Up Control for PF11 IO.
26 (R/W)	PF10_SPU	Strong Pull Up Control for PF10 IO.
25 (R/W)	PF9_SPU	Strong Pull Up Control for PF9 IO.
24 (R/W)	PF8_SPU	Strong Pull Up Control for PF8 IO.
23 (R/W)	PF7_SPU	Strong Pull Up Control for PF7 IO.
22 (R/W)	PF6_SPU	Strong Pull Up Control for PF6 IO.
21 (R/W)	PF5_SPU	Strong Pull Up Control for PF5 IO.
20 (R/W)	PF4_SPU	Strong Pull Up Control for PF4 IO.
19 (R/W)	PF3_SPU	Strong Pull Up Control for PF3 IO.
18 (R/W)	PF2_SPU	Strong Pull Up Control for PF2 IO.
17 (R/W)	PF1_SPU	Strong Pull Up Control for PF1 IO.
16 (R/W)	PF0_SPU	Strong Pull Up Control for PF0 IO.

Table 13-90: PADS_PORT_EF_SPU Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PE15_SPU	Strong Pull Up Control for PE15 IO.
14 (R/W)	PE14_SPU	Strong Pull Up Control for PE14 IO.
13 (R/W)	PE13_SPU	Strong Pull Up Control for PE13 IO.
12 (R/W)	PE12_SPU	Strong Pull Up Control for PE12 IO.
11 (R/W)	PE11_SPU	Strong Pull Up Control for PE11 IO.
10 (R/W)	PE10_SPU	Strong Pull Up Control for PE10 IO.
9 (R/W)	PE9_SPU	Strong Pull Up Control for PE9 IO.
8 (R/W)	PE8_SPU	Strong Pull Up Control for PE8 IO.
7 (R/W)	PE7_SPU	Strong Pull Up Control for PE7 IO.
6 (R/W)	PE6_SPU	Strong Pull Up Control for PE6 IO.
5 (R/W)	PE5_SPU	Strong Pull Up Control for PE5 IO.
4 (R/W)	PE4_SPU	Strong Pull Up Control for PE4 IO.
3 (R/W)	PE3_SPU	Strong Pull Up Control for PE3 IO.
2 (R/W)	PE2_SPU	Strong Pull Up Control for PE2 IO.
1 (R/W)	PE1_SPU	Strong Pull Up Control for PE1 IO.
0 (R/W)	PE0_SPU	Strong Pull Up Control for PE0 IO.

PORTG and PORTH Pull Enable Register

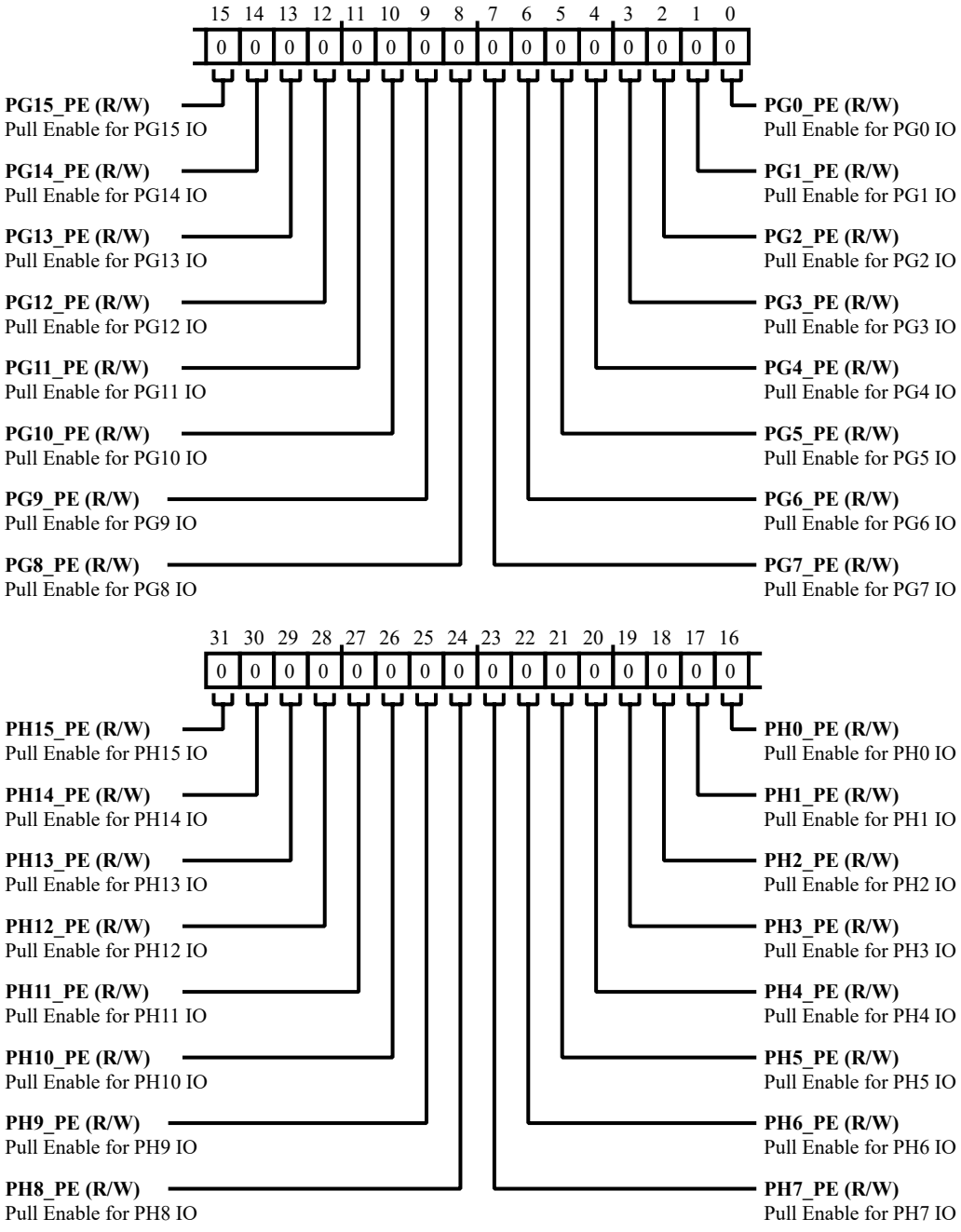


Figure 13-86: PADS_PORT_GH_PE Register Diagram

Table 13-91: PADS_PORT_GH_PE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PH15_PE	Pull Enable for PH15 IO.
30 (R/W)	PH14_PE	Pull Enable for PH14 IO.
29 (R/W)	PH13_PE	Pull Enable for PH13 IO.
28 (R/W)	PH12_PE	Pull Enable for PH12 IO.
27 (R/W)	PH11_PE	Pull Enable for PH11 IO.
26 (R/W)	PH10_PE	Pull Enable for PH10 IO.
25 (R/W)	PH9_PE	Pull Enable for PH9 IO.
24 (R/W)	PH8_PE	Pull Enable for PH8 IO.
23 (R/W)	PH7_PE	Pull Enable for PH7 IO.
22 (R/W)	PH6_PE	Pull Enable for PH6 IO.
21 (R/W)	PH5_PE	Pull Enable for PH5 IO.
20 (R/W)	PH4_PE	Pull Enable for PH4 IO.
19 (R/W)	PH3_PE	Pull Enable for PH3 IO.
18 (R/W)	PH2_PE	Pull Enable for PH2 IO.
17 (R/W)	PH1_PE	Pull Enable for PH1 IO.
16 (R/W)	PH0_PE	Pull Enable for PH0 IO.

Table 13-91: PADS_PORT_GH_PE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PG15_PE	Pull Enable for PG15 IO.
14 (R/W)	PG14_PE	Pull Enable for PG14 IO.
13 (R/W)	PG13_PE	Pull Enable for PG13 IO.
12 (R/W)	PG12_PE	Pull Enable for PG12 IO.
11 (R/W)	PG11_PE	Pull Enable for PG11 IO.
10 (R/W)	PG10_PE	Pull Enable for PG10 IO.
9 (R/W)	PG9_PE	Pull Enable for PG9 IO.
8 (R/W)	PG8_PE	Pull Enable for PG8 IO.
7 (R/W)	PG7_PE	Pull Enable for PG7 IO.
6 (R/W)	PG6_PE	Pull Enable for PG6 IO.
5 (R/W)	PG5_PE	Pull Enable for PG5 IO.
4 (R/W)	PG4_PE	Pull Enable for PG4 IO.
3 (R/W)	PG3_PE	Pull Enable for PG3 IO.
2 (R/W)	PG2_PE	Pull Enable for PG2 IO.
1 (R/W)	PG1_PE	Pull Enable for PG1 IO.
0 (R/W)	PG0_PE	Pull Enable for PG0 IO.

PORTG and PORTH Pull Selector Register

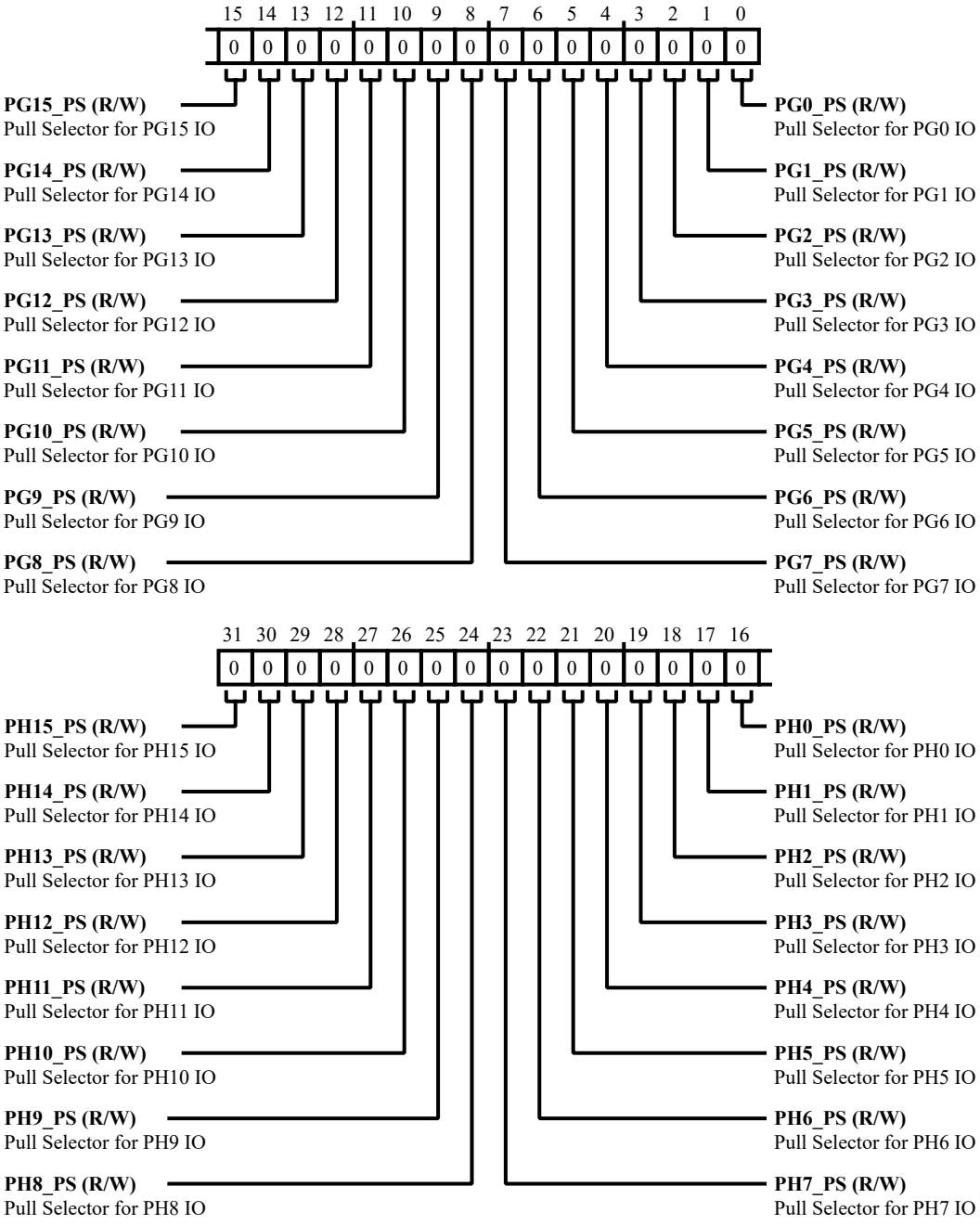


Figure 13-87: PADS_PORT_GH_PS Register Diagram

Table 13-92: PADS_PORT_GH_PS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PH15_PS	Pull Selector for PH15 IO.
30 (R/W)	PH14_PS	Pull Selector for PH14 IO.
29 (R/W)	PH13_PS	Pull Selector for PH13 IO.
28 (R/W)	PH12_PS	Pull Selector for PH12 IO.
27 (R/W)	PH11_PS	Pull Selector for PH11 IO.
26 (R/W)	PH10_PS	Pull Selector for PH10 IO.
25 (R/W)	PH9_PS	Pull Selector for PH9 IO.
24 (R/W)	PH8_PS	Pull Selector for PH8 IO.
23 (R/W)	PH7_PS	Pull Selector for PH7 IO.
22 (R/W)	PH6_PS	Pull Selector for PH6 IO.
21 (R/W)	PH5_PS	Pull Selector for PH5 IO.
20 (R/W)	PH4_PS	Pull Selector for PH4 IO.
19 (R/W)	PH3_PS	Pull Selector for PH3 IO.
18 (R/W)	PH2_PS	Pull Selector for PH2 IO.
17 (R/W)	PH1_PS	Pull Selector for PH1 IO.
16 (R/W)	PH0_PS	Pull Selector for PH0 IO.

Table 13-92: PADS_PORT_GH_PS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PG15_PS	Pull Selector for PG15 IO.
14 (R/W)	PG14_PS	Pull Selector for PG14 IO.
13 (R/W)	PG13_PS	Pull Selector for PG13 IO.
12 (R/W)	PG12_PS	Pull Selector for PG12 IO.
11 (R/W)	PG11_PS	Pull Selector for PG11 IO.
10 (R/W)	PG10_PS	Pull Selector for PG10 IO.
9 (R/W)	PG9_PS	Pull Selector for PG9 IO.
8 (R/W)	PG8_PS	Pull Selector for PG8 IO.
7 (R/W)	PG7_PS	Pull Selector for PG7 IO.
6 (R/W)	PG6_PS	Pull Selector for PG6 IO.
5 (R/W)	PG5_PS	Pull Selector for PG5 IO.
4 (R/W)	PG4_PS	Pull Selector for PG4 IO.
3 (R/W)	PG3_PS	Pull Selector for PG3 IO.
2 (R/W)	PG2_PS	Pull Selector for PG2 IO.
1 (R/W)	PG1_PS	Pull Selector for PG1 IO.
0 (R/W)	PG0_PS	Pull Selector for PG0 IO.

PORTG and PORTH Slew Rate Control Register

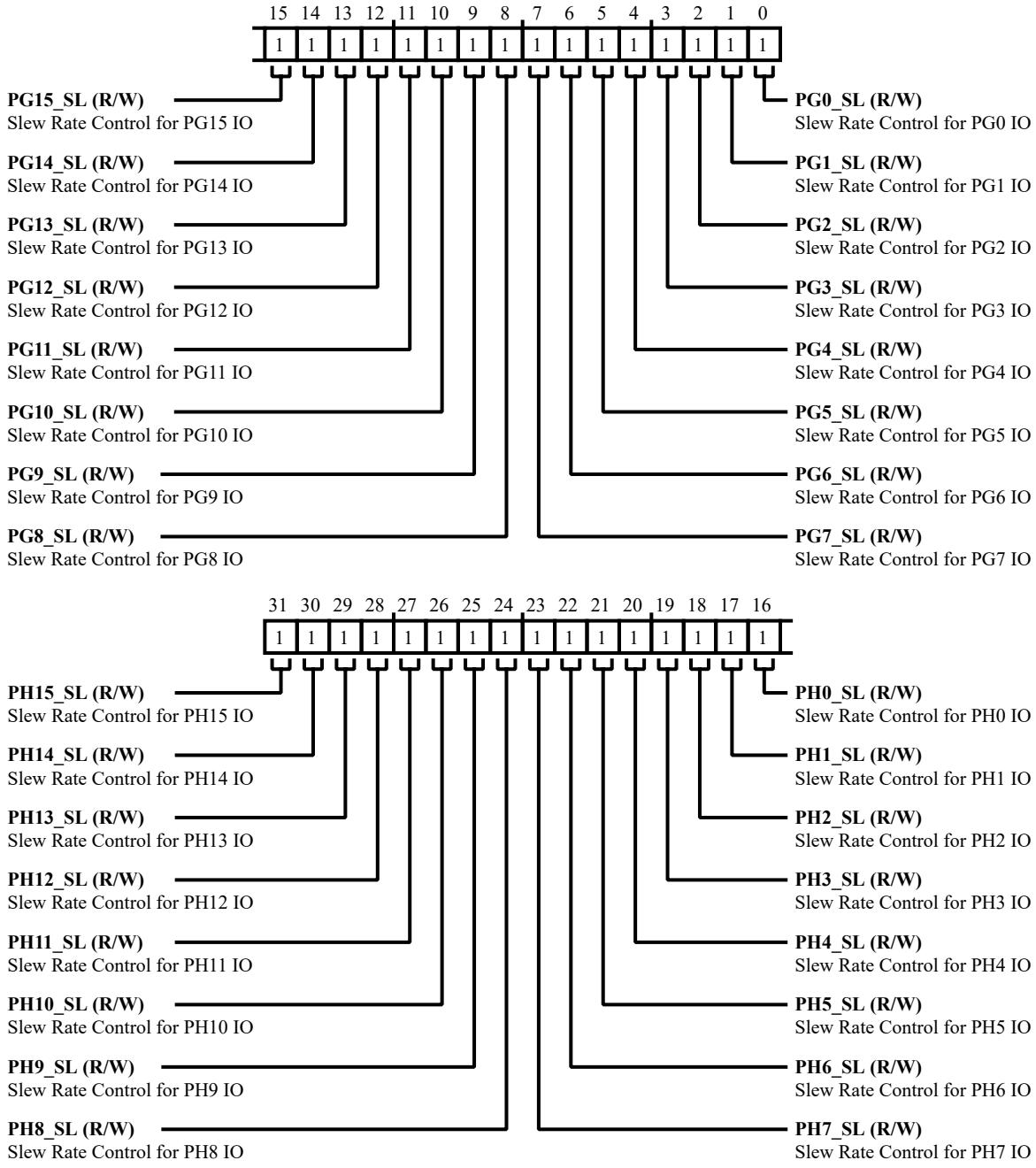


Figure 13-88: PADS_PORT_GH_SL Register Diagram

Table 13-93: PADS_PORT_GH_SL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PH15_SL	Slew Rate Control for PH15 IO.
30 (R/W)	PH14_SL	Slew Rate Control for PH14 IO.
29 (R/W)	PH13_SL	Slew Rate Control for PH13 IO.
28 (R/W)	PH12_SL	Slew Rate Control for PH12 IO.
27 (R/W)	PH11_SL	Slew Rate Control for PH11 IO.
26 (R/W)	PH10_SL	Slew Rate Control for PH10 IO.
25 (R/W)	PH9_SL	Slew Rate Control for PH9 IO.
24 (R/W)	PH8_SL	Slew Rate Control for PH8 IO.
23 (R/W)	PH7_SL	Slew Rate Control for PH7 IO.
22 (R/W)	PH6_SL	Slew Rate Control for PH6 IO.
21 (R/W)	PH5_SL	Slew Rate Control for PH5 IO.
20 (R/W)	PH4_SL	Slew Rate Control for PH4 IO.
19 (R/W)	PH3_SL	Slew Rate Control for PH3 IO.
18 (R/W)	PH2_SL	Slew Rate Control for PH2 IO.
17 (R/W)	PH1_SL	Slew Rate Control for PH1 IO.
16 (R/W)	PH0_SL	Slew Rate Control for PH0 IO.

Table 13-93: PADS_PORT_GH_SL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PG15_SL	Slew Rate Control for PG15 IO.
14 (R/W)	PG14_SL	Slew Rate Control for PG14 IO.
13 (R/W)	PG13_SL	Slew Rate Control for PG13 IO.
12 (R/W)	PG12_SL	Slew Rate Control for PG12 IO.
11 (R/W)	PG11_SL	Slew Rate Control for PG11 IO.
10 (R/W)	PG10_SL	Slew Rate Control for PG10 IO.
9 (R/W)	PG9_SL	Slew Rate Control for PG9 IO.
8 (R/W)	PG8_SL	Slew Rate Control for PG8 IO.
7 (R/W)	PG7_SL	Slew Rate Control for PG7 IO.
6 (R/W)	PG6_SL	Slew Rate Control for PG6 IO.
5 (R/W)	PG5_SL	Slew Rate Control for PG5 IO.
4 (R/W)	PG4_SL	Slew Rate Control for PG4 IO.
3 (R/W)	PG3_SL	Slew Rate Control for PG3 IO.
2 (R/W)	PG2_SL	Slew Rate Control for PG2 IO.
1 (R/W)	PG1_SL	Slew Rate Control for PG1 IO.
0 (R/W)	PG0_SL	Slew Rate Control for PG0 IO.

PORTG and PORTH Strong Pull Up Control Register

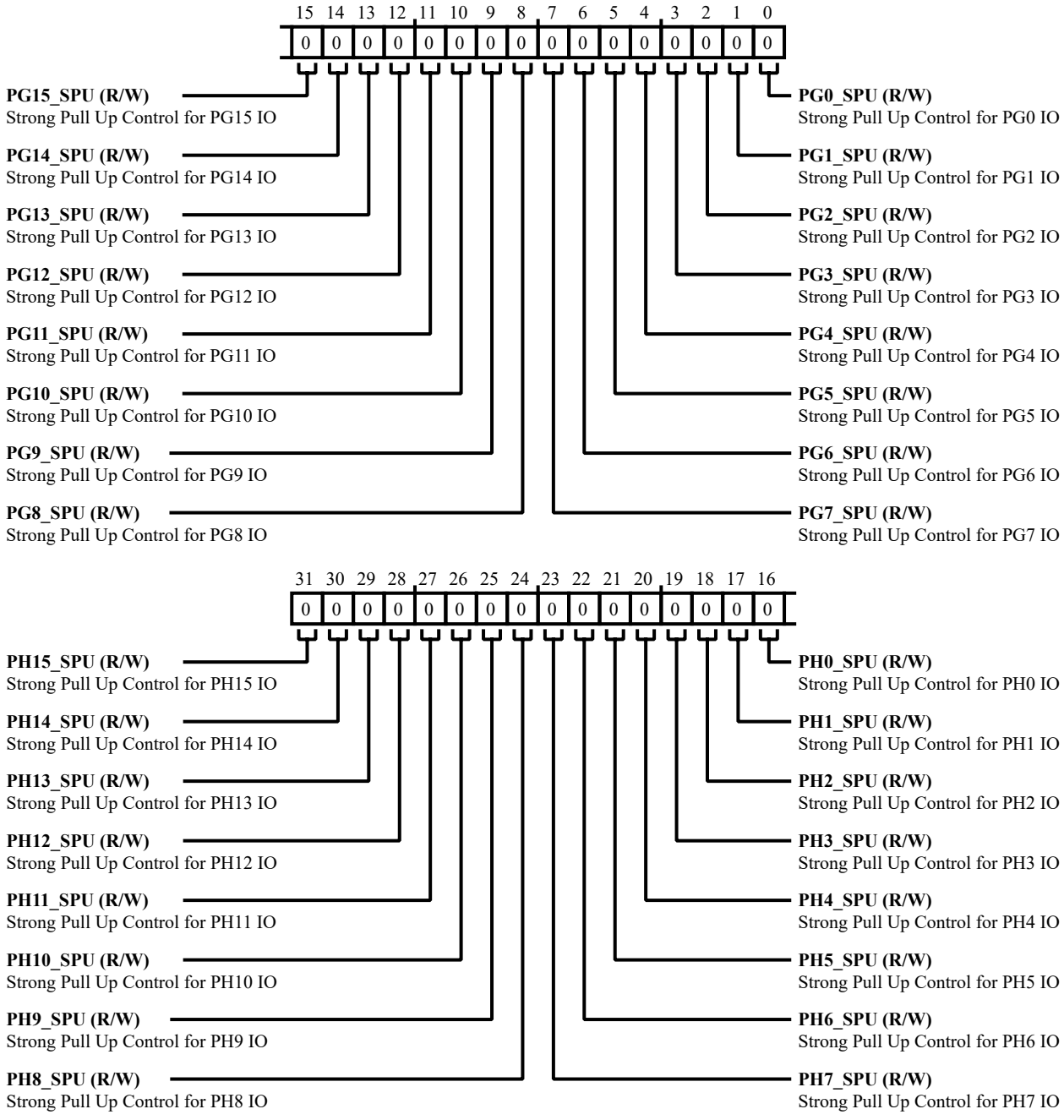


Figure 13-89: PADS_PORT_GH_SPU Register Diagram

Table 13-94: PADS_PORT_GH_SPU Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	PH15_SPU	Strong Pull Up Control for PH15 IO.
30 (R/W)	PH14_SPU	Strong Pull Up Control for PH14 IO.
29 (R/W)	PH13_SPU	Strong Pull Up Control for PH13 IO.
28 (R/W)	PH12_SPU	Strong Pull Up Control for PH12 IO.
27 (R/W)	PH11_SPU	Strong Pull Up Control for PH11 IO.
26 (R/W)	PH10_SPU	Strong Pull Up Control for PH10 IO.
25 (R/W)	PH9_SPU	Strong Pull Up Control for PH9 IO.
24 (R/W)	PH8_SPU	Strong Pull Up Control for PH8 IO.
23 (R/W)	PH7_SPU	Strong Pull Up Control for PH7 IO.
22 (R/W)	PH6_SPU	Strong Pull Up Control for PH6 IO.
21 (R/W)	PH5_SPU	Strong Pull Up Control for PH5 IO.
20 (R/W)	PH4_SPU	Strong Pull Up Control for PH4 IO.
19 (R/W)	PH3_SPU	Strong Pull Up Control for PH3 IO.
18 (R/W)	PH2_SPU	Strong Pull Up Control for PH2 IO.
17 (R/W)	PH1_SPU	Strong Pull Up Control for PH1 IO.
16 (R/W)	PH0_SPU	Strong Pull Up Control for PH0 IO.

Table 13-94: PADS_PORT_GH_SPU Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PG15_SPU	Strong Pull Up Control for PG15 IO.
14 (R/W)	PG14_SPU	Strong Pull Up Control for PG14 IO.
13 (R/W)	PG13_SPU	Strong Pull Up Control for PG13 IO.
12 (R/W)	PG12_SPU	Strong Pull Up Control for PG12 IO.
11 (R/W)	PG11_SPU	Strong Pull Up Control for PG11 IO.
10 (R/W)	PG10_SPU	Strong Pull Up Control for PG10 IO.
9 (R/W)	PG9_SPU	Strong Pull Up Control for PG9 IO.
8 (R/W)	PG8_SPU	Strong Pull Up Control for PG8 IO.
7 (R/W)	PG7_SPU	Strong Pull Up Control for PG7 IO.
6 (R/W)	PG6_SPU	Strong Pull Up Control for PG6 IO.
5 (R/W)	PG5_SPU	Strong Pull Up Control for PG5 IO.
4 (R/W)	PG4_SPU	Strong Pull Up Control for PG4 IO.
3 (R/W)	PG3_SPU	Strong Pull Up Control for PG3 IO.
2 (R/W)	PG2_SPU	Strong Pull Up Control for PG2 IO.
1 (R/W)	PG1_SPU	Strong Pull Up Control for PG1 IO.
0 (R/W)	PG0_SPU	Strong Pull Up Control for PG0 IO.

Status Register

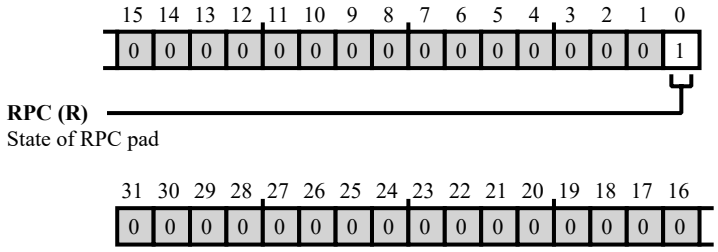


Figure 13-90: PADS_STAT Register Diagram

Table 13-95: PADS_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	RPC	State of RPC pad.

14 Thermal Monitoring Unit (TMU)

The TMU provides on-chip temperature measurement which is important in applications that have substantial power consumption. The TMU is integrated into the processor die and digital infrastructure using an MMR-based system access to measure the die temperature variations in real-time.

TMU Features

The TMU supports the following features:

- On-chip temperature sensing
- Programmable over-temperature and under-temperature limits
- Programmable conversion rate
- Averaging feature available
- Temperature gain and offset correction options
- Uses dedicated channel of HADC for conversion
- Programmable blanking and refresh period for TMU conversion

TMU Functional Description

Following sections provide the functional description of TMU.

ADSP-2184x TMU Register List

Thermal monitoring unit

Table 14-1: ADSP-2184x TMU Register List

Name	Description
TMU_ALRT_LIM_HI	Alert High Limit Register
TMU_ALRT_LIM_LO	Alert Low Limit Register
TMU_AVG	Averaging Register

Table 14-1: ADSP-2184x TMU Register List (Continued)

Name	Description
TMU_CNV_BLANK	Temperature Conversion Blank Register
TMU_CTL	TMU Control Register
TMU_FLT_LIM_HI	Fault High Limit Register
TMU_FLT_LIM_LO	Fault Low Limit Register
TMU_GAIN	Gain Value Register
TMU_IMSK	Interrupt Mask Register
TMU_OFFSET	Offset Register
TMU_REFR_CNTR	Temperature Refresh Counter
TMU_STAT	Status Register
TMU_TEMP	Temperature Value Register

ADSP-2184x TMU Interrupt List

Table 14-2: ADSP-2184x TMU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
347	TMU0_FAULT	TMU0 Fault		
348	TMU0_ALERT	TMU0 Alert		

ADSP-2184x TMU Trigger List

Table 14-3: ADSP-2184x TMU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
189	TMU0_FAULT	TMU0 TM0 Fault Event	
190	TMU0_ALERT	TMU0 TM0 Alert Event	

Table 14-4: ADSP-2184x TMU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

TMU Definitions

The following definitions are helpful when using the TMU module.

Thermal Diode

A special type of diode whose electrical properties change with the temperature in a defined way.

DTM

Dynamic thermal management is a set of techniques that adapt the run-time behavior of a processor to achieve the highest performance under thermal constraints.

TMU Block Diagram

The ***Thermal Monitoring Unit Block Diagram*** shows the main block inside TMU.

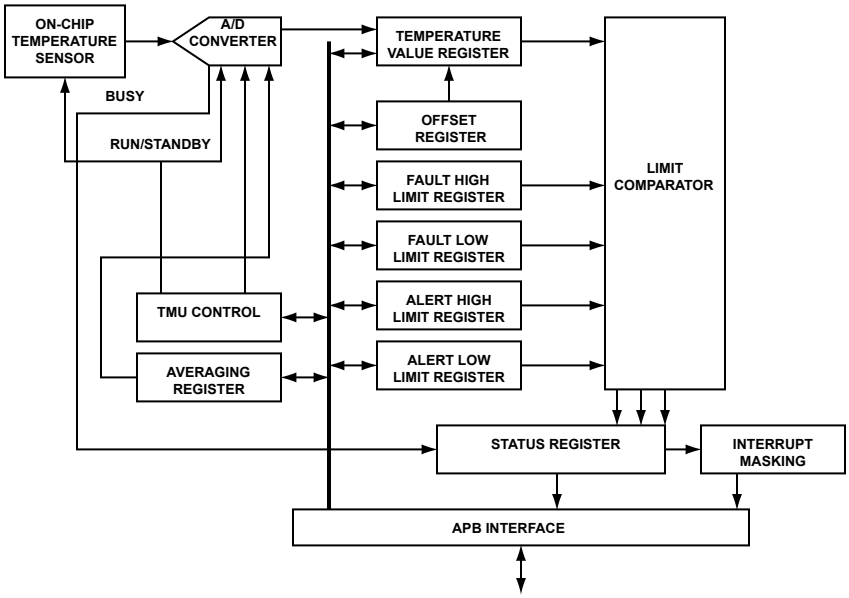


Figure 14-1: Thermal Monitoring Unit Block Diagram

TMU Architectural Concepts

The following sections provide information on the architecture and system integration of the TMU.

System Integration

Dynamic thermal management (DTM) techniques adapt the run-time behavior of a processor to achieve the highest performance under thermal constraints. One of the most important aspects of DTM is to capture the run-time variations in the temperature caused by power consumption variations due to workload changes.

The accuracy of thermal measurements directly affects the efficiency of thermal management as well as the performance of the processor. Temperature estimations lower or higher than the actual temperature may cause late or early activation of DTM techniques.

- Late activation of DTM can result in degraded reliability because the temperature may exceed the designated thresholds.
- Early activation of DTM can have significant impact on performance.

The TMU is an analog thermal sensor that consists of a temperature-sensing diode, a calibrated reference current source, and a current comparator. The sensor placement error is one of the most important sources of inaccuracy in values obtained from thermal sensors.

The TMU module provides the temperature monitoring capability to the chip and implements thermal management in the end system. It provides many features which ensure the minimum load on the software and also minimal or no external components for a flexible temperature monitoring system.

Digital Thermometer

The TMU functions as a digital thermometer with an MMR-based system access. The on-die temperature value is measured and digitized through an A/D converter. The temperature value is stored and updated periodically in an MMR register. The TMU can be configured to generate an interrupt as it crosses the upper temperature limit (`TMU_FLT_LIM_HI` register). A thermal event is also routed to the interrupt port of the sensor. The event is then routed to the SEC, to ensure that core intervention is not required in the event of overheating.

Temperature Sensor Averaging

The temperature sensor averaging feature enhances the accuracy of the temperature measurements. This feature is enabled by default after power-up. In this mode, the averaging reduces the effect of noise on the temperature result. The temperature is measured each time a conversion is performed. A moving average method is used to determine the result in the temperature value register. The total time to measure a temperature channel is typically 1 ms.

TMU and HADC

The TMU modules uses one of the ADC channels to digitize the temperature reading. Therefore, the HADC module is shared by the TMU. If the TMU is not enabled, the HADC operates at its full throughput. TMU can be enabled periodically or permanently. In periodic enable mode, it is enabled once in a specific refresh period. After enabling, the TMU measures the temperature for a specific blanking period. During the blanking period, the TMU uses the ADC for temperature measurement, so that the ADC is not available for data conversion.

The typical value of blanking period is 1.1×10^6 system clock (SCLK) cycles.

The typical value of refresh period is 21×10^6 SCLK cycles.

So, for every 21×10^6 cycles, 1.1×10^6 cycles are used for temperature measurement. Program these values for blanking period and refresh period through the TMU registers.

When TMU is enabled, the HADC modules is not available for external input conversion. This state is indicated by the `HADC_STAT.TMUHADC_BUSY` status bit. All the requests for HADC conversion are ignored when `HADC_STAT.TMUHADC_BUSY` bit is high.

In the periodic enable mode, the ongoing HADC conversion is completed activating the TMU to measure temperature as shown. If HADC is in autoscan mode, the TMU is activated after completing the ongoing sequence. If HADC is in fixed-conversion mode, all the fixed conversions are completed and followed by TMU activation.

TMU Event Control

The TMU generates different events depending on the state of the TMU temperature measurement and the different thresholds set in thresh hold registers. These events are reported in the TMU_STAT register as shown below. It can generate an event for each of following conditions.

- The fault high limit is configured in the TMU_FLT_LIM_HI register. The interrupt is generated when the temperature value is greater than or equal to this value.
- The alert high limit (0,1,2) is configured in the TMU_ALRT_LIM_HI register. The interrupt is generated when the temperature value is greater than or equal to these value.
- The fault low limit is configured in the TMU_FLT_LIM_LO register. The TMU_STAT.FLTLO status bit is set when the temperature value is less than or equal to this value.
- The alert low limit is configured in the TMU_ALRT_LIM_LO register. The TMU_STAT.FLTLO status bit is set when the temperature value is less than or equal to this value.

Interrupts and status conditions can be masked (disabled) or unmasked (enabled) by setting and clearing bits in the TMU_IMSK register.

NOTE: Only TMU_FLT_LIM_HI and TMU_ALRT_LIM_HI generate an interrupt to core. Other events only result in status change in the TMU_STAT register.

Status and Error Signals

When the measured temperature value exceeds the high or low limits that are configured in the TMU_FLT_LIM_HI/TMU_FLT_LIM_LO and the TMU_ALRT_LIM_HI/TMU_ALRT_LIM_LO registers, corresponding thermal events are triggered.

The fault and alert events are sent to the core through the status register (TMU_STAT). Both the TMU_STAT.FLTHI and the TMU_STAT.ALRT1HI/TMU_STAT.ALRT2HI/TMU_STAT.ALRT3HI bits are sticky bits and are cleared by a W1C operation by the core.

The alert and fault registers can be programmed through the MMR bus interface as shown in the TMU block diagram. A write into the TMU_FLT_LIM_HI/TMU_FLT_LIM_LO and TMU_ALRT_LIM_HI/TMU_ALRT_LIM_LO registers or the TMU_TEMP register is not allowed when the events are being triggered.

TMU Programming Guidelines

The following section provides basic programming information for the TMU module.

To get the best performance and accuracy from the TMU, initialize the TMU_GAIN and TMU_OFFSET registers before enabling the module. After the registers are programmed, the rest of TMU initialization can take place. This includes setting up the fault and alert limits and then enabling the TMU.

Reading Temperature Data

After enabling the TMU, the temperature data can only be read after a finite period of time referred to as startup time.

If the HADC module is used, apply the following mechanism:

1. Initialize the HADC module.
2. Wait for the ADC ready interrupt.
3. Perform TMU conversions.

The minimum time required before reading the first valid data is ~100us for a typical SCLK frequency of 125Mhz and using the default register values.

If the HADC module is not used, complete the following steps

1. Initialize the HADC module (dummy initialization).
2. Wait for the ADC ready interrupt.
3. Perform TMU conversions.

TMU Calibration

Calibration settings are comprised of gain and offset settings. These settings must be programmed into the corresponding device registers before the temperature is read from the TMU temperature value register. The calibration setting ensures that the reported temperature approximates the actual die temperature (within the error specified in the data sheet). The specified accuracy of the reported temperature cannot be met if gain and offset settings are not used.

The TMU configuration software programs gain and offset values as part of the TMU initialization process. The gain value is programmed in the `TMU_GAIN.VALUE` field. The offset value is programmed in the `TMU_OFFSET.VALUE` bit field. The *TMU Calibration Settings* table provides the calibration setting for the processor.

Table 14-5: TMU Calibration Settings

Bit Field	Programmed Value
`TMU_GAIN.VALUE`	TBD
`TMU_OFFSET.VALUE`	TBD

NOTE: Refer to the product data sheet for the expected TMU accuracy achieved using the settings.

NOTE: The calibration settings provide expected TMU performance at temperatures above room temperature. The accuracy is not guaranteed for temperatures below room temperature.

ADSP-2184x TMU Register Descriptions

Thermal monitoring unit (TMU) contains the following registers.

Table 14-6: ADSP-2184x TMU Register List

Name	Description
TMU_ALRT_LIM_HI	Alert High Limit Register
TMU_ALRT_LIM_LO	Alert Low Limit Register
TMU_AVG	Averaging Register
TMU_CNV_BLANK	Temperature Conversion Blank Register
TMU_CTL	TMU Control Register
TMU_FLT_LIM_HI	Fault High Limit Register
TMU_FLT_LIM_LO	Fault Low Limit Register
TMU_GAIN	Gain Value Register
TMU_IMSK	Interrupt Mask Register
TMU_OFFSET	Offset Register
TMU_REFR_CNTR	Temperature Refresh Counter
TMU_STAT	Status Register
TMU_TEMP	Temperature Value Register

Alert High Limit Register

The TMU_ALRT_LIM_HI register configures the alert temperature high limit.

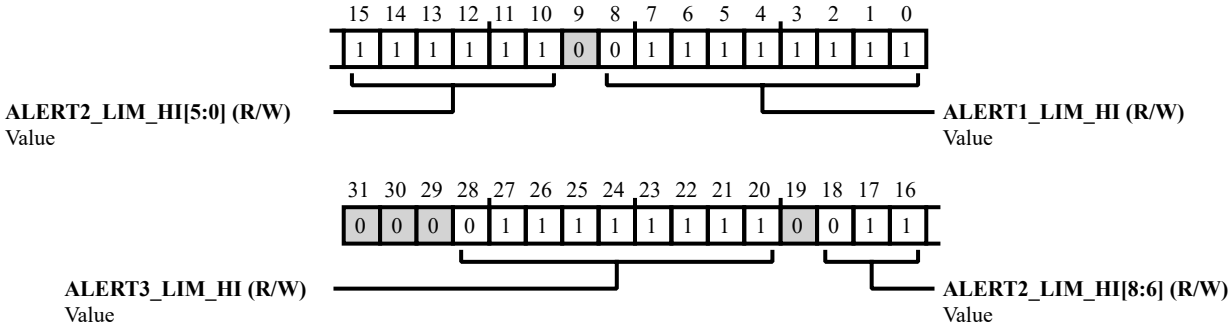


Figure 14-2: TMU_ALRT_LIM_HI Register Diagram

Table 14-7: TMU_ALRT_LIM_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:20 (R/W)	ALERT3_LIM_HI	Value. The TMU_ALRT_LIM_HI.ALERT3_LIM_HI bit field configures the alert temperature high limit as an integer value stored in two's complement format. If the temperature value is greater than or equal to the high limit, the TMU_STAT.ALRT3HI bit is set.
18:10 (R/W)	ALERT2_LIM_HI	Value. The TMU_ALRT_LIM_HI.ALERT2_LIM_HI bit field configures the alert temperature high limit as an integer value stored in two's complement format. If the temperature value is greater than or equal to the high limit, the TMU_STAT.ALRT2HI bit is set.
8:0 (R/W)	ALERT1_LIM_HI	Value. The TMU_ALRT_LIM_HI.ALERT1_LIM_HI bit field configures the alert temperature high limit as an integer value stored in two's complement format. If the temperature value is greater than or equal to the high limit, the TMU_STAT.ALRT1HI bit is set.

Alert Low Limit Register

The TMU_ALRT_LIM_LO register configures the alert temperature low limit.

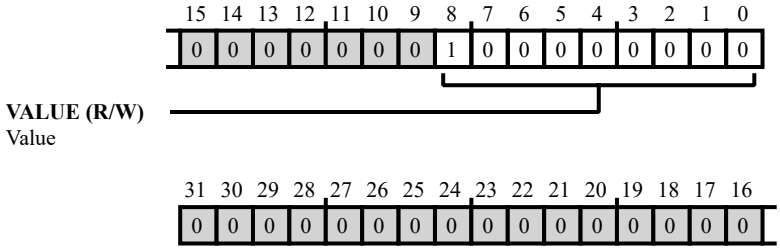


Figure 14-3: TMU_ALRT_LIM_LO Register Diagram

Table 14-8: TMU_ALRT_LIM_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_ALRT_LIM_LO.VALUE bit field configures the alert temperature low limit as an integer value stored in two's complement format. If the temperature value is less than or equal to the low limit the TMU_STAT.ALRTLO bit is set.

Averaging Register

The TMU_AVG register enables averaging on the TMU.

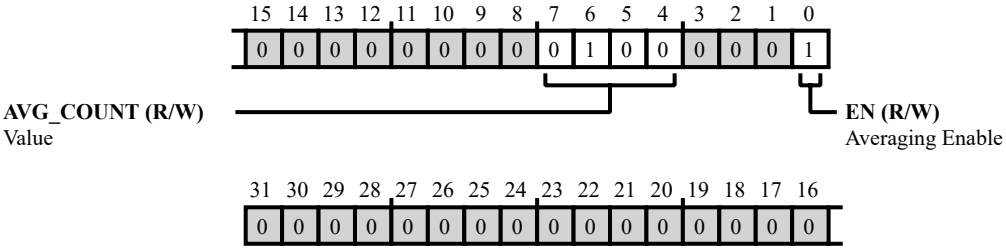


Figure 14-4: TMU_AVG Register Diagram

Table 14-9: TMU_AVG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/W)	AVG_COUNT	Value. The TMU_AVG.AVG_COUNT bit field indicates the number of values (N) to be averaged where N = 2^{AVG_COUNT}. The minimum value of TMU_AVG.AVG_COUNT is 1 and maximum value is 12. The fault value is 5, hence 2^5 = 32 values are averaged, by default. A maximum 2^{12}=4096 values can be averaged.
0 (R/W)	EN	Averaging Enable. The TMU_AVG.EN bit enables averaging on the TMU. Moving average is calculated for the past N values, where N is a power of 2. N value can be programmed by a register. N is defined by the TMU_AVG.AVG_COUNT register The average value is calculated using the equation: {(N-1)*previous_average+current_sampe}/N. The first N samples are summed and divided by N to calculate the first average value. Hence, the temperature value will not be updated for the first N conversions, when averaging is enabled. Averaging is enabled by default. It can be disabled when it is not needed.

Temperature Conversion Blank Register

The TMU_CNV_BLANK register specifies blanking period in which TMU utilizes HADC for temperature conversion.

The conversion time is calculated as: (VALUE+1)*50k SCLK cycles. So, the minimum conversion time is 50k SCLK cycles. Default conversion time is 200k SCLK cycles. For this duration, the system cannot use HADC for data conversion and HADC does take input for data-conversion. The status is indicated by TMUHADC_BUSY bit of the HADC block.

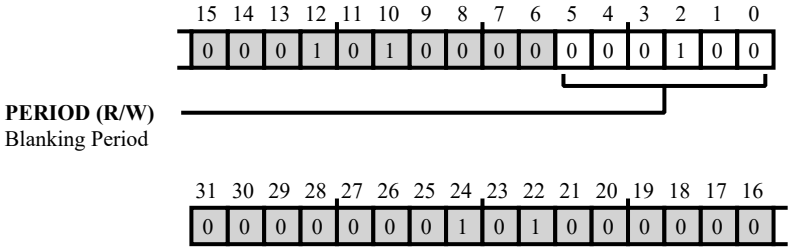


Figure 14-5: TMU_CNV_BLANK Register Diagram

Table 14-10: TMU_CNV_BLANK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	PERIOD	Blanking Period. Blanking period is (CNV_BLANK+1)*150k SCLK cycles. So the minimum blanking period is 150k SCLK cycles. Default blanking period is 1650k (1.65 million) SCLK cycles. For this duration, The system cannot use HADC for data conversion and HADC will not take input for data-conversion. The status will be indicated by TMHADC_BUSY bit of HADC block. More details can be found in TMU Programming model section.

TMU Control Register

The TMU_CTL register contains bits that allow programs to configure and enable the TMU.

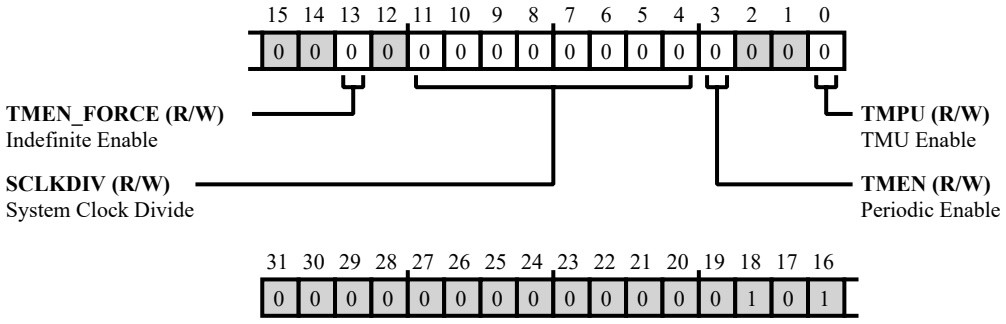


Figure 14-6: TMU_CTL Register Diagram

Table 14-11: TMU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	TMEN_FORCE	Indefinite Enable. Asserting the TMU_CTL.TMEN_FORCE bit enables the TMU indefinitely. HADC is always used by the TMU, so the HADC cannot be used to convert channel inputs.
11:4 (R/W)	SCLKDIV	System Clock Divide. The TMU_CTL.SCLKDIV bit selects the division ratio for the system clock (SCLK). SCLK is divided by (11+2*SCLKDIV). Thus, by default the SCLK is divided by 11.
3 (R/W)	TMEN	Periodic Enable. Asserting the TMU_CTL.TMEN bit enables the TMU periodically. TMU is enabled once in every refresh period defined by the TMU_REFR_CNTR register. And in each refresh period, TMU measures temperature for the blanking period (Tblank). It is defined by CNV_BLANK bit.
0 (R/W)	TMPU	TMU Enable. The TMU_CTL.TMPU bit enables the TMU. By default, the module is in power-down mode. The peripheral interface is active even in power-down mode (the TMU registers can be read/written).

Fault High Limit Register

The TMU_FLT_LIM_HI register sets the temperature fault high limit as an integer value. The value is stored in two's complement format. Asserts TMU_STAT.FLTHI if the TMU_TEMP value is greater than or equal to TMU_FLT_LIM_HI. The TMU_FLT_LIM_HI value should be programmed for value greater than 8'h3C (60 degC).

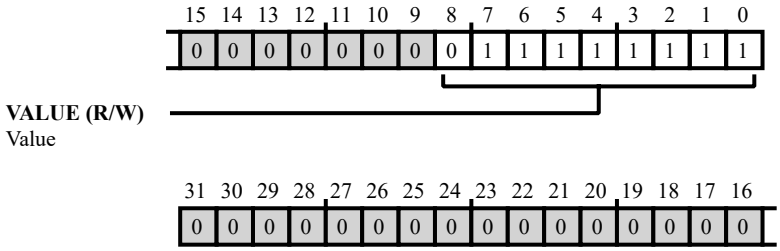


Figure 14-7: TMU_FLT_LIM_HI Register Diagram

Table 14-12: TMU_FLT_LIM_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_FLT_LIM_HI.VALUE bit field sets the temperature high limit as an integer in two's complement format. If the limit value is greater than or equal to the high limit the TMU_STAT.FLTHI bit is set.

Fault Low Limit Register

The TMU_FLT_LIM_LO register configures the fault temperature low limit.

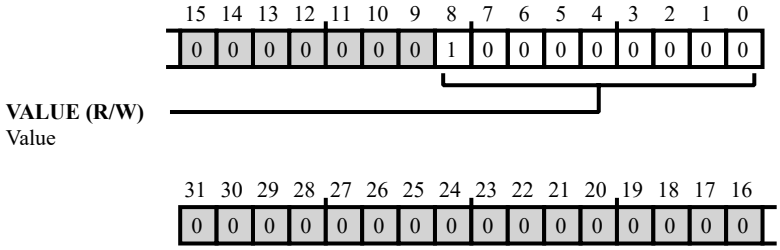


Figure 14-8: TMU_FLT_LIM_LO Register Diagram

Table 14-13: TMU_FLT_LIM_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_FLT_LIM_LO.VALUE bit field configures the fault temperature low limit as an integer which is stored in two's complement format. If the temperature value is less than or equal to the low limit, the TMU_STAT.FLTLO bit is set.

Gain Value Register

The TMU_GAIN register is used to configure the gain value in two's complement format. This value is used to correct the gain error in the TMU_TEMP register.

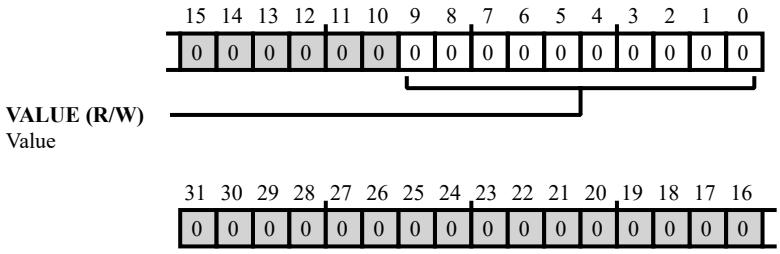


Figure 14-9: TMU_GAIN Register Diagram

Table 14-14: TMU_GAIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Value. The TMU_GAIN.VALUE bit field configures the gain value in two's complement format. This value is used to correct the gain error in the TMU_TEMP register.

Interrupt Mask Register

The TMU_IMSK register provides bits that are used to mask and unmask the interrupts associated with the TMU module.

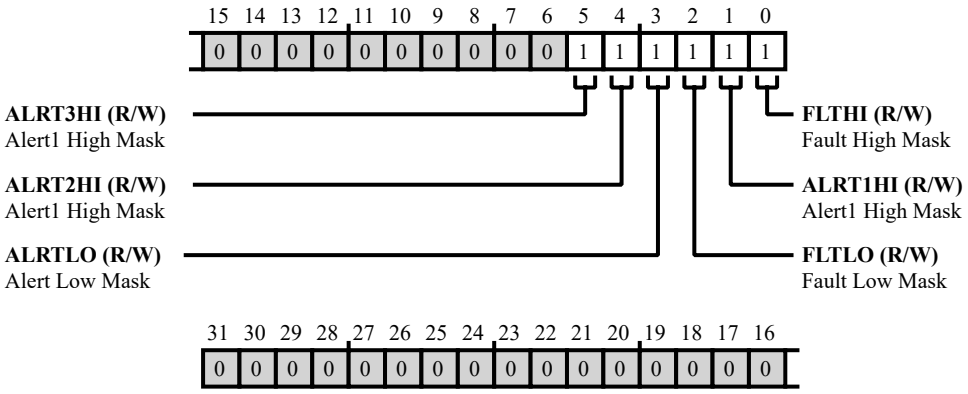


Figure 14-10: TMU_IMSK Register Diagram

Table 14-15: TMU_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	ALRT3HI	Alert1 High Mask.
4 (R/W)	ALRT2HI	Alert1 High Mask.
3 (R/W)	ALRTLO	Alert Low Mask. The TMU_IMSK.ALRTLO bit masks or unmasks the alert low status change (TMU_STAT.ALRTLO).
2 (R/W)	FLTLO	Fault Low Mask. The TMU_IMSK.FLTLO bit masks or unmasks the fault low status change (TMU_STAT.FLTLO).
1 (R/W)	ALRT1HI	Alert1 High Mask.
0 (R/W)	FLTHI	Fault High Mask. The TMU_IMSK.FLTHI bit masks or unmasks the fault high interrupt (TMU_STAT.FLTHI).

Offset Register

The value programmed in the TMU_OFFSET register is used to correct the offset error in TMU_TEMP register.

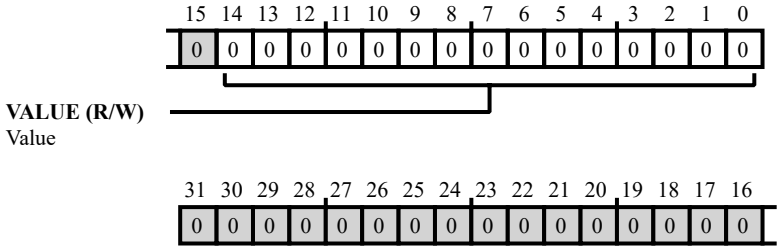


Figure 14-11: TMU_OFFSET Register Diagram

Table 14-16: TMU_OFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/W)	VALUE	Value. The TMU_OFFSET.VALUE bit field provides the offset value which is used to correct the offset error in the TMU_TEMP register. This value is in a two's complement fixed point Q7.7 format i.e 8 MSB's represent integer part in 2's complement format. and remaining 7 LSB's represent decimal part. Offset will be applied after multiplying gain.

Temperature Refresh Counter

The TMU_REFR_CNTR register defines the refresh-period to refresh temperature value.

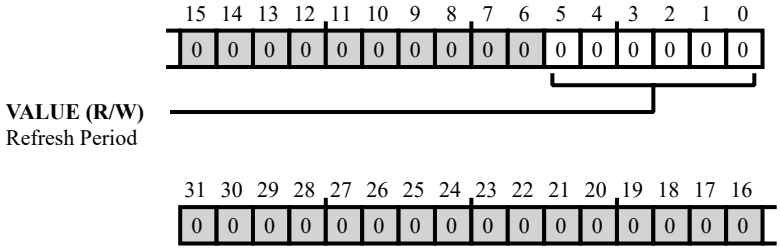


Figure 14-12: TMU_REFR_CNTR Register Diagram

Table 14-17: TMU_REFR_CNTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	VALUE	Refresh Period. The TMU_REFR_CNTR.VALUE bit field defines the period to refresh the temperature value. Temperature is refreshed for every (VALUE+1)*(21+4*SCLKDIV)*1M system-clock cycles. SCLKDIV is defined in TMU_CTL register. So, by default, the period is 21 million system-clock cycles.

Status Register

The TMU_STAT register bits indicate when an error or fault is detected.

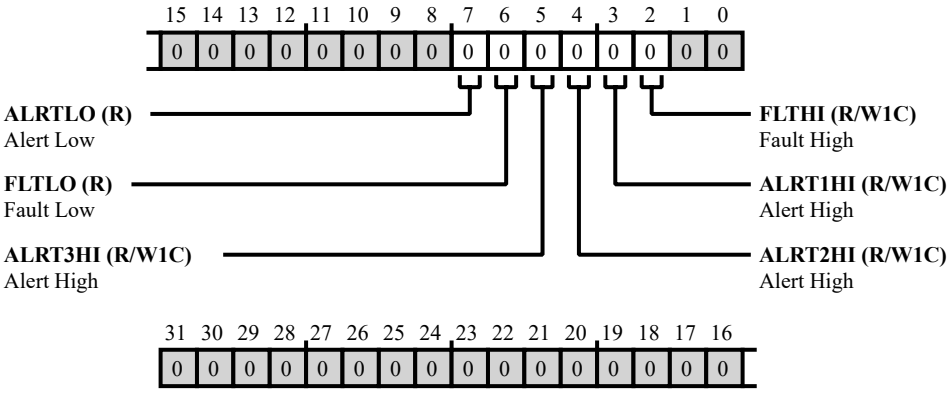


Figure 14-13: TMU_STAT Register Diagram

Table 14-18: TMU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	ALRTLO	Alert Low. The TMU_STAT.ALRTLO bit is set when the temperature value is less than or equal to the setting in the TMU_ALRT_LIM_LO register. The bit assertion is masked if TMU_IMSK.ALRTLO is HIGH.
6 (R/NW)	FLTLO	Fault Low. The TMU_STAT.FLTLO bit is set when the temperature value is less than or equal to the setting in the TMU_FLT_LIM_LO register. The bit assertion is masked if TMU_IMSK.FLTLO is HIGH.
5 (R/W1C)	ALRT3HI	Alert High. The TMU_STAT.ALRT3HI bit is set when the temperature value is greater than or equal to the setting in the TMU_ALRT_LIM_HI register. A Write1toClear is needed to reset the bit if it is asserted. The bit assertion is masked when TMU_IMSK.ALRT3HI =1.
4 (R/W1C)	ALRT2HI	Alert High. The TMU_STAT.ALRT2HI bit is set when the temperature value is greater than or equal to the setting in the TMU_ALRT_LIM_HI register. A Write1toClear is needed to reset the bit if it is asserted. The bit assertion is masked when TMU_IMSK.ALRT2HI =1.

Table 14-18: TMU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W1C)	ALRT1HI	Alert High. The TMU_STAT.ALRT1HI bit is set when the temperature value is greater than or equal to the setting in the TMU_ALRT_LIM_HI register. A Write1toClear is needed to reset the bit if it is asserted. The bit assertion is masked when TMU_IMSK.ALRT1HI =1.
2 (R/W1C)	FLTHI	Fault High. The TMU_STAT.FLTHI bit is set when the temperature value is greater than or equal to the setting in the TMU_FLT_LIM_HI register. The temperature value register is set to 16'h0D80 (effective temperature value of 27 degrees C) when a W1C is done on this bit. The bit assertion is masked if TMU_IMSK.FLTHI is HIGH.

Temperature Value Register

The TMU_TEMP register provides the temperature value from the A/D converter and the status.

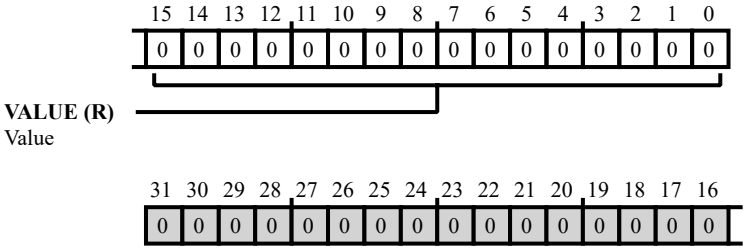


Figure 14-14: TMU_TEMP Register Diagram

Table 14-19: TMU_TEMP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	Value. The TMU_TEMP.VALUE bit field is the temperature value from A/D converter. This value is stored in two's complement fixed point Q8.7 format where the 9 MSB's represent the integer part in 2's complement, and the remaining 7 LSB's represent the decimal part.

15 Housekeeping ADC (HADC)

The Housekeeping ADC is a 12-bit (with 10-bit accuracy), successive approximation ADC. It operates from single supply and features throughput rates up to 1 MSPS. The HADC can be used for the collection of housekeeping parameters like voltages, temperatures in the system or for any general-purpose use as well.

NOTE: HADC is used by the TMU for temperature monitor, so it may be unavailable during temperature conversion. For details, refer to the TMU and HADC section.

HADC Features

The HADC supports following features:

- 12-bit ADC core with built-in sample and hold
- ENOB = 10 bit
- 8 input channels which can be extended to 15 channels by putting an external channel multiplexer
- Throughput rates up to 1 MSPS
- Single ended operation
- External reference nominal at 1.8 V
- Analog input 0 V to 1.8 V.
- Selectable ADC clock frequency through a pre-scaler
- Conversion type adaptable to each application–allows single or continuous conversion with option of auto-scan
- Auto sequencing capability provides up to 15 *auto-conversions* in a single session. Each conversion can be programmed to select any of the available input channels.

HADC Functional Description

The HADC provides the analog to digital conversion capability for general-purpose housekeeping tasks, such as voltage and temperature monitoring. The core of HADC is a 12-bit SAR ADC, providing multiple analog input channels.

The HADC has the following functionality:

Fixed and continuous conversion modes

ADC converts the input channel sequence for a fixed number of times or continuously converts an input channel sequence.

Auto scanning

All the input channels can be sampled in a sequential manner.

Channel sequence programming

The sequence of a channel can be selected by programming the channel mask register. If the bit corresponding to the channel is programmed to zero, that channel is included in the auto-scan chain.

ADSP-2184x HADC Register List

The Housekeeping ADC (HADC) provides a general purpose, multi-channel successive approximation A-to-D converter. A set of registers governs HADC operations. For more information on HADC functionality, see the HADC register descriptions.

Table 15-1: ADSP-2184x HADC Register List

Name	Description
HADC_CHAN_MSK	Channel Mask Register
HADC_CTL	Control Register
HADC_DATA[nn]	Channel Data Registers
HADC_IMSK	Interrupt Mask Register
HADC_STAT	Status Register

ADSP-2184x HADC Interrupt List

Table 15-2: ADSP-2184x HADC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
118	HADC0_EVT	HADC0 Event	Edge	
346	HADC0_FIFO_EVT	HADC0 FIFO Event	Edge	

ADSP-2184x HADC Trigger List

Table 15-3: ADSP-2184x HADC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
57	HADC0_EOC	HADC0 HADC0 End of Conversion	Edge

Table 15-4: ADSP-2184x HADC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

HADC Definitions

The following definitions are helpful when using the HADC module.

Auto-scan

Auto-scan is a feature which allows the multiple channels to be scanned and converted in sequence one after the other.

HADC Wakeup Time

It is the time required by the module after coming out of power down before it can start converting.

Signal-to-Noise and Distortion Ratio (SINAD)

The measured ratio of signal-to-noise and distortion at the output of the ADC. The signal is the rms amplitude of the fundamental. Noise is the sum of all non-fundamental signals up to half the sampling frequency ($f_S/2$), excluding dc. The ratio depends on the number of quantization levels in the digitization process; the more levels, the smaller the quantization noise. The theoretical signal-to-noise and distortion ratio for an ideal N-bit converter with a sine wave input is given by:

Signal-to-(Noise + Distortion) = (6.02 N + 1.76) dB

Total Harmonic Distortion (THD)

The ratio of the rms sum to the harmonics to the fundamental.

Peak Harmonic or Spurious Noise

The ratio of the rms value of the next largest component in the ADC output spectrum (up to $f_S/2$ and excluding dc) to the rms value of the fundamental. Typically, the value of this specification is determined by the largest harmonic in the spectrum, but for ADCs where the harmonics are buried in the noise floor, it is a noise peak.

Integral Nonlinearity

The maximum deviation from a straight line passing through the endpoints of the ADC transfer function. The endpoints are zero scale, a point 1 LSB below the first code transition, and full scale, a point 1 LSB above the last code transition.

Differential Nonlinearity

The difference between the measured and the ideal 1 LSB change between any two adjacent codes in the ADC.

Offset Error

The deviation of the first code transition (00...000) to (00...001) from the ideal—that is, GND1 + 1 LSB.

Offset Error Match

The difference in offset error between any two channels.

Gain Error

The deviation of the last code transition (111...110) to (111...111) from the ideal (that is, REFIN – 1 LSB) after the offset error has been adjusted out.

Gain Error Matching

The difference in gain error between any two channels.

Power Supply Rejection Ratio (PSRR)

PSRR is defined as the ratio of the power in the ADC output at full-scale frequency, f, to the power of a 100 mV p-p sine wave applied to the ADC VDD supply of frequency, f_S. The frequency of the input varies from 5 kHz to 25 MHz. PSRR (dB) = 10 log(Pf/Pf_S) where: Pf is the power at frequency, f, in the ADC output. Pf_S is the power at frequency, f_S, in the ADC output.

HADC Block Diagram

The ***HADC Block Diagram*** figure shows the functional blocks within the HADC and the interface to the processor core and the peripherals. The HADC has an internal channel multiplexer that is controlled by a programmable sequencer which selects the desired channel or sequence of channels.

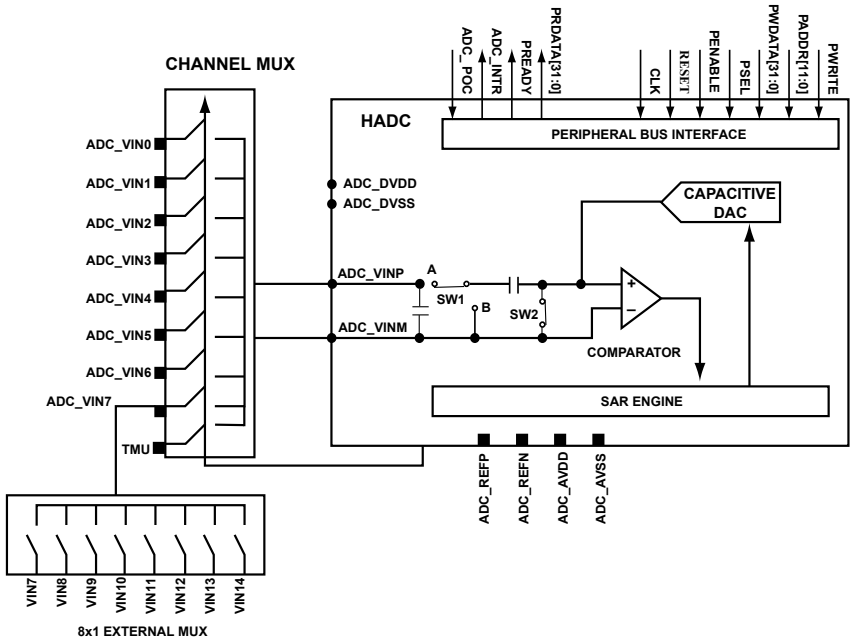


Figure 15-1: HADC Block Diagram

HADC Signal Descriptions

The ***HADC Signal Descriptions*** table provides descriptions of the signals used by the HADC.

Table 15-5: HADC Signal Descriptions

Signal Name	Signal Description
AVDD	ADC I/O supply
G	I/O ground for analog blocks - shorted with other ground pins within the package
VREFP	External reference for ADC
VREFN	Ground reference for ADC
VINn	Analog input at channel n

HADC Architectural Concepts

The HADC is based on a 12-bit SAR ADC that provides a simple register-based access model to obtain the results of conversion. The digital front end of the HADC provides a set of registers to configure the mode of operation, sampling frequency, and input channel selection control. The ADC supports multiple input analog channels which can be individually selected or deselected for conversion. The results of each analog channel are stored in a register. The core can access the register to read the conversion results once the conversion is complete. The HADC also provides the interrupts on completion of each channel conversion to avoid polling by the core. The following sections provide more details about the architecture of the HADC.

Converter Operation

The housekeeping ADC is a 12-bit successive approximation ADC based around a capacitive DAC. The ***ADC Acquisition Phase*** figure and the ***ADC Conversion Phase*** figure show simplified schematics of the ADC. The ADC is comprised of control logic, SAR, and a capacitive DAC. The components are used to add and subtract fixed amounts of charge from the sampling capacitor to bring the comparator back into a balanced condition. The ***ADC Conversion Phase*** figure shows the ADC during its acquisition phase. SW2 is closed and SW1 is in Position A. The comparator is held in a balanced condition and the sampling capacitor acquires the signal on the selected V_{IN} channel.

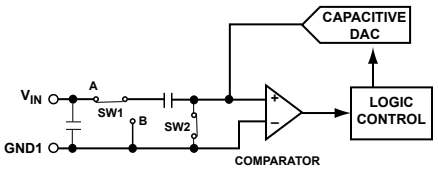


Figure 15-2: ADC Acquisition Phase

When the ADC starts a conversion (see the ***ADC Conversion Phase*** figure), SW2 opens and SW1 moves to Position B, causing the comparator to become unbalanced. The control logic and the capacitive DAC are used to add and subtract fixed amounts of charge to bring the comparator back into a balanced condition. When the comparator is rebalanced, the conversion is complete. The control logic generates the ADC output code.

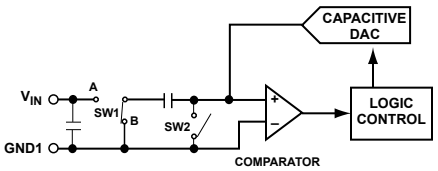


Figure 15-3: ADC Conversion Phase

Auto-Scan

The HADC features auto-scan mode where all the input channels can be sampled in a sequential manner. The number of channels enabled in auto-scan mode can be selected by programming the HADC_CHAN_MSK register. If the bit corresponding to the channel is set high, that particular channel is masked, and is not included in the auto-scan chain. In this way programs can sample all, none, or a selected set of channels by writing a high or a low for the individual channel. Auto sequencing allows the system to convert the same channel multiple times, allowing programs to perform oversampling algorithms.

For example, if the HADC_CHAN_MSK register bits [3:0] are set to 1101, then channel 0, channel 2 and channel 3 are not included in the auto-scan chain. Whether the conversion is a single or fixed number or continuous depends on the status of HADC_CTL.CONT bit. If this bit is low, the HADC_CTL.FIXEDCNV bits determine the number of sequence conversions.

The maximum number of fixed sequence conversions is 15. By default, the first eight channels of the HADC are enabled and extended channels are masked. An extended channel is the increased number of channels when a single channel of ADC is externally multiplexed, effectively increasing the total number of available channels. The total

number of channels supported is 15. The program must configure the HADC_CHAN_MSK register to enable any desired channel.

Channel Sequence Programming

The sequence of a channel can be selected by programming the HADC_CHAN_MSK register. If the bit corresponding to the channel is programmed to zero, that channel is included in the auto-scan chain. If the program must get the conversion results from a particular channel, then the bit corresponding to that channel should be zero. Channels 8-15 are virtual channels; in that they can connect an external multiplexer and increase the effective number of channels. As the output of the external multiplexer is routed to channel 7, this channel is not available (or converted) separately when the virtual channels are not masked in the HADC_CHAN_MSK register. The auto-scan section has more details.

ADC Transfer Function

The output coding is straight binary for the analog input channel conversion. The designed code transitions occur at successive LSB values (that is, 1 LSB, 2 LSBs, and so forth). The LSB size is $V_{REF}/4096$ for the HADC. The *ADC Transfer Function* figure shows the ideal transfer characteristic for straight binary coding.

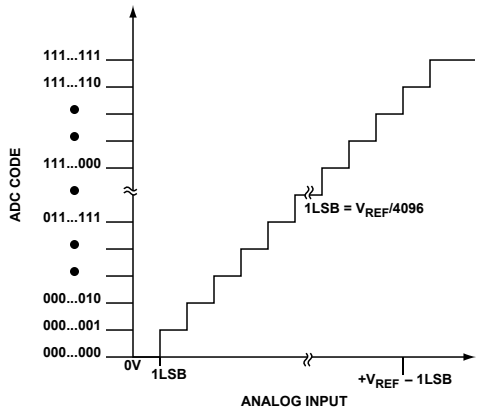


Figure 15-4: ADC Transfer Function

Results

The HADC takes 20 cycles of f_{SAMPLE} for one channel conversion. (The value of the HADC_CTL.FDIV bit field determines f_{SAMPLE}). The time taken to complete one sequence depends on the number of channels in the auto-scan chain. There is a latency of 1 cycle from the time the channel is selected internally to sample to the time the data is ready. After the end of each channel conversion, the data is written into the corresponding data register. An interrupt is generated (if the interrupt mask is not enabled) to signal that the data is ready for that particular channel.

HADC Operating Modes

The HADC has two modes of operation described in the following sections.

Fixed Conversion Mode

In this mode, the ADC converts the input channel sequence for a fixed number of times. The frequency is configured in the HADC_CTL.FIXEDCNV bit field. To use this mode, clear the HADC_CTL.CONT bit.

Continuous Conversion Mode

In this mode ADC continuously converts an input channel sequence, when the HADC_CTL.STARTCNV bit is held high. To use this mode, set the HADC_CTL.CONT bit.

HADC Event Control

The HADC generates different events depending on the state of the ADC and the status of channel conversions. It can generate an event for each of the following conditions:

- When ADC is ready for conversion
- At the end of sequence conversion
- At the end of each individual channel conversion

Each of these events can generate an interrupt. To generate an interrupt on any desired event, clear the respective bit in the HADC_IMSK register.

HADC Programming Model

Following sections provide some guidelines for HADC programming.

Powering Up the HADC

To power-up the HADC, program the following bits in the HADC_CTL register.

- Deassert the HADC_CTL.PD bit (HADC power down)
- Set the HADC_CTL.NRST bit (Reset)
- Set the HADC_CTL.ENLS bit (Enable level shifters)

After deasserting HADC_CTL.PD, the HADC requires a finite wake-up time (t_{WAKEUP}) before it can start converting. The HADC requires only two f_{SAMPLE} clocks from the assertion of the HADC_CTL.NRST bit before the module is ready to convert. (HADC_CTL.PD is low). Poll the HADC_STAT.RDY bit. A 1 on this bit indicates that the HADC is ready to convert data.

Enabling the HADC

Setting the HADC_CTL.STARTCNV bit enables the HADC. When this bit is kept high, the HADC can work in either continuous or fixed conversion mode. After the HADC_CTL.STARTCNV bit is set =1, the HADC_CHAN_MSK can still be re-programmed, but the new sequence only comes into effect after the current sequence conversion is complete.

ADSP-2184x HADC Register Descriptions

Housekeeping ADC (HADC) contains the following registers.

Table 15-6: ADSP-2184x HADC Register List

Name	Description
HADC_CHAN_MSK	Channel Mask Register
HADC_CTL	Control Register
HADC_DATA[nn]	Channel Data Registers
HADC_IMSK	Interrupt Mask Register
HADC_STAT	Status Register

Channel Mask Register

The HADC_CHAN_MSK register provides bits that mask each channel. The LSB corresponds to channel 0 , the second LSB to channel 1 and so on. If a mask bit is set, the corresponding channel is not converted. By default, channels 0-7 are not masked and the extended channels 8-15 are masked.

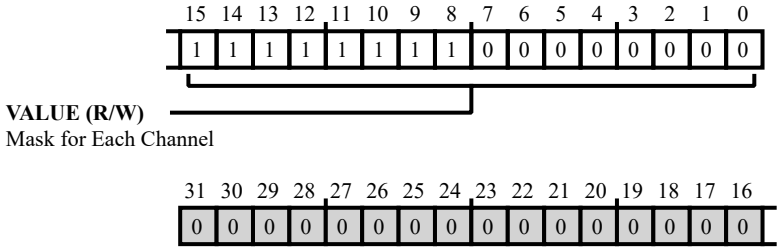


Figure 15-5: HADC_CHAN_MSK Register Diagram

Table 15-7: HADC_CHAN_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Mask for Each Channel. The HADC_CHAN_MSK.VALUE bit field is the mask bit for each channel. The first MSB corresponds to channel 15, the second MSB to channel 14 and so on. If the mask is set for a particular channel, that channel is not converted. By default, channels 0-7 are not masked and the extended channels 8-15 are masked. The first MSB corresponds to channel 7, the second MSB to channel 6 and so on. If the mask is set for a particular channel, that channel is not converted. By default, channels 0-7 are not masked.

Control Register

The HADC_CTL register contains control bits that configure various module settings start or reset the module.

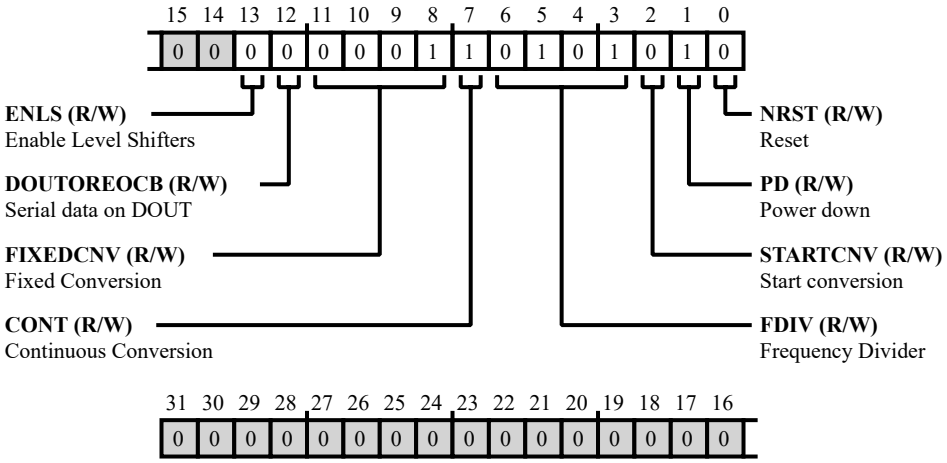


Figure 15-6: HADC_CTL Register Diagram

Table 15-8: HADC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	ENLS	Enable Level Shifters. Setting the HADC_CTL.ENLS bit enables the level shifters, allowing the HADC analog side (which works in the VDD_EXT domain) to work with the digital core and interface is (in the VDD_INT domain).	
		0	Disable down level shifters
		1	Enable down level shifters
12 (R/W)	DOUTOREOCB	Serial data on DOUT. If the HADC_CTL.DOUTOREOCB bit =1, serial data arrives on the EOC_DOUT pin. If this bit =0 (default) it acts as an EOC only if the external multiplexer is connected.	
		0	Reserved
		1	Reserved
11:8 (R/W)	FIXEDCNV	Fixed Conversion. The HADC_CTL.FIXEDCNV bit configures the number of conversions = FIX-EDCNV. This value determines how many times a sequence is converted when the HADC is in fixed conversion mode. This only applies when the HADC_CTL.CONT bit =0. Before changing the HADC_CTL.FIXEDCNV bit, clear the HADC_CTL.NRST bit (=0).	

Table 15-8: HADC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	CONT	Continuous Conversion. When the HADC_CTL.CONT bit =0, the ADC converts a sequence for a fixed number of times. This number is configured using the HADC_CTL.FIXEDCNV bit field. When the HADC_CTL.CONT bit =1, the ADC continuously converts a given sequence, provided the HADC_CTL.STARTCNV is held high.	
		0	ADC converts sequence for fixed number of times
		1	ADC continuously converts given sequence
6:3 (R/W)	FDIV	Frequency Divider. The HADC_CTL.FDIV bit field configures the f_SAMPLE=f_CLK/(FDIV+1). Select f_CLK and HADC_CTL.FDIV values so that f_SAMPLE is in range of 50 kHz to 22.5 MHz. The minimum value for HADC_CTL.FDIV is 1. Before changing the HADC_CTL.FDIV bits, clear the HADC_CTL.NRST bit.	
2 (R/W)	STARTCNV	Start conversion. The HADC_CTL.STARTCNV bit needs to be set for the ADC to start converting data. If the ADC is running in non continuous mode, it is reset by hardware after the desired number of conversions is completed.	
		0	No action
		1	Start converting
1 (R/W)	PD	Power down. The HADC_CTL.PD bit powers down the analog circuitry of the ADC. After this bit returns to 0 a finite power-up time is required before the ADC can start converting data.	
		0	No action
		1	Power down the analog circuitry of the ADC
0 (R/W)	NRST	Reset. The HADC_CTL.NRST bit resets the ADC.	
		0	Reset the ADC
		1	No action

Channel Data Registers

The HADC_DATA[nn] registers NN ranges from 0-14. Each corresponding to an ADC channel. ADC_DATA_0 corresponds to channel 0, ADC_DATA_1 to channel 1 and so on.

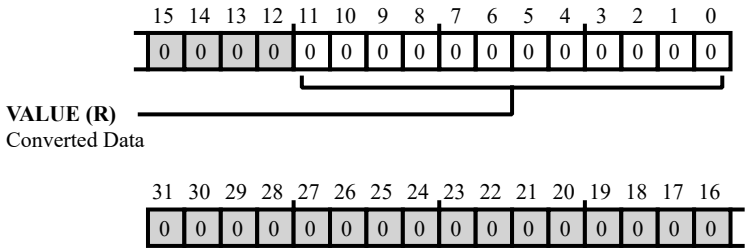


Figure 15-7: HADC_DATA[nn] Register Diagram

Table 15-9: HADC_DATA[nn] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/NW)	VALUE	Converted Data. The HADC_DATA[nn].VALUE bit field contains the digital code for the sampled analog value. Each channel has its own data register.

Interrupt Mask Register

The HADC_IMSK register masks (disables) or unmasks (enables) the interrupts as programmed.

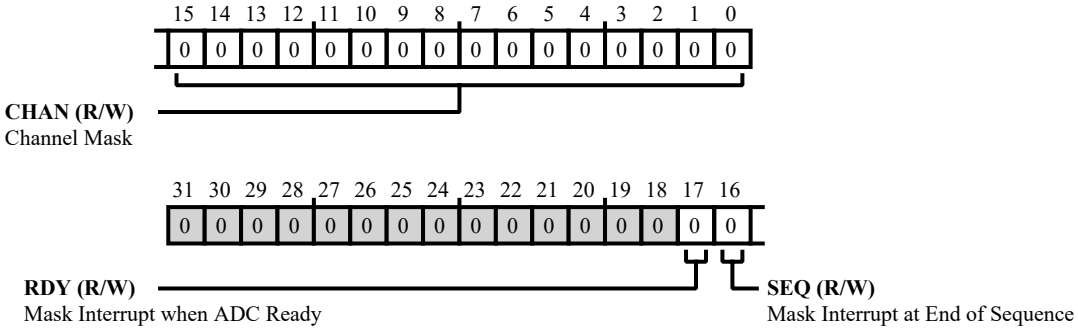


Figure 15-8: HADC_IMSK Register Diagram

Table 15-10: HADC_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RDY	Mask Interrupt when ADC Ready. The HADC_IMSK.RDY bit masks the interrupt generated when ADC is ready to convert.	
16 (R/W)	SEQ	Mask Interrupt at End of Sequence. The HADC_IMSK.SEQ bit masks the interrupt which is generated at the end of sequence completion.	
		0	Interrupt is unmasked
		1	Interrupt is masked
15:0 (R/W)	CHAN	Channel Mask. The HADC_IMSK.CHAN bit field provides the interrupt mask bit for each channel. N ranges from 0-15. The MSB corresponds to channel 15, the second MSB to channel 14 and so on. If the bit is SET, interrupt is masked for corresponding channel.	

Status Register

The HADC_STAT register contains bits that provide status information on the HADC module.

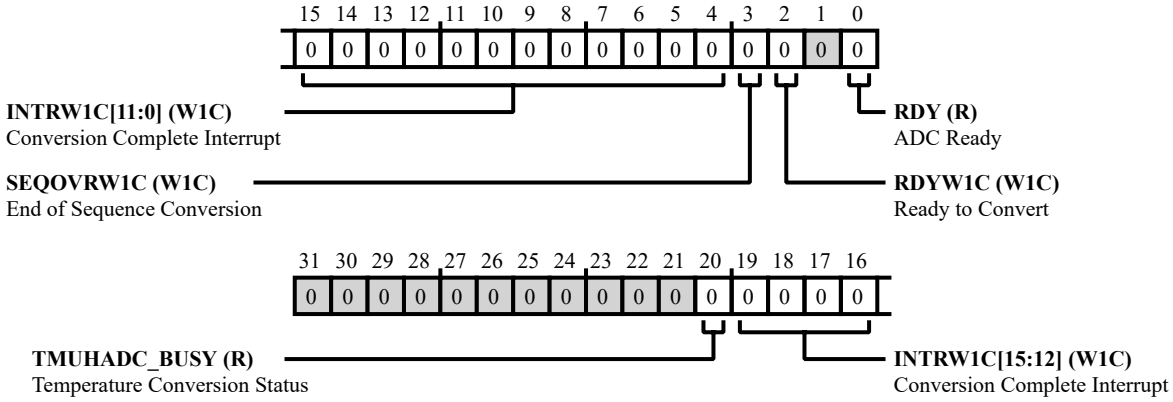


Figure 15-9: HADC_STAT Register Diagram

Table 15-11: HADC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20 (R/NW)	TMUHADC_BUSY	Temperature Conversion Status. The HADC_STAT.TMUHADC_BUSY bit, if high, indicates to the system that temperature conversion is ongoing. All ADC conversion requests are ignored.
19:4 (RX/W1C)	INTRW1C	Conversion Complete Interrupt. The HADC_STAT.INTRW1C bit field indicates when the corresponding ADC channel completes conversion. N ranges from 0-15 and the MSB corresponds to channel 15, the second MSB to channel 14 and so on. N ranges from 0-7 and the MSB corresponds to channel 7, the second MSB to channel 6 and so on. These bits are sticky and are W1C.
3 (RX/W1C)	SEQOVRW1C	End of Sequence Conversion. The HADC_STAT.SEQOVRW1C bit indicates the end of a sequence conversion and is a sticky status bit which is W1C.
2 (RX/W1C)	RDYW1C	Ready to Convert. The HADC_STAT.RDYW1C bit is the stick version of the HADC_STAT.RDY bit.
0 (R/NW)	RDY	ADC Ready. The HADC_STAT.RDY bit is set (=1) when the ADC is ready to convert data.

16 Controller Area Network Flexible Data Rate (CANFD)

The controller area network (CAN) protocol is primarily for use as a vehicle serial data bus. It meets the specific requirements of this field, including real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

The CANFD module is a full implementation of the CAN protocol specification, the CAN with flexible data rate (CAN FD) protocol, and the CAN 2.0 Part B protocol. It supports both standard and extended message frames and long payloads up to 64 bytes, transferred at rates up to 8 Mbps. The message buffers are stored in an embedded RAM dedicated to the CANFD module.

NOTE: This document assumes familiarity with the CAN standard. For more information, refer to Version 2.0 of the CAN specification from Robert Bosch GmbH.

CANFD Features

Key features of the CANFD module include:

- Full implementation of the CAN FD protocol and the CAN specification 2.0 (Part B) including:
  - Standard data frames
  - Extended data frames
  - Zero to sixty-four bytes data length
  - Programmable bit rate
  - Content-related addressing
- ISO 11898-1 standard compliance
- Flexible mailboxes
  - Configurable data lengths from 0 to 64 bytes
  - Configurable as receive or transmit, supporting standard and extended messages
  - Individual Rx mask registers per mailbox

- Full-featured Rx FIFO
    - Storage capacity for up to six frames
    - Automatic internal pointer handling with DMA support
- Transmission abort capability
- 28 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Independence from the transmission medium (an external transceiver is required)
- Low-power modes, with programmable wakeup on bus activity or matching with received frames (Pretended Networking)
- Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be handled automatically or by software
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- Status of synchronization with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Rx FIFO ID filtering, matching incoming IDs against 128 extended, 256 standard, or 512 partial (8 bit) IDs
    - Up to 32 ID filter table elements
- Detection and correction of memory read access errors, with five parity bits for each byte of CANFD memory
- Pretended networking functionality in low-power doze mode

CANFD Functional Description

The CANFD module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system consists of a set of message buffers (MBs) that store configuration and control data, time stamp, message ID, and data. The memory corresponding to the first 38 MBs is configurable to support a FIFO reception scheme with a powerful ID filtering mechanism. The ID filtering mechanism is capable of checking

incoming frames against a table of IDs (up to 128 extended IDs, 256 standard IDs, or 512 8-bit ID slices), with an individual mask register for up to 32 ID filter table elements.

Classical CAN frames support simultaneous reception through a FIFO and mailboxes. CAN FD frames support reception only through mailboxes. For mailbox reception, there is a matching algorithm to store received frames into the MBs that have the same ID programmed in the ID field. There is a masking scheme to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

The CANFD module can also receive and transmit messages in CAN FD format. The MBs are sized to store the quantity of data bytes selected in the MBDSRn fields of the CANFD_FD_CTL register.

The following sections provides listings of the CANFD registers, interrupts, and triggers. It section also provides information on the architectural concepts and functional operation of the CANFD module.

ADSP-2184x CANFD Register List

CANFD module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. Features : Flexible DataRate, PretendedNetwork, DMA support, ECC for RAM.

Table 16-1: ADSP-2184x CANFD Register List

Name	Description
CANFD_TIMING	Can Bit Timing Register
CANFD_CRC	CRC Register
CANFD_CTL1	Control 1 Register
CANFD_PN_CTL1	Pretended Networking Control1 Register
CANFD_CTL2	Control 2 Register
CANFD_PN_CTL2	Pretended Networking Control2 Register
CANFD_ECR	Error Counter Register
CANFD_ERR_IADDR	Error Injection Address Register
CANFD_ERR_IDP	Error Injection Data Pattern Register
CANFD_ERR_IPP	Error Injection Parity Pattern Register
CANFD_ERR_STAT	Error Status Register
CANFD_ESR1	Error and Status 1 Register
CANFD_ESR2	Error and Status 2 Register
CANFD_FD_TIMING	CANFD Bit Timing Register
CANFD_FD_CRC	CANFD CRC Register
CANFD_FD_CTL	CANFD Control Register

Table 16-1: ADSP-2184x CANFD Register List (Continued)

Name	Description
CANFD_FLTR_DLC	Pretended Networking DLC Filter Register
CANFD_FLTR_ID1	Pretended Networking ID Filter1 Register
CANFD_FLTR_ID2_IDMSK	Pretended Networking ID Filter2 / IDMask Register
CANFD_IFLG1	Mailbox Interrupt Flag 1 Register
CANFD_IFLG2	Mailbox Interrupt Flag 2 Register
CANFD_IMSK1	Mailbox Interrupt Mask 1 Register
CANFD_IMSK2	Mailbox Interrupt Mask 2 Register
CANFD_CFG	Module Configuration Register
CANFD_MEC	Memory Error Control Register
CANFD_FLTR_DATA1_HI	Pretended Networking Payload Low Filter2 Register
CANFD_FLTR_DATA1_LO	Pretended Networking Payload Low Filter1 Register
CANFD_FLTR_DATA2_DMSK_HI	Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register
CANFD_FLTR_DATA2_DMSK_LO	Pretended Networking Payload Low Filter2 / Payload Low Mask Register
CANFD_ERR_RADDR	Error Report Address Register
CANFD_ERR_RDAT	Error Report Data Register
CANFD_ERR_RSYN	Error Report Syndrome Register
CANFD_RX_14_MSK	Receive Mailbox14 Mask Register
CANFD_RX_15_MSK	Receive Mailbox15 Mask Register
CANFD_RX_FIFO_GMSK	Receive FIFO Global Mask Register
CANFD_RX_FIFO	Receive FIFO Information Register
CANFD_RX_IMSK[n]	Receive Individual Mask Register
CANFD_RX_MB_GMSK	Receive Mailbox Global Mask Register
CANFD_TMR	Free Running Timer Register
CANFD_WMB0_CS	Pretended Networking Wake Up Message Buffer 0 Control Status Register
CANFD_WMB0_D03	Pretended Networking Wake Up Message Buffer 0 Data 0-3 Register
CANFD_WMB0_D47	Wakeup Message Buffer Data 4-7 Register
CANFD_WMB0_ID	Pretended Networking Wake Up Message Buffer ID 0 Register
CANFD_WMB1_CS	Pretended Networking Wake Up Message Buffer 1 Control Status Register
CANFD_WMB1_D03	Pretended Networking Wake Up Message Buffer 1 Data 0-3 Register
CANFD_WMB1_D47	Pretended Networking Wake Up Message Buffer 1 Data 4-7 Register
CANFD_WMB1_ID	Pretended Networking Wake Up Message Buffer ID 1 Register

Table 16-1: ADSP-2184x CANFD Register List (Continued)

Name	Description
CANFD_WMB2_CS	Pretended Networking Wake Up Message Buffer 2 Control Status Register
CANFD_WMB2_D03	Pretended Networking Wake Up Message Buffer 2 Data 0-3 Register
CANFD_WMB2_D47	Pretended Networking Wake Up Message Buffer 2 Data 4-7 Register
CANFD_WMB2_ID	Pretended Networking Wake Up Message Buffer ID 2 Register
CANFD_WMB3_CS	Pretended Networking Wake Up Message Buffer 3 Control Status Register
CANFD_WMB3_D03	Pretended Networking Wake Up Message Buffer 3 Data 0-3 Register
CANFD_WMB3_D47	Pretended Networking Wake Up Message Buffer 3 Data 4-7 Register
CANFD_WMB3_ID	Pretended Networking Wake Up Message Buffer ID 3 Register
CANFD_WUM	Pretended Networking Wakeup Match Register

ADSP-2184x CANFD Interrupt List

Table 16-2: ADSP-2184x CANFD Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
1	CANFD0_WU	CANFD0 Wakeup Interrupt	None	
2	CANFD0_IRQ	CANFD0 CAN Interrupt	None	
3	CANFD0_MSG	CANFD0 Message Receive/Transmit Interrupt	None	
4	CANFD1_WU	CANFD1 Wakeup Interrupt	None	
5	CANFD1_IRQ	CANFD1 CAN Interrupt	None	
6	CANFD1_MSG	CANFD1 Message Receive/Transmit Interrupt	None	

ADSP-2184x CANFD Trigger List

Table 16-3: ADSP-2184x CANFD Trigger List Generators

Trigger ID	Name	Description	Sensitivity
1	CANFD0_IPD_REQ	CANFD0 DMA Request	None
2	CANFD1_IPD_REQ	CANFD1 DMA Request	None

Table 16-4: ADSP-2184x CANFD Trigger List Receivers

Trigger ID	Name	Description	Sensitivity

Table 16-4: ADSP-2184x CANFD Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
None			

CANFD Architectural Concepts

The following sections provide information about the CANFD architecture.

Block Diagram

The *CANFD Block Diagram* figure shows a block diagram of the CANFD module.

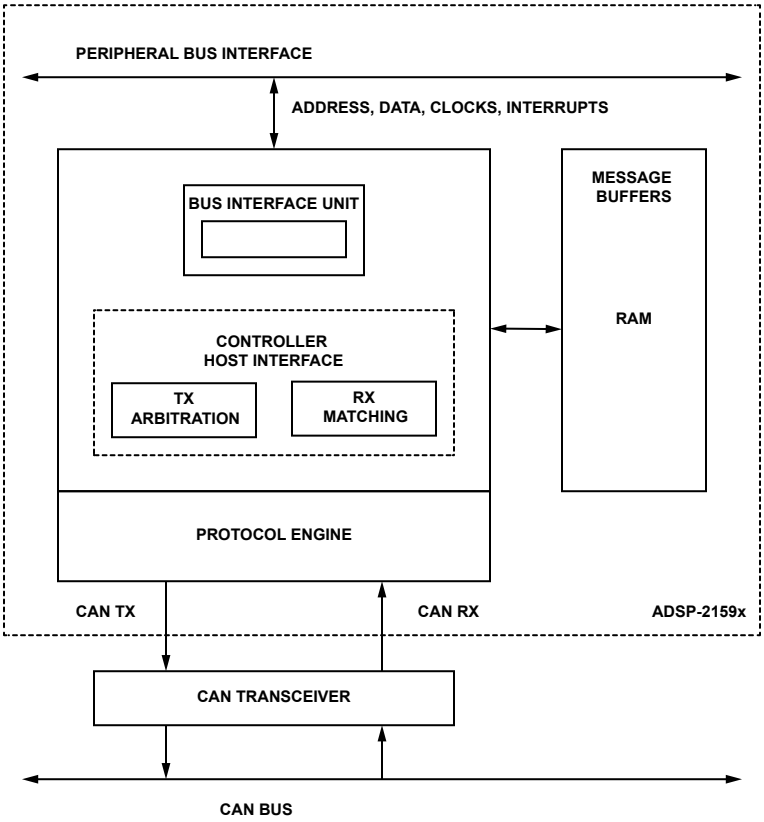


Figure 16-1: CANFD Controller Block Diagram

The CANFD module has the following three submodules and message buffers in an embedded RAM:

Protocol Engine (PE)

The PE submodule manages the serial communication on the CAN bus. It requests RAM access for receiving and transmitting message frames, validates received messages, preforms error handling, and detects CAN FD messages.

Controller Host Interface (CHI)

The CHI submodule handles message buffer selection for reception and transmission, including arbitration, and ID matching algorithms for both flexible data rate (FD) and non-FD message formats.

Bus Interface Unit (BIU)

The BIU submodule controls the access to and from the internal interface bus to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, DMA, and test signals are accessed through the BIU.

Message Buffer (MB)

The CANFD module uses a message buffer structure that represents both extended (29-bit identifier) and standard (11-bit identifier) frames from the CAN specification (Version 2.0, Part B).

Each individual message buffer is 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area for mailboxes is from address offset 0x80 to 0x47F. When CAN FD is enabled, the exact address for each memory byte depends on the size of the payload.

The *Message Buffer Structure 64-Byte Payload* figure shows an example of the message buffer structure with a 64-byte payload.

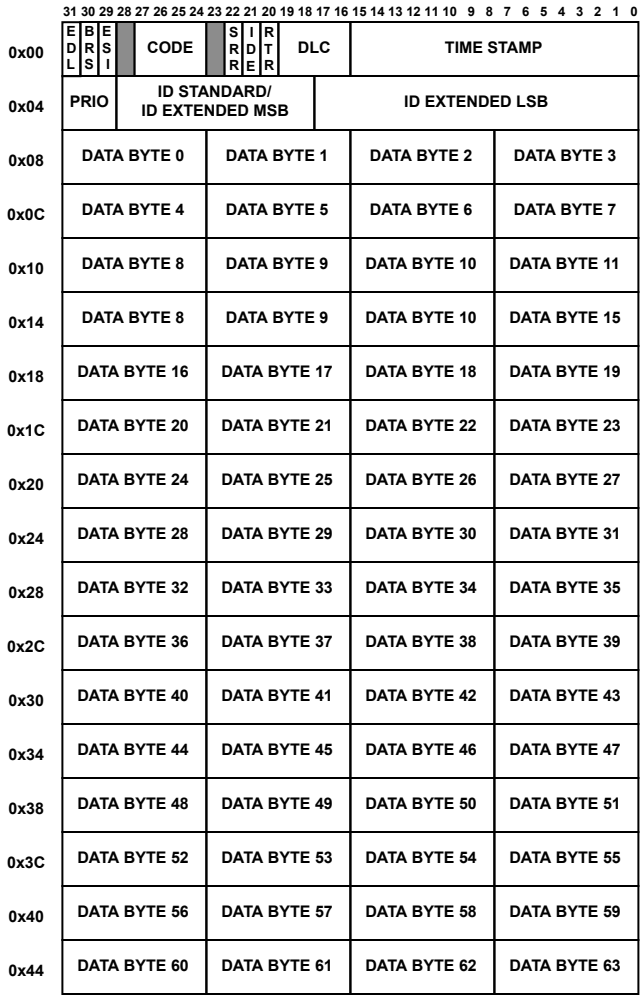


Figure 16-2: Message Buffer Structure 64-Byte Payload

In the example, the first eight bytes of the message buffer is control information associated with the data, followed by the bytes of data in the payload. The control information is defined in the following sections:

Extended Data Length (EDL)

The EDL bit distinguishes between CAN and CAN FD format frames. Do not set EDL when message buffers are configured to RANSWER with code field 0b1010.

Bit Rate Switch (BRS)

The BRS bit defines whether the bit rate is switched inside a CAN FD format frame.

Error State Indicator (ESI)

The ESI bit indicates if the transmitting node is error active or error passive.

Message Buffer Code (CODE)

The CODE field is accessible (read or write) to the processor and the CANFD module, as part of the message buffer matching and arbitration process. The *Message Buffer Code – Rx Buffer* and *Message Buffer Code – Tx Buffer* tables show the encoding.

Table 16-5: Message Buffer Code – Rx Buffer

CODE Description	Rx Code Before New Frame	SRV[*1]	Rx Code After Successful Reception [*2]	RRS [*3]	Comment
0000: INACTIVE – MB is not active.	INACTIVE	N/A	N/A	N/A	MB does not participate in the matching process.
0100: EMPTY – MB is active and empty	EMPTY	N/A	FULL	N/A	When a frame is received successfully, the CODE field is automatically updated to FULL.
0010: FULL – MB is full.	FULL	Yes	FULL	N/A	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching Process.
		No	OVERRUN	N/A	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching Process.
0110: OVERRUN – MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	N/A	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL.
		No	OVERRUN	N/A	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching Process.
1010: RANSWER[*4] – A frame was configured to recognize a remote request frame and transmit a response frame in return.	RANSWER	N/A	TANSWER(1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that, an MB is set to transmit a response frame. The code is automatically changed to TANSWER (1110). See Matching Process. If the `CANFD_CTL2.RRS` bit is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		N/A	N/A	1	This code is ignored during matching and arbitration process.

Table 16-5: Message Buffer Code – Rx Buffer (Continued)

CODE Description	Rx Code Before New Frame	SRV*1	Rx Code After Successful Reception *2	RRS *3	Comment
CODE[0]=1; BUSY – CANFD is updating the contents of the MB. The CPU must not access the MB.	BUSY*5	N/A	FULL	N/A	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		N/A	OVERRUN	N/A	

*1 Serviced MB. The MB has been read and unlocked by reading the CANFD_TMR register or other MB.

*2 A frame has a successful reception after it is moved to an MB. See Move Process.

*3 Remote Request Stored bit, see the CANFD_CTL2 register.

*4 Code 1010 is not considered Tx and an MB with this code should not be aborted.

*5 6. For Tx MBs, the BUSY bit is ignored upon read, except when the CANFD_CFG.ABORTEN bit is set. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 16-6: Buffer Message Code – Tx Buffer

CODE Description	Tx Code Before Tx Frame	MB RTR	Tx Code After Successful Transmission	Comment
1000: INACTIVE – MB is not active.	INACTIVE	N/A	N/A	MB does not participate in arbitration process.
1001: ABORT – MB is aborted.	ABORT	N/A	N/A	MB does not participate in arbitration process.
1100: DATA – MB is a Tx Data Frame (MB RTR must be 0).	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1100: REMOTE – MB is a Tx Remote Request Frame (MB RTR must be 1).	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.

Table 16-6: Buffer Message Code – Tx Buffer (Continued)

CODE Description	Tx Code Before Tx Frame	MB RTR	Tx Code After Successful Transmission	Comment
1110: TANSWER – MB is a Tx Response Frame from an incoming Remote Request Frame.	TANSWER	N/A	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching Process and Arbitration Process.

Substitute Remote Request (SRR)

SRR is a fixed recessive bit, only for the extended format. The SRR bit is set to one for transmission (Tx buffers) and is stored with the value received on the CAN bus for Rx receiving buffers. It is received as either recessive or dominant. If the CANFD module receives this bit as dominant, then it is interpreted as an arbitration loss.

When the bit is set, the recessive value is compulsory for transmission in extended format frames. When it is clear, dominant is not a valid value for transmission in extended format frames.

ID Extended Bit (IDE)

IDE identifies whether the frame format is standard or extended.

When the bit is set, the frame format is extended. When the bit is clear, the frame format is standard.

Remote Transmission Request (RTR)

The RTR bit affects the behavior of remote frames and is part of the reception filter:

When the bit is set, the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx, then incoming remote request frames may be stored. When the RTR bit is clear, the current MB has a data frame to be transmitted. In an Rx MB, it may be considered in matching processes.

See the *Message Buffer Code – Rx Buffer* table, the *Message Buffer Code – Tx Buffer* table, and the description of the `CANFD_CTL2.RRS` bit for additional details.

If the CANFD module transmits RTR as 1 (recessive) and receives it as 0 (dominant), it is interpreted as an arbitration loss. If RTR is transmitted as 0 (dominant) and received as 1 (recessive), the CANFD treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

NOTE: When configuring CAN FD frames, the RTR bit must be negated.

Length of Data in Bytes (DLC)

The 4-bit DLC field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space. In reception, DLC is written by the CANFD module, copied from the DLC (Data Length Code) field of the received frame. In transmission, DLC is written by the processor and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field. See the *Data Byte Validity* table.

Free-Running Counter Time Stamp (TIME STAMP)

The 16-bit TIME STAMP field is a copy of the free-running timer, captured for Tx and Rx frames when the beginning of the identifier field appears on the CAN bus.

Local Priority (PRIO)

The 3-bit PRIO field is used only when the CANFD_CFG.LPRIO_EN bit is set and only makes sense for Tx mailboxes. The PRIO field is not transmitted. It is appended to the regular ID to define the transmission priority. See the Arbitration Process section.

Frame Identifier (ID)

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification for both receive and transmit, and the 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification for both receive and transmit.

Data Field (DATA BYTE 0 to 63)

Up to 64 bytes can be used for a data frame, depending on the size of payload selected for the Message Buffers.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE(n) is valid only if n is less than DLC as shown in the *Data Byte Validity* table.

Table 16-7: Data Byte Validity

DLC	Valid Data Bytes
0	None
1	Data Byte 0
2	Data Byte 0 to 1
3	Data Byte 0 to 2

Table 16-7: Data Byte Validity (Continued)

DLC	Valid Data Bytes
4	Data Byte 0 to 3
5	Data Byte 0 to 4
6	Data Byte 0 to 5
7	Data Byte 0 to 6
8	Data Byte 0 to 7
9	Data Byte 0 to 11
10	Data Byte 0 to 15
11	Data Byte 0 to 19
12	Data Byte 0 to 23
13	Data Byte 0 to 31
14	Data Byte 0 to 47
15	Data Byte 0 to 63

Message Buffer Memory Map

The CANFD memory buffers are allocated in memory according to the *8-Byte Message Buffers*, *16-Byte Message Buffers*, *32-Byte Message Buffers*, and *64-Byte Message Buffers* tables.

Table 16-8: 8-Byte Message Buffers

Address Offset	MBDSR = 00 (8-Byte Payload)
0x0080	MB0
0x0090	MB1
0x00A0	MB2
0x00B0	MB3
0x00C0	MB4
0x00D0	MB5
0x00E0	MB6
0x00F0	MB7
0x0100	MB8
0x0110	MB9
0x0120	MB10
0x0130	MB11
0x0140	MB12

Table 16-8: 8-Byte Message Buffers (Continued)

Address Offset	MBDSR = 00 (8-Byte Payload)
0x0150	MB13
0x0160	MB14
0x0170	MB15
0x0180	MB16
0x0190	MB17
0x01A0	MB18
0x01B0	MB19
0x10C0	MB20
0x10D0	MB21
0x10E0	MB22
0x10F0	MB23
0x0200	MB24
0x0210	MB25
0x0220	MB26
0x0230	MB27
0x0240	MB28
0x0250	MB29
0x0260	MB30
0x0270	MB31
0x0280	MB32
0x0290	MB33
0x02A0	MB34
0x02B0	MB35
0x02C0	MB36
0x02D0	MB37
0x02E0	MB38
0x02F0	MB39
0x0300	MB40
0x0310	MB41
0x0320	MB42
0x0330	MB43

Table 16-8: 8-Byte Message Buffers (Continued)

Address Offset	MBDSR = 00 (8-Byte Payload)
0x0340	MB44
0x0350	MB45
0x0360	MB46
0x0370	MB47
0x0380	MB48
0x0390	MB49
0x03A0	MB50
0x03B0	MB51
0x03C0	MB52
0x03D0	MB53
0x03E0	MB54
0x03F0	MB55
0x0400	MB56
0x0410	MB57
0x0420	MB58
0x0430	MB59
0x0440	MB60
0x0450	MB61
0x0460	MB62
0x0470	MB63

Table 16-9: 16-Byte Message Buffers

Address Offset	MBSR = 01 (16-Byte Payload)
0x0080	MB0
0x0098	MB1
0x00B0	MB2
0x00C8	MB3
0x00E0	MB4
0x00F8	MB5
0x0100	MB6
0x0128	MB7
0x0140	MB8

Table 16-9: 16-Byte Message Buffers (Continued)

Address Offset	MBSR = 01 (16-Byte Payload)
0x0158	MB9
0x0170	MB10
0x0188	MB11
0x01A0	MB12
0x01B8	MB13
0x01D0	MB14
0x01E8	MB15
0x0200	MB16
0x0218	MB17
0x0230	MB18
0x0248	MB19
0x0260	MB20
0x0280	MB21
0x0298	MB22
0x02B0	MB23
0x02C8	MB24
0x02E0	MB25
0x02F8	MB26
0x0310	MB27
0x0328	MB28
0x0340	MB29
0x0358	MB30
0x0370	MB31
0x0388	MB32
0x03A0	MB33
0x03B8	MB34
0x03D0	MB35
0x03E8	MB36
0x0400	MB37
0x0418	MB38
0x0430	MB39

Table 16-9: 16-Byte Message Buffers (Continued)

Address Offset	MBSR = 01 (16-Byte Payload)
0x0448	MB40
0x0460	MB41

Table 16-10: 32-Byte Message Buffers

Address Offset	MBDSR = 10 (32-Byte Payload)
0x0080	MB0
0x00A8	MB1
0x00D0	MB2
0x00F8	MB3
0x0120	MB4
0x0248	MB5
0x0170	MB6
0x0198	MB7
0x01C0	MB8
0x01E8	MB9
0x0210	MB10
0x0238	MB11
0x0280	MB12
0x02A8	MB13
0x02D0	MB14
0x02F8	MB15
0x0320	MB16
0x0348	MB17
0x0270	MB18
0x0298	MB19
0x03C0	MB20
0x03E8	MB21
0x0410	MB22
0x0438	MB23

Table 16-11: 64-Byte Message Buffers

Address Offset	MBSR = 11 (64-Byte Payload)
0x0080	MB0
0x0C8	MB1
0x0110	MB2
0x0158	MB3
0x01A0	MB4
0x01E8	MB5
0x0230	MB6
0x0280	MB7
0x02C8	MB8
0x0310	MB9
0x0258	MB10
0x03A0	MB11
0x03E8	MB12
0x0430	MB13

Memory Partition

When CAN FD is enabled, the CANFD RAM can be partitioned in two blocks of 512 bytes. Each block can accommodate a number of message buffers. This depends on the configuration of the size control bit fields in the `CANFD_FD_CTL` register as shown in the *RAM Partition* table.

Table 16-12: RAM Partition

RAM Block	Number of MBs with 8 Bytes[*1]	Size Control Bit Field	Number of MBs of Different Sizes, per Block
0	0 to 31	`CANFD_FD_CTL.MBDSIZR0`	MBDSIZR0=00, 32 MBs with 8 bytes payload MBDSIZR0=01, 21 MBs with 16 bytes payload MBDSIZR0=10, 12 MBs with 32 bytes payload, MBDSIZR0=11, 7 MBs with 64 bytes payload
1	32 to 63	`CANFD_FD_CTL.MBDSIZR1`	MBDSIZR1=00, 32 MBs with 8 bytes payload MBDSIZR1=01, 21 MBs with 16 bytes payload MBDSIZR1=10, 12 MBs with 32 bytes payload MBDSIZR1=11, 7 MBs with 64 bytes payload

*1 default range

When payload sizes of 16, 32, or 64 bytes are configured in some or all RAM blocks, the total number of MBs, and the respective number order may differ from the default configuration of 8 bytes. For example, if block 0 is configured to 8 bytes payload, block 1 to 16 bytes, the following table indicates how the message buffers are arranged into RAM.

Table 16-13: RAM Partition Example

RAM Block	Payload Size	Number of MBs in the RAM Block	MB Range
0	CANFD_FD_CTL.MBDSIZR0 =00, 8 bytes payload	32	0 to 31
1	CANFD_FD_CTL.MBDSIZR1 =01, 16 bytes payload	21	32 to 52

Rx FIFO Structure

When the CANFD_CFG.RFEN bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0-5) is used by the Rx FIFO engine.

The region 0x80 to 0x8C contains the output of the FIFO, which must be read by the processor as a message buffer. The FIFO output contains the oldest message that has been received but not yet read. The region 0x90–0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6 to 37) depending on the CANFD_CTL2.RFFNUM bit field setting, contains the ID filter table (configurable from 8 to 128 table elements), which specifies filtering criteria for accepting frames into the FIFO. Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for the CANFD_CTL2.RFFNUM bit field setting equal to zero.

The *Rx FIFO Structure* figure shows the Rx FIFO data structure.

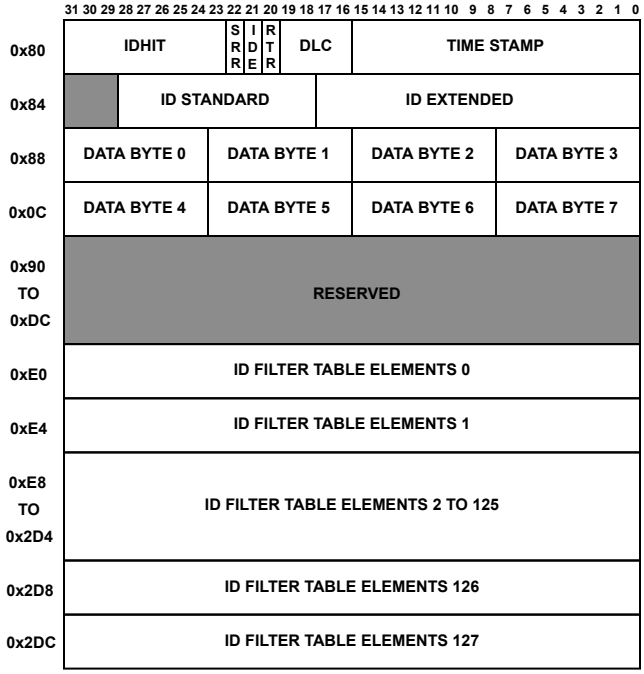


Figure 16-3: Rx FIFO Structure

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the `CANFD_CFG.IDAM` bit field setting.

The *ID Filter Table Structure* figure shows the three different formats of the ID table elements. All elements of the table must have the same format, as selected by the `CANFD_CFG.IDAM` bit field.

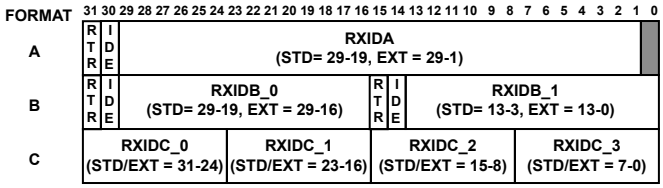


Figure 16-4: ID Filter Table Structure

The *Rx FIFO Filters* table shows the number of Rx FIFO filters configured with the `CANFD_CTL2.RFFNUM` bit field setting.

Table 16-14: Rx FIFO Filters

`CANFD_CTL2.RFFNUM`	**Number of Rx FIFO Filter Elements**	**Message Buffers Occupied by Rx FIFO and ID filter table**	**Remaining Mailboxes**	**Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks**	**Rx FIFO ID filter table elements affected by Rx FIFO global mask**
0x0	8	MB 0 to 7	MB 8 to 63	Elements 0 to 7	None
0x1	16	MB 0 to 9	MB 10 to 63	Elements 0 to 9	Elements 10 to 15
0x2	24	MB 0 to 11	MB 12 to 63	Elements 0 to 11	Elements 12 to 23

Table 16-14: Rx FIFO Filters (Continued)

CANFD_CTL2.RFFNUM	Number of Rx FIFO Filter Elements	Message Buffers Occupied by Rx FIFO and ID filter table	Remaining Mailboxes	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Rx FIFO ID filter table elements affected by Rx FIFO global mask
0x3	32	MB 0 to 13	MB 14 to 63	Elements 0 to 13	Elements 14 to 31
0x4	40	MB 0 to 15	MB 16 to 63	Elements 0 to 15	Elements 16 to 39
0x5	48	MB 0 to 17	MB 18 to 63	Elements 0 to 17	Elements 18 to 47
0x6	56	MB 0 to 19	MB 20 to 63	Elements 0 to 19	Elements 20 to 55
0x7	64	MB 0 to 21	MB 22 to 63	Elements 0 to 21	Elements 22 to 63
0x8	72	MB 0 to 23	MB 24 to 63	Elements 0 to 23	Elements 24 to 71
0x9	80	MB 0 to 25	MB 26 to 63	Elements 0 to 25	Elements 26 to 79
0xA	88	MB 0 to 27	MB 28 to 63	Elements 0 to 27	Elements 28 to 87
0xB	96	MB 0 to 29	MB 30 to 63	Elements 0 to 29	Elements 30 to 95
0xC	104	MB 0 to 31	MB 32 to 63	Elements 0 to 31	Elements 32 to 103
0xD	112	MB 0 to 33	MB 34 to 63	Elements 0 to 31	Elements 32 to 111
0xE	120	MB 0 to 35	MB 36 to 63	Elements 0 to 31	Elements 32 to 119
0xF	128	MB 0 to 37	MB 38 to 63	Elements 0 to 31	Elements 32 to 127

Remote Frame (RTR)

The RTR bit specifies if remote frames are accepted into the FIFO if they match the target ID. If the bit is set, remote frames are accepted and data frames are rejected. If the bit is clear, remote frames are rejected and data frames are accepted.

Extended Frame (IDE)

The IDE bit specifies whether extended or standard frames are accepted into the FIFO if they match the target ID. If the bit is set, extended frames are accepted and standard frames are rejected. If the bit is clear, extended frames are rejected and standard frames are accepted.

Rx Frame Identifier Format A (RXIDA)

Specifies the ID for as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

Rx Frame Identifier Format B (RXIDB_0, RXIDB_1)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

Rx Frame Identifier Format C (RXIDC_1, RXIDC_2, RXIDC_3)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

Identifier Acceptance Filter Hit Indicator (IDHIT)

The 9-bit IDHIT field indicates which identifier acceptance filter is hit by the received message that is in the output of the Rx FIFO. See the Rx FIFO section for more information.

CANFD Processes

The following sections describe the how to transmit and receive CAN and CAN FD frames using the mailbox system.

Transmit Process

To transmit a CAN frame, the processor must prepare a message buffer for transmission by executing the following procedure:

1. Check if the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (1001) to the CODE field of the control and status (C/S) word to request an abortion of the transmission. Wait for the corresponding IFLAG bit to be asserted by polling the `CANFD_IFLG1` or `CANFD_IFLG2` register or by the interrupt request if enabled in the respective IMASK bit in the `CANFD_IMSK1` or `CANFD_IMSK2` register. Then, read back the CODE field to check if the transmission was aborted or transmitted (see the Transmission Abort Mechanism section).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the C/S word to activate the MB. When `CANFD_CFG.FDEN` is set, also write the EDL, BRS and ESI bits.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than the respective MB payload size, the CANFD adds the necessary number of bytes with constant 0xCC pattern to complete the expected DLC.

At the end of the successful transmission, the value of the `CANFD_TMR` register is written into the TIME STAMP field, the CODE field in the C/S word is updated, both `CANFD_CRC` and `CANFD_FD_CRC` registers are updated, a status flag is set in the `CANFD_IFLG1` or `CANFD_IFLG2` register, and an interrupt is generated if allowed by the corresponding CANFD_IFLAGn bit. The new CODE field after transmission depends on the code that was used to activate the MB. See the *Message Buffer Code – Rx Buffer* table.

When the Abort feature is enabled (`CANFD_CFG.ABORTEN` is asserted), after the interrupt flag is asserted for a mailbox configured as transmit buffer, the mailbox is blocked. Therefore, the processor is not able to update it until the interrupt flag is disabled by the processor. This means that the processor must clear the corresponding CANFD_IFLAGn bit before starting to prepare this MB for a new transmission or reception.

Arbitration Process

The arbitration process scans the mailboxes, searching for the Tx mailbox that holds the message to be sent in the next opportunity. This mailbox is called the arbitration winner.

The scan starts from the lowest number mailbox and runs toward the higher ones. The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the `CANFD_CTL2.TXASDLY` field value.
- During the error delimiter field of a CAN frame.
- During the overload delimiter field of a CAN frame.
- When the winner is inactivated, and the CAN bus has still not reached the first bit of the intermission field.
- When there is a processor write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the intermission field.
- When CHI is in idle state and the processor writes to the C/S word of any MB.
- When CANFD module exits the bus off state.
- Upon leaving freeze mode or low power mode.

If the arbitration process does not manage to evaluate all mailboxes before the CAN bus has reached the first bit of the intermission field, the temporary arbitration winner is invalidated and the CANFD module does not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx mailboxes at the end of the scan according to the `CANFD_CTL1.LBUF` and `CANFD_CFG.LPRIOEN` bit settings.

Lowest Number Mailbox First

If the `CANFD_CTL1.LBUF` bit is enabled, the first (lowest number) active Tx mailbox found is the arbitration winner. The `CANFD_CFG.LPRIOEN` bit has no effect when the `CANFD_CTL1.LBUF` bit is enabled.

Highest-Priority Mailbox First

If the `CANFD_CTL1.LBUF` bit is disabled, then the arbitration process searches the active Tx mailbox with the highest priority. This mailbox frame with the highest priority has a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the arbitration value of the mailbox. The highest-priority Tx mailbox is the one that has the lowest arbitration value among all Tx mailboxes.

If two or more mailboxes have equivalent arbitration values, the mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on the `CANFD_CFG.LPRIOEN` bit.

Local Priority Disabled

When the CANFD_CFG.LPRIOEN is disabled, local priority is disabled, and the arbitration value is built in the exact sequence of bits transmitted in a CAN frame as shown in the *Arbitration Value Priority Disabled* table.

Table 16-15: Arbitration Value Priority Disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1bit)	IDE (1 bit)	Extended ID[17:0] (11 bits)	RTR (1bit)

Local Priority Enabled

If the CANFD_CFG.LPRIOEN is enabled, local priority is enabled. In this case, the mailbox PRIO field is included at the very left of the arbitration value as shown in the *Arbitration Value Priority Enabled table.*

Table 16-16: Arbitration Value Priority Enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1bit)	IDE (1 bit)	Extended ID[17:0] (11 bits)	RTR (1bit)

As the PRIO field is the most significant part of the arbitration value, mailboxes with low PRIO values have higher priority than mailboxes with high PRIO values, regardless of the rest of their arbitration values.

NOTE: The PRIO field is not part of the frame on the CAN bus and only affects the internal arbitration process.

Arbitration Completion

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and, after it is done, write access to the C/S word of the corresponding MB is blocked (if CANFD_CFG.ABORTEN is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the processor.
- The CANFD module enters freeze mode or bus off.
- The CANFD module loses the bus arbitration or there is an error during the transmission.

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration Start and Stop Conditions

The arbitration process is triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. The `CANFD_CTL2.TXASDLY` value may be changed to optimize the arbitration start point.
- During CAN bus off state from TX_ERR_CNT = 124 to 128. The `CANFD_CTL2.TXASDLY` value may be changed to optimize the arbitration start point.
- During C/S write by the processor in bus idle mode. The first C/S write starts the arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, the Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, the Tx arbitration machine begins a new arbitration process. If there is a pending arbitration and the bus idle state starts, an arbitration process is triggered. In this case, the first and second C/S write in the bus idle state do not restart the arbitration process. It is possible that there is not enough time to finish arbitration in the wait for bus idle state and the next state is idle. In this case, the scan is not interrupted, and it is completed during the bus idle state. During this arbitration, a C/S write does not cause an arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting freeze mode (first bit of the wait for bus idle state). If there is a re- synchronization during wait for bus idle state, the arbitration process restarts.

Arbitration process stops in the following situations:

- All mailboxes were scanned.
- A Tx active mailbox is found when the lowest buffer feature is enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish the Tx arbitration process (for example, when a deactivation was performed near the end of frame). In this case, the arbitration process is pending.
- There is an error or overload flag in the bus.
- There is a low power or freeze mode request in the idle state.

Arbitration is considered pending as described below:

- It was not possible to finish the arbitration process in time.
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx arbitration process in progress.
- Rx Match has just updated an Rx code to Tx code
- Entering the bus off state.

C/S write during arbitration has the following effect:

- If the C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If the C/S write is performed in an MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

Receive Process

To receive CAN frames into a mailbox, the processor must prepare the mailbox for reception by executing the following steps:

1. If the mailbox is active (either Tx or Rx), inactivate the mailbox (See Mailbox Inactivation), preferably with a safe inactivation (See Transmission Abort Mechanism).
2. Write the ID word.
3. Write the EMPTY code (0b0100) to the CODE field of the C/S word to activate the mailbox. No setup is required for the EDL, BRS, and ESI bits; they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the mailbox is updated by the move-in process (See Move-In) as follows:

1. The received data field (8 bytes at most for classical CAN message format and up to 64 bytes for the CAN FD message format) is stored.
2. The received ID field is stored.
3. The value of the `CANFD_TMR` register at the time of the second bit of the frame identifier field is written into the mailbox TIME STAMP field.
4. The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5. The CODE field in the C/S word is updated according to the *Message Buffer Code – Rx Buffer* and *Message Buffer Code – Tx Buffer* tables.
6. A status flag is set in the `CANFD_IFLG1` or `CANFD_IFLG2` register and an interrupt is generated if allowed by the corresponding `CANFD_IMSK1` or `CANFD_IMSK2` register bit.

The recommended procedure for the processor to service (read) the frame received in a mailbox is:

The processor polls for frame reception by checking the status flag bit for the specific mailbox in the respective CANFD_IFLAGn register and not by polling the CODE field of that mailbox. Polling the CODE field does not work because, once a frame is received and the processor services the mailbox (by reading the C/S word followed by unlocking the mailbox), the CODE field does not return to EMPTY. It remains FULL, as explained in the *Message Buffer Code – Rx Buffer* table. If the processor tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the mailbox without a prior safe inactivation, a newly received frame matching the filter of that mailbox may be lost.

NOTE: Never poll by reading the C/S word directly from the mailboxes. Always read the `CANFD_IFLG1` or `CANFD_IFLG2` register.

The identifier field of the receive frame is always stored in the matching mailbox. Therefore, the contents of the ID field in a mailbox may change if the match was due to masking.

When the CANFD_CFG.SRXDIS bit is enabled, the CANFD module does not store frames transmitted by itself in any MB, even if it contains a matching Rx mailbox, and no interrupt flag or interrupt signal is generated. Otherwise, when the CANFD_CFG.SRXDIS bit is disabled, the CANFD module can receive frames transmitted by itself if there is a matching Rx mailbox.

To be able to receive CAN frames through the Rx FIFO, the processor must enable and configure the Rx FIFO during freeze mode (See Rx FIFO). Upon receiving the frames available in Rx FIFO interrupt (see the CANFD_IFLG1.MB05 bit description), the processor services the received frame using the following procedure:

1. Read the C/S word (optional: needed only if a mask was used for the IDE and RTR bits).
2. Read the ID field (optional: needed only if a mask was used).
3. Read the Data field.
4. Read the CANFD_RX_FIFO register (optional).
5. Clear the frames available in Rx FIFO interrupt by writing a one to the CANFD_IFLG1.MB05 bit to release the MB and allow the processor to read the next Rx FIFO entry.

When the CANFD_CFG.DMAEN bit is enabled, upon receiving a frame in the Rx FIFO, the CANFD_IFLG1.MB05 bit generates a DMA request and does not generate a processor interrupt (See Rx FIFO Under DMA Operation). The CANFD_IMSK1 bits in the Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the C/S word (read 0x80 address, optional).
2. Read the ID field (read 0x84 address, optional).
3. Read all data bytes (start read at 0x88 address, optional).
4. Read the last data bytes (read 0x8C address is mandatory).

Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between mailboxes and FIFO filters. The matching starts from the lowest number MB toward the higher ones. When no match is found within the first structure, then the other is scanned subsequently. When the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- When the received frame is a remote frame, the start point is the CRC field of the frame.

- When the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame.
- When the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame.

When a matching ID is found in the FIFO table or in one of the mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched mailbox by the move-in process. When any CAN protocol error is detected, then no match results are transferred to the FIFO or to the matched mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx mailboxes (CODE is EMPTY, FULL, OVERRUN, or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure as a mailbox. The reception structures (Rx FIFO or mailboxes) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures and depends on conditions described below.

Table 16-17: Matching Architecture

Structure	SMB.RTR	CTRL2.RSS	CTRL2.EA-CEN	MB.IDE	MB.RTR	MB.ID*1	MB.CODE
Mailbox	0	N/A	0	cmp*2	no_cmp*3	cmp_msk*4	
Mailbox	0	N/A	1	cmp_msk	cmp_msk	cmp_msk	
Mailbox	1	0	N/A	cmp	no_cmp	cmp	
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	
FIFO*5	N/A	N/A	N/A	cmp_msk	cmp_msk	cmp_msk	

*1 1. For mailbox structure, If SMB.IDE is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB.IDE is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on CANFD_CFG.IDAM.

*2 cmp: Compares the Rx SMB contents with the MB contents regardless the masks.

*3 no_cmp: The Rx SMB contents are not compared with the MB contents.

*4 cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.

*5 SMB.IDE and SMB.RTR. are not considered when CANFD_CFG.IDAM is format C.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the mailbox is EMPTY.
- The CODE field of the mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the processor and unlocked as described in the Mailbox Lock Mechanism section).
- The CODE field of the mailbox is either FULL or OVERRUN and an inactivation is performed. See the Mailbox Inactivation section.
- The Rx FIFO is not full.

The scan order for mailboxes and the Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for mailboxes is affected by the CANFD_CFG.IRMQEN bit. If the CANFD_CFG.IRMQEN bit is disabled, the matching winner is the first matched mailbox regardless if it is free-to-receive or not. If the CANFD_CFG.IRMQEN bit is enabled, the matching winner is selected according to the priority below:

1. The first free-to-receive matched mailbox.
2. The last non free-to-receive matched mailbox.

It is possible to select the priority of scan between mailboxes and the Rx FIFO by using the-CANFD_CTL2.MRPRIO bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for mailboxes.
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched for among mailboxes as described above.

If the selected priority is mailboxes first:

- If a free-to-receive matched mailbox is found, it is the matching winner regardless of the scan for the Rx FIFO.
- If no matched mailbox is found, then the matching winner is searched for in the scan for the Rx FIFO.

If both conditions above are not satisfied and a non free-to-receive matched mailbox is found, then the matching winner determination is conditioned by the CANFD_CFG.IRMQEN bit:

- If the CANFD_CFG.IRMQEN bit is disabled, the matching winner is the first matched mailbox.
- If the CANFD_CFG.IRMQEN bit is enabled, the matching winner is the Rx FIFO if it is a free-to- receive matched structure; otherwise, the matching winner is the last non free-to- receive matched mailbox.

See the *Matching Possibilities and Resulting Reception Structures* table for a summary of matching possibilities.

Table 16-18: Matching Possibilities and Resulting Reception Structures

RFEN	IRMQEN	MRPRIO	Matched in MB[*1]	Matched in FIFO[*2]	Reception Structure	Description
No FIFO, only MB, match is always MB first:						
0	0	X[*3]	None	_[*4]	None	Frame lost by no match
0	0	X	Free	–	FirstMB	
0	1	X	None	–	None	Frame lost by no match

Table 16-18: Matching Possibilities and Resulting Reception Structures (Continued)

RFEN	IRMQEN	MRPRIO	Matched in MB*1	Matched in FIFO*2	Reception Structure	Description
0	1	X	Free	–	FirstMB	
0	1	X	Not Free	–	LastMB	Overrun
FIFO enabled, no match in FIFO (as if FIFO does not exist):						
1	0	X	None	None	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMB	
1	1	X	Not Free	None	LastMB	Overrun
FIFO enabled, queue disabled:						
1	0	0	X	NotFull	FIFO	
1	0	0	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	Not Free	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	0	1	Free	X	FirstMB	
1	0	1	Not Free	X	FirstMB	Overrun
FIFO enabled, queue enabled:						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	Not Free	Full	LastMB	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	Not Free	NotFull	FIFO	

Table 16-18: Matching Possibilities and Resulting Reception Structures (Continued)

RFEN	IRMQEN	MRPRIO	Matched in MB*1	Matched in FIFO*2	Reception Structure	Description
1	1	1	Not Free	Full	LastMB	Overrun

*1 In the Matched in MB column, the term "None" means the frame has not matched an MB (free-to-receive or non-free-to receive). The term "Free" means the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.

*2 In the Matched in FIFO column, the term "None" means the frame has not matched any filter for the Rx FIFO. It is as if the Rx FIFO does not exist (CANFD_CTL2.RFEN = 0). The term "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it. The term "Full" means that the frame has matched a FIFO filter, but the FIFO couldn't store it because the FIFO has no empty slots to receive it.

*3 The X denotes a don't-care condition.

*4 The – denotes a forbidden condition.

When a non-safe mailbox inactivation (See Mailbox Inactivation) occurs during the matching process and the mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted; it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the Rx FIFO is disabled, the `CANFD_CFG.IRMQEN` is enabled, there are two MBs with the same ID, and the CANFD starts receiving messages with that ID. If the two MBs are the second and the fifth in the array, then when the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive," so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive," so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full-featured FIFO) to allow more time for the processor to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the TIME STAMP field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID acceptance masks. The CANFD module supports individual masking per MB. During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care." The `CANFD_IMSK1` and `CANFD_IMSK2` registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the CANFD module is in freeze mode; otherwise, they are blocked by hardware.

The CANFD also supports an alternate masking scheme with only four mask registers (`CANFD_RX_FIFO_GMSK`, `CANFD_RX_MB_GMSK`, `CANFD_RX_14_MSK`, and `CANFD_RX_15_MSK`) for backward compatibility with legacy applications. This alternate masking scheme is enabled when `CANFD_CFG.IRMQEN` is enabled.

Receive Process Pretending Network Mode

Pretended networking mode adds specific wake up functionality in low power modes (doze mode). When pretended network (PN) mode is enabled by asserting the CANFD_CFG.PNETEN bit, the CANFD module continues processing Rx CAN messages under low power mode, able to detect specific wake up messages by filtering them against ID and payload target values using preselected matching criteria. Wake up functionality is not available for messages in CAN FD format. While in pretended networking mode, CAN FD format messages are ignored.

PN registers are in the 0x0B00–0x0B7C address range and can only be written only in freeze mode. These registers are used for writing PN configuration (both control and target values) prior entering into pretended networking mode, and for reading wake up flags and the received message ID and data when returning to normal mode after waking. The processor waits for the CANFD_CFG.LPMACK bit to be disabled before performing any access to CANFD PN registers.

PN control registers are CANFD_PN_CTL1 and CANFD_PN_CTL2. The control bit fields that configure the filtering criteria are:

- CANFD_PN_CTL1.PLFSEL: Payload filtering selection.
- CANFD_PN_CTL1.IDFSEL: ID filtering selection.
- CANFD_PN_CTL1.FCSEL: Filtering combination selection.

PN target values are:

- CANFD_FLTR_ID1.IDE: IDE target value used to filter the incoming message by its format (standard or extended).
- CANFD_FLTR_ID1.RTR: RTR target value used to filter the incoming message by its type (data or remote frame).
- CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO: Target DLC range used to filter the size of payload part of an incoming message.
- CANFD_FLTR_ID1.VALUE: ID target value used to filter the incoming message ID (equal to, smaller than or equal, greater than or equal, or the lower limit value in an ID range).
- CANFD_FLTR_ID2_IDMSK: ID target value used as the upper limit in an ID range.
- PL1: Payload target value used to filter the incoming message payload (equal to, smaller than or equal, greater than or equal, or the lower limit value in a payload range).
- PL2: Payload target value used as the upper limit in a payload range.

The IDE, RTR, ID, and payload filters have their respective masks. These masks determine which bits are considered in equality comparisons (1 in certain mask positions) and which ones are don't care (0 in other mask positions). ID and payload masks are used only for exact ID and/or exact payload comparisons.

The ID of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact ID value by detecting the equality between the ID field of the incoming message and the content of target CANFD_FLTR_ID1 register. The ID mask is used.
- A match with the maximum range of ID; in other words, any message with ID value smaller than or equal to the content of target CANFD_FLTR_ID11 register is accepted. The ID mask is not used.
- A match with the minimum range of ID; in other words, any message with ID value greater than or equal to the content of target CANFD_FLTR_ID1 register is accepted. The ID mask is not used.
- A match inside a range of IDs; in other words, any message with an ID value that is greater than or equal to the content of target CANFD_FLTR_ID1 register and smaller than or equal to the content of target CANFD_FLTR_ID2_IDMSK register is accepted. The ID mask is not used.

The above criteria for ID filtering must be coherent with CANFD_FLTR_ID1.IDE and CANFD_FLTR_ID1.RTR target values. Only Rx frames that match the respective IDE and RTR bits to the contents of FLT_IDE and FLT_RTR bit fields are compared. When range of IDs is selected (CANFD_PN_CTL1.IDFSEL = 11), both the CANFD_FLTR_ID1.VALUE field and the CANFD_FLTR_ID2_IDMSK register are referred to the same CANFD_FLTR_ID1.IDE and CANFD_FLTR_ID1.RTR bits.

The ID mask is applied only to the exact ID comparison filtering option (CANFD_PN_CTL1.IDFSEL = 00) to determine which bits are considered in the comparison. For the exact match option, the mask can select any bit within the ID field. For maximum range, minimum range and inside range comparisons, the ID mask is not considered.

The IDE and RTR masks are applied in both exact and range ID comparison filtering options to determine which bits are considered in comparison.

Similarly to the ID criteria, 64-bit data or payloads (PL) of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact payload value by detecting the equality between the payload field of the incoming message and the content of PL1 register. The payload mask is used.
- A match with the maximum range of payload–in other words, any message with payload value smaller than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match with the minimum range of payload–in other words, any message with payload value greater than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match inside a range of payloads–in other words, any message with a payload value that is greater than or equal to the content of PL1 register and smaller than or equal to the content of PL2 register is accepted. The payload mask is not used.

The above criteria for payload filtering must be coherent with CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO limit values. The payload of a Rx incoming message is filtered in accordance with the selected criteria only when the DLC value of the Rx incoming message is inside a DLC range:

- Greater than or equal to the CANFD_FLTR_DLC.LO (lower limit) and

- Lower than or equal to the CANFD_FLTR_DLC.HI (upper limit)

Conversely, a DLC value out of the specified range results in mismatch. By making CANFD_FLTR_DLC.LO = CANFD_FLTR_DLC.HI, only payloads of specified quantity of bytes will be filtered. DLC is not maskable.

When the inside range of payloads option is selected (CANFD_PN_CTL1.PLFSEL = 11), both PL1 and PL2 are considered with the 8-byte data length. All the data bytes excluded by the DLC of the received message are considered with value zero.

Payload mask is only used in the exact match option (CANFD_PN_CTL1.PLFSEL = 00) to select which bits or bytes in the 8-byte data field of both Rx incoming message and the contents of PL1 register are selected for matching. Mask length must be in accordance with the expected range of DLC values. For maximum range, minimum range, and inside range comparisons, the payload mask is not considered.

When a remote frame is received by the CANFD module and the CANFD_PN_CTL1.FCSEL bit is configured to select the payload comparison, the payload filtering is not considered and the comparison results in a mismatch.

Rx incoming messages can also be filtered based upon the quantity and rate of message reception, specifically:

- Several messages that match the filtering criteria for ID or payload a predefined quantity of times. This quantity can be configured in the 1 to 255 range. See the CANFD_CTL1 register.
- No message matching the filtering criteria for ID or payload up to a timeout trigger.

  That is, non-reception of a matching message for a defined quantity of time. See the CANFD_CTL2 register.

The CANFD module can generate a wakeup timeout event from an internal timer with associated comparator circuitry capable of generating a timeout flag when the counting reaches the predefined timeout value, as specified in the CANFD_PN_CTL2.MATCHTO bit field.

The above filtering criteria can be used together as follows:

- Message ID filtering only.
- Message ID filtering and payload filtering.
- Message ID filtering only occurring N times.
- Message ID filtering and payload filtering occurring N times.

The timeout counter runs concurrently with the reception filtering process. Both engines, timeout counter and message filtering, are independent. If an incoming message matches the selected filter criteria, the timeout counter keeps counting until the processor wakes up.

Conversely, if the timeout counter reaches the target value, then the message filtering process continue to filter incoming messages until the processor wakes up. The CANFD_WUM.MCNT field reports the number of matched messages that occurred under pretended networking mode up to the moment the processor wakes up.

Under pretended networking mode, a wakeup event that occurs sets the respective wake up flag (see the CANFD_WUM register.

- In case of a successful match with the selected filtering criteria, the CANFD_WUM.WUMF bit field.

- In case of a timeout trigger, the CANFD_WUM.WTOF bit field.

Either of these flags generates an interrupt to the processor if the respective mask bit is asserted (CANFD_PN_CTL1.WUMFMSK or CANFD_PN_CTL1.WTOFMSK).

There are four wake up message buffers (WMBs) for storing incoming messages in pretended networking mode. They stored in the following registers: CANFD_WMB0_ID, CANFD_WMB1_ID, CANFD_WMB2_ID, and CANFD_WMB3_ID When the CANFD_PN_CTL1.MATCHCNT value is one, just one message is received if matching the filtering criteria, and this message is stored in the CANFD_WMB0 register. If the CANFD_PN_CTL1.MATCHCNT value is between two and four, CANFD_WMB1, CANFD_WMB2, and CANFD_WMB3 are used to store the second, third, and fourth matching messages, respectively. If CANFD_PN_CTL1.MATCHCNT is greater than four, the last four matching messages are stored in the WMBs; respecting the WMB index to indicate the arrival order, the latest is stored in CANFD_WMB3. Only the valid data bytes of the incoming match message is stored in data field of WMBs. The non-valid data bytes are read as zero. In case the of DLC = 0 and RTR = 1, the data field is filled with zero. In any of the above cases, the wake up interrupt is generated just when the filtering criteria is completed and the CANFD_PN_CTL1.WUMFMSK bit is set.

When a non-match wakeup event occurs (timeout or external) and the CANFD_WUM.MCNT value is equal or greater than four, the message stored in WMB0 does not have a valid content. The CANFD_WMB0 is used as the buffer for the current message on the CAN bus. Messages received during pretended networking mode do not have timestamps and respective field in the WMB structure must be ignored.

Under low-power mode (doze or stop), all processes are shut down except for the PN functionality in the PE submodule, which is kept clocked by the oscillator clock. The CANFD continues to receive Rx incoming messages and just compares them against the predefined target values and according to the selected filtering criteria. The matching, arbitration, move-in, and move-out processes, normally available in normal mode, are not performed under pretended networking mode.

Under pretended networking, the CANFD reacts to messages on the CAN bus in the same manner as in normal mode (generates acknowledge bits, detects and counts errors, etc.).

Move Process

There are two types of move process: move-in and move-out.

Move-In

The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the CANFD_RX_FIFO register. Each Rx SMB has its own move-in process, but only one is performed at a given time as described below. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (See Matching Process) and all of the following conditions are true:

- The CAN bus has reached or let past either of the following:
  - The second bit of the Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB

- There is no ongoing matching process.
- The destination mailbox is not locked by the processor.
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time, both are performed and the newest substitutes the oldest.

The term pending *move-in* is used throughout the documentation and stands for a *move-to- be* that still does not satisfy all the previously mentioned conditions.

The move-in is canceled and the Rx SMB can receive another message when any of the following conditions is satisfied:

- The destination mailbox is inactivated after the CAN bus has reached the first bit of intermission field next to the frame that carried the message and its matching process has finished.
- There is a previous pending move-in to the same destination mailbox.
- The Rx SMB is receiving a frame transmitted by the CANFD itself and the self-reception is disabled (`CANFD_CFG.SRXDIS` bit is enabled)
- Any CAN protocol error is detected.

Note that the pending move-in is not canceled if the CANFD module enters freeze or low-power mode. The pending move-in stays on hold waiting for exit from freeze or low-power mode and to be unlocked. If an MB is unlocked during freeze mode, the move-in happens immediately.

The move-in process is the execution by the CANFD module of the following steps:

1. Push `CANFD_RX_FIFO.IDHIT` into the `CANFD_RX_FIFO` FIFO if the message is destined to the Rx FIFO.
2. Read all data words from the Rx SMB according to the selected payload size for the Rx storage element.
3. Write all data words to the Rx mailbox according to the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4. Read the C/S and ID words from the Rx SMB.
5. Write C/S and ID words to the Rx mailbox and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately canceled by the inactivation of the destination mailbox (See Mailbox Inactivation). In this case, the mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO message buffer, in which case the process cannot be canceled.

The BUSY bit (least significant bit of the CODE field) of the destination message buffer is asserted while the move-in is being performed to alert the processor that the message buffer content is temporarily incoherent.

Move-Out

The move-out process is the copy of the content from a Tx mailbox to the Tx SMB when a message for transmission is available (See Arbitration Process). The move-out occurs in the following conditions:

- The first bit of intermission field.
- During the bus off state when Tx error counter is in the 124 to 128 range.
- During bus idle state.
- During the wait for bus idle state.

The move-out process is not atomic. Only the processor has priority to access the memory concurrently out of bus idle state. In bus idle, the move-out has the lowest priority to the concurrent memory accesses.

Data Coherence

In order to maintain data coherency and proper operation of the CANFD module, the processor must obey the rules described in the Transmit Process and Receive Process sections.

Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the processor if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- The `CANFD_CFG.ABORTEN` bit is enabled.
- The first processor action must be the writing of abort code (1001) into the CODE field of the C/S word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a mailbox that is currently being transmitted or to a mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked, and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The CANFD module loses the bus arbitration.
- There is an error during the transmission.
- The CANFD module is put into freeze mode.
- The CANFD enters the bus off state.
- There is an overload frame.

When none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the respective `CANFD_IFLG1` or `CANFD_IFLG2` register, and an interrupt to the processor is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, when one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the

interrupt flag is set in the respective CANFD_IFLG1 or CANFD_IFLG2 register, and an interrupt is (optionally) generated.

When the processor writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated, and the interrupt flag is set. The processor just needs to read the abort code to make sure the active MB was safely inactivated. Although the CANFD_CFG.ABORTEN bit is asserted and the processor wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- The processor checks the corresponding IFLAG and clears it, if asserted.
- The processor writes 1001 into the CODE field of the C/S word.
- The processor waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- The processor reads the CODE field to check if the frame was either transmitted (CODE = 1000) or aborted (CODE = 1001).
- It is necessary to clear the corresponding IFLAG to allow the MB to be reconfigured.
- It is necessary to reconfigure the EDL, BRS, and ESI fields of the aborted MB before transmitting it again.

Mailbox Inactivation

Inactivation is a mechanism provided to protect the mailbox against updates by the CANFD internal processes, thus allowing the processor to rely on mailbox data coherence after having updated it, even in normal mode.

Inactivation of transmission mailboxes must be performed only when the CANFD_CFG.ABORTEN bit is deasserted.

If a mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated.

To inactivate a mailbox, the processor must update its CODE field to INACTIVE (either 0000 or 1000).

Because the processor is not able to synchronize the CODE field update with the CANFD internal processes, an inactivation has the following consequences:

- A frame on the bus that matches the filtering of the inactivated Rx mailbox may be lost without notice, even if there are other mailboxes with the same filter.
- A frame containing the message within the inactivated Tx mailbox may be transmitted without setting the respective IFLAG.

To perform a safe inactivation and avoid the above consequences for Tx mailboxes, the processor must use the transmission abort mechanism .

The inactivation automatically unlocks the mailbox (See Mailbox Lock Mechanism).

NOTE: Message buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region. The processor must maintain data coherency in the FIFO region when the CANFD_CFG.RFEN bit is enabled.

Mailbox Lock Mechanism

Other than mailbox inactivation, the CANFD module has another data coherence mechanism for the receive process. When the processor reads the C/S word of an Rx MB with codes FULL or OVERRUN, the CANFD assumes that the processor wants to read the whole MB in an atomic operation and therefore it sets an internal lock flag for that MB. The lock is released when the processor reads the CANFD_TIMING register (global unlock operation), or when it reads the C/S word of another MB regardless of its code. A processor write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the processor is reading it.

NOTE: The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code other than INACTIVE (0000) or EMPTY (0100). Tx MBs can not be locked. When the CANFD_CFG.IRMQEN bit is disabled, reading the C/S word locked the MB even if it was EMPTY.

When the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and the CANFD module has already received and stored messages into these two MBs. If the processor then attempts to read MB number five and at the same time another message with the same ID is arriving, then when the processor reads the C/S word of MB number five, this MB is locked. The new message arrives, and the matching algorithm finds out that there are no “free-to-receive” MBs, it decides to override MB number five. However, this MB is locked, so the new message cannot be written there. It remains in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

When the MB is not unlocked in time and yet another new message with the same ID arrives, the new message overwrites the message in the Rx SMB and there is no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the Rx SMB to the MB, the BUSY bit on the CODE field is enabled. If the processor reads the C/S word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is cleared.

NOTE: If the BUSY bit is enabled or if the MB is empty, then reading the C/S word does not lock the MB.

Inactivation takes precedence over locking. If the processor inactivates a locked Rx MB, then the lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB is transferred to the MB. An MB is unlocked when the processor reads the CANFD_TMR register, or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both normal and freeze modes.

An unlock during normal or freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode, and it takes place only when the module resumes to normal or freeze mode.

Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the CANFD_CFG.RFEN bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the CANFD module that did not have the FIFO feature.

NOTE: The Rx FIFO must not be enabled when the CAN FD feature is enabled.

The Rx FIFO is 6 messages deep. The memory region occupied by the FIFO structure (both message buffers and FIFO engine) is described in the Rx FIFO Structure section. The processor can read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the FIFO.

The frames available bit (CANFD_IFLG1.MB05)is enabled when there is at least one frame available to be read from the Rx FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the processor reads the message (accessing the output of the FIFO as a message buffer) and the CANFD_RX_FIFO register, then clears the interrupt. If there are more messages in the Rx FIFO, the act of clearing the interrupt updates the output of the FIFO with the next message and updates the CANFD_RX_FIFO register with the attributes of that message, reissuing the interrupt to the processor. Otherwise, the flag remains negated. The output of the Rx FIFO is only valid while the CANFD_IFLG1.MB05 bit enabled.

The Rx FIFO warning bit (CANFD_IFLG1.MB06) is enabled when the number of unread messages in the Rx FIFO is increased to five from four due to the reception of a new message, meaning that the Rx FIFO is almost full. The flag remains enabled until the processor clears it.

The Rx FIFO overflow bit (CANFD_IFLG1.MB07) is enabled when an incoming message is lost because the Rx FIFO is full. The CANFD_IFLG1.MB07 bit is not set when the Rx FIFO is full, and the message is captured by a mailbox. The CANFD_IFLG1.MB07 remains set until the processor clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated when an IFLAG bit asserts and the corresponding mask bit is enabled.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to the CANFD_CTL2.RFFNUM bit setting, which is configurable to one of the following formats:

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

NOTE: A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding identifier acceptance filter hit indicator (IDHIT) that can read in the IDHIT field from C/S word. Another way the processor can obtain this information is by accessing the CANFD_RX_FIFO register. The CANFD_RX_FIFO.IDHIT field refers to the message at the output of the FIFO

and is valid while the CANFD_IFLG1.MB05 flag is enabled. The CANFD_RX_FIFO register must be read only before clearing the flag, to guarantee that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the individual mask registers (CANFD_IMSK1 and CANFD_IMSK2), according to the setting of the CANFD_CTL2.RFFNUM bit, allowing very powerful filtering criteria to be defined. If the CANFD_CFG.IRMQEN bit is disabled, then the FIFO filter table is affected by the CANFD_RX_FIFO_GMSK register value.

Rx FIFO Under DMA Operation

The Rx FIFO can support DMA. Using the Rx FIFO with the DMA feature is enabled by enabling both CANFD_CFG.RFEN and CANFD_CFG.DMAEN bits. The DMA controller reads the received message by reading a message buffer structure at the FIFO output port at the 0x80–0x8C address range.

The reset value of the CANFD_CFG.DMAEN bit is zero. When the CANFD_CFG.DMAEN is enabled, the processor must not access the FIFO output port address range. Before enabling the CANFD_CFG.DMAEN bit, the processor must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling the CANFD_CFG.DMAEN, the processor must perform a clear FIFO operation.

The Frames available in Rx FIFO bit (CANFD_IFLG1.MB05) is asserted when there is at least one frame available to be read from the FIFO; consequently, a DMA request is generated simultaneously. Upon receiving the request, the DMA controller reads the message (accessing the output of the FIFO as a message buffer). The DMA reading process must end by reading address 0x8C, which clears the CANFD_IFLG1.MB05 bit and updates both the FIFO output with the next message (if FIFO is not empty) and the CANFD_RX_FIFO register with the attributes of the new message. If there are more messages stored in the FIFO, the CANFD_IFLG1.MB05 bit is re-enabled, and another DMA request is issued.

NOTE: The CANFD_RX_FIFO register contents cannot be read after the DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 0x080.

The CANFD_IFLG1.MB06 and CANFD_IFLG1.MB07 flags are not used when the DMA feature is enabled.

When the CANFD module is working with DMA, the processor does not receive any Rx FIFO interrupts and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

There is no dedicated DMA channel for reading the CANFD FIFO, however, the MDMA along with the TRU can read from the FIFO RAM. This works by configuring the CANFD module to generate a trigger whenever the CANFD_IFLG1.MB05 bit is set. The MDMA is configured to wait for the trigger from the CANFD module and on receiving the trigger, read from the CANFD FIFO start address at 0x80 offset. Once 16 bytes of FIFO data are read, the CANFD_IFLG1.MB05 bit is cleared.

Only MDMA2 and MDMA6 streams can access the CANFD FIFO.

Clear FIFO Operation

When the CANFD_CFG.RFEN is enabled, the clear FIFO operation empties the FIFO contents. With the CANFD_CFG.RFEN bit enabled, the clear FIFO occurs when the processor writes a one to the

CANFD_IFLG1.MB00 bit. This operation only can only occur in freeze mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs; consequently, the processor must service all FIFO IFLAGs before executing the clear FIFO task.

When the Rx FIFO is working with DMA, the clear FIFO operation clears the CANFD_IFLG1.MB05 flag and the DMA request is canceled.

NOTE: The clear FIFO operation does not clear IFLAGs, except when the CANFD_CFG.DMAEN bit is enabled. When the CANFD_CFG.DMAEN bit is enabled, only the CANFD_IFLG1.MB05 flag is cleared.

CAN Protocol Features

This section describes the CAN protocol related features.

CAN FD Frames

The ISO 11898-1 standard specifies the classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN flexible data rate frame format. The classical frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The flexible data rate frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 bytes per frame. The CANFD module can receive and transmit CAN FD messages interleaved with classical CAN messages.

There are three additional control bits in the CAN FD frame:

- The extended data length (EDL) bit enables a longer data payload with different data length coding.
- The bit rate switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame.
- The error state indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes.

There are no remote frames in the CAN FD format. A message configured to transmit a remote frame is always sent out in the classical CAN format. When a FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is cleared. The RTR bit is considered in only in classical CAN frames.

CAN FD messages can be formatted as long frames, in which the data field exceeds 8 bytes and may range from 12 up to 64 bytes. CAN FD messages can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

Messages in the classical CAN format are limited to transport a maximum payload of 8 bytes at nominal rate. The *CAN Message Format* figure illustrates the message formats for classical and FD frames with either a standard or extended ID.

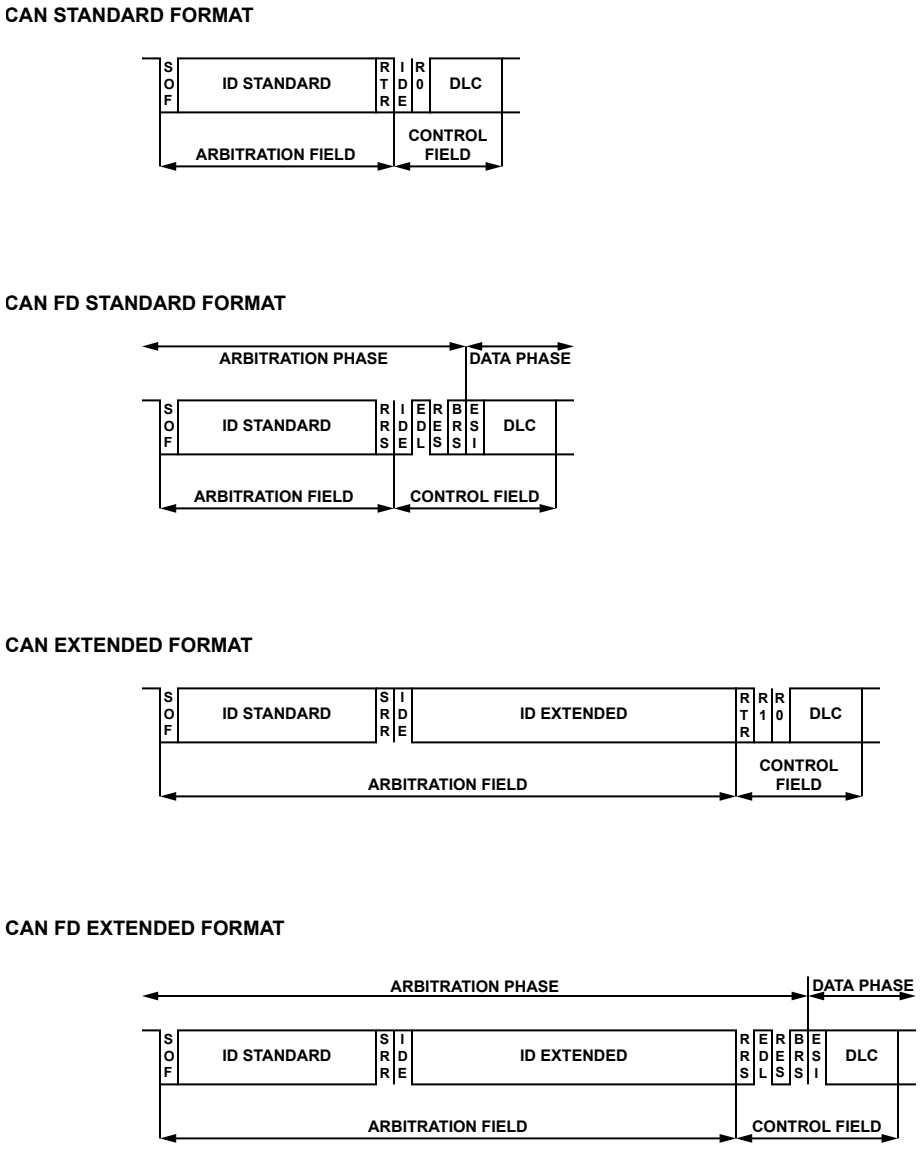


Figure 16-5: CAN Message Format

The ability to receive and transmit CAN FD messages is enabled by the `CANFD_CFG.FDEN` bit. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN FD frame is recognized by a recessive EDL bit, while a classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded according to the DLC field value.

CANFD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the start of frame (SOF) bit until the bit rate switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC delimiter bit, also named the data phase, is transmitted with the data bit rate defined by a second set of CAN data bit timing configuration values. Finally, from the CRC delimiter until the intermission bits, the transmission resumes to nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the

CAN FD arbitration phase, the nominal CAN bit timing defined by the CANFD_TIMING register and by the CANFD_CTL1 register for backward compatibility. Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by the CANFD_FD_TIMING register.

NOTE: If the length of the time quantum in the nominal bit timing and the length of the time quantum in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase and lasts until the next synchronization event. Thus, the length of the time quantum should be the same in nominal and data bit timing to minimize the chance of error frames on the CAN bus, and to optimize the clock tolerance in networks that use CAN FD frames.

The CANFD_FD_CTL.BRSEN bit enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. When the bit is cleared, the transmission is performed at nominal rate regardless of the BRS bit value. The CANFD_FD_CTL.BRSEN bit can be written any time but takes effect only for the next message transmitted or received.

The nominal bit timing is resumed at either the sample point of the CRC delimiter bit or when an error is detected, whichever occurs first. The *CAN FD Message Bit Rate Switching* figure describes the mechanism for entering and leaving the data phase when the BRS bit is recessive.

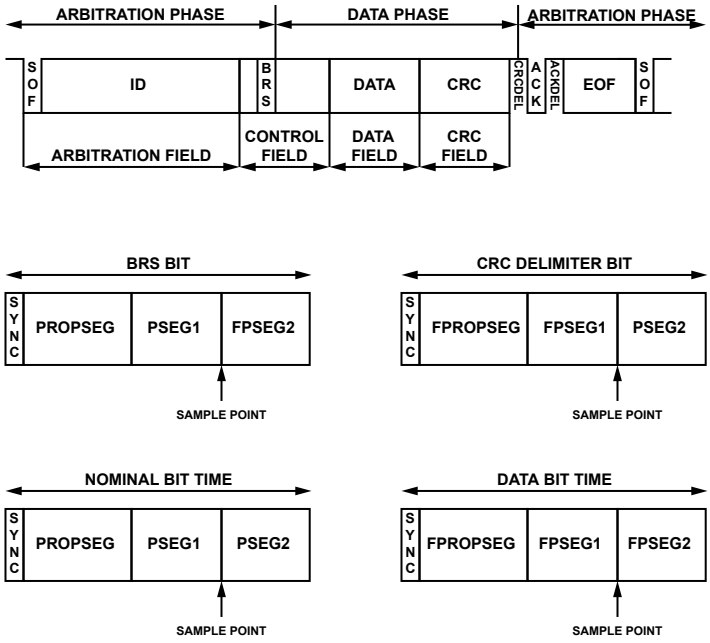


Figure 16-6: CAN FD Message Bit Rate Switching

NOTE: In classical CAN frames, the CRC delimiter is one single recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. The CANFD module sends only one recessive bit as the CRC delimiter, but it accepts two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. As a receiver, the CANFD module sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, the CANFD module accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The value of the ESI bit is determined either by the error state of the transmitter at the start of the transmission, if the frame is originated in the CANFD node, or by the original transmitting node if the CANFD module is acting as a gateway for the message. If the transmitter is error passive, the ESI is transmitted recessive; otherwise, it is transmitted dominant.

There are different CRC polynomials for different CAN frame formats:

- The first polynomial, CRC_15, is for all frames in classical CAN format.
- The second, CRC_17, is for frames in CAN FD format with a data field up to 16 bytes long.
- The third, CRC_21, is for frames in CAN FD format with a data field longer than 16 bytes.

Each polynomial results in a hamming distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. When receiving a message, the CANFD module decodes the EDL and DLC bits to select the proper CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the CANFD_FD_CRC register stores the calculated CRC for the transmitted message, with the adequate length for the type of message, for both CAN FD and non-FD messages. The CANFD_CRC register reports a valid CRC for classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When the CANFD module is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When the CANFD module is receiving a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A stuff error is detected if the fixed stuff bit has the same value as its preceding bit.

The CANFD module detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters CANFD_ECR.RXERRCNT and CANFD_ECR.TXERRCNT in the CANFD_ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indistinctly. There are two extra error counters (CANFD_ECR.RXERRCNTF and CANFD_ECR.TXERRCNTF) that accumulate Rx and Tx errors occurring in the data phase of CAN FD frames with the BRS bit set. The rules for updating the error counters are the same for both CAN FD and non-FD frames.

The error flags CANFD_ESR1.B1ERR, CANFD_ESR1.B0ERR, CANFD_ESR1.ACKERR, CANFD_ESR1.CRCERR, CANFD_ESR1.FRMERR, and CANFD_ESR1.STFERR report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if the CANFD_CTL1.ERRMSK bit is enabled. The CANFD_ESR1 register has additional error flags (CANFD_ESR1.B1ERRF, CANFD_ESR1.B0ERRF, CANFD_ESR1.CRCERRF, CANFD_ESR1.FRMERRF, and CANFD_ESR1.STFERRF) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set. There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported with the CANFD_ESR1.FLTCONF bit is the same for both CAN FD and classical CAN frames, and is based on CANFD_ECR.RXERRCNT

and CANFD_ECR.TXERRCNT error counters only. Information contained in CANFD_ECR.RXERRCNTF and CANFD_ECR.TXERRCNTF counters may be considered as status to help detect the error nature related to the bit rate value. The *CANFD State Related Bit Encoding* table shows how the CANFD state is related to the CANFD_ESR1.SYNC, CANFD_ESR1.IDLE, CANFD_ESR1.TXINPROG, and CANFD_ESR1.RXINPROG bits.

Table 16-19: CANFD State Related Bit Encoding

SYNC	IDLE1	TXINPROG	RXINPROG	CANFD State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

When the CANFD is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an error flag

Resynchronization and hard synchronization occur in CAN FD frames in the same way as in classical CAN frames. A hard synchronization is also performed at the recessive-to-dominant edge from EDL to R0 in CAN FD format frames. The CANFD does not resynchronize while transmitting in the CAN FD data phase.

Transceiver Delay Compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the BRS bit of the message is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled, the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

The CANFD module supports an optional transceiver delay compensation (TDC) mechanism that defines a secondary sample point where the transmitted bit is correctly compared with the received bit to check for bit errors.

The TDC mechanism is enabled by the CANFD_FD_CTL.TDCOMPEN bit and is effective only during the data phase of CAN FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at normal bit rate. The TDC mode is active from the sample point of the BRS bit until the sample point of the CRC delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE: The actual value of the CRC delimiter bit is disregarded by transmitters using the TDC mechanism. A global error at the end of the CRC field will cause the receivers to send error frames that the transmitter will detect during an acknowledge or end of frame.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit, as shown in the *Transceiver Loop Delay Measurement* figure.

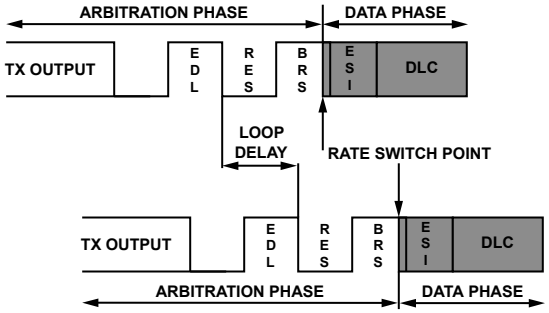


Figure 16-7: Transceiver Loop Delay Measurement

The loop delay is measured in PE clock periods (CANCLK) , from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified with the `CANFD_FD_CTL.TDCOMPOFF` bit. The `CANFD_FD_CTL.TDCOMPVAL` bit field stores the result of this calculation. The `CANFD_FD_CTL.TDCOMPVAL` value saturates at the maximum value of 15 CANCLK when the delay measurement is too long.

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset (`CANFD_FD_CTL.TDCOMPOFF`) shifts the secondary sample point from the edge to an intermediate point inside the bit time (for example, half of the bit time in the data phase), far away from its edges. Therefore, the `CANFD_FD_CTL.TDCOMPOFF` value cannot be larger than the CAN bit duration in the data phase.

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on the `CANFD_FD_CTL.TDCOMPVAL` value is started which ends with the comparison of the received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the CANFD module issues an error flag to the CAN bus at the next sample point.

During the arbitration phase, the delay compensation is always disabled. The maximum delay that can be compensated by the transceiver delay compensation during the data phase is 3 CAN bit times - 2 Tq. Beyond this limit, the `CANFD_FD_CTL.TDCOMPFAIL` flag is set to indicate when the TDC mechanism is out of range and unable to compensate the transceiver loop delay.

Remote Frames

A remote frame is a special kind of frame. The processor can program a mailbox to be a remote request frame by configuring the mailbox to transmit with the RTR bit set to one. After the remote request frame is transmitted successfully, the mailbox becomes a receive message buffer, with the same ID as before.

When a remote request frame is received by the CANFD module, it is treated in one of the following ways, depending on the `CANFD_CTL2.RRS` and `CANFD_CFG.RFEN` bits:

- When RRS is disabled, the frame ID is compared to the IDs of the transmit message buffers with the CODE field 1010. If there is a matching ID, then this mailbox frame is transmitted. Note that if the matching mailbox has the RTR bit set, then the CANFD module transmits a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- When RRS is enabled, the frame ID is compared to the IDs of the receive mailboxes with the CODE field 0100, 0010, or 0110. If there is a matching ID, then this mailbox stores the remote frame in the same fashion of a data frame. No automatic remote response frame is generated. The mask registers are used in the matching process.
- When RFEN is asserted, the CANFD does not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it is stored in the FIFO and presented to the processor. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote request frames are considered normal frames,and generate a FIFO overflow when a successful reception occurs, and the FIFO is already full.

NOTE: There is no remote frame in the CAN FD format. The RTR bit is replaced by a fixed dominant RRS bit. The CANFD receives and transmits remote frames in the classical CAN format.

Overload Frames

The CANFD module transmits overload frames due to the detection of the following CAN bus conditions:

- Detection of a dominant bit in the first/second bit of intermission.
- Detection of a dominant bit at the 7th bit (last) of end of frame field (Rx frames).
- Detection of a dominant bit at the 8th bit (last) of error frame delimiter or overload frame delimiter.

Time Stamp

The value of the CANFD_TMR register is sampled at the beginning of the identifier field on the CAN bus and stored at the end of the move-in process in the TIME STAMP field, providing network behavior with respect to time. The free running timer is clocked by the CANFD bit-clock, which defines the baud rate on the CAN bus. During a message transmission or reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The CANFD_TMR register is not incremented during module disable, doze, stop, and freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization.

Protocol Timing

The *CAN Engine Clocking Scheme* figure shows the structure of the clock generation circuitry that prescales the CAND module clock to generate the Sclock.

Figure 16-8: CAN Engine Clocking Scheme

The CANFD module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The CANFD_CTL1 register has fields to control bit timing parameters: CANFD_CTL1.PRESDIV, CANFD_CTL1.PROPSEG, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2, and CANFD_CTL1.RJW.

The CANFD_TIMING register extends the range of the CAN bit timing variables in the CANFD_CTL1 register. The CANFD_TIMING register provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the BRS bit set.

NOTE: When the CAN FD feature is enabled, always set the CANFD_TIMING.BTF bit and configure the CAN bit timing variables in the CANFD_TIMING register.

The CANFD_CTL1.PRESDIV bit field (as well as the extended range CANFD_TIMING.EPRESDIV and CANFD_FD_TIMING.FPRESDIV for the data phase bits of CAN FD messages) defines the prescaler value that generates Sclock. The Sclock period defines the time quantum used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$Tq = (PRESDIV + 1) / f_{CANCLK}$

The bit rate, which defines the rate at which the CAN message is either received or transmitted, is given by the formula:

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

Bit Rate = 1/CAN Bit Time

A bit time is subdivided into three segments:

- SYNC_SEG – Has a fixed length of one time quantum. Signal edges are expected to occur within this segment.
- Time segment 1 – Includes the propagation segment and the phase segment 1 of the CAN standard. Time segment 1 is programmed by setting the CANFD_CTL1.PROPSEG and the CANFD_CTL1.PSEG1 fields so that their sum (plus 2) is in the range of 2 to 16 time quanta. When the CANFD_TIMING.BTF bit is enabled, the CANFD module uses the CANFD_TIMING.EPROPSEG and CANFD_TIMING.EPSEG1 bit fields so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, the CANFD uses CANFD_FD_TIMING.FPROPSEG and CANFD_FD_TIMING.FPSEG1 bit fields, so that their sum (plus 1) is in the range of 2 to 39 time quanta.
- Time Segment 2 – Represents the phase segment 2 of the CAN standard. It can be programmed by setting the CANFD_CTL1.PSEG2 bit field (plus 1) to be 2 to 8 time quanta long. When the CANFD_TIMING.BTF

bit is enabled, the CANFD module configures the CANFD_TIMING.EPSEG2 bit field with a value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, the CANFD uses the CANFD_FD_TIMING.FPSEG2 bit field instead, so that the value (plus 1) is in the range of 2 to 8 time quanta. The time segment 2 cannot be smaller than the information processing time (IPT), which is 2 time quanta.

See the *Bit Time Segment Example One* figure for an example using CANFD_CTL1 register bit timing variables for classical can format. See the *Bit Time Segment Example Two* figure for an example using CANFD_TIMING and CANFD_FD_TIMING register bit timing variables for CAN FD format. See the *Time Segment Syntax* table for syntax descriptions.

NOTE: For further explanation of the underlying concepts, see the ISO 11898-1 standard and the CAN 2.0A/B protocol specification for bit timing.

The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an IPT of 2, which is the value implemented in the CANFD module.

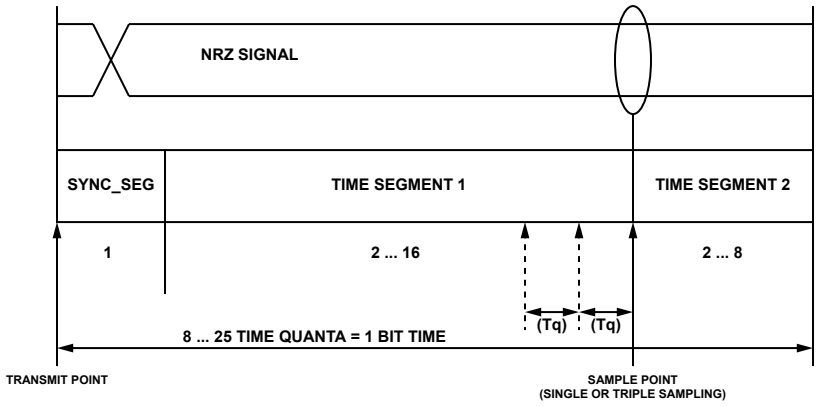


Figure 16-9: Bit Time Segment Example One

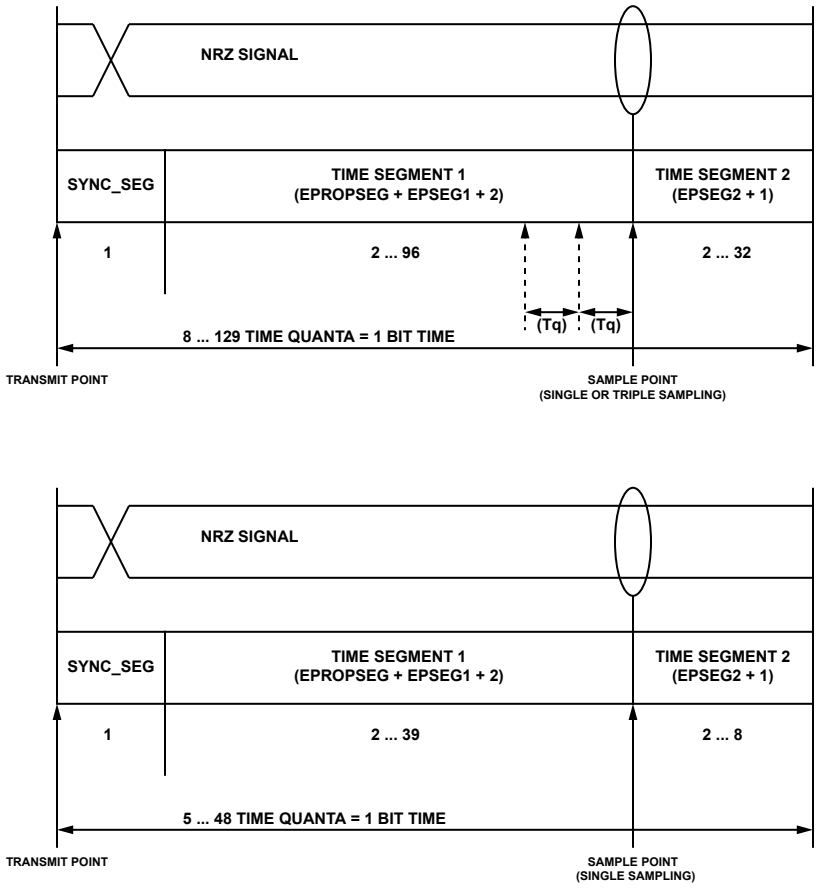


Figure 16-10: Bit Time Segment Example Two

Table 16-20: Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The *BOSCH CAN 2.0B Standard Compliant Bit Time Segment Settings* table gives some examples of the CAN compliant segment settings for classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 16-21: BOSCH CAN 2.0B Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Resynchronization Jump Width
	2	1 .. 2

Table 16-21: BOSCH CAN 2.0B Standard Compliant Bit Time Segment Settings (Continued)

Time Segment 1	Time Segment 2	Resynchronization Jump Width
5 .. 10		
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

NOTE: Ensure the bit time settings comply with the CAN Protocol standard (ISO 11898-1).

Whenever a CAN bit is used as a measure of time duration (for example, estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) is calculated as NumClkBit = (f_{SYS}/f_{CANCLK}) x (PRESDIV + 1) x (PROPSEG + PSEG1 + PSEG2 + 4), where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- f_{CANCLK} is the PE clock in Hz
- f_{SYS} is the frequency of operation of the system CHI clock in Hz.
- PSEG1 is the `CANFD_CTL1.PSEG1` value.
- PSEG2 is the `CANFD_CTL1.PSEG2` value.
- PROPSEG is the `CANFD_CTL1.PROPSEG` value.
- PRESDIV is the `CANFD_CTL1.PRESDIV` value.

The formula above is also applicable to the alternative CAN bit timing variables described in the `CANFD_TIMING` register and the `CANFD_FD_TIMING` register. For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

Arbitration and Matching Timing

During normal reception and transmission, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

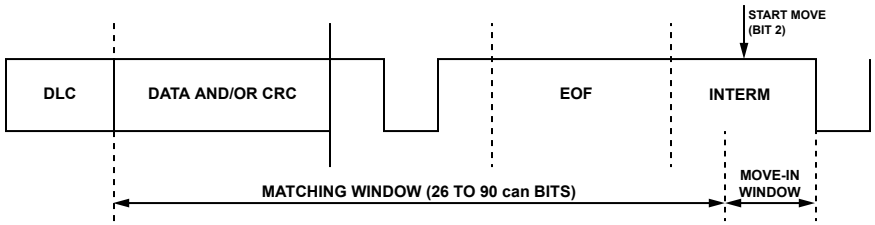


Figure 16-11: Matching and Move-In Time Windows

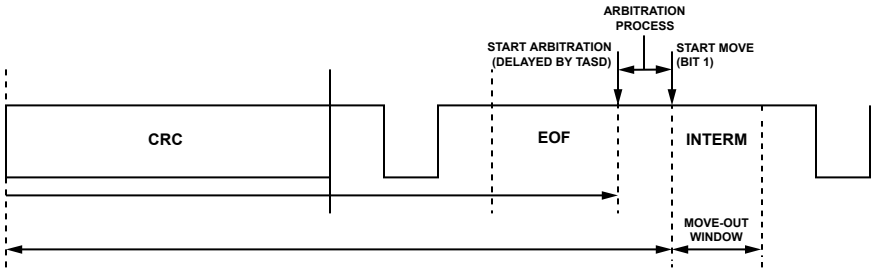


Figure 16-12: Arbitration And Move-out Time Windows

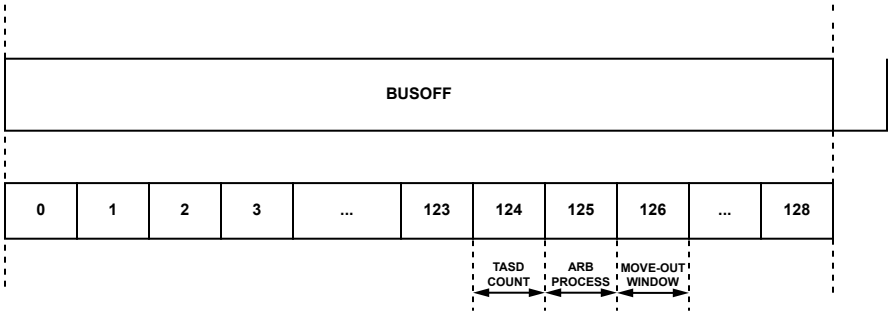


Figure 16-13: Arbitration At The End Of Bus Off And Move-out Time Windows

NOTE: In the previous figures, the matching and arbitration timing does not consider the delay caused by the concurrent memory access due to the processor or other internal CANFD submodules.

Tx Arbitration Start Delay

The Tx arbitration start delay (`CANFD_CTL2.TXASDLY` bit field) is a variable that indicates the number of CAN bits used by the CANFD to delay the Tx arbitration process start point from the first bit of the CRC field of the current frame.

The `CANFD_CTL2.TXASDLY` bit field can only be written only in freeze mode and is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the processor to reconfigure MBs for transmission after the end of the internal arbitration process, in which the CANFD module finds the winner MB for transmission according to the standard arbitration process. If the arbitration ends before the first bit of the intermission field, then there is a chance the processor reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

The `CANFD_CTL2.TXASDLY` bit field is useful to optimize the transmission performance by defining the arbitration start point, as shown in the next figure, based on factors such as the CAN bit timing variables that determine the CAN bit rate and the number of MBs in use by the matching and arbitration processes.

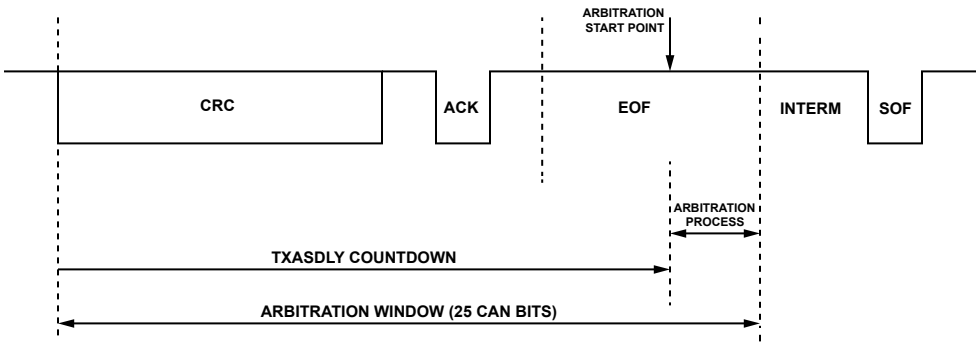


Figure 16-14: Optimal Tx Arbitration Start Point

The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the arbitration is placed closer to the frame's end, adding more delay to its start point, and vice-versa.

When the `CANFD_CTL2.TXASDLY` field is set to zero, the arbitration start is not delayed, and more time is reserved for arbitration. On the other hand, when the `CANFD_CTL2.TXASDLY` value is close to 24, then the processor can configure a Tx MB later and less time is reserved for arbitration. When too little time is reserved for arbitration, the CANFD module may not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal `CANFD_CTL2.TXASDLY` value is calculated as follows:

For CAN FD frames and (MAXMB + 1) $\leq NMB_{END}$, TXASDLY = 31 – ((2*(MAXMB + 1) + 4)/$CPCB_N$)

For CAN FD frames and (MAXMB + 1) > NMB_{END}, TXASDLY = 22 – ((2*(MAXMB + 1) - NMB_{END})/ $CPCB_F$)

For non-FD frames, TXASDLY = 25 – (((2*(MAXMB + 1) + 4)) / CPCB) where:

NMB_{END} = ((9 * CPCBN) - 4) / 2

$BITRATE_N$= f_{CANCLK} / ([1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] x (EPRESDIV + 1))

$BITRATE_F$= f_{CANCLK} / ([1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] x (FPRESDIV + 1))

$CPCB_N$ = f_{SYS} / $BITRATE_N$

$CPCB_F$ = f_{SYS} / $BITRATE_F$

CPCB = $CPCB_N$

- MAXMB is the `CANFD_CFG.MAXMB` value.

- NMBEND is the number of MBs that can be scanned by the arbitration process during the 9 last CAN bits at the end of a frame.
- BITRATEN is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables.
- BITRATEF is the CAN bit rate in bits per second calculated by the data CAN bit time variables.
- CPCBN is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames.
- CPCBF is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames.
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames.
- f_{CANCLK} is the oscillator clock, in Hz.
- f_{SYS} is the peripheral clock, in Hz.
- EPSEG1 is the `CANFD_TIMING.EPSEG1` or `CANFD_CTL1.PSEG1` value.
- EPSEG2 is the `CANFD_TIMING.EPSEG2` or `CANFD_CTL1.PSEG2` value.
- EPROPSEG is the `CANFD_TIMING.EPROPSEG` or `CANFD_CTL1.PROPSEG` value.
- EPRESDIV is the `CANFD_TIMING.EPRESDIV` or `CANFD_CTL1.PRESDIV` value.
- FPSEG1 is the `CANFD_FD_TIMING.FPSEG1` value.
- FPSEG2 is the `CANFD_FD_TIMING.FPSEG2` value.
- FPROPSEG is the `CANFD_FD_TIMING.FPROPSEG` value.
- FPRESDIV is the `CANFD_FD_TIMING.FPRESDIV` value.

NOTE: The f_{SYS} and f_{CANCLK} is same clocked from CDU CLKO4.

The following table give the TXASDLY value calculated for some configuration cases.

- f_{CANCLK} = 40 MHz
- Bit rate in arbitration phase = 1 Mbaud

Table 16-22: TXASDLY Values For Case Example

Number of MBs	TXASDLY Value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	6.67
54	22	5.0
64	21	3.33
96	20	1.6

CANFD Clock Domains and Restrictions

When doing matching and arbitration, the CANFD module needs to scan the entire MB memory during the time slot of one CAN frame, comprising a number of CAN bits. To have sufficient time to do that, there must be a minimum number of peripheral clocks per CAN bit, as specified in the *CAN Bit Minimum Peripheral Clock* table.

Table 16-23: CAN Bit Minimum Peripheral Clock

Number of Mailboxes	**CANFD_CFG.RFEN** Bit Value	Minimum Peripheral Clocks per CAN Bit
16	0	16
32	0	16
64	0	25
96	0	37
128	0	49
16	1	16
32	1	17
64	1	30

For the classical CAN frame format, the minimum number of peripheral clocks per CAN bit specified in the *TXASDLY Values For Case Example* table determines the minimum peripheral clock frequency for a given number of mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, that can be defined by adjusting one or more of the bit timing values contained in either the CANFD_CTL1 register or the CANFD_TIMING register. The time quantum (Tq) is defined in the Protocol Timing section. The minimum number of time quanta per CAN bit must be 8, so the CAN clock frequency is at least 8 times the CAN bit rate.

For the CAN FD frame format, the number of peripheral clocks per CAN bit in nominal bit rate (NumClkNom-Bit) is calculated with the following equation:

NumClkNomBit =(fSYS/fCANCLK) X (PRESDIV + 1) X (PROPSEG + PSEG1 + PSEG2 + 4)

= fSYS/NomBitRate

where PRESDIV, PSEG1 and PSEG2 are CAN bit time values in the CANFD_CTL1 register. Alternatively, the EPRESDIV, EPSEG1 and EPSEG2 values in the CANFD_TIMING register can be used instead. NumClkNomBit is also calculated as a function of the expected nominal bit rate used in the arbitration phase (NomBitRate).

The number of CAN bits in the data phase of a FD frame with the BRS bit set depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) is determined as shown in the *CAN FD Frame Fast CAN Bits* table. The less the number of data bytes, the less the number of fast CAN bits, and less time is available for the CANFD module to scan the whole MB memory during the internal matching and arbitration processes.

Table 16-24: CAN FD Frame Fast CAN Bits

Minimum Data Bytes	DLC Field	Number of Fast Bits
0	0x0	21
1	0x1	29
2	0x2	37
3	0x3	45
4	0x4	53
5	0x5	61
6	0x6	69
7	0x7	77
8	0x8	85
12	0x9	117
16	0xA	149
20	0xB	186
24	0xC	218
32	0xD	282
48	0xE	410
64	0xF	538

The critical part of a CAN FD frame is during the data phase, where the CAN bit rate is faster than in the arbitration phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is calculated so that enough time is available for the CANFD module to scan the MB memory during reception and transmission. The equation below calculates this constraint:

MinNumClkFastBit = ((8.5 x MaxNumOfMb) + 64 - (9 x NumClkNomBit)) / NumOfFastBits

where MaxNumOfMb is the maximum number of available mailboxes defined in the `CANFD_CFG.MAXMB` bit field.

The clock domain crossing circuit between the CHI and PE submodules also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without losing status information through the interface, as shown in the following equation:

$$MinNumClkFastBit = 3 \times (1 + (f_{SYS}/f_{CANCLK}))$$

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above:

MinNumClkFastBit = Maximum (MinNumClkFastBit , MinNumClkFastBit)

Then, the maximum CAN bit rate in the data phase of CAN FD frames (DataBitRateMAX) is calculated as follows:

DataBitRate = f_{CANCLK} / (ROUNDUP ((MinNumClkFastBit x f_{CANCLK}) / f_{SYS}

The peripheral and oscillator clock frequencies, the maximum number of mailboxes, and the expected nominal bit rate affect the maximum data bit rate attainable by the CANFD module in CAN FD mode. In addition, the data bit rate depends on the minimum payload size of FD frames used in each application.

To illustrate how the CAN FD bit rate is affected by the configuration of CANFD variables, an consider an application example with the peripheral and oscillator clock frequencies set to 50 MHz.

1. Considering the nominal bit rate as 1 Mbps, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as follows: NumClkNomBit = (50 x 10^6)/(1 x 10^6) = 50
2. The number of fast CAN bits (NumOfFastBits) is determined as shown in the *CAN FD Frames Maximum CAN Bit Rate In Data Phase* table. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the data phase.
3. Assuming the maximum number of mailboxes is 96, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated as follows:

   MinNumClkFastBit = ((8.5 X 96) + 64 - (9 X 50)) / 85 = 5.06

   MinNumClkFastBit = 3 x (1 + 50/50) = 6.0

   MinNumClkFastBit = Maximum (5.06 , 6.0) = 6.0

As demonstrated in this example, even though the oscillator clock frequency (50 MHz) is adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific CANFD configuration limits this rate to 6.667 Mbps. This limitation is due to the low peripheral clock frequency that imposes the $MinNumClkFastBit_B$ limit.

The *CAN FD Frames Maximum CAN Bit Rate In Data Phase* table shows the maximum data rate for CAN FD according to the clock frequencies, payload size, and number of available mailboxes. As shown in the *CAN FD Frames Maximum CAN Bit Rate In Data Phase* table, for some cases, if the number of available mailboxes is reduced, the CANFD module achieves a data rate up to 8 Mbps.

Table 16-25: CAN FD Frames Maximum CAN Bit Rate In Data Phase

Peripheral Clock Frequency (MHz)	Payload Size	Number of Available Mailboxes	Maximum Data Rate (Mbps)
40	8	94	6.667
40	8	114	5.0
40	12	117	6.667
40	12	128	5.714
50	12 to 64	128	6.667
60	8	128	8.0
60	12	128	8.0

Table 16-25: CAN FD Frames Maximum CAN Bit Rate In Data Phase (Continued)

Peripheral Clock Frequency (MHz)	Payload Size	Number of Available Mailboxes	Maximum Data Rate (Mbps)
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

CANFD Operating Modes

The CANFD module has functional modes and low-power modes.

CAUTION: The CANFD does not support the permanent dominant failure on the CAN bus line. If a low-power request or freeze mode request is done during a permanent dominant, the corresponding acknowledge can never be asserted.

The CANDFD module has the following main functional modes:

- Normal Mode
- Freeze Mode
- Loop-Back Mode
- Listen-Only Mode
- CAN FD Active Mode

For low-power operation, the CANDFD module has the following modes:

- Module Disable Mode
- Doze Mode
- Stop Mode
- Pretended Network Mode

NOTE: The CANFD does not support the permanent dominant failure on the CAN bus line. If there is a low-power request or freeze mode request during a permanent dominant, the corresponding acknowledge will not be asserted.

Normal Mode

In normal mode, the CANFD module operates by receiving and transmitting message frames. In this mode, errors are handled normally, and all CAN protocol functions are enabled.

Freeze Mode

In freeze mode, there is no transmission or reception of frames and synchronicity to the CAN bus is lost. Freeze mode is enabled by setting the CANFD_CFG.FRZ bit.

Freeze mode is requested either by the processor by enabling the CANFD_CFG.HALT bit or when it is put into debug mode. In both cases, when the CANFD_CFG.FRZ bit is enabled, the CANFD module must not be in a low-power mode. Freeze mode is also requested through the automatic assertion of both the CANFD_CFG.HALT and CANFD_CFG.FRZACK bits when the CANFD_MEC.NCERRFRZEN bit is set and a non-correctable error is detected in a memory read access performed by CANFD internal processes. The acknowledgement is obtained through the assertion by the CANFD_CFG.FRZACK bit. The CANFD module is only in freeze mode when both the request and acknowledge conditions are satisfied.

In response to a freeze mode request, the CANFD module performs the following operations:

- Waits to be in either the intermission, passive error, bus off, or idle state.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent going to freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive.
- Stops the prescaler, thus halting all CAN protocol activities.
- Grants write access to the CANFD_ECR register, which is read-only in other modes.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.FRZACK bits.

After requesting freeze mode, wait for the CANFD_CFG.FRZACK bit to be set before executing any other action, or the CANFD module will not operate properly.

To exit freeze mode, disable the CANFD_CFG.FRZ bit, remove the processor from debug mode, or disable the CANFD_CFG.HALT bit.

The CANFD_CFG.FRZACK bit is disabled after the PE recognizes the negation of the freeze request. Once out of freeze mode, the CANFD module tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

Loop-Back Mode

The CANFD module enters loop-back mode when the CANFD_CTL1.LBEN is set. In loop-back mode, the CANFD module performs an internal loop back for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). The CANFD module behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In loop-back mode, the CANFD ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

Listen-Only Mode

The CANFD module enters listen-only mode when the CANFD_CTL1.LOMEN bit is enabled. In listen-only mode, transmission is disabled, all error counters are frozen, and the CANFD module operates in a CAN error passive mode. The module only receives messages acknowledged by another CAN station. If the CANFD detects a message that has not been acknowledged, it flags a BIT0 error (without changing the receive error counter), as if it were trying to acknowledge the message.

CAN FD Active Mode

In CAN FD active mode, the CANFD module can transmit and receive all messages formatted according to the CAN FD protocol and CAN 2.0 (protocol 2.0) in an interleaved fashion. The CANFD module is put into CAN FD active mode by setting the CANFD_CFG.FDEN bit when in freeze mode.

Module Disable Mode

The module disable mode is a low-power mode normally used to temporarily disable a complete CANFD block, with no power consumption. To enter module disable mode, enable the CANFD_CFG.DIS bit. Once the processor is in module disable mode, the CANFD_CFG.LPMACK bit is set.

Do not use module disable mode under pretended networking mode. Disable the CANFD_CFG.DIS bit and wait for CANFD_CFG.LPMACK bit to clear before enabling the CANFD_CFG.PNETEN bit.

If the CNAFD module is disabled during transmission or reception, it will perform the following operations:

- Waits to be in either the idle or bus off state, or else waits for the third bit of intermission and then checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not considered.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.LPMACK bits.

The bus interface unit continues to operate, enabling the processor to access memory mapped registers, except the CANFD_RX_FIFO_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers. The CANFD_RX_FIFO, the message buffers, the Rx individual mask registers, and the reserved words within RAM are not accessible when the CANFD module is in the module disable mode.

To exit module disable mode, disable the CANFD_CFG.DIS bit. If the CANFD module is disabled during freeze mode, it sends a requests to disable the clocks to the PE and CHI submodules, sets the CANFD_CFG.LPMACK bit, and clears the CANFD_CFG.FRZACK bit.

Doze Mode

The doze mode is a system low-power mode in which the processor bus is kept alive, and a global doze mode request is sent to all peripherals asking them to enter low-power mode. Before sending a global doze mode request, set the CANFD_CFG.DOZEN bit. The global doze request is set with the MISCREG_CAN_SYSCTL.CAN1_IPG_DOZE or MISCREG_CAN_SYSCTL.CAN0_IPG_DOZE bit. Once the processor requests the doze mode, the CANFD_CFG.LPMACK bit is set.

To exit doze mode, disable the CANFD_CFG.DOZEN bit or clear the IPG_DOZE bits in the MISCREG_CAN_SYSCTL register. The CANFD module also exits doze mode when activity is detected on the CAN bus and the self-wake up mechanism is enabled using the CANFD_CFG.SLFWAKEN bit.

If the doze mode is triggered during freeze mode, the CANFD module requests to disable the clocks to the PE and CHI submodules, sets the CANFD_CFG.LPMACK bit, and clears the CANFD_CFG.FRZACK bit.

If doze mode is triggered during transmission or reception, the CANFD module performs the following operations:

- Waits to be in either idle or bus off state, or else waits for the third bit of intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not considered.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.LPMACK bits.

The bus interface unit continues to operate, enabling the processor to access memory mapped registers, except the CANFD_RX_FIFO_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers. The CANFD_RX_FIFO, the message buffers, the Rx individual mask registers, and the reserved words within RAM are not accessible when the CANFD module is in the doze mode.

Exiting Doze mode is done in one of the following ways:

- Disabling the doze mode request in the MISCREG_CAN_SYSCTL register.
- Disabling the CANFD_CFG.DOZEN bit.
- When the CANFD module detects activity on the CAN bus and the self-wake up mechanism is enabled using the CANFD_CFG.SLFWAKEN bit.

When the self-wake up mechanism is enabled at the time that the CANFD module entered doze mode, then upon detection of a recessive-to-dominant transition on the CAN bus, the CANFD module clears the CANFD_CFG.DOZEN bit, requests to resume clocks, and disables the CANFD_CFG.LPMACK bit after the CAN PE recognizes the doze mode is disabled.

The CANFD module also sets the CANFD_ESR1.WAKINT bit, so if the CANFD_CFG.WAKMSK bit is enabled, it generates a wake up interrupt to the processor. The CANFD module then waits for 11 consecutive recessive

bits to synchronize to the CAN bus. Consequently, the CANFD module does not receive the frame that woke it up. The *Wake Up From Doze Mode* table details the effect of the CANFD_CFG.SLFWAKEN and the CANFD_CFG.WAKMSK bits upon waking from the doze mode.

Table 16-26: Wake Up From Doze Mode

SLFWAKEN	WAKINT	WAKMSK	CANFD Clocks Enabled	Wakeup Interrupt Generated
0	N/A	N/A	No	No
0	N/A	N/A	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	Yes	No
1	1	1	Yes	Yes

Applying a low-pass filter function to the Rx CAN input line while in doze mode will modify sensitivity to CAN bus activity as described in the CANFD_CFG.WSFLTREN bit description. This feature also protects the CANFD module from waking up due to short glitches on the CAN bus lines that can result from electromagnetic interference in noisy environments.

Stop Mode

The stop mode is a system low-power mode in which all clocks are stopped for maximum power savings. A global stop request is set with the MISCREG_CAN_SYSCTL.CAN0_IPG_STOP or MISCREG_CAN_SYSCTL.CAN1_IPG_STOP bit. Once the processor is in the stop mode, the CANFD_CFG.LPMACK bit is set.

If the CANFD module receives a global stop mode request during freeze mode, it sets the CANFD_CFG.LPMACK bit, clears the CANFD_CFG.FRZACK bit, and then sends the stop acknowledge signal to the processor to shut down the clocks globally.

If stop mode is requested during transmission or reception, the CANFD module performs the following operations:

- Waits to be in either the idle or bus off state, or else waits for the third bit of intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not considered.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.LPMACK bits.
- Sends a stop acknowledge signal to the processor to shut down the clocks globally.

Stop mode is exited when the processor resumes the clocks and removes the stop mode request, or when activity is detected on the CAN bus and the self-wake up mechanism is enabled using the CANFD_CFG.SLFWAKEN bit.

When the self-wake up mechanism is enabled at the time that the CANFD module entered stop mode, then upon detection of a recessive-to-dominant transition on the CAN bus, the CANFD module sets the CANFD_ESR1.WAKINT bit, so if the CANFD_CFG.WAKMSK bit is enabled, it generates a wake up interrupt to the processor. Upon receiving the interrupt, the processor resumes the clocks and removes the stop mode request. The CANFD module then waits for 11 consecutive recessive bits to synchronize to the CAN bus. Consequently, the CANFD module does not receive the frame that woke it up. The *Wakeup From Stop Mode* table details the effect of the CANFD_CFG.SLFWAKEN and the CANFD_CFG.WAKMSK bits upon wakeup from the stop mode. Note that wakeup from the stop mode only works when both bits are asserted.

Table 16-27: Wakeup From Stop Mode

SLFWAKEN	WAKINT	WAKMSK	Chip Clocks Enabled	Wakeup Interrupt Generated
0	N/A	N/A	No	No
0	N/A	N/A	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	No	No
1	1	1	Yes	Yes

Applying a low-pass filter function to the Rx CAN input line while in stop mode will modify sensitivity to CAN bus activity as described in the CANFD_CFG.WSFLTREN bit description. This feature also protects the CANFD module from waking up due to short glitches on the CAN bus lines that can result from electromagnetic interference in noisy environments.

Pretended Network Mode

Pretended networking is a special low-power mode used to receive wake up messages with low power consumption. Pretended networking mode can be selected to operate together with doze mode. Before entering low-power mode, the CANFD_CFG.PNETEN bit must be asserted. Once in low-power mode, the CHI submodule clocks are shut down and the PE submodule is kept clocked, so that the receive process is still active to filter incoming messages as defined by the configuration.

Upon detecting a wakeup event, a wake up interrupt is issued to the system. When the CANFD_CFG.PNETEN bit is enabled, the processor disables the self-wake up feature by disabling the CANFD_CFG.SLFWAKEN bit. Wake up from matched message or time-out in pretended networking modes happens only when pretended networking mode is used along with doze mode. Wakeup is not supported in stop mode.

- Waits to be in the idle state, or else waits for the third bit of intermission, and then checks it to be recessive.
- Sets the CANFD_CFG.LPMACK bit.
- Requests the shutdown of the CHI submodule clock, while keeping the PE submodule clock active.

The CANFD module exits pretended networking mode in the following ways:

- The processor removes the doze mode request.
- The processor disables the CANFD_CFG.DOZEN bit.

The CANFD will wait until bus idle or the third bit of the intermission state to disable the CANFD_CFG.LPMACK bit.

The above exit methods are triggered either by the CANFD module detecting a wake up event and issuing the respective interrupt, or by the processor itself upon being woken up. The CANFD will wait until the bus idle state or until the third bit of the intermission state to disable the CANFD_CFG.LPMACK bit and resume normal mode. This procedure ensures that the CANFD module is synchronized to the CAN bus after exiting pretended networking mode. The processor must wait for the CANFD_CFG.LPMACK bit to be disabled before performing any access to the CANFD.

CANFD Event Control

The following section describe how the CANFD module generates and controls events.

Interrupts

The CANFD module has many interrupt sources: interrupts due to MBs and interrupts due to the OR'd interrupts from MBs, bus off, bus off done, error, error fast (errors detected in the data phase of CAN FD format messages with the CANFD_FD_CTL.BRSEN bit set), wake up, wake up match, wake up timeout, Tx warning, and Rx warning states.

Each one of the MBs can be an interrupt source if the corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the CANFD_IFLG1 and CANFD_IFLG2 registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the processor writes it to 1 (unless another interrupt is generated at the same time).

NOTE: The processor must only clear the bit causing the current interrupt, so bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause the accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (CANFD_CFG.RFEN = 1) and DMA is disabled (CANFD_CFG.DMAEN = 0), the interrupts corresponding to MBs 0 to 7 have different meanings:

- Bit 7 of the CANFD_IFLG1 register becomes the "FIFO Overflow" flag.
- Bit 6 of the CANFD_IFLG1 register becomes the "FIFO Warning" flag.
- Bit 5 of the CANFD_IFLG1 register becomes the "Frames Available in FIFO" flag.
- Bit 4-0 of the CANFD_IFLG1 register are unused.

If both Rx FIFO and DMA are enabled (CANFD_CFG.RFEN =1 and CANFD_CFG.DMAEN = 1), the CANFD does not generate any FIFO interrupt. Bit 5 of the CANFD_IFLG1 register still indicates “Frames Available in FIFO” and generates a DMA request. Bits 7, 6, and 4–0 are unused.

NOTE: The Rx FIFO cannot be enabled when the CAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR’d together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the processor must read the CANFD_IFLG1 and CANFD_IFLG2 registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for bus off, bus off done, error, error fast, wake up, Tx warning ,and Rx warning generate interrupts like the MB interrupt sources, and are read from the CANFD_ESR1 register. The bus off, error, Tx warning, and Rx warning interrupt mask bits are located in the CANFD_CTL1 register; the wake up interrupt mask bit is located in CANFD_CFG register.

The interrupt sources for pretended networking (wake up by match flag and wake up by timeout flag) are read in the CANFD_WUM register and the respective interrupts masks bits are located in CANFD_PN_CTL1 register.

Bus Interface

The processor accesses to CANFD registers are subject to the following rules:

- Read and write accesses to reserved address space result in access errors.
- Write accesses to positions with bits that are all currently read-only results in access errors.

  If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to the register and bit descriptions for details.
- Read and write accesses to unimplemented address space results in access errors.
- Read and write accessed to RAM located positions during low-power modes results in access errors.
- If CANFD_CFG.MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Reserved words within RAM cannot be used.

  For example, suppose the CANFD RAM can support up to 16 MBs, the CANFD_CTL2.RFFNUM bit field is 0x0, and the CANFD_CFG.MAXMB bit field is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one individual mask and the available memory in the mask registers space is from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF, there are reserved words for internal use which cannot be used as general purpose RAM. Generally, free memory space for general-purpose usage depends only on the CANFD_CFG.MAXMB bit field.

Detection and Correction of Memory Errors

The CANFD module supports detection and correction of errors in memory read accesses. Each byte of CANFD memory is associated with five parity bits, which ensures a Hamming distance of four. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors), and errors in 2 bits can be detected but not corrected (non-correctable errors). Errors in more than 2 bits may not be detected. In case of non-correctable errors, the corrupted data is not changed by the error correction logic. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

The CANFD module detects a non-correctable error in the event either an all-zeros or an all-ones read occurs and updates the CANFD_ERR_RSYN register. The CANFD_ERR_RSYN register holds the syndrome detected in a memory read with an error. It reports the bytes that are read in the 32-bit read transaction.

The *Syndrome Definition* table shows the SYNDn field in the CANFD_ERR_RSYN register, indicates the type of error, and the bit in byte (n) affected by the error. Each BEn field in the CANFD_ERR_RSYN register indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated and reported as described in the Error Indication and Error Reporting sections.

Table 16-28: Syndrome Definition

SYNDn	Type	Bit Affected
0x00	N/A	None (no error)
0x01	Code	0
0x02	Code	1
0x04	Code	2
0x07	Data	5
0x08	Code	3
0x0E	Data	7
0x10	Code	4
0x13	Data	2
0x15	Data	6
0x16	Data	1
0x19	Data	3
0x1A	Data	4
0x1C	Data	0
0x06	N/A	All-zeros non-correctable error
0x1F	N/A	All-ones non-correctable error
All others	N/A	Non-correctable error

Memory errors are indicated to the host through the CANFD_ERR_STAT register and bus transfer errors, and reported through the CANFD_ERR_RADDR,CANFD_ERR_RDAT, and CANFD_ERR_RSYN registers.

The error detection and correction mechanism is activated by the CANFD_MEC.ECCDIS bit. When error detection and correction is disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated to the data.

NOTE: All CANFD memory must be initialized before any operation for the parity bits in memory to properly update. The CANFD_CTL2.WRMFRZEN bit grants write access to all memory positions that require initialization, ranging from 0x080 to 0xADF (and from 0xF28 to 0xFFF when the CAN FD feature is enabled). The CANFD_RX_MB_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers also need to be initialized. The CANFD_CFG.RFEN bit must not be set during memory initialization. The *Identifier Acceptance Filter Fields for Global Bits* table shows the configurations for the CANFD_RX_FIFO_GMSK register.

Table 16-29: Identifier Acceptance Filter Fields for Global Mask Bits

Rx FIFO ID Filter Table Element Format **CANFD_CFG.IDAM**	**Identifier Acceptance Filter Fields**					
RTR	IDE	RXIDA	RXIDB	RXIDC	Reserved	VALUE[0]
A	VALUE[31]	VALUE[30]	VALUE[29:1]	N/A	N/A	
B	VALUE[31], VALUE[15]	VALUE[30], VALUE[14]	N/A	VALUE[29:16], VALUE[13:0]	N/A	N/A
C	N/A	N/A	N/A	N/A	VALUE[31:24], VALUE[23:16], VALUE[15:8], VALUE[7:0]	N/A

To avoid accidentally changing the critical error correction configuration, use the following protocol to enable the CANFD_MEC register update:

1. By default, the CANFD_CTL2.ECRWREN bit is zero and the CANFD_MEC.ECRWRDIS bit is one.
2. Set the CANFD_CTL2.ECRWREN bit.
3. Clear the CANFD_MEC.ECRWRDIS bit.
4. All writes to the CANFD_MEC register must keep the CANFD_MEC.ECRWRDIS bit cleared.
5. After configuration is done, lock the CANFD_MEC register by either setting the CANFD_MEC.ECRWRDIS bit or clearing the CANFD_CTL2.ECRWREN bit.

Sources of Memory Access

The CANFD memory can be accessed by two major sources (or requesters):

- Host (processor): The largest word accessed is 32 bits wide.
- CANFD internals processes (Rx matching, Tx arbitration, move-in on reception, move-out on transmission): The largest word accessed is 64 bits wide.

The way that non-correctable errors are indicated and reported depends on the source of access.

Error Indication

Memory errors are indicated by the CANFD_ERR_STAT.HNCEI, CANFD_ERR_STAT.INCEI, and CANFD_ERR_STAT.CEI flag bits. Non-correctable errors detected in memory reads requested by the host are indicated separately from the errors detected in requests by CANFD internal processes. The CANFD makes no distinction of the source of the access when correctable errors are detected. There are three independent flags for these three cases, and each flag raises an interrupt unless it is masked in the CANFD_MEC register. If both non-correctable and correctable errors are found in different bytes of the same read operation, both flags are set.

A non-correctable error detected in host access is also indicated as a bus transfer error. A bus wait request asserts to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the CANFD_ERR_STAT.HNCEI bit is not masked, the same non-correctable error will raise a bus transfer error and an interrupt request.

Each indication flag has one overrun flag in the CANFD_ERR_STAT register. The overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. The recommended handling sequence for error indication is as follows:

1. Get the error report information from the report registers CANFD_ERR_RADDR, CANFD_ERR_RDAT, and CANFD_ERR_RSYN.
2. Use the error report information to take proper measures in the application.
3. Clear the CANFD_ERR_STAT.HNCEI, CANFD_ERR_STAT.INCEI, CANFD_ERR_STAT.CEI flag bits.
4. If the overrun flag is active:
    a. Alert that application that at least one error could not be handled.
    b. Clear the overrun flag bit.

The CANFD internal processes can access memory in transactions larger than 32 bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words, the interrupt and overrun flags are set simultaneously.

Error Reporting

The report registers CANFD_ERR_RADDR, CANFD_ERR_RDAT, and CANFD_ERR_RSYN provide detailed information about the address read, raw data, and syndrome read with error and indicated by the flags.

The address, data, and syndrome registers are updated simultaneously along with the error flags, according to these rules:

1. If any of the two non-correctable error flags is currently set, the report registers are not updated (the previous non-correctable error reporting is preserved).
2. Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error, or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected.

Reporting of errors detected in accesses larger than 32 bits follows the rules described in the CANFD_ERR_RADDR register.

The address reported in the CANFD_ERR_RADDR register and defined in the CANFD_ERR_IADDR register are not the same as those listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in the Error Injection section.

Addresses reported when reading memory portions organized as FIFOs, such as the CANFD_RX_FIFO register, refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address.

To ensure coherence of the error report registers, turn off the report update by setting the CANFD_MEC.RERDIS bit before reading the report registers.

Response to Errors

Correctable errors have no effect on CANFD operation because affected data is corrected before use by the host or CANFD internal processes.

For host-initiated reads, a non-correctable error affects the host, but does not affect CANFD operation.

Non-correctable errors detected on memory reads requested by the CANFD internal processes result in incorrect operation depending on the state of the CANFD_MEC.NCERRFRZEN bit, as follows:

- During reception (either matching or move-in processes), when a non-correctable error occurs, an incorrect destination is selected to store the incoming frame, a corrupted frame is stored in the correct destination, or both. If the CANFD_MEC.NCERRFRZEN is set, the CANFD module stops operation automatically and enters freeze mode to prevent corrupted data from being treated as valid by CANFD internal processes. If the CANFD_MEC.NCERRFRZEN bit is cleared, the CANFD continues working and a corrupted frame is received.
- During the arbitration process, when a non-correctable error occurs, either a non-highest priority Tx message buffer is mistakenly selected for transmission or the data is corrupted. If the CANFD_MEC.NCERRFRZEN bit is set, the CANFD module stops operation automatically and enters freeze mode before starting the move-out. If the CANFD_MEC.NCERRFRZEN bit is cleared, the CANFD proceeds to move-out with a corrupted frame that will transmit on the CAN bus.

- During the move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the arbitration to the Tx SMB for transmission. If the CANFD_MEC.NCERRFRZEN bit is set, the CANFD module stops operation automatically and enters freeze mode before starting the transmission. If the CANFD_MEC.NCERRFRZEN bit is cleared, the corrupted frame is transferred from the Tx SMB to the PE submodule and transmits on the CAN bus.
- A non-correctable error can be detected beyond the move-out process, when Tx data is read from the Tx SMB (buffer located in RAM) to be transferred to the PE submodule for transmission. In this case, a frame with corrupted ID and/or data is transmitted on the CAN bus. To prevent the frame from being successfully received by the external nodes, the CANFD module inverts all bits in the CRC field (CRC sequence plus CRC Delimiter) and transmits an error flag just after CRC delimiter because of a self-detecting bit1 error and a form error due to the CRC field inversion. When the CANFD_MEC.NCERRFRZEN bit is set, the CANFD module stops operation automatically and enters freeze mode just after the error frame. If the CANFD_MEC.NCERRFRZEN bit is negated, the CANFD module attempts to re-transmit the same frame, when no other higher priority Tx MB is subsequently configured for transmission. In the event the non-correctable error persists, the CANFD module eventually reaches the bus off state because of consecutive error detections. The CANFD_ECR.TXERRCNT field is updated every time the CANFD module inverts the CRC field causing errors as described above.

When the CANFD_MEC.NCERRFRZEN bit is set and the CANFD module enters freeze mode, only the host processor can cause then CANFD to exit freeze mode and resume normal mode. Assertion of the CANFD_MEC.NCERRFRZEN bit is the only way to prevent corrupted frames from transmitting on the CAN bus up to the move-out internal process.

The error report registers provide information to the application for customized handling of these situations.

Error Injection

The error injection registers CANFD_ERR_IADDR, CANFD_ERR_IDP, and CANFD_ERR_IPP are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers.

The *Error Injection Address Mapping* table shows the relationship between the error injection addresses and the ones in the module memory map. Use this table to convert from the memory map address to the correct location in the physical CANFD RAM.

NOTE: Where pairs of values are provided in the table, the first is the address when the CANFD_CFG.FDEN bit is disabled and the second is when the CANFD_CFG.FDEN bit is enabled.

Table 16-30: Error Injection Address Mapping

RAM Contents	Injection Address	Memory Map
CANFD registers	Not mapped	N/A
MBs	0x0000	0x0080
Reserved	N/A	0x0480
RXIMRs	0x0400	0x0880

Table 16-30: Error Injection Address Mapping (Continued)

RAM Contents	Injection Address	Memory Map
Reserved	N/A	0x0980
RXFIR_0	0x0500	0x0A80
RXFIR_1	0x0504	0x0A84
RXFIR_2	0x0508	0x0A88
RXFIR_3	0x050C	0x0A8C
RXFIR_4	0x0510	0x0A90
RXFIR_5	0x0514	0x0A94
Reserved	N/A	0x0A98
CANFD_RX_MB_GMSK	0x0520	0x0010
CANFD_RX_FIFO_GMSK	0x0524	0x0048
CANFD_RX_14_MSK	0x0528	0x0014
CANFD_RX_15_MSK	0x052C	0x0018
Tx_SMB	0x0530	0x0AB0 / 0x0F28
Rx_SMB0	0x0540 / 0x0578	0x0AC0 / 0x0F70
Rx_SMB1	0x0550 / 0x05C0	0x0AD0 / 0x0FB8
ECC registers	Not mapped	0x0AE0
Pretended networking registers	Not mapped	0x0B00
CAN FD registers	Not mapped	0x0C00
Reserved	N/A	0x0C0C

The injection is done by flipping the data and parity bits corresponding to the bits set to 1 in the CANFD_ERR_IDP and CANFD_ERR_IPP registers. Injection can be selected specifically for memory accesses requested by host or by CANFD internal processes.

In case of accesses larger than 32 bits, the CANFD_MEC.EXTERRIEN bit extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE: It is very unlikely, but error injection may correct a bit with an error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the CANFD_MEC.HAERRIEN and CANFD_MEC.IAERRIEN bits must be cleared while configuring the memory injection.

CANFD Programming Model

Following sections provide some guidelines for CANFD programming.

Initialization

The CANFD module is reset in one of the following ways:

- A chip level hard reset that resets all memory mapped registers asynchronously.
- Setting the CANFD_CFG.SOFTRST bit , which resets some of the memory mapped registers synchronously. The *Register Reset States* table shows the registers affected by a soft reset.
- Chip level soft reset, which has the same effect as setting the CANFD_CFG.SOFTRST bit.

A soft reset is synchronous and must follow an internal request and acknowledge procedure across clock domains and takes time to fully propagate the effects. The CANFD_CFG.SOFTRST bit remains set while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset cannot be applied while clocks are shut down in a low-power mode. Exit the low-power mode and resume the clocks before applying a soft reset.

Table 16-31: Register Reset States

Register	Affected by Hard Reset	Affected by Soft Reset
CANFD_TIMING	Yes	No
CANFD_CRC	Yes	Yes
CANFD_CTL1	Yes	No
CANFD_PN_CTL1	Yes	Yes
CANFD_CTL2	Yes	No
CANFD_PN_CTL2	Yes	Yes
CANFD_ECR	Yes	Yes
CANFD_ERR_IADDR	Yes	Yes
CANFD_ERR_IDP	Yes	Yes
CANFD_ERR_IPP	Yes	Yes
CANFD_ERR_STAT	Yes	Yes
CANFD_ESR1	Yes	Yes
CANFD_ESR2	Yes	Yes
CANFD_FD_TIMING	Yes	No
CANFD_FD_CRC	Yes	Yes
CANFD_FD_CTL	Yes	No
CANFD_FLTR_DLC	Yes	Yes
CANFD_FLTR_ID1	Yes	Yes
CANFD_FLTR_ID2_IDMSK	Yes	Yes

Table 16-31: Register Reset States (Continued)

Register	Affected by Hard Reset	Affected by Soft Reset
CANFD_IFLG1	Yes	Yes
CANFD_IFLG2	Yes	Yes
CANFD_IMSK1	Yes	Yes
CANFD_IMSK2	Yes	Yes
CANFD_CFG	Yes	Yes
CANFD_MEC	Yes	Yes
CANFD_FLTR_DATA1_HI	Yes	Yes
CANFD_FLTR_DATA1_LO	Yes	Yes
CANFD_FLTR_DATA2_DMSK_HI	Yes	Yes
CANFD_FLTR_DATA2_DMSK_LO	Yes	Yes
CANFD_ERR_RADDR	Yes	Yes
CANFD_ERR_RDAT	Yes	Yes
CANFD_ERR_RSYN	Yes	Yes
CANFD_RX_14_MSK	No	No
CANFD_RX_15_MSK	No	No
CANFD_RX_FIFO_GMSK	No	No
CANFD_RX_FIFO	No	No
CANFD_RX_IMSK[n]	No	No
CANFD_RX_MB_GMSK	No	No
CANFD_TMR	Yes	Yes
CANFD_WUM	Yes	Yes

Note that the MBs and the Rx individual mask registers are not affected by reset, so they are not automatically initialized.

CANFD Initialization Sequence

For any configuration change or initialization, the CANFD must be is put into freeze mode. The following steps are a generic initialization sequence applicable to the CANFD module:

1. Initialize the Module Configuration Register (CANFD_CFG):
    - Enable the individual filtering per MB and reception queue features by setting the CANFD_CFG.IRMQEN bit.
    - Enable the warning interrupts by setting the CANFD_CFG.WRNEN bit.
    - Optionally disable frame self reception by setting the CANFD_CFG.SRXDIS bit.

- Enable the Rx FIFO by setting the CANFD_CFG.RFEN bit.
- If Rx FIFO is enabled and DMA is required, set the CANFD_CFG.DMAEN bit.
- If Pretended Networking mode is required, set the CANFD_CFG.PNETEN bit.
- Enable the abort mechanism by setting the CANFD_CFG.ABORTEN bit.
- Enable the local priority feature by setting the CANFD_CFG.LPRIOEN bit.

2. Initialize the Control 1 Register (CANFD_CTL1) and optionally the CAN Bit Timing Register (CANFD_TIMING). Also, initialize the CAN FD CAN Bit Timing Register (CANFD_FD_TIMING):
   - Determine the bit timing parameters: CANFD_CTL1.PROPSEG, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2, CANFD_CTL1.RJW.
   - Optionally determine the extended bit timing parameters: CANFD_TIMING.EPROPSEG, CANFD_TIMING.EPSEG1, CANFD_TIMING.EPSEG2, CANFD_TIMING.ERJW.
   - Determine the CAN FD bit timing parameters: CANFD_FD_TIMING.FPROPSEG, CANFD_FD_TIMING.FPSEG1, CANFD_FD_TIMING.FPSEG2, CANFD_FD_TIMING.FRJW.
   - Determine the bit rate by programming the CANFD_CTL1.PRESDIV field and optionally the CANFD_TIMING.EPRESDIV field.
   - Determine the CAN FD bit rate by programming the CANFD_FD_TIMING.FPRESDIV field. Determine the internal arbitration mode (CANFD_CTL1.LBUF bit).
3. Initialize the Message Buffers:
   - Initialize the C/S word of all MBs.
   - If the Rx FIFO is enabled, initialize the ID filter table.
   - Initialize other entries in each MB.
4. Initialize the individual masks in the CANFD_IMSK1 and CANFD_IMSK2 registers.
5. Set required interrupt mask bits in the CANFD_IMSK1 and CANFD_IMSK2 registers (for all MB interrupts), in the CANFD_CFG register for the wake up interrupt, and in the CANFD_CTL1 and CANFD_CTL2 registers for the bus off and error interrupts.
6. If pretended networking mode is enabled, configure the necessary registers for selective wake up.
7. Disable the CANFD_CFG.HALT bit.

After the last step listed, the CANFD module attempts to synchronize to the CAN bus.

Pretended Network - Doze Mode

Complete the following sequence to enter the pretended networking mode in doze mode:

1. Set the CANFD_CFG.PNETEN bit to enable the controller to enter the pretended networking mode, when one of the low power modes is requested.
2. Clear the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.
3. Request the CANFD to enter Doze mode.
4. Wait for the CANFD to set the CANFD_CFG.LPMACK bit.
5. Clear the CANFD_WUM.WTOF and/or CANFD_WUM.WUMF bits (if either bit is set).
6. Set the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.

Pretended Network - Stop Mode

Complete the following sequence to enter the pretended networking mode in stop mode:

1. Set the CANFD_CFG.PNETEN bit to enable the controller to enter pretended networking mode, when one of the low power modes is requested.
2. Clear the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.
3. Request the CANFD to enter Stop mode. Keep CAN0_IPG_STOP_CLK_EXT/ CAN1_IPG_STOP_CLK_EXT bits set to keep the clock to register block running.
4. Wait for the CANFD to set the CANFD_CFG.LPMACK bit.
5. Clear the CANFD_WUM.WTOF and/or CANFD_WUM.WUMF bits (if either bit is set).
6. Set the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.
7. Disable CANFD register clocks by clearing bits CAN0_IPG_STOP_CLK_EXT/ CAN1_IPG_STOP_CLK_EXT for CANFD0/CANFD1 respectively.

ADSP-2184x CANFD Register Descriptions

Controller Access Network with Flexible Data Rate (CANFD) (CANFD) contains the following registers.

Table 16-32: ADSP-2184x CANFD Register List

Name	Description
CANFD_TIMING	Can Bit Timing Register
CANFD_CRC	CRC Register
CANFD_CTL1	Control 1 Register
CANFD_PN_CTL1	Pretended Networking Control1 Register
CANFD_CTL2	Control 2 Register
CANFD_PN_CTL2	Pretended Networking Control2 Register

Table 16-32: ADSP-2184x CANFD Register List (Continued)

Name	Description
CANFD_ECR	Error Counter Register
CANFD_ERR_IADDR	Error Injection Address Register
CANFD_ERR_IDP	Error Injection Data Pattern Register
CANFD_ERR_IPP	Error Injection Parity Pattern Register
CANFD_ERR_STAT	Error Status Register
CANFD_ESR1	Error and Status 1 Register
CANFD_ESR2	Error and Status 2 Register
CANFD_FD_TIMING	CANFD Bit Timing Register
CANFD_FD_CRC	CANFD CRC Register
CANFD_FD_CTL	CANFD Control Register
CANFD_FLTR_DLC	Pretended Networking DLC Filter Register
CANFD_FLTR_ID1	Pretended Networking ID Filter1 Register
CANFD_FLTR_ID2_IDMSK	Pretended Networking ID Filter2 / IDMask Register
CANFD_IFLG1	Mailbox Interrupt Flag 1 Register
CANFD_IFLG2	Mailbox Interrupt Flag 2 Register
CANFD_IMSK1	Mailbox Interrupt Mask 1 Register
CANFD_IMSK2	Mailbox Interrupt Mask 2 Register
CANFD_CFG	Module Configuration Register
CANFD_MEC	Memory Error Control Register
CANFD_FLTR_DATA1_HI	Pretended Networking Payload Low Filter2 Register
CANFD_FLTR_DATA1_LO	Pretended Networking Payload Low Filter1 Register
CANFD_FLTR_DATA2_DMSK_HI	Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register
CANFD_FLTR_DATA2_DMSK_LO	Pretended Networking Payload Low Filter2 / Payload Low Mask Register
CANFD_ERR_RADDR	Error Report Address Register
CANFD_ERR_RDAT	Error Report Data Register
CANFD_ERR_RSYN	Error Report Syndrome Register
CANFD_RX_14_MSK	Receive Mailbox14 Mask Register
CANFD_RX_15_MSK	Receive Mailbox15 Mask Register
CANFD_RX_FIFO_GMSK	Receive FIFO Global Mask Register
CANFD_RX_FIFO	Receive FIFO Information Register
CANFD_RX_IMSK[n]	Receive Individual Mask Register

Table 16-32: ADSP-2184x CANFD Register List (Continued)

Name	Description
CANFD_RX_MB_GMSK	Receive Mailbox Global Mask Register
CANFD_TMR	Free Running Timer Register
CANFD_WMB0_CS	Pretended Networking Wake Up Message Buffer 0 Control Status Register
CANFD_WMB0_D03	Pretended Networking Wake Up Message Buffer 0 Data 0-3 Register
CANFD_WMB0_D47	Wakeup Message Buffer Data 4-7 Register
CANFD_WMB0_ID	Pretended Networking Wake Up Message Buffer ID 0 Register
CANFD_WMB1_CS	Pretended Networking Wake Up Message Buffer 1 Control Status Register
CANFD_WMB1_D03	Pretended Networking Wake Up Message Buffer 1 Data 0-3 Register
CANFD_WMB1_D47	Pretended Networking Wake Up Message Buffer 1 Data 4-7 Register
CANFD_WMB1_ID	Pretended Networking Wake Up Message Buffer ID 1 Register
CANFD_WMB2_CS	Pretended Networking Wake Up Message Buffer 2 Control Status Register
CANFD_WMB2_D03	Pretended Networking Wake Up Message Buffer 2 Data 0-3 Register
CANFD_WMB2_D47	Pretended Networking Wake Up Message Buffer 2 Data 4-7 Register
CANFD_WMB2_ID	Pretended Networking Wake Up Message Buffer ID 2 Register
CANFD_WMB3_CS	Pretended Networking Wake Up Message Buffer 3 Control Status Register
CANFD_WMB3_D03	Pretended Networking Wake Up Message Buffer 3 Data 0-3 Register
CANFD_WMB3_D47	Pretended Networking Wake Up Message Buffer 3 Data 4-7 Register
CANFD_WMB3_ID	Pretended Networking Wake Up Message Buffer ID 3 Register
CANFD_WUM	Pretended Networking Wakeup Match Register

Can Bit Timing Register

The CANFD_TIMING register is an alternative way to store the CAN bit timing variables described in the CANFD_CTL1 register. The CANFD_TIMING.EPRESDIV, CANFD_TIMING.EPROPSEG, CANFD_TIMING.EPSEG1, CANFD_TIMING.EPSEG2, and CANFD_TIMING.ERJW bit fields are extended versions of the CANFD_CTL1 register CANFD_CTL1.PRESDIV, CANFD_CTL1.PROPSEG, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2 and CANFD_CTL1.RJW bit fields respectively.

The CANFD_TIMING.BTF bit selects the use of the timing variables defined in the CANFD_TIMING register. The contents of the CANFD_TIMING register are not affected by a soft reset.

NOTE: The CAN bit variables in the CANFD_CTL1 and CANFD_TIMING registers are stored in the same register.

NOTE: When the CAN FD feature is enabled (CANFD_CFG.FDEN is set), always set CANFD_TIMING.BTF.

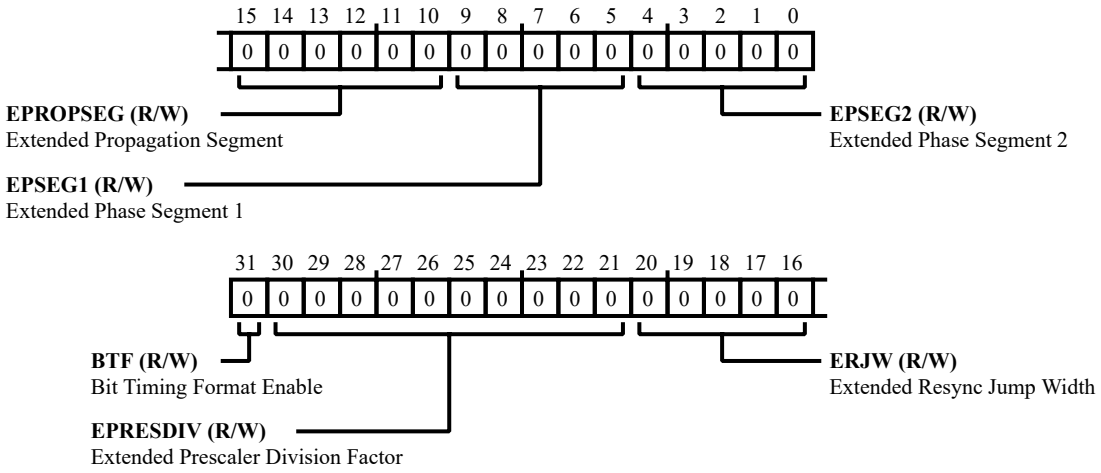


Figure 16-15: CANFD_TIMING Register Diagram

Table 16-33: CANFD_TIMING Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	BTF	Bit Timing Format Enable. The CANFD_TIMING.BTF bit enables the use of the extended CAN bit timing fields in the CANFD_TIMING register to replace the CAN bit time variables defined in the CANFD_CTL1 register. The CANFD_TIMING.BTF bit can only be written in freeze mode.	
		0	Disable
		1	Enable

Table 16-33: CANFD_TIMING Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30:21 (R/W)	EPRESDIV	Extended Prescaler Division Factor. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPRESDIV bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency. The CANFD_TIMING.EPRESDIV bit field extends the CANFD_CTL1.PRESDIV value range. Sclock frequency = PE clock frequency / (EPRESDIV + 1) The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPRESDIV bit field has no effect. The CANFD_TIMING.EPRESDIV bit field can only be written in freeze mode and is blocked by hardware in other modes.
20:16 (R/W)	ERJW	Extended Resync Jump Width. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.ERJW bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization. The CANFD_TIMING.ERJW bit field extends the CANFD_CTL1.RJW value range. Resync Jump Width = ERJW + 1 One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.ERJW bit field has no effect. The CANFD_TIMING.ERJW bit field can only be written in freeze mode and is blocked by hardware in other modes.
15:10 (R/W)	EPROPSEG	Extended Propagation Segment. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPROPSEG bit field defines the length of the propagation segment in the bit time. The CANFD_TIMING.EPROPSEG bit field extends the CANFD_CTL1.PROPSEG value range. Propagation Segment Time = (EPROPSEG + 1) * Time-Quanta One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPROPSEG bit field has no effect. The CANFD_TIMING.EPROPSEG bit field can only be written in freeze mode and is blocked by hardware in other modes.

Table 16-33: CANFD_TIMING Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	EPSEG1	Extended Phase Segment 1. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPSEG1 bit field defines the length of phase segment 1 in the bit time. The CANFD_TIMING.EPSEG1 bit field extends the CANFD_CTL1.PSEG1 value range. Phase Buffer Segment 1 = (EPSEG1 + 1) * Time-Quanta One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPSEG1 bit field has no effect. The CANFD_TIMING.EPSEG1 bit field can only be written in freeze mode and is blocked by hardware in other modes.
4:0 (R/W)	EPSEG2	Extended Phase Segment 2. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPSEG2 bit field defines the length of phase segment 2 in the bit time. The CANFD_TIMING.EPSEG2 bit field extends the CANFD_CTL1.PSEG2 value range. Phase Buffer Segment 1 = (EPSEG1 + 1) * Time-Quanta One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPSEG2 bit field has no effect. The CANFD_TIMING.EPSEG2 bit field can only be written in freeze mode and is blocked by hardware in other modes.

CRC Register

The CANFD_CRC register provides information about the CRC of transmitted messages for non-FD messages. For messages in CAN FD format that require either 17 or 21 bits, the CANFD_CRC register only reports the 15 low order bits of CRC calculations. For CAN FD format frames, the CANFD_FD_CRC register must be used. The CANFD_CRC register updates at the same time the Tx Interrupt Flag is set.

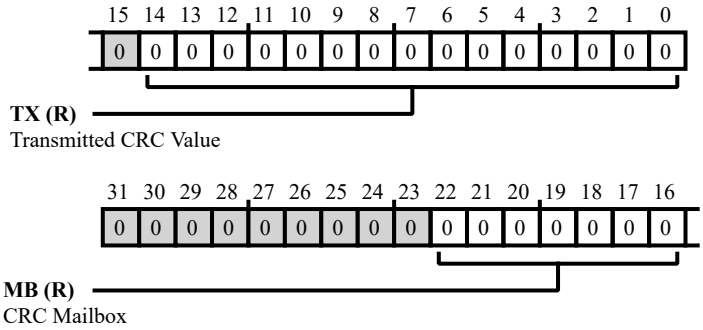


Figure 16-16: CANFD_CRC Register Diagram

Table 16-34: CANFD_CRC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22:16 (R/NW)	MB	CRC Mailbox. The CANFD_CRC.MB bit field indicates the number of the mailbox corresponding to the value in the CANFD_CRC.TX field.
14:0 (R/NW)	TX	Transmitted CRC Value. The CANFD_CRC.TX bit field indicates the CRC value of the last transmitted message for non-FD frames. For CAN FD frames, the CRC value is reported in the CANFD_FD_CRC register.

Control 1 Register

The CANFD_CTL1 register is defined for specific CANFD control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, loop-back mode, listen-only mode, bus off recovery behavior, and interrupt enabling (bus off, error, warning). It also determines the division factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW) are also configurable in the CANFD_TIMING register, which extends the range of all these variables. If the CANFD_TIMING.BTF bit is set, the PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields of the CANFD_CTL1 register become read-only.

NOTE: When the CAN FD feature is enabled, do not use the CANFD_CTL1.PRESDIV, CANFD_CTL1.RJW, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2, and CANFD_CTL1.PROPSEG fields of the CANFD_CTL1 register for CAN bit timing. Instead, use the CANFD_TIMING.EPRESDIV, CANFD_TIMING.ERJW, CANFD_TIMING.EPSEG1, CANFD_TIMING.EPSEG2, and CANFD_TIMING.EPROPSEG fields of the CANFD_TIMING register.

The contents of the CANFD_CTL1 register are not affected by a soft reset.

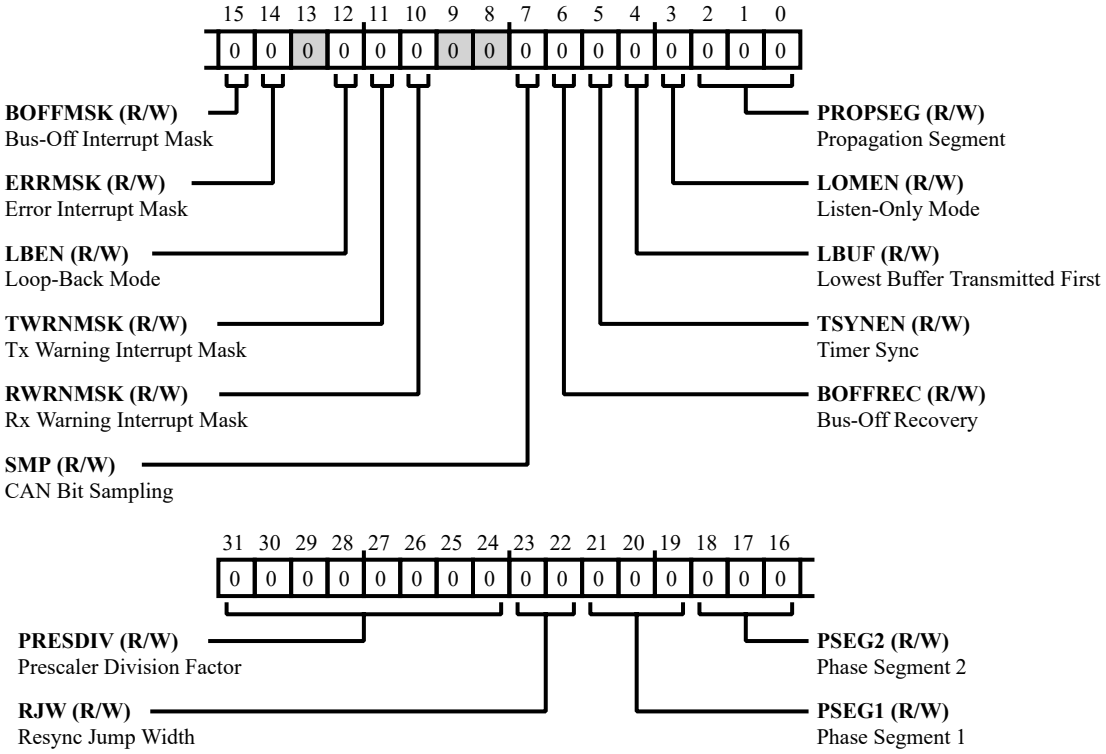


Figure 16-17: CANFD_CTL1 Register Diagram

Table 16-35: CANFD_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	PRESDIV	Prescaler Division Factor. The CANFD_CTL1.PRESDIV bit field defines the ratio between the PE clock frequency and the serial clock (SCLOCK) frequency. The SCLOCK period defines the time quantum of the CAN protocol. For the reset value, the SCLOCK frequency is equal to the PE clock frequency. The maximum value of PRESDIV is 0xFF, which gives a minimum SCLOCK frequency equal to the PE clock frequency divided by 256. The CANFD_CTL1.PRESDIV bit can only be written in freeze mode and is blocked by hardware in other modes. SCLOCK Frequency = PE Clock Frequency / (PRESDIV + 1)	
23:22 (R/W)	RJW	Resync Jump Width. The CANFD_CTL1.RJW bit field defines the maximum number of time quanta that a bit time can change with one re-synchronization. One time quantum is equal to the SCLOCK period. The valid programmable values for the CANFD_CTL1.RJW bits are 0-3. This bit field can only be written in freeze mode and is blocked by hardware in other modes. Resync Jump Width = RJW + 1	
21:19 (R/W)	PSEG1	Phase Segment 1. The CANFD_CTL1.PSEG1 bit field defines the length of the phase segment 1 of the bit time. The valid programmable values are 0-7. This bit field can only be written in freeze mode and is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta	
18:16 (R/W)	PSEG2	Phase Segment 2. The CANFD_CTL1.PSEG2 bit field defines the length of phase segment 2 of the bit time. The valid programmable values are 1-7. This bit field can only be written in freeze mode and is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta	
15 (R/W)	BOFFMSK	Bus-Off Interrupt Mask. The CANFD_CTL1.BOFFMSK bit provides a mask for the bus off interrupt (CANFD_ESR1.BOFFINT).	
		0	Disable
		1	Enable

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	ERRMSK	Error Interrupt Mask. The CANFD_CTL1.ERRMSK bit provides a mask for the error interrupt (CANFD_ESR1.ERRINT).	
		0	Disable
		1	Enable
12 (R/W)	LBEN	Loop-Back Mode. The CANFD_CTL1.LBEN bit configures the CANFD module to operate in loop-back mode. In loop-back mode, the CANFD modules performs an internal loop back to use for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). The CANFD module behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In loop-back mode, the CANFD module ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. The CANFD_CTL1.LBEN bit can only be written in freeze mode and is blocked by hardware in other modes. Note that in loop-back mode, the CANFD_CFG.SRXDIS bit cannot be asserted, because it impedes the self reception of a transmitted message. Also, note that the CANFD_FD_CTL.TDCOMPEN bit must be 0 (transceiver delay compensation feature disabled) when the CANFD_CTL1.LBEN bit is asserted.	
11 (R/W)	TWRNMSK	Tx Warning Interrupt Mask. The CANFD_CTL1.TWRNMSK bit provides a mask for the Tx warning interrupt (CANFD_ESR1.TXWRNINT) The CANFD_CTL1.TWRNMSK bit is read as zero when CAN_MCR.WRNEN is negated. The CANFD_CTL1.TWRNMSK bit can only be written when the CANFD_CFG.WRNEN bit is enabled.	
		0	Tx Warning Interrupt Disabled
		1	Tx Warning Interrupt Enabled

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	RWRNMSK	Rx Warning Interrupt Mask. The CANFD_CTL1.RWRNMSK bit provides a mask for the Rx warning interrupt (CANFD_ESR1.RXWRNINT) The CANFD_CTL1.RWRNMSK bit is read as zero when the CANFD_CFG.WRNEN bit is negated. It can only be written only if the CANFD_CFG.WRNEN bit is enabled.	
		0	Rx Warning Interrupt Disabled
		1	Rx Warning Interrupt Enabled
7 (R/W)	SMP	CAN Bit Sampling. The CANFD_CTL1.SMP bit defines the sampling mode of CAN bits at the Rx input. When the CANFD_CTL1.SMP bit is disabled, one sample determines the bit value. When it is enabled three samples determine the value of the received bit: the regular sample (sample point) and the two preceding samples; a majority rule is used. The CANFD_CTL1.SMP bit can only be written in freeze mode and is blocked by hardware in other modes. Note that for proper operation, when enabling CAN bit sampling, guarantee a minimum value of 2 time quanta in the CANFD_CTL1.PSEG1 (or CANFD_TIMING.EPSEG1) bit field. SMP cannot be asserted when CAN FD is enabled (CANFD_CFG.FDEN = 1).	
		0	Disable
		1	Enable

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	BOFFREC	Bus-Off Recovery. The CANFD_CTL1.BOFFREC bit defines how the CANFD module recovers from bus off state. When the CANFD_CTL1.BOFFREC bit is enabled, automatic recovering from the bus off state occurs according to the CAN Specification 2.0B. When the CANFD_CTL1.BOFFREC bit is disabled, automatic recovering from the bus off state is disabled and the CANFD module remains in the bus off state until the CANFD_CTL1.BOFFREC bit is enabled by software. If this occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then the bus off recovery happens as if the CANFD_CTL1.BOFFREC bit was never disabled. If the bit is enabled after 128 sequences of 11 recessive bits occurred, then the CANFD module will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After being enabled, the CANFD_CTL1.BOFFREC bit can be disabled again during the bus off state, but it will be effective only the next time the CANFD module enters the bus off state. If CANFD_CTL1.BOFFREC bit was enabled when the CANFD module entered the bus off state, disabling the CANFD_CTL1.BOFFREC bit during the bus off state will not be effective for the current bus off recovery.	
		0	Enable
		1	Disable
5 (R/W)	TSYNEN	Timer Sync. The CANFD_CTL1.TSYNEN bit enables a mechanism that resets the free-running timer each time a message is received in message buffer 0. The timer sync feature provides a means to synchronize multiple CANFD stations with a special "SYNC" message (global network time). If the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the first available mailbox, according to the CANFD_CTL2.RFFNUM setting, is used for timer synchronization instead of MB0. The CANFD_CTL1.TSYNEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	LBUF	Lowest Buffer Transmitted First. The CANFD_CTL1.LBUF bit defines the ordering mechanism for message buffer transmission. When the CANFD_CTL1.LBUF bit is enabled, the lowest number buffer is transmitted first and when it is disabled the buffer with highest priority is transmitted first. When the CANFD_CTL1.LBUF bit is enabled, the CANFD_CFG.LPRIOEN bit does not affect the arbitration priority. The CANFD_CTL1.LBUF bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
3 (R/W)	LOMEN	Listen-Only Mode. The CANFD_CTL1.LOMEN bit configures the CANFD module to operate in listen-only mode. When the CANFD_CTL1.LOMEN is enabled to be in listen-only mode, transmission is disabled, all error counters in the CANFD_ECR register are frozen, and the CANFD module operates in the CAN error passive mode. Only messages acknowledged by another CAN station will be received. If the CANFD module detects a message that has not been acknowledged, it flags a BIT0 error without changing the receive error counter (CANFD_ECR.RXERRCNT), as if it were trying to acknowledge the message. Listen-Only mode is acknowledged by the state of the CANFD_ESR1.FLTCONF bit field indicating a passive error. There can be some delay between the listen-only mode request and acknowledge. The CANFD_CTL1.LOMEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
2:0 (R/W)	PROPSEG	Propagation Segment. The CANFD_CTL1.PROPSEG bit field defines the length of the propagation segment in the bit time. The valid programmable values are 0-7. The CANFD_CTL1.PROPSEG bit can only be written in freeze mode and is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) x Time-Quanta. Time-Quantum = One SCLOCK Period.	

Pretended Networking Control1 Register

The CANFD_PN_CTL1 register contains control bits for pretended networking mode filtering selection. Configure this register with the filter criteria to receive wakeup messages. Bits other than CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK can only be written in freeze mode.

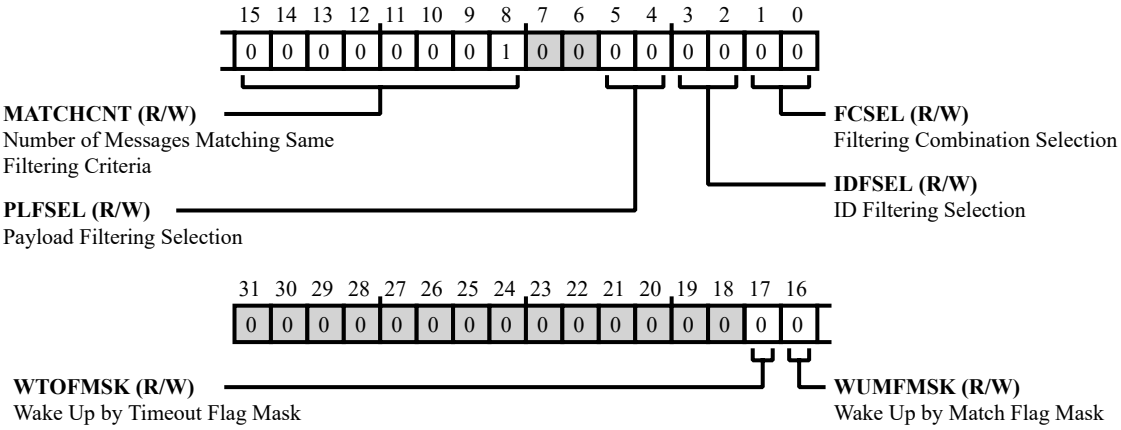


Figure 16-18: CANFD_PN_CTL1 Register Diagram

Table 16-36: CANFD_PN_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	WTOFMSK	Wake Up by Timeout Flag Mask. The CANFD_PN_CTL1.WTOFMSK bit masks the generation of a wakeup event originated by a timeout:	
		0	Timeout Event Disabled
		1	Timeout Event Enabled
16 (R/W)	WUMFMSK	Wake Up by Match Flag Mask. The CANFD_PN_CTL1.WUMFMSK bit masks the generation of a wakeup event originated by a successfully filtered Rx message.	
		0	Wakeup match disabled
		1	Wakeup match enabled

Table 16-36: CANFD_PN_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/W)	MATCHCNT	Number of Messages Matching Same Filtering Criteria. The CANFD_PN_CTL1.MATCHCNT bit field defines the number of times a given message must match the predefined filtering criteria for ID and/or PL before generating a wakeup event. This quantity is configured in the 1 to 255 range. For example, if the received message must match the predefined filtering criteria for ID and/or PL once before generating a wakeup event, the CANFD_PN_CTL1.MATCHCNT bit field value is 0x01; if it must match twice before generating a wakeup event, the bit field value is 0x02; and if it must match 255 times generating a wakeup event, the CANFD_PN_CTL1.MATCHCNT bit field value is 0xFF.	
5:4 (R/W)	PLFSEL	Payload Filtering Selection. The CANFD_PN_CTL1.PLFSEL bit field selects the level of payload filtering to be applied when the CANFD module is under pretended networking mode. Filtering does not accept remote messages (RTR=1) when payload filtering is active.	
		0	Match the payload contents against an exact target value
		1	Match if a payload value is greater than or equal to a specific target value
		2	Match if a payload value is less than or equal to a specific target value
		3	Match if a payload value is inside a range, greater than or equal to a specified lower limit and smaller than or equal a specified upper limit
3:2 (R/W)	IDFSEL	ID Filtering Selection. The CANFD_PN_CTL1.IDFSEL bit field selects the level of ID filtering to be applied when the CANFD module is under pretended networking mode. In ID filtering, if the CANFD_FLTR_ID2_IDMSK.IDE and CANFD_FLTR_ID2_IDMSK.RTR bits are set, the IDE and RTR bits are also considered part of the reception filter.	
		0	Match the ID contents against an exact target value
		1	Match if the ID value is greater than or equal to a specific target value
		2	Match if the ID value is less than or equal to a specific target value
		3	Match if the ID value is inside a range, greater than or equal to a specified lower limit and smaller than or equal a specified upper limit

Table 16-36: CANFD_PN_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	FCSEL	Filtering Combination Selection. The `CANFD_PN_CTL1.FCSEL` bit field selects elects the filtering criteria to be applied when the CANFD module is in pretended networking mode.	
		0	Message ID filtering only
		1	Message ID filtering and payload filtering
		2	Message ID filtering occurring a specified number of times
		3	Message ID filtering and payload filtering a specified number of times

Control 2 Register

The CANFD_CTL2 register complements CANFD_CTL1, providing control bits for memory write accesses in freeze mode, for extending FIFO filter quantity, and for adjusting the operation of internal CANFD processes like matching and arbitration.

The contents of the CANFD_CTL2 register are not affected by soft reset.

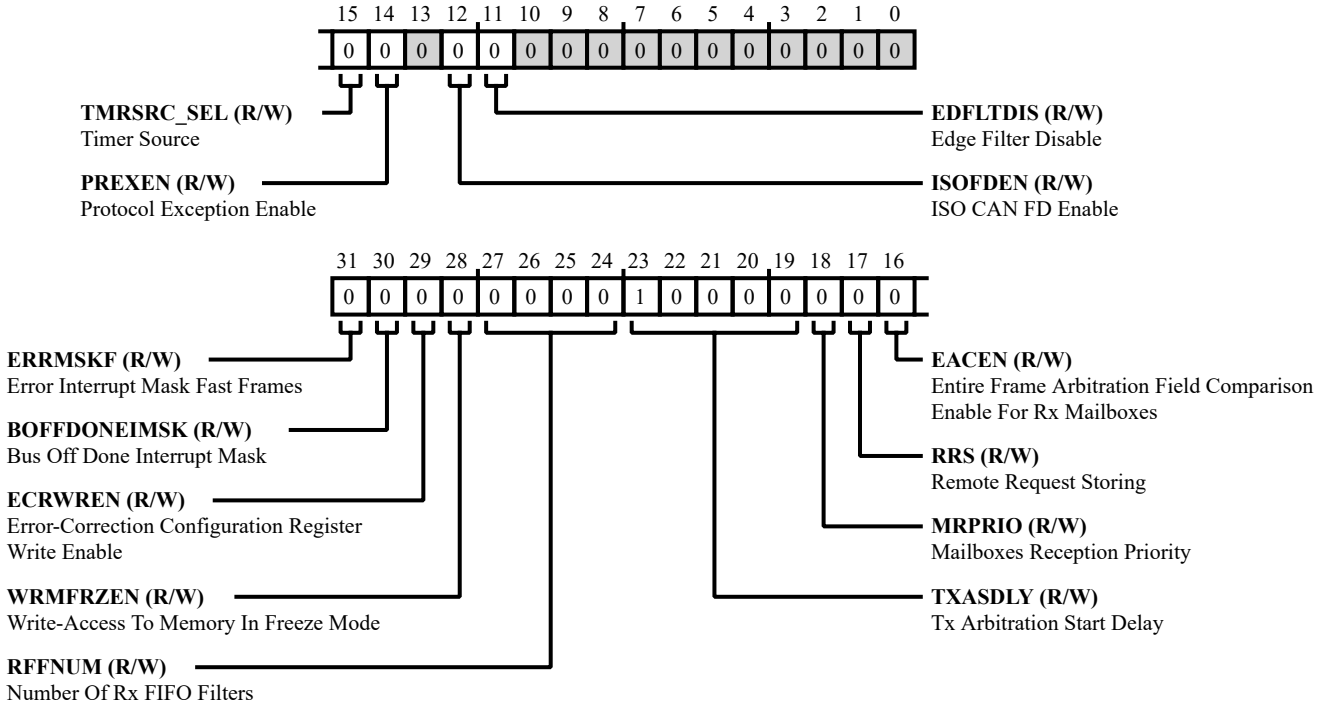


Figure 16-19: CANFD_CTL2 Register Diagram

Table 16-37: CANFD_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	ERRMSKF	Error Interrupt Mask Fast Frames. The CANFD_CTL2.ERRMSKF bit is the error interrupt mask for errors detected in the data phase of fast CAN FD frames. It provides a mask for the CANFD_ESR1.ERRINTF interrupt. When the CANFD_CTL2.ERRMSKF bit is set, the error interrupt is enabled. When the bit is cleared the error interrupt is disabled.
30 (R/W)	BOFFDONEIMSK	Bus Off Done Interrupt Mask. The CANFD_CTL2.BOFFDONEIMSK bit provides a mask for the bus off done interrupt in the CANFD_ESR1 register. When the CANFD_CTL2.BOFFDONEIMSK bit is set, the bus off done interrupt is enabled. When the bit is cleared the bus off done interrupt is disabled.

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	ECRWREN	Error-Correction Configuration Register Write Enable. The CANFD_CTL2.ECRWREN bit enables the MECR register to be updated. Setting the CANFD_CTL2.ECRWREN bit is set, enables updates. Clearing the bit disables updates. The CANFD_CTL2.ECRWREN bit is automatically set to zero if the protocol described in the "Detection and Correction of Memory Errors" section of the CANFD chapter is not followed.
28 (R/W)	WRMFRZEN	Write-Access To Memory In Freeze Mode. The CANFD_CTL2.WRMFRZEN bit enables unrestricted write access to the CANFD memory in freeze mode. Clear the CANFD_CTL2.WRMFRZEN bit to maintain the write access restrictions. Set the CANFD_CTL2.WRMFRZEN bit to enable unrestricted write access to the CANFD memory. The CANFD_CTL2.WRMFRZEN bit can only be written in freeze mode and has no effect out of freeze mode. The CANFD_CFG.RFEN bit must not be set during CANFD memory initialization.

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
27:24 (R/W)	RFFNUM	Number Of Rx FIFO Filters. The CANFD_CTL2.RFFNUM bit field defines the number of Rx FIFO filters. For more details, see the Number of Rx FIFO Filters table in the CANFD chapter. The maximum selectable number of filters is a function of configuration parameter NUMBER_OF_MB. Do not program the CANFD_CTL2.RFFNUM bit field with values that make the number of message buffers occupied by the Rx FIFO and ID Filter exceed the number of mailboxes present, defined by the CANFD_CFG.MAXMB bit field. Each group of eight filters occupies a memory space equivalent to two message buffers, which means that as more filters are implemented fewer mailboxes are available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, program the CANFD_CTL2.RFFNUM bit with a value corresponding to a number of filters not greater than the number of available memory words. Calculate this as follows: (SETUP_MB - 6) x 4 where SETUP_MB is the lower value between parameter NUMBER_OF_MB and register field CANFD_CFG.MAXMB. The number of remaining mailboxes available is (SETUP_MB - 8) - (RFFN x 2) If the number of Rx FIFO filters programmed through the CANFD_CTL2.RFFNUM bit field exceeds the SETUP_MB value (memory space available), the exceeding ones are not functional. Note that the number of the last remaining available mailbox is defined by the least value between (NUMBER_OF_MB - 1) and the CANFD_CFG.MAXMB field. If the individual Rx mask registers are not enabled, all Rx FIFO filters are affected by the Rx FIFO global mask. The CANFD_CTL2.RFFNUM bit field can only be written in freeze mode as it is blocked by hardware in other modes.
23:19 (R/W)	TXASDLY	Tx Arbitration Start Delay. The CANFD_CTL2.TXASDLY bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on the CAN bus. The CANFD_CTL2.TXASDLY bit field can only be written in freeze mode and it is blocked by hardware in other modes.
18 (R/W)	MRPRIO	Mailboxes Reception Priority. The CANFD_CTL2.MRPRIO bit defines the mailboxes reception priority. If the CANFD_CTL2.MRPRIO bit is set, the matching process starts from the mailboxes and if no match occurs the matching continues on the Rx FIFO. If the bit is cleared, matching starts from Rx FIFO and continues on mailboxes. The CANFD_CTL2.MRPRIO bit can only be written only in freeze mode and it is blocked by hardware in other modes.

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RRS	Remote Request Storing. If the CANFD_CTL2.RRS bit is set, a remote request frame is submitted to a matching process and stored in the corresponding message buffer in the same fashion of a data frame. No automatic remote response frame is generated. If the CANFD_CTL2.RRS bit is cleared, the remote request frame is submitted to a matching process and an automatic remote response frame is generated if a message buffer with CODE=0b1010 is found with the same ID. The CANFD_CTL2.RRS bit can only be written in freeze mode and it is blocked by hardware in other modes.	
16 (R/W)	EACEN	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes. The CANFD_CTL2.EACEN bit controls the comparison of IDE and RTR bits within Rx mailboxes filters with their corresponding bits in the incoming frame by the matching process. The CANFD_CTL2.EACEN bit does not affect matching for the Rx FIFO. When the CANFD_CTL2.EACEN bit is set it enables the comparison of both the IDE and RTR bits of an Rx mailbox filter with the corresponding bits of the incoming frame. Mask bits do apply. When the CANFD_CTL2.EACEN bit is cleared the IDE bit of the Rx mailbox filter is always compared and RTR is never compared, regardless of mask bits. EACEN can be only written only in freeze mode and it is blocked by hardware in other modes.	
15 (R/W)	TMRSRC_SEL	Timer Source. The CANFD_CTL2.TMRSRC_SEL bit selects the time tick source used for incrementing the free running timer counter. When the CANFD_CTL2.TMRSRC_SEL bit is set, the free running timer is clocked by an external time tick. The period can be either adjusted to be equal to the baud rate on the CAN bus or a different value as required. When the CANFD_CTL2.TMRSRC_SEL bit is cleared, the free running timer is clocked by the CAN bit clock, which defines the baud rate on the CAN bus. The CANFD_CTL2.TMRSRC_SEL bit can only be written in freeze mode.	
14 (R/W)	PREXEN	Protocol Exception Enable. The CANFD_CTL2.PREXEN bit enables the protocol exception feature. The CANFD_CTL2.PREXEN bit can only be written in freeze mode.	
		0	Disable
		1	Enable

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	ISOFDEN	ISO CAN FD Enable. The `CANFD_CTL2.ISOFDEN` bit controls ISO CAN FD compliant operation. Setting the bit enables ISO CAN FD compliant operation by enabling the following features, which are part of the ISO 11898 standard and not included in the original (Bosch) CAN FD protocol specification: The count of variable stuff bits inserted from the start of frame bit to the last bit of data field. Also, the modulo 8 count of variable stuff bits plus the respective parity bit (even parity calculated over the 3-bit modulo 8 count) are combined as the 4-bit stuff count field and inserted before the CRC sequence field. CRC calculation extends beyond the end of data field and takes the stuff count field bits into account. Clearing the `CANFD_CTL2.ISOFDEN` bit disables ISO CAN FD specific features (non-ISO CAN FD operation). The `CANFD_CTL2.ISOFDEN` bit is only writable in freeze mode.	
11 (R/W)	EDFLTDIS	Edge Filter Disable. The `CANFD_CTL2.EDFLTDIS` bit controls the edge filter used during the bus integration state. When the edge filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The edge filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition. The `CANFD_CTL2.EDFLTDIS` bit is only writable in freeze mode.	
		0	Enable
		1	Disable

Pretended Networking Control2 Register

The CANFD_PN_CTL2 register contains configuration bits for the timeout value under pretended networking mode. Only write to this register in freeze mode.

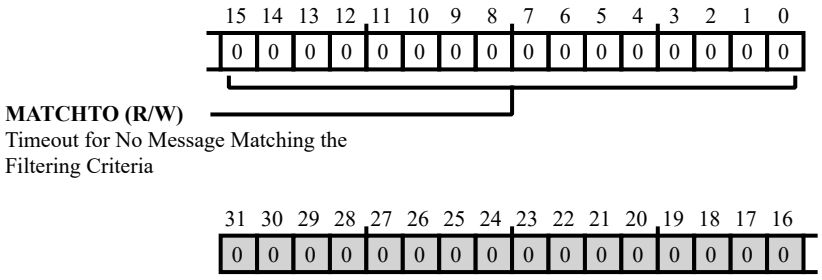


Figure 16-20: CANFD_PN_CTL2 Register Diagram

Table 16-38: CANFD_PN_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	MATCHTO	Timeout for No Message Matching the Filtering Criteria. The CANFD_PN_CTL2.MATCHTO bit field defines a timeout value that generates a wakeup event when the CANFD_CFG.PNETEN bit is enabled. If the timeout counter reaches the target value when the CANFD module is in pretended networking mode, a wakeup event is generated. The timeout limit value is configured from 1 to 65535 to control an internal 16-bit up-count timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, with a period 64 times the CAN bit time unit. When the CANFD_PN_CTL2.MATCHTO bit field is 0x0000, the timeout is disabled.

Error Counter Register

The CANFD_ECR register has four 8-bit fields reflecting the value of the CANFD error counters:

• Transmit error counter (CANFD_ECR.TXERRCNT field)

• Receive error counter (CANFD_ECR.RXERRCNT field)

• Transmit error counter for errors detected in the data phase of CAN FD messages with the BRS bit set (CANFD_ECR.TXERRCNTF field)

• Receive error counter for errors detected in the data phase of CAN FD messages with the BRS bit set (CANFD_ECR.RXERRCNTF field)

The TXERRCNT and RXERRCNT counters take into account all errors in both CAN FD and non-FD message formats. The TXERRCNTF and RXERRCNT_FAST counters only count the errors that occur in the data phase of CAN FD frames with the BRS bit set.

The fault confinement state (CANFD_ESR1.FLTCONF) is only updated based on the TXERRCNT and RXERRCNT counters. The TXERRCNT and RXERRCNT counters can only be written in freeze mode. The TXERRCNTF and RXERRCNT_FAST counters are read-only except in freeze mode, where the processor can write a zero value. The rules for incrementing and decrementing these counters are described in the CAN protocol and are completely implemented in the CANFD module.

The basic rules for the CANFD bus state transitions are as follows:

• If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the CANFD_ESR1.FLTCONF field updates to reflect the error passive state.

• If the CANFD state is error passive, and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the CANFD_ESR1.FLTCONF field updates to reflect the error active state.

• If the value of TXERRCNT increases to be greater than 255, the CANFD_ESR1.FLTCONF field updates to reflect the bus off state and an interrupt may be issued. The value of TXERRCNT is then reset to zero.

• If the CANFD is in the bus off state, then TXERRCNT is cascaded together with another internal counter to count the 128 occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing TXERRCNT. When TXERRCNT reaches the value of 128, the CANFD_ESR1.FLTCONF field is updated to be error active and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNTF counter is frozen during bus off.

• If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the CANFD_ESR1.ACKERR bit). After the transition to the error passive state, TXERRCNT is no longer incremented by acknowledge errors. Therefore, the device never goes to the bus off state.

• If RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to the error active state.

• The TXERRCNTF and RXERRCNT_FAST error counters values increment and decrement based on errors detected only in the data phase of CAN FD frames with the BRS bit set, following the same increment and decrement rules as the TXERRCNT and RXERRCNT counters. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen while the CANFD is in the bus off state. They are reset when the CANFD leaves the bus off state and restart counting once the CANFD resumes to the error active state.

• When the CANFD module is in pretended networking mode, RXERRCNT and RXERRCNT_FAST keep counting errors and error flags are stored. TXERRCNT and TXERRCNTF preserve their values and do not change since no transmission occurs under pretended networking mode. Error counters and error flags that changed values while in pretended networking mode are updated in the CANFD_ECR and CANFD_ESR1 registers when the CANFD module resumes normal mode. The FAST error flags in the CANFD_ESR1 register are not set if the CANFD is in pretended networking mode.

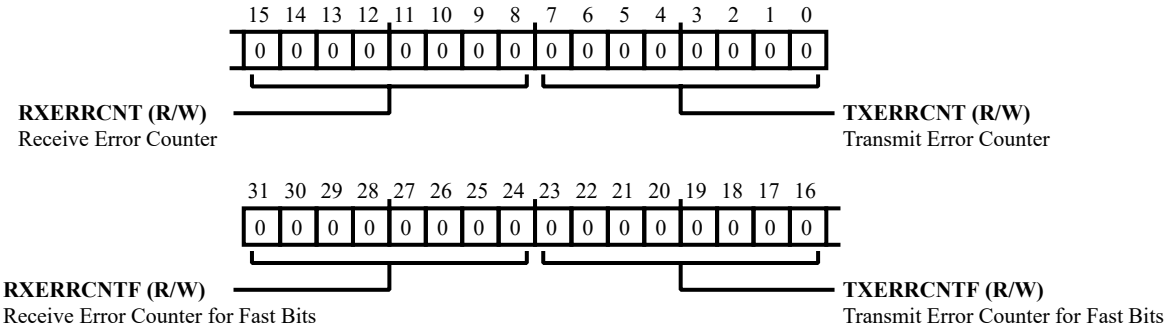


Figure 16-21: CANFD_ECR Register Diagram

Table 16-39: CANFD_ECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RXERRCNTF	Receive Error Counter for Fast Bits. The CANFD_ECR.RXERRCNTF bit field is for errors detected in the data phase of received CAN FD messages with the BRS bit set. The CANFD_ECR.RXERRCNTF bit field is read-only except in freeze mode. In freeze mode the processor can only write a 8-bit zero value.
23:16 (R/W)	TXERRCNTF	Transmit Error Counter for Fast Bits. The CANFD_ECR.TXERRCNTF bit field is for errors detected in the data phase of received CAN FD messages with the BRS bit set. The CANFD_ECR.TXERRCNTF bit field is read-only except in freeze mode. In freeze mode the processor can only write a 8-bit zero value.

Table 16-39: CANFD_ECR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	RXERRCNT	Receive Error Counter. The `CANFD_ECR.RXERRCNT` bit field is for all errors detected in received messages. The `CANFD_ECR.RXERRCNT` bit field is read-only except in freeze mode, where the processor can write it.
7:0 (R/W)	TXERRCNT	Transmit Error Counter. The `CANFD_ECR.TXERRCNT` bit field is for all errors detected in transmitted messages. The `CANFD_ECR.TXERRCNT` bit field is read-only except in freeze mode, where the processor can write it.

Error Injection Address Register

The CANFD_ERR_IADDR register holds the address where a memory read error is injected.

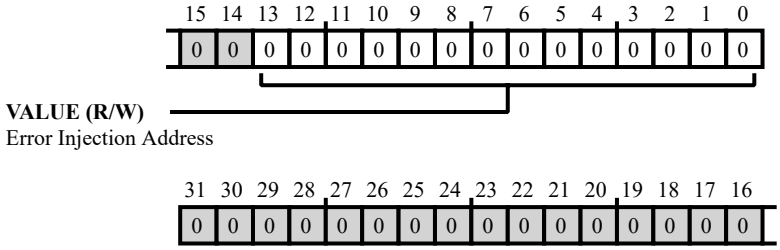


Figure 16-22: CANFD_ERR_IADDR Register Diagram

Table 16-40: CANFD_ERR_IADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:0 (R/W)	VALUE	Error Injection Address. The CANFD_ERR_IADDR.VALUE bit field defines the physical RAM address where a memory read error is to be injected. See the Error Injection Address Mapping table in the CANFD chapter for more details. Note that the two least significant bits are always read as zero.

Error Injection Data Pattern Register

The CANFD_ERR_IDP register holds the error pattern injected in the data word that is read from memory.

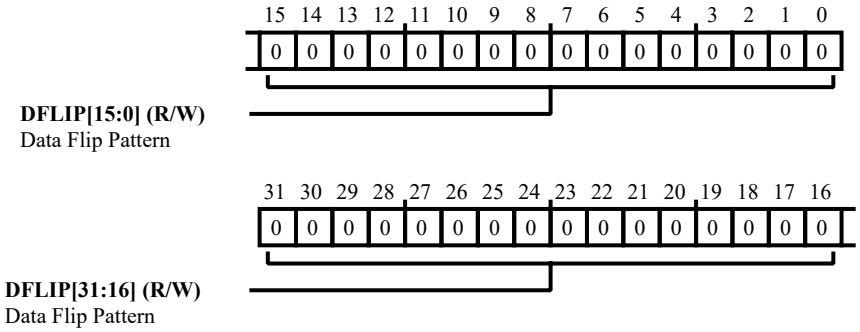


Figure 16-23: CANFD_ERR_IDP Register Diagram

Table 16-41: CANFD_ERR_IDP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DFLIP	Data Flip Pattern. The CANFD_ERR_IDP.DFLIP bit field determines the data flip pattern. Bits set to one in the flip pattern cause the corresponding data bit in the word read from memory to invert.

Error Injection Parity Pattern Register

The CANFD_ERR_IPP register holds the error pattern that is injected in the parity bits read from memory along with data word. Bits set to one in the flip pattern cause the corresponding parity bit, in the word read from memory, to invert.

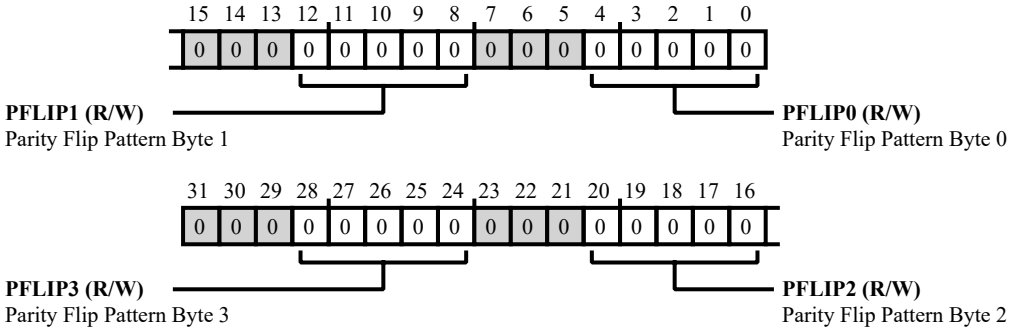


Figure 16-24: CANFD_ERR_IPP Register Diagram

Table 16-42: CANFD_ERR_IPP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:24 (R/W)	PFLIP3	Parity Flip Pattern Byte 3. The CANFD_ERR_IPP.PFLIP3 bit field is for the parity flip pattern for byte 3 (most significant).
20:16 (R/W)	PFLIP2	Parity Flip Pattern Byte 2. The CANFD_ERR_IPP.PFLIP2 bit field is for the parity flip pattern for byte 2.
12:8 (R/W)	PFLIP1	Parity Flip Pattern Byte 1. The CANFD_ERR_IPP.PFLIP1 bit field is for the parity flip pattern for byte 1.
4:0 (R/W)	PFLIP0	Parity Flip Pattern Byte 0. The CANFD_ERR_IPP.PFLIP0 bit field is for the parity flip pattern for byte 0 (least significant).

Error Status Register

The CANFD_ERR_STAT register holds the status bits of the error correction and detection operations. Clear these flags by writing them with a one. Writing a zero to the flags has no effect.

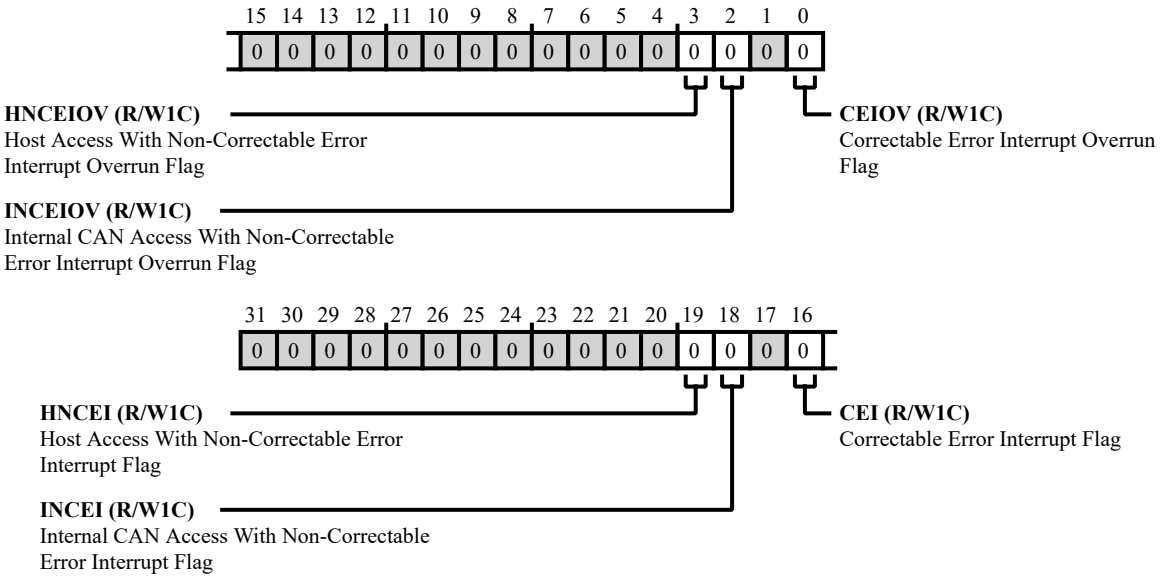


Figure 16-25: CANFD_ERR_STAT Register Diagram

Table 16-43: CANFD_ERR_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W1C)	HNCEI	Host Access With Non-Correctable Error Interrupt Flag. The CANFD_ERR_STAT.HNCEI bit indicates if a non-correctable error is detected in a host access. If the CANFD_ERR_STAT.HNCEI bit is set, a non-correctable error was detected in a memory read initiated by the host. A bus transfer error is asserted for that access. If the CANFD_MEC.HNCIMSK bit is set, the interrupt is asserted. If the CANFD_ERR_STAT.HNCEI bit is cleared, no non-correctable errors were detected in host accesses so far.	
		0	No Non-Correctable Error Detected
		1	Non-Correctable Error Detected

Table 16-43: CANFD_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W1C)	INCEI	Internal CAN Access With Non-Correctable Error Interrupt Flag. The CANFD_ERR_STAT.INCEI bit indicates if a non-correctable error is detected in a CAN access. If the CANFD_ERR_STAT.INCEI bit is set, a non-correctable error was detected in a memory read initiated by a CANFD internal process. A bus transfer error is asserted for that access. If the CANFD_MEC.INCEMSK bit is set, the interrupt is asserted. If the CANFD_ERR_STAT.INCEI bit is cleared, no non-correctable errors were detected in CAN accesses so far.	
		0	No Non-Correctable Error Detected
		1	Non-Correctable Error Detected
16 (R/W1C)	CEI	Correctable Error Interrupt Flag. The CANFD_ERR_STAT.CEI bit indicates if a correctable error is detected. If the CANFD_ERR_STAT.CEI bit is set, a correctable error was detected in a memory read. If the CANFD_MEC.CEIMSK bit is set, the interrupt is asserted. If the CANFD_ERR_STAT.CEI bit is cleared, no correctable errors were detected so far.	
		0	No Correctable Error Detected
		1	Correctable Error Detected
3 (R/W1C)	HNCEIOV	Host Access With Non-Correctable Error Interrupt Overrun Flag. The CANFD_ERR_STAT.HNCEIOV bit indicates if an overrun on non-correctable error is detected in a host access. If the CANFD_ERR_STAT.HNCEIOV bit is set, a non-correctable error was detected in a memory read initiated by the host when the CANFD_ERR_STAT.HNCEI bit is set. No interrupt is associated with this flag. If the CANFD_ERR_STAT.HNCEIOV bit is cleared, no overrun on non-correctable errors were detected in a host access.	
		0	No Overrun Detected
		1	Overrun Detected

Table 16-43: CANFD_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	INCEIOV	Internal CAN Access With Non-Correctable Error Interrupt Overrun Flag. The `CANFD_ERR_STAT.INCEIOV` bit indicates if an overrun on non-correctable error is detected in a CAN access. If the `CANFD_ERR_STAT.INCEIOV` bit is set, a non-correctable error was detected in a memory read initiated by a CANFD internal process when the `CANFD_ERR_STAT.INCEI` bit is set. No interrupt is associated with this flag. If the `CANFD_ERR_STAT.INCEIOV` bit is cleared, no overrun on non-correctable errors were detected in a CAN access.	
		0	No Overrun Detected
		1	Overrun Detected
0 (R/W1C)	CEIOV	Correctable Error Interrupt Overrun Flag. The `CANFD_ERR_STAT.CEIOV` bit indicates if an overrun on correctable errors is detected. If the `CANFD_ERR_STAT.CEIOV` bit is set, a overrun on a correctable error was detected in a memory read while the `CANFD_ERR_STAT.CEI` bit was set. No interrupt is associated with this flag. If the `CANFD_ERR_STAT.CEIOV` bit is cleared, no overrun on correctable errors were detected.	
		0	No Overrun Detected
		1	Overrun Detected

Error and Status 1 Register

The CANFD_ESR1 register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device, and is the source of some interrupts to the processor.

The reported error conditions are:

• CANFD_ESR1.B1ERR, CANFD_ESR1.B0ERR, CANFD_ESR1.ACKERR, CANFD_ESR1.CRCERR, CANFD_ESR1.FRMERR, and CANFD_ESR1.STFERR, for errors detected in CAN frames of any format.

• CANFD_ESR1.B1ERRF, CANFD_ESR1.B0ERRF, CANFD_ESR1.CRCERRF, CANFD_ESR1.FRMERRF, and CANFD_ESR1.STFERRF for errors detected in the data phase of CAN FD frames with the BRS bit set.

An error detected in a single CAN frame is reported by one or more error flags. Error reporting is cumulative in case additional error events occur in subsequent frames before the processor reads the CANFD_ESR1 register.

CANFD_ESR1.TXWRN, CANFD_ESR1.RXWRN, CANFD_ESR1.IDLE, CANFD_ESR1.TXINPROG, CANFD_ESR1.FLTCONF, CANFD_ESR1.RXINPROG, and CANFD_ESR1.SYNC are status bits.

CANFD_ESR1.BOFFINT, CANFD_ESR1.BOFFDONEINT, CANFD_ESR1.ERRINT, CANFD_ESR1.ERRINTF, CANFD_ESR1.WAKINT,CANFD_ESR1.TXWRNINT, and CANFD_ESR1.RXWRNINT are interrupt bits. When servicing interrupt requests generated by these bits use the following procedure:

• Read the CANFD_ESR1 register to capture all error condition and status bits. This action clears the respective bits set since the last read access.

• Write 1 to clear the interrupt bit that triggered the interrupt request.

• Write 1 to clear the CANFD_ESR1.ERROVR bit if it is set.

Starting from all error flags cleared, a first error event sets either the CANFD_ESR1.ERRINT or the CANFD_ESR1.ERRINTF bits (provided the corresponding mask bit is asserted). If other error events in subsequent frames occur before the processor services the interrupt request, the CANFD_ESR1.ERROVR bit is set to indicate that errors from different frames have accumulated.

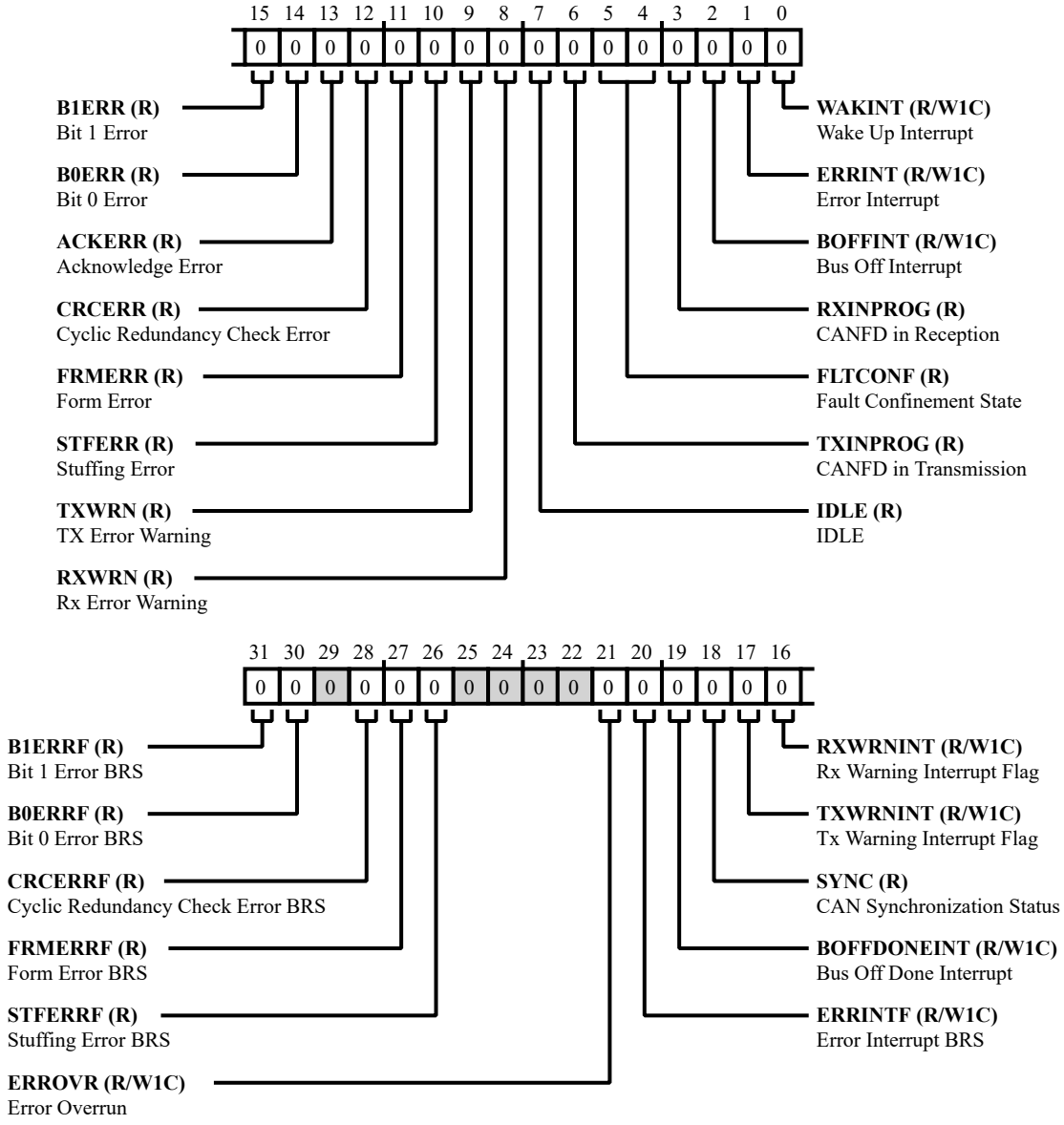


Figure 16-26: CANFD_ESR1 Register Diagram

Table 16-44: CANFD_ESR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	B1ERRF	Bit 1 Error BRS. The CANFD_ESR1.B1ERRF bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.B1ERRF bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
30 (R/NW)	B0ERRF	Bit 0 Error BRS. The CANFD_ESR1.B0ERRF bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.B0ERRF bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
28 (R/NW)	CRCERRF	Cyclic Redundancy Check Error BRS. The CANFD_ESR1.CRCERRF bit indicates if a CRC error is detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set. The calculated CRC is different from the received CRC. When the CANFD_ESR1.CRCERRF bit is enabled, a CRC error occurred since the last read of the register. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
27 (R/NW)	FRMERRF	Form Error BRS. The CANFD_ESR1.FRMERRF bit indicates that a form error is detected by the receiver node in the data phase of CAN FD frames with the BRS bit set. A fixed-form bit field contains at least one illegal bit. When the CANFD_ESR1.FRMERRF bit is enabled, a form error occurred since the last read of the register. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	STFERRF	Stuffing Error BRS. The CANFD_ESR1.STFERRF bit indicates that a stuffing error is detected in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.STFERRF bit is enabled, a stuffing error occurred since the last read of the register. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
21 (R/W1C)	ERROVR	Error Overrun. The CANFD_ESR1.ERROVR bit indicates that an error condition occurred when any error flag is already set. When the CANFD_ESR1.ERROVR bit is enabled an overrun has occurred. When it is disabled, an overrun has not occurred. Clear the CANFD_ESR1.ERROVR bit by writing a one to it.	
		0	Disable
		1	Enable
20 (R/W1C)	ERRINTF	Error Interrupt BRS. The CANFD_ESR1.ERRINTF bit indicates an error interrupt is detected in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.ERRINTF bit is enabled it indicates setting the of any error bit detected in the data phase of CAN FD frames with the BRS bit set. When it is disabled, there is no such occurrence. The CANFD_ESR1.ERRINTF bit indicates that at least one of the error bits detected in the data phase of CAN FD frames with the BRS bit set (CANFD_ESR1.B1ERRF, CANFD_ESR1.B0ERRF, CANFD_ESR1.CRCERRF, CANFD_ESR1.FRMERRF, or CANFD_ESR1.STFERRF) is set. If the CANFD_CTL2.ERRMSKF is set, an interrupt is generated to the processor. Clear the CANFD_ESR1.ERRINTF bit by writing a one to it. Writing a zero has no effect.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W1C)	BOFFDONEINT	Bus Off Done Interrupt. When the CANFD_ESR1.BOFFDONEINT bit is enabled it indicates the CANFD module has completed the bus off process. When it is disabled, there is no such occurrence. The CANFD_ESR1.BOFFDONEINT bit is set when the Tx Error Counter (CANFD_ECR.TXERRCNT) finishes counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave bus off mode. If the CANFD_CTL2.BOFFDONEIMSK is set, an interrupt is generated to the processor. Clear the CANFD_ESR1.BOFFDONEINT bit by writing a one to it. Writing a zero has no effect.	
		0	Disable
		1	Enable
18 (R/NW)	SYNC	CAN Synchronization Status. The CANFD_ESR1.SYNC bit is a read-only flag that indicates whether the CANFD module is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the CANFD module. When the CANFD_ESR1.SYNC bit is enabled, the CANFD module is synchronized to the CAN bus. When it is disabled, the CANFD module is not synchronized to the CAN bus.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	TXWRNINT	Tx Warning Interrupt Flag. When the CANFD_ESR1.TXWRNINT bit is enabled, the Tx error counter transitioned from less than 96 to greater than or equal to 96. When the bit is disabled, there is no such occurrence. If the CANFD_CFG.WRNEN bit is enabled, the CANFD_ESR1.TXWRNINT bit is set when the CANFD_ESR1.TXWRN bit transitions from 0 to 1, meaning that the Tx error counter reached 96. If the CANFD_CTL1.TWRNMSK bit is set, an interrupt is generated to the processor. The CANFD_ESR1.TXWRNINT bit is cleared by writing a one to it. Writing a zero has no effect. When the CANFD_CFG.WRNEN bit is disabled, the CANFD_ESR1.TXWRNINT flag is masked. The processor must clear the CANFD_ESR1.TXWRNINT flag before disabling the CANFD_CFG.WRNEN bit or the CANFD_ESR1.TXWRNINT bit will be set when the CANFD_CFG.WRNEN bit is enabled again. The CANFD_ESR1.TXWRNINT bit is not generated during the bus off state and is not updated during freeze mode. When the CANFD module returns to normal mode from pretended networking mode, the CANFD_ESR1.TXWRNINT bit does not update.	
		0	Disable
		1	Enable
16 (R/W1C)	RXWRNINT	Rx Warning Interrupt Flag. When the CANFD_ESR1.RXWRNINT bit is enabled, it indicates that the Rx error counter transitioned from less than 96 to greater than or equal to 96. When the bit is disabled, there is no such occurrence. If the CANFD_CFG.WRNEN bit is enabled, the CANFD_ESR1.RXWRNINT bit is set when the CANFD_ESR1.RXWRN bit transitions from 0 to 1, meaning that the Rx error counter reached 96. If the CANFD_CTL1.RWRNMSK bit is set, an interrupt is generated to the processor. The CANFD_ESR1.RXWRNINT bit is cleared by writing a one to it. Writing a zero has no effect. When the CANFD_CFG.WRNEN bit is disabled, the CANFD_ESR1.RXWRNINT flag is masked. The processor must clear the CANFD_ESR1.RXWRNINT flag before disabling the CANFD_CFG.WRNEN bit or the CANFD_ESR1.RXWRNINT bit will be set when the CANFD_CFG.WRNEN bit is enabled again. The CANFD_ESR1.RXWRNINT bit is not updated during freeze mode. When the CANFD module returns to normal mode from pretended networking mode, the CANFD_ESR1.RXWRNINT bit does not update to reflect the Rx error counter state.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	B1ERR	Bit 1 Error. The CANFD_ESR1.B1ERR bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. When the CANFD_ESR1.B1ERR bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.B1ERR bit updates when the CANFD module returns to normal mode from pretended networking mode. Note that the CANFD_ESR1.B1ERR bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.	
		0	Disable
		1	Enable
14 (R/NW)	B0ERR	Bit 0 Error. The CANFD_ESR1.B0ERR bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. When the CANFD_ESR1.B0ERR bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.B0ERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
13 (R/NW)	ACKERR	Acknowledge Error. The CANFD_ESR1.ACKERR bit indicates an acknowledge error is detected by the transmitter node (a dominant bit detected during the ACK SLOT) When the CANFD_ESR1.ACKERR bit is enabled an ACK error occurred since the last read of this register.When the bit is disabled, there is no such occurrence. The CANFD_ESR1.ACKERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/NW)	CRCERR	Cyclic Redundancy Check Error. The CANFD_ESR1.CRCERR bit indicates if a CRC error is detected by the receiver node in a non-FD message or in the arbitration or data phase of a frame in CAN FD format (the calculated CRC is different from the received CRC) When the CANFD_ESR1.CRCERR bit is enabled, a CRC error occurred since the last read of the register. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.CRCERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
11 (R/NW)	FRMERR	Form Error. The CANFD_ESR1.FRMERR bit indicates that a form error is detected by the receiver node in a non-FD message or in a CAN FD message in the arbitration or data phase. A fixed-form bit field contains at least one illegal bit. When the CANFD_ESR1.FRMERR bit is enabled, a form error occurred since the last read of the register. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.FRMERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
10 (R/NW)	STFERR	Stuffing Error. The CANFD_ESR1.STFERR bit indicates that a stuffing error is detected by the receiver node in a non-FD message or in a CAN FD message in the arbitration or data phase. When the CANFD_ESR1.STFERR bit is enabled, a stuffing error occurred since the last read of the register. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.STFERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	TXWRN	TX Error Warning. The CANFD_ESR1.TXWRN bit indicates when repetitive errors are occurring during message transmission. When the CANFD_ESR1.TXWRN bit is enabled, the CANFD_ECR.TXERRCNT value is greater than or equal to 96. When it is disabled, there is no such occurrence. The CANFD_ESR1.TXWRN bit is only affected by the CANFD_ECR.TXERRCNT value. It does not update during freeze mode.	
		0	Disable
		1	Enable
8 (R/NW)	RXWRN	Rx Error Warning. The CANFD_ESR1.RXWRN bit indicates when repetitive errors are occurring during message transmission. When the CANFD_ESR1.RXWRN bit is enabled, the CANFD_ECR.TXERRCNT value is greater than or equal to 96. When it is disabled, there is no such occurrence. The CANFD_ESR1.RXWRN bit is only affected by the CANFD_ECR.RXERRCNT value. It does not update during freeze mode. It updates when the CANFD returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
7 (R/NW)	IDLE	IDLE. The CANFD_ESR1.IDLE bit indicates when the CAN bus is in the IDLE state. When the CANFD_ESR1.IDLE bit is enabled, the CAN bus is IDLE. When it is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
6 (R/NW)	TXINPROG	CANFD in Transmission. The CANFD_ESR1.TXINPROG bit indicates if the CANFD module is transmitting a message. When the CANFD_ESR1.TXINPROG bit is enabled, the CANFD module is transmitting a message. When it is disabled, the CANFD module is not transmitting a message.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	FLTCONF	Fault Confinement State. The CANFD_ESR1.FLTCONF bit field indicates the fault confinement state of the CANFD module. If the CANFD_CTL1.LOMEN bit is enabled, after some delay that depends on the CAN bit timing, the CANFD_ESR1.FLTCONF bit field will indicate the error passive state. The same delay affects how the CANFD_ESR1.FLTCONF bit field reflects an update to the CANFD_ECR register by the processor. It may take up to one CAN bit time to get them coherent again. The CANFD_ESR1.FLTCONF bit field is affected by soft reset. However, if the CANFD_CTL1.LOMEN bit is enabled, the CANFD_ESR1.FLTCONF reset value lasts just one CAN bit. After this time, the CANFD_ESR1.FLTCONF bit field indicates the error passive state.	
		0	Error Active
		1	Error Passive
		2	Bus Off
		3	Bus Off
3 (R/NW)	RXINPROG	CANFD in Reception. The CANFD_ESR1.RXINPROG bit indicates if the CANFD module is receiving a message. When the CANFD_ESR1.RXINPROG bit is enabled the CANFD module is receiving a message and when it is disabled the CANFD module is not receiving a message.	
		0	Disable
		1	Enable
2 (R/W1C)	BOFFINT	Bus Off Interrupt. The CANFD_ESR1.BOFFINT bit is enabled when the CANFD module enters the bus off state. If the corresponding mask bit (CAN_CTRL1.BOFFMSK) is set, an interrupt is generated to the processor. When the CANFD_ESR1.BOFFINT bit is disabled, there is no such occurrence. The CANFD_ESR1.BOFFINT bit is cleared by writing one to it. Writing zero has no effect.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	ERRINT	Error Interrupt. The CANFD_ESR1.ERRINT bit indicates that at least one of the error bits (CANFD_ESR1.B1ERR, CANFD_ESR1.B0ERR, CANFD_ESR1.CRCERR, CANFD_ESR1.FRMERR, or CANFD_ESR1.STFERR) is set. If the corresponding mask bit (CANFD_CTL1.ERRMSK) is set, an interrupt is generated to the processor. The CANFD_ESR1.ERRINT bit is cleared by writing a one to it. Writing a zero has no effect.	
		0	Disable
		1	Enable
0 (R/W1C)	WAKINT	Wake Up Interrupt. The CANFD_ESR1.WAKINT bit applies when the CANFD module is in low-power mode under a self wake up mechanism (doze or stop mode). When the CANFD_ESR1.WAKINT bit is enabled it indicates a recessive-to-dominant transition was received on the CAN bus. When a recessive-to-dominant transition is detected on the CAN bus and the CANFD_CFG.WAKMSK bit is set, an interrupt is generated to the processor. This bit is cleared by writing it to one. When the bit is disabled, there is no such occurrence. When the CANFD_CFG.SLFWAKEN bit is disabled, the CANFD_ESR1.WAKINT flag is masked. The processor must clear the CANFD_ESR1.WAKINT flag before disabling the CANFD_CFG.SLFWAKEN bit. Otherwise, he CANFD_ESR1.WAKINT is set when the CANFD_CFG.SLFWAKEN bit is set again. Writing zero has no effect. The CANFD_ESR1.WAKINT bit is cleared by writing one to it. Writing zero has no effect.	
		0	Disable
		1	Enable

Error and Status 2 Register

The CANFD_ESR2 register reports general status information.

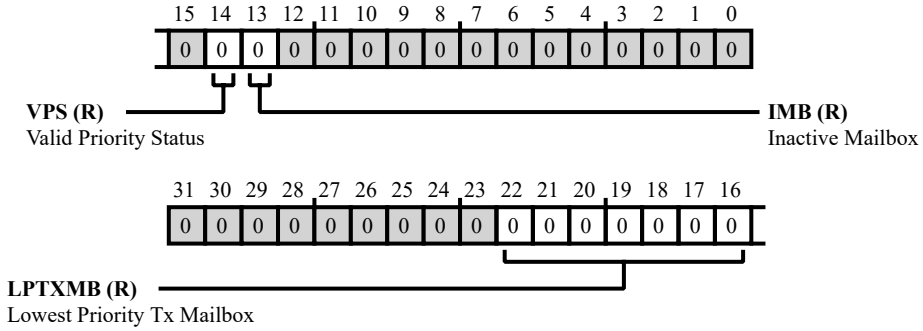


Figure 16-27: CANFD_ESR2 Register Diagram

Table 16-45: CANFD_ESR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22:16 (R/NW)	LPTXMB	Lowest Priority Tx Mailbox. If the CANFD_ESR2.VPS bit is set, the CANFD_ESR2.LPTXMB bit field indicates the lowest number inactive mailbox, per the CANFD_ESR2.IMB bit. If there is no inactive mailbox, then the mailbox indicated depends on the CANFD_CTL1.LBUF bit value. If the CANFD_CTL1.LBUF bit bit is cleared, the mailbox indicated is the one that has the greatest arbitration value. If the CANFD_CTL1.LBUF bit bit is set, the mailbox indicated is the highest number active Tx mailbox. If a Tx mailbox is being transmitted, it is not considered in the CANFD_ESR2.LPTXMB bit field calculation. If the CANFD_ESR2.IMB bit is not set and a frame is transmitted successfully, the CANFD_ESR2.LPTXMB bit field is updated with the mailbox number.

Table 16-45: CANFD_ESR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	VPS	Valid Priority Status. The CANFD_ESR2.VPS bit indicates whether the CANFD_ESR2.IMB and CANFD_ESR2.LPTXMB contents are currently valid. If the CANFD_ESR2.VPS bit is set, the contents of the CANFD_ESR2.IMB and CANFD_ESR2.LPTXMB bits are valid. If the bit is cleared, the contents of the CANFD_ESR2.IMB and CANFD_ESR2.LPTXMB bits are invalid. The CANFD_ESR2.VPS bit is set upon every complete Tx arbitration process unless the processor writes to the C/S word of a mailbox that has already been scanned (is behind Tx Arbitration Pointer during the Tx arbitration process). If there is no inactive mailbox and only one Tx mailbox that is being transmitted, then the CANFD_ESR2.VPS bit is not set. The CANFD_ESR2.VPS bit is cleared upon the start of every Tx arbitration process or upon a write to the C/S word of any mailbox. Note that the CANFD_ESR2.VPS bit is not affected by any processor write into the C/S of a MB that is blocked by the abort mechanism. When the CANFD_CFG.ABORTEN bit is set, the abort code write into the C/S of an MB that is being transmitted (pending abort), or any write attempt into a Tx MB with the interrupt flag set is blocked.
13 (R/NW)	IMB	Inactive Mailbox. If the CANFD_ESR2.VPS bit is set, the CANFD_ESR2.IMB bit indicates whether there is any inactive mailbox (CODE field is either 0b1000 or 0b0000). If the CANFD_ESR2.IMB bit is set and the CANFD_ESR2.VPS bit is set and there is at least one inactive mailbox. The CANFD_ESR2.LPTXMB bit field content is the number of the first one. If the CANFD_ESR2.IMB bit is cleared and the CANFD_ESR2.VPS bit is set, the CANFD_ESR2.LPTXMB bit field is not an inactive mailbox. The CANFD_ESR2.IMB bit is set during arbitration, if a CANFD_ESR2.LPTXMB value is found and it is inactive. The CANFD_ESR2.IMB bit is also set if CAN_ESR2.MB is not set and a frame is transmitted successfully. The CANFD_ESR2.IMB bit is cleared at the start of arbitration. Note that if an MB is successfully transmitted and The CANFD_ESR2.IMB bit is zero (no inactive mailbox), then the CANFD_ESR2.VPS and CANFD_ESR2.IMB bits are set and the index related to the MB just transmitted is loaded into the CANFD_ESR2.LPTXMB bit field.

CANFD Bit Timing Register

The CANFD_FD_TIMING register stores the CAN bit timing variables for use in the data phase of CAN FD messages when the CANFD_FD_CTL.BRSEN bit is set, compatible with CAN FD specification. The CANFD_FD_TIMING.FPRESDIV, CANFD_FD_TIMING.FPROPSEG, CANFD_FD_TIMING.FPSEG1, CANFD_FD_TIMING.FPSEG2, and CANFD_FD_TIMING.FRJW bit fields define the time quantum duration, the number of time quanta per CAN bit, and the sample point position for the data bit rate portion of a CAN FD message with the BRS bit set.

The contents of the CANFD_FD_TIMING register are not affected by soft reset.

Note that the sum of values in the CANFD_FD_TIMING.FPROPSEG and CANFD_FD_TIMING.FPSEG1 bit fields must be at least two time quanta.

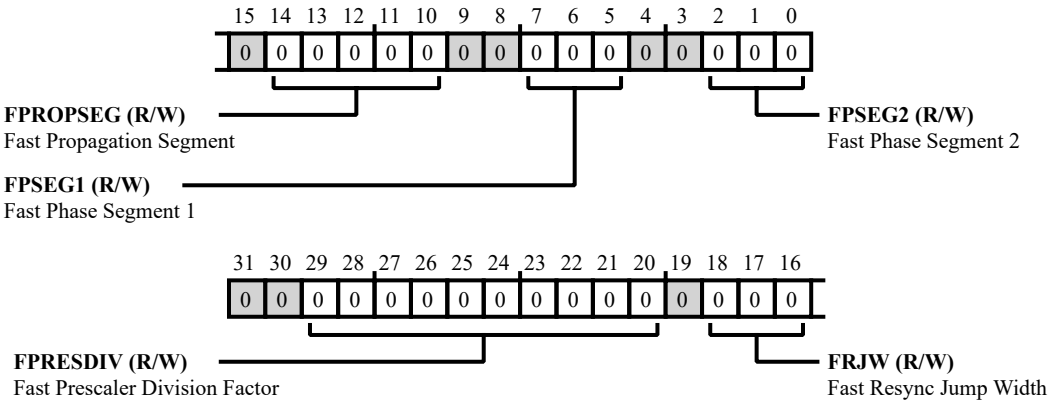


Figure 16-28: CANFD_FD_TIMING Register Diagram

Table 16-46: CANFD_FD_TIMING Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:20 (R/W)	FPRESDIV	Fast Prescaler Division Factor. The CANFD_FD_TIMING.FPRESDIV bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. This is in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. Sclock frequency = PE clock frequency / (FPRESDIV + 1). To minimize errors when processing FD frames, use the same value for the CANFD_FD_TIMING.FPRESDIV and CANFD_CTL1.PRESDIV bit fields The CANFD_FD_TIMING.FPRESDIV bit field can only be written in freeze mode and is blocked by hardware in other modes.

Table 16-46: CANFD_FD_TIMING Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
18:16 (R/W)	FRJW	Fast Resync Jump Width. The `CANFD_FD_TIMING.FRJW` bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Resync Jump Width = FSJW + 1. One time quantum is equal to the Sclock period. The `CANFD_FD_TIMING.FRJW` bit field can only be written in freeze mode and is blocked by hardware in other modes.
14:10 (R/W)	FPROPSEG	Fast Propagation Segment. The `CANFD_FD_TIMING.FPROPSEG` bit field defines the length of the propagation segment in the bit time. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Propagation Segment Time = FPROPSEG x Time-Quanta One time quantum is equal to the Sclock period. The `CANFD_FD_TIMING.FPROPSEG` bit field can only be written in freeze mode and is blocked by hardware in other modes.
7:5 (R/W)	FPSEG1	Fast Phase Segment 1. The `CANFD_FD_TIMING.FPSEG1` bit field defines the length of phase segment 1 in the bit time. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Phase Segment 1 = (FPSEG1 + 1) x Time-Quanta One time quantum is equal to the Sclock period. FPSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes.
2:0 (R/W)	FPSEG2	Fast Phase Segment 2. The `CANFD_FD_TIMING.FPSEG2` bit field defines the length of phase segment 2. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Phase Segment 2 = (FPSEG2 + 1) x Time-Quanta One time quantum is equal to the Sclock period. FPSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes.

CANFD CRC Register

The CANFD_FD_CRC register provides information about the CRC of transmitted messages. The CANFD module uses different CRC polynomials for different frame formats as follows:

- The CRC_15 polynomial is used for all frames in CAN format.
- The CRC_17 polynomial is used for frames in CAN FD format with a data field up to sixteen bytes.
- The CRC_21 polynomial is used for frames in CAN FD format with a data field longer than sixteen bytes.

Each polynomial shown below results in a Hamming distance of 6. The CANFD_FD_CRC register updates when the Tx interrupt flag is asserted.

CRC_15 = 0xC599:$(x^{15}+x^{14}+x^{10}+x^{8}+ x^{7}+x^{4}+x^{3}+1)$

CRC_17 = 0x3685B:$(x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^{6}+x^{4}+x^{3}+x^{1}+1)$

CRC_21 = 0x302899:$(x^{21}+x^{20}+x^{13}+x^{11}+x^{7}+x^{4}+x^{3}+1)$

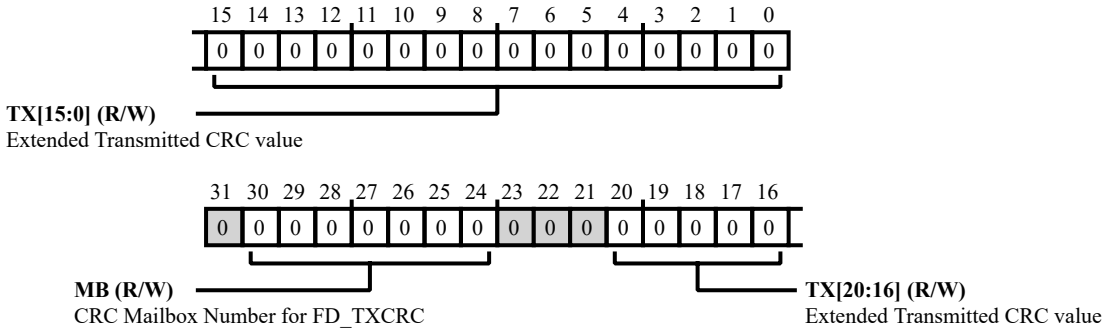


Figure 16-29: CANFD_FD_CRC Register Diagram

Table 16-47: CANFD_FD_CRC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:24 (R/W)	MB	CRC Mailbox Number for FD_TXCRC. The CANFD_FD_CRC.MB bit field indicates the number of the mailbox corresponding to the value in the FD_TXCRC field, for both FD and non-FD frames.

Table 16-47: CANFD_FD_CRC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20:0 (R/W)	TX	Extended Transmitted CRC value. The `CANFD_FD_CRC.TX` bit field contains the CRC value calculated over the most recent transmitted message. There are different CRC polynomials for different frame formats. All frames in CAN format use a 15-bit polynomial, CRC_15. CRC_17 is for frames in CAN FD format with a data field up to sixteen bytes long. The 21-bit polynomial, CRC_21, is for frames in CAN FD format with a data field longer than sixteen bytes. For CRC_15 and CRC_17, the six most significant bits and the four most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as the `CANFD_CRC` register.

CANFD Control Register

The CANFD_FD_CTL register contains control bits for CAN FD operation. It also defines the data size of message buffers allocated in different partitions of RAM (memory blocks).

When 8 bytes of payload is selected, block R0 allocates MB0 to MB31 and block R1 allocates MB32 to MB63.

When more than 8 bytes of payload is selected, the maximum number of MBs in a block is limited as described as follows:

Payload Size => Maximum Number of Message Buffers per RAM Block

8 bytes => 32

16 bytes => 21

32 bytes => 12

64 bytes => 7

Note that one memory block fits exactly 32 MBs with 8 bytes payload. For the other options of payload sizes, empty memory may exist between last MB in a block and the beginning of the next block. This empty memory corresponds to less than one MB, and must not be used.

The contents of CANFD_FD_CTL register are not affected by soft reset.

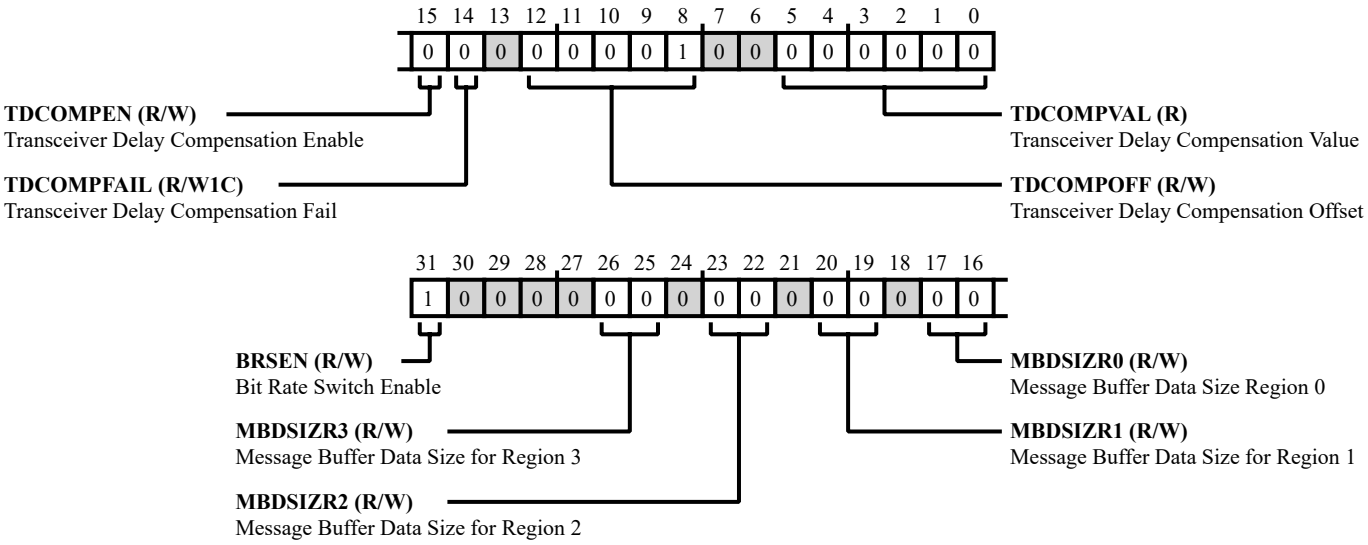


Figure 16-30: CANFD_FD_CTL Register Diagram

Table 16-48: CANFD_FD_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	BRSEN	Bit Rate Switch Enable. The CANFD_FD_CTL.BRSEN bit enables the effect of the bit rate switch (BRS) bit during the data phase of Tx messages. When the CANFD_FD_CTL.BRSEN bit is set, if the BRS bit in the Tx MB is recessive, the CANFD module transmits a frame with bit rate switching. When the CANFD_FD_CTL.BRSEN bit is clear, the BRS bit in the Tx MB has no effect and the CANFD module transmits a frame with the nominal rate. The processor can write the CANFD_FD_CTL.BRSEN bit any time, however, the effect only becomes active when the CAN bus is in the wait for bus idle, bus idle, bus off state, or when the current frame under reception or transmission reaches the inter-frame space. By negating the CANFD_FD_CTL.BRSEN bit, the processor can force all bits in CAN FD messages to be transmitted with a nominal bit rate, despite the value in the BRS bit of the Tx MBs.	
		0	Nominal Rate
		1	Bit Rate Switching
26:25 (R/W)	MBDSIZR3	Message Buffer Data Size for Region 3. The CANFD_FD_CTL.MBDSIZR3 field selects the data size for the region R3 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR3 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer
23:22 (R/W)	MBDSIZR2	Message Buffer Data Size for Region 2. The CANFD_FD_CTL.MBDSIZR2 field selects the data size for the region R2 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR2 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer

Table 16-48: CANFD_FD_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20:19 (R/W)	MBDSIZR1	Message Buffer Data Size for Region 1. The CANFD_FD_CTL.MBDSIZR1 bit field selects the data size for the region R1 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR1 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer
17:16 (R/W)	MBDSIZR0	Message Buffer Data Size Region 0. The CANFD_FD_CTL.MBDSIZR0 bit field selects the data size for the region R0 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR0 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer
15 (R/W)	TDCOMPEN	Transceiver Delay Compensation Enable. The CANFD_FD_CTL.TDCOMPEN bit enables or disables the TDC feature. The CANFD_FD_CTL.TDCOMPEN bit can only be written in freeze mode. Note that when loop-back mode is enabled (CANFD_CTL1.LBEN), the CANFD_FD_CTL.TDCOMPEN bit must be disabled.	
		0	Disable
		1	Enable
14 (R/W1C)	TDCOMPFAIL	Transceiver Delay Compensation Fail. The CANFD_FD_CTL.TDCOMPFAIL bit indicates when the transceiver delay compensation (TDC) mechanism is out of range. When the TDC is out of range it is unable to compensate for the loop delay of the transceiver and successfully compare the delayed received bits to the transmitted ones. The CANFD_FD_CTL.TDCOMPFAIL bit is set the first time the CANFD module detects the out of range condition. To clear the CANFD_FD_CTL.TDCOMPFAIL bit, write a one to it.	
		0	Measured loop delay is in range
		1	Measured loop delay is out of range

Table 16-48: CANFD_FD_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
12:8 (R/W)	TDCOMPOFF	Transceiver Delay Compensation Offset. The `CANFD_FD_CTL.TDCOMPOFF` bit field contains the offset value that must be added to the loop delay of the measured transceiver. This defines the position of the delayed comparison point when bit rate switching is active. The `CANFD_FD_CTL.TDCOMPOFF` bit field can only be written in freeze mode. The bit field value is defined in PE clock periods and must be smaller than the CAN bit duration in the data bit rate. Do not set the `CANFD_FD_CTL.TDCOMPOFF` bit field to zero.
5:0 (R/NW)	TDCOMPVAL	Transceiver Delay Compensation Value. The `CANFD_FD_CTL.TDCOMPVAL` bit field contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the value in the `CANFD_FD_CTL.TDCOMPOFF` bit field. This value is an integer multiple of the PE clock period.

Pretended Networking DLC Filter Register

The CANFD_FLTR_DLC register contains the DLC inside range target values (CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO) for filtering incoming messages. The DLC range is only for payload filtering. The CANFD_FLTR_DLC register can only be written in freeze mode.

Note that when a fixed quantity of data bytes is required, both the CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO bit fields must have the same value; otherwise, a range of DLC is considered for filtering.

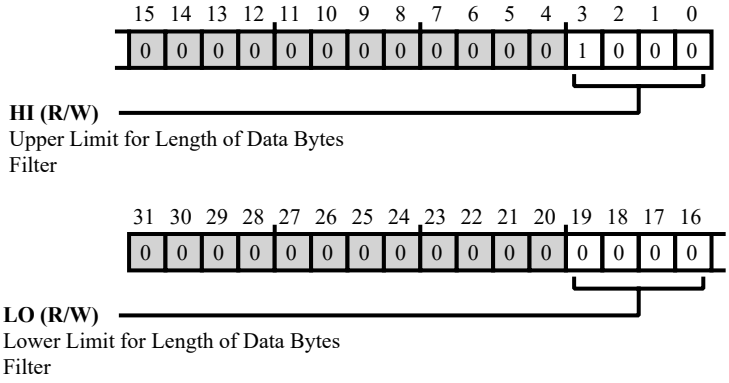


Figure 16-31: CANFD_FLTR_DLC Register Diagram

Table 16-49: CANFD_FLTR_DLC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	LO	Lower Limit for Length of Data Bytes Filter. The CANFD_FLTR_DLC.LO bit field specifies the lower limit on the number of data bytes considered valid for payload comparison. The CANFD_FLTR_DLC.LO bit field is part of payload reception filter.
3:0 (R/W)	HI	Upper Limit for Length of Data Bytes Filter. The CANFD_FLTR_DLC.HI bit field specifies the upper limit on the number of data bytes considered valid for payload comparison. The CANFD_FLTR_DLC.HI bit field is part of payload reception filter.

Pretended Networking ID Filter1 Register

The CANFD_FLTR_ID1 register contains the FLT_ID1 target value and the IDE and RTR target values for filtering the incoming message ID. This register is for equal to, smaller than, greater than comparisons, or as the lower limit value in an ID range detection. The CANFD_FLTR_ID1 register is only written in freeze mode.

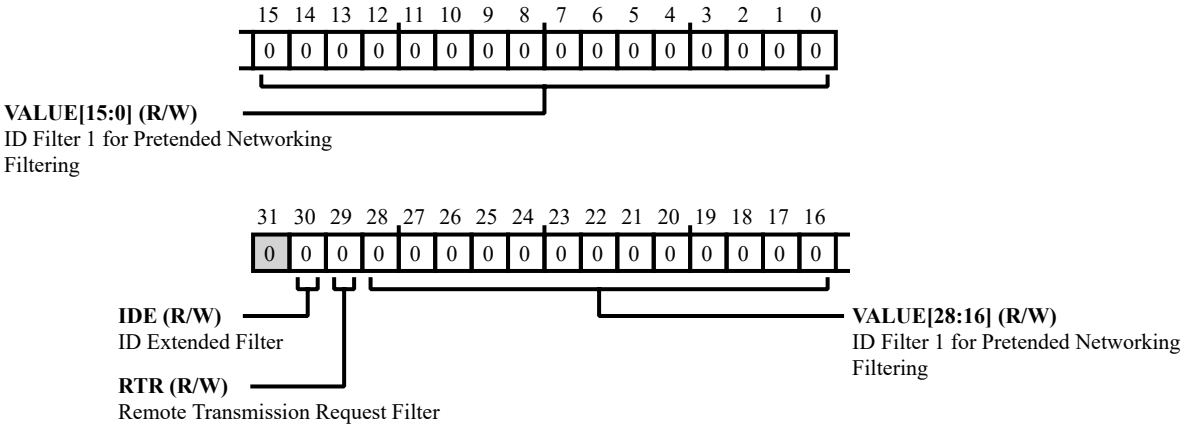


Figure 16-32: CANFD_FLTR_ID1 Register Diagram

Table 16-50: CANFD_FLTR_ID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	IDE	ID Extended Filter. The CANFD_FLTR_ID1.IDE bit identifies if the frame format is standard or extended. It is part of the ID reception filter.	
		0	Accept Standard Frame Format
		1	Accept Extended Frame Format
29 (R/W)	RTR	Remote Transmission Request Filter. The CANFD_FLTR_ID1.RTR bit identifies if the frame is remote. It is part of the ID reception filter.	
		0	Reject remote frame (accept data frame)
		1	Accept remote frame
28:0 (R/W)	VALUE	ID Filter 1 for Pretended Networking Filtering. The CANFD_FLTR_ID1.VALUE bit field defines either the 29 bits of a extended frame format, considering all bits, or the 11 bits of a standard frame format, considering just the 11 leftmost bits.	

Pretended Networking ID Filter2 / IDMask Register

The CANFD_FLTR_ID2_IDMSK register contains the FLT_ID2 target value for use as the upper limit value in ID range detection. When exact ID filtering criteria is selected, the CANFD_FLTR_ID2_IDMSK register is also for storing the ID mask. The CANFD_FLTR_ID2_IDMSK.IDE and CANFD_FLTR_ID2_IDMSK.RTR bits are for ID filtering (exact and range) as part of the ID reception filter.

The CANFD_FLTR_ID2_IDMSK register can only be written in freeze mode.

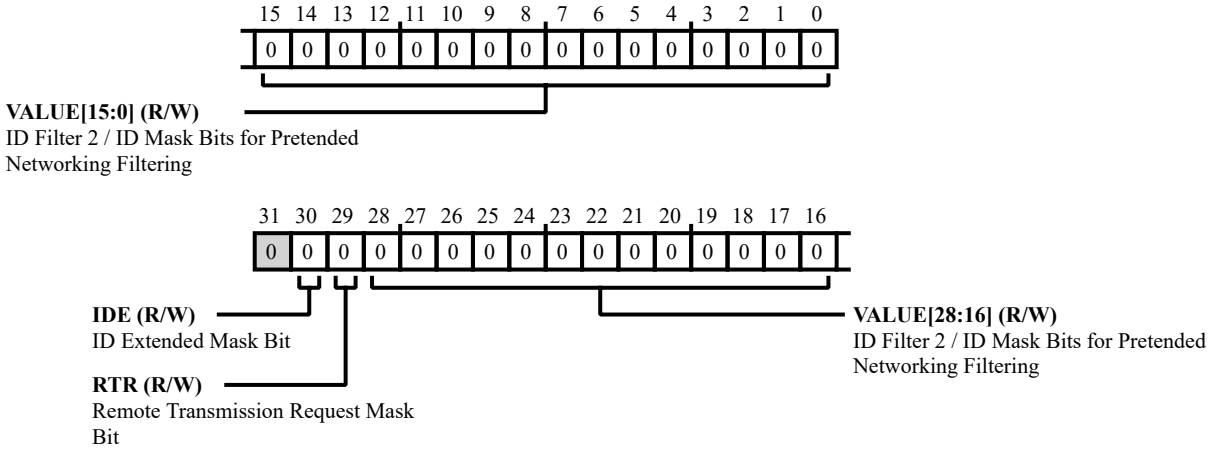


Figure 16-33: CANFD_FLTR_ID2_IDMSK Register Diagram

Table 16-51: CANFD_FLTR_ID2_IDMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	IDE	ID Extended Mask Bit. The CANFD_FLTR_ID2_IDMSK.IDE bit indicates whether the frame format (standard/extended) is used as part of the ID reception filter. If the CANFD_FLTR_ID2_IDMSK.IDE bit is set, the corresponding bit in the filter is checked. If it is cleared, the corresponding bit in the filter is a don't care.	
		0	Don't Care
		1	Checked
29 (R/W)	RTR	Remote Transmission Request Mask Bit. The CANFD_FLTR_ID2_IDMSK.RTR bit indicates if the frame type (data/remote) is part of the ID reception filter. If the CANFD_FLTR_ID2_IDMSK.RTR bit is set, the corresponding bit in the filter is checked. If it is cleared, the corresponding bit in the filter is a don't care.	
		0	Don't Care
		1	Checked

Table 16-51: CANFD_FLTR_ID2_IDMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	VALUE	ID Filter 2 / ID Mask Bits for Pretended Networking Filtering. The `CANFD_FLTR_ID2_IDMSK.VALUE` bit field defines the ID filter value (FLT_ID2). In extended frame format, use all 29 bits (FLT_ID2[28:0]) in the field. In standard frame format, use the upper 11 bits (FLT_ID2[28:18]) and the lower 18 bits (FLT_ID2[17:0]) have no meaning. The `CANFD_FLTR_ID2_IDMSK.VALUE` bit field can also be IDMASK in exact ID filtering to define the mask value. In extended frame format, use all 29 bits (IDMASK[28:0]) in the field. In standard frame format, use the upper 11 bits (IDMASK[28:18]) and the lower 18 bits (IDMASK[17:0]) have no meaning.

Mailbox Interrupt Flag 1 Register

The CANFD_IFLG1 register defines the flags for the 32 message buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding bit in the CANFD_IFLG1 register. If the corresponding bit in the CANFD_IMSK1 register is set, an interrupt is generated. The interrupt flag must be cleared by writing a 1 to it. Writing a zero has no effect.

Before updating the CANFD_CFG.MAXMB field, the processor must service the CANFD_IFLG1 bits whose MB value is greater than the CANFD_CFG.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

There is an exception is when DMA for Rx FIFO is enabled, as described below.

Rx FIFO:

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the CANFD_CFG.RFEN bit is set and the CANFD_CFG.DMAEN bit is cleared, the function of the 8 least significant interrupt flags changes as follows:

- BUF7I, BUF6I, and BUF5I indicate operating conditions of the FIFO.
- BUF0I is used to empty the FIFO.
- BUF4I to BUF1I are reserved.

Before setting the CANFD_CFG.RFEN bit, the processor must service the IFLAG bits asserted in the Rx FIFO region or these IFLAG bits will mistakenly show the related MBs now belonging to the FIFO as having contents to be serviced. When the CANFD_CFG.RFEN bit is negated, the FIFO flags must be cleared. The same care must be taken when a CANFD_CTL2.RFFNUM value is selected extending Rx FIFO filters beyond MB7. For example, when the CANFD_CTL2.RFFNUM value is 0x8, the MB0–23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

The Rx FIFO must be disabled when the CANFD_CFG.FDEN bit is enabled.

Rx FIFO with DMA:

When both the CANFD_CFG.RFEN bit and the CANFD_CFG.DMAEN bits are set (DMA feature for Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I–BUF0I) are changed to support the DMA operation:

- BUF7I, BUF6I, and BUF4I–BUF1I are not used.
- BUF5I indicates operating condition of FIFO.
- BUF5I does not generate a CPU interrupt, but generates a DMA request.
- BUF0I is used to empty FIFO.

The CANFD_IMSK1 register bits in the Rx FIFO region are not considered when the CANFD_CFG.DMAEN bit is asserted. In addition, the processor must not clear the flag BUF5I when DMA is enabled. Before setting the

CANFD_CFG.DMAEN bit, the processor must service the IFLAGs asserted in the Rx FIFO region. When the CANFD_CFG.DMAEN bit is cleared, the FIFO must be empty.

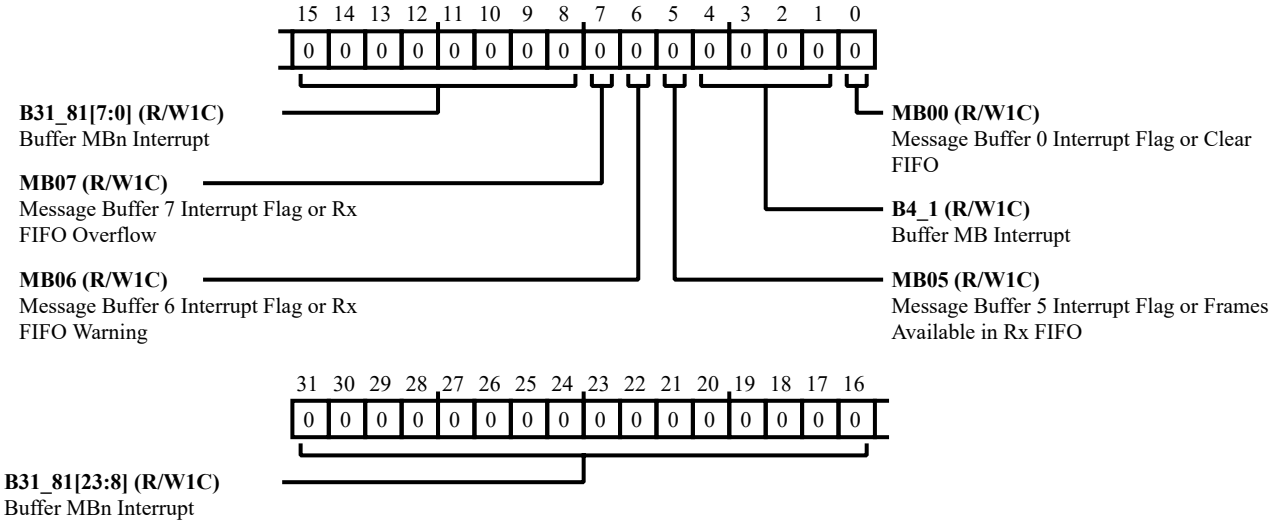


Figure 16-34: CANFD_IFLG1 Register Diagram

Table 16-52: CANFD_IFLG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W1C)	B31_81	Buffer MBn Interrupt. Each bit in the CANFD_IFLG1.B31_81 bit field flags the corresponding CANFD Message Buffer Interrupt for MB31 to MB8. Only the lower 8 bits are available when parameter NUMBER_OF_MB = 16.	
		0	No occurrence of successful transmission or reception
		1	Successful transmission or reception

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W1C)	MB07	Message Buffer 7 Interrupt Flag or Rx FIFO Overflow. The CANFD_IFLG1.MB07 bit flags the CANFD module message buffer interrupt for MB7 or Rx FIFO overflow. When the CANFD_IFLG1.MB07 bit is set and CANFD_CFG.RFEN is disabled, MB7 completed transmission or reception. When the CANFD_IFLG1.MB07 bit is set and CANFD_CFG.RFEN is enabled, there is an Rx FIFO overflow. When the CANFD_IFLG1.MB07 bit is cleared and CANFD_CFG.RFEN is disabled, the corresponding buffer has no occurrence of successfully completed transmission or reception.When the CANFD_IFLG1.MB07 bit is cleared and CANFD_CFG.RFEN is enabled, there is no occurrence of Rx FIFO overflow. When the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled), the CANFD_IFLG1.MB07 bit flags the interrupt for MB7. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB07 bit represents the Rx FIFO overflow. In this case, the CANFD_IFLG1.MB07 bit indicates that a message was lost because the Rx FIFO is full. Note that the CANFD_IFLG1.MB07 bit is not asserted when the Rx FIFO is full and the message was captured by a mailbox. Note that the CANFD_IFLG1.MB07 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.
6 (R/W1C)	MB06	Message Buffer 6 Interrupt Flag or Rx FIFO Warning. The CANFD_IFLG1.MB06 bit flags the CANFD module message buffer interrupt for MB6 or Rx FIFO warning. When the CANFD_IFLG1.MB06 bit is set and CANFD_CFG.RFEN is disabled, MB6 completed transmission or reception. When the CANFD_IFLG1.MB06 bit is set and CANFD_CFG.RFEN is enabled, the Rx FIFO is almost full. When the CANFD_IFLG1.MB06 bit is cleared and CANFD_CFG.RFEN is disabled, the corresponding buffer has no occurrence of successfully completed transmission or reception.When the CANFD_IFLG1.MB06 bit is cleared and CANFD_CFG.RFEN is enabled, the Rx FIFO is not almost full. When the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled), the CANFD_IFLG1.MB06 bit flags the interrupt for MB6. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB06 bit represents the Rx FIFO warning. In this case, BUF6I indicates when the number of unread messages in the Rx FIFO increases from 4 to 5 due to the reception of a new message, meaning that the Rx FIFO is almost full. Note that if the CANFD_IFLG1.MB06 bit is cleared while the number of unread messages is greater than 4, the CANFD_IFLG1.MB06 bit is not asserted again until the number of unread messages in the Rx FIFO decreases to 4 or less. Note that the CANFD_IFLG1.MB06 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1C)	MB05	Message Buffer 5 Interrupt Flag or Frames Available in Rx FIFO. The CANFD_IFLG1.MB05 bit flags the CANFD module message buffer interrupt for MB5 or Rx FIFO frames available. When the CANFD_IFLG1.MB05 bit is set and CANFD_CFG.RFEN is disabled, MB5 completed transmission or reception. When the CANFD_IFLG1.MB05 bit is set and CANFD_CFG.RFEN is enabled, the Rx FIFO has frames available. BUF5I generates a DMA request when both the CANFD_CFG.RFEN and CANFD_CFG.DMAEN bits are set. When the CANFD_IFLG1.MB05 bit is cleared and CANFD_CFG.RFEN is disabled, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_IFLG1.MB05 bit is cleared and CANFD_CFG.RFEN is enabled, no frames are available in the Rx FIFO. When the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled), CANFD_IFLG1.MB05 bit flags the interrupt for MB5. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB05 bit represents the frames available in the Rx FIFO and indicates that at least one frame is available to be read from the Rx FIFO. When the CANFD_CFG.DMAEN bit is set, the CANFD_IFLG1.MB05 bit generates a DMA request and the processor must not clear the CANFD_IFLG1.MB05 bit by writing one to it. Note that the CANFD_IFLG1.MB05 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.	
4:1 (R/W1C)	B4_1	Buffer MB Interrupt. The CANFD_IFLG1.B4_1 bit field indicates whether the corresponding buffer has successfully completed transmission or reception (when CAN_MCR.RFEN=0). When CAN_MCR.RFEN is cleared (=0) (Rx FIFO disabled), the BUF4TO1I flags the interrupts for MB4 to MB1. When CAN_MCR.RFEN is set (=1)(Rx FIFO enabled), BUF4TO1I flags are reserved. Note: BUF4TO1I are cleared by the CANFD module whenever CAN_MCR.RFEN is changed by a CPU write.	
		0	No occurrence of successful completed transmission or reception
		1	Successful transmission or reception

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W1C)	MB00	Message Buffer 0 Interrupt Flag or Clear FIFO. The CANFD_IFLG1.MB00 bit flags the interrupt for MB0, when the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled). When the CANFD_IFLG1.MB00 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB00 bit triggers the clear FIFO operation. This operation empties FIFO contents. Before performing this operation, the processor must service all FIFO related IFLAGs. When the CANFD_CFG.DMAEN bit is set (DMA is enabled), this operation also clears the CANFD_IFLG1.MB05 flag and consequently aborts the DMA request. The clear FIFO operation occurs when the processor writes a one to the CANFD_IFLG1.MB00 bit, which is only allowed in freeze mode and is blocked by hardware in other conditions.

Mailbox Interrupt Flag 2 Register

The CANFD_IFLG2 register defines the flags for the 32 message buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective bit in the CANFD_IFLG2 bit. If the corresponding bit is set, an interrupt is generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating the CANFD_CFG.MAXMB field, the processor must service the CANFD_IFLG2 bits whose MB value is greater than the CANFD_CFG.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

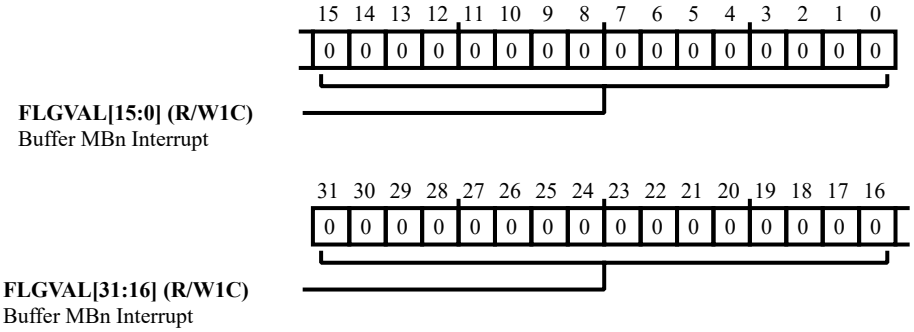


Figure 16-35: CANFD_IFLG2 Register Diagram

Table 16-53: CANFD_IFLG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W1C)	FLGVAL	Buffer MBn Interrupt. The CANFD_IFLG2.FLGVAL bit field flags the corresponding buffer (MB63 to MB32) indicating whether there has been a successfully completed transmission or reception.	
		0	No occurrence of successful transmission or reception
		1	Successful transmission or reception

Mailbox Interrupt Mask 1 Register

The CANFD_IMSK1 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB31 to MB. The CANFD_IMSK1 register contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding bit in the register).

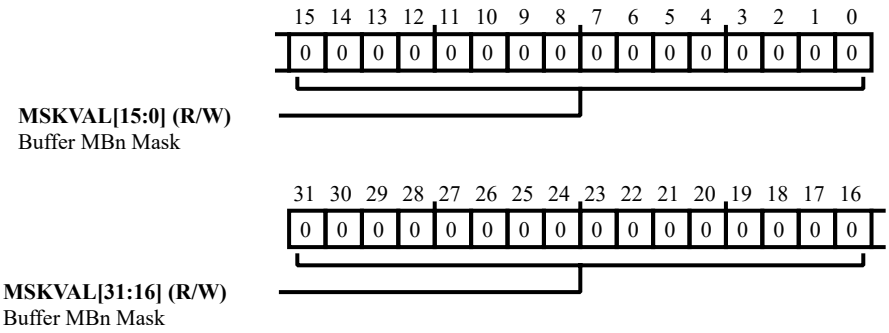


Figure 16-36: CANFD_IMSK1 Register Diagram

Table 16-54: CANFD_IMSK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	MSKVAL	Buffer MBn Mask. Each CANFD_IMSK1.MSKVAL bit enables or disables the corresponding CANFD Message Buffer Interrupt for MB31 to MB0. Note: Setting or clearing a bit in CAN_IMASK1 can assert or negate an interrupt request if the corresponding CAN_IFLAG1 bit is set.	
		0	Disabled
		1	Enabled

Mailbox Interrupt Mask 2 Register

The CANFD_IMSK2 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB63 to MB32. The CANFD_IMSK2 register contains one interrupt mask bit per buffer, enabling the processor to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding bit in the register).

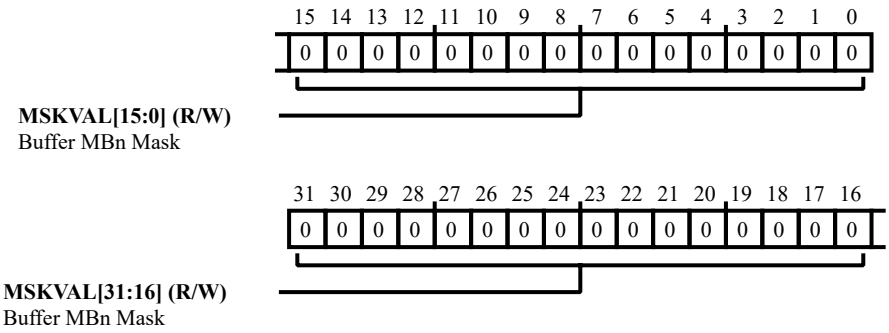


Figure 16-37: CANFD_IMSK2 Register Diagram

Table 16-55: CANFD_IMSK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSKVAL	Buffer MBn Mask. Each CANFD_IMSK2.MSKVAL bit enables or disables the corresponding CANFD Message Buffer Interrupt for MB63 to MB32. : Note: Setting or clearing a bit in CAN_IMASK2 can assert or negate an interrupt request if the corresponding CAN_IFLAG2 bit is set.

Module Configuration Register

The CANFD_CFG register configures the operation modes and settings for the CANFD module. This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

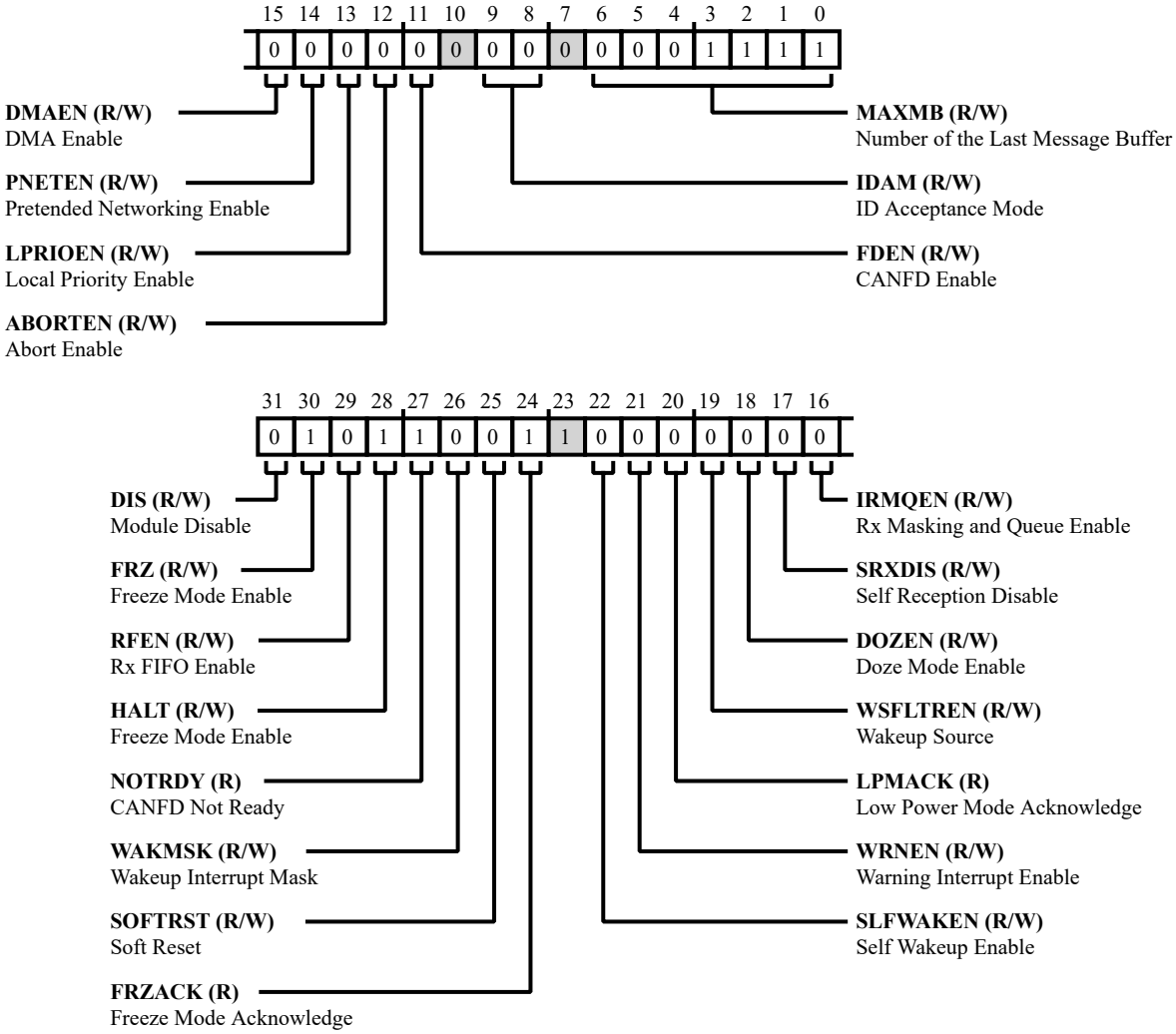


Figure 16-38: CANFD_CFG Register Diagram

Table 16-56: CANFD_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	DIS	Module Disable. The CANFD_CFG.DIS bit disables or enables the CANFD module. When the CANFD_CFG.DIS bit is set, the CANFD module disables the clocks to the PE and CHI submodules. This bit is not affected by a soft reset.	
		0	Enable
		1	Disable
30 (R/W)	FRZ	Freeze Mode Enable. The CANFD_CFG.FRZ bit specifies the CANFD module behavior when the CANFD_CFG.HALT bit is set or when there is a debug mode request at the chip level. When the CANFD_CFG.FRZ bit is set, the CANFD module enters freeze mode. Negation of the CANFD_CFG.FRZ bit causes the CANFD module to exit from freeze mode. The CANFD_CFG.FRZ bit is set by hardware when a non-correctable error is detected and the CANFD_MEC.NCERRFRZEN bit is set.	
		0	Disable
		1	Enable
29 (R/W)	RFEN	Rx FIFO Enable. The CANFD_CFG.RFEN bit enables the Rx FIFO. When the CANFD_CFG.RFEN bit is set, do not use MBs 0-5 for normal reception and transmission. The corresponding memory region (0x80-0xDC) is used by the FIFO engine and additional MBs (up to 32, depending on the CANFD_CTL2.RFFNUM bit setting) are used as Rx FIFO ID filter table elements. The CANFD_CFG.RFEN bit also impacts the minimum number of peripheral clocks per CAN bit. The CANFD_CFG.RFEN bit can only be written in freeze mode because it is blocked by hardware in other modes. Note that the CANFD_CFG.RFEN bit cannot be set when the CANFD_CFG.FDEN bit is set.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	HALT	Freeze Mode Enable. Setting the CANFD_CFG.HALT bit puts the CANFD module into freeze mode. The processor clears the CANFD_CFG.HALT bit after initializing the message buffers, the CANFD_CTL1 register, and the CANFD_CTL2 register. The CANFD module does not receive or transmit data before the CANFD_CFG.HALT bit is cleared. The CANFD module cannot enter freeze mode while it is in a low-power mode. The CANFD_CFG.HALT bit is set by hardware when a non-correctable error is detected and the CANFD_MEC.NCERRFRZEN bit is set.	
		0	Disable
		1	Enable
27 (R/NW)	NOTRDY	CANFD Not Ready. The CANFD_CFG.NOTRDY bit is a read-only bit that indicates the CANFD module is either in module disable, doze, stop, or freeze mode. The CANFD_CFG.NOTRDY bit is negated once the CANFD module exits these modes. Soft reset does not affect the CANFD_CFG.NOTRDY bit.	
		0	Normal, Listen-Only, or Loop-Back Mode
		1	Module Disable, Doze, Stop, or Freeze Mode
26 (R/W)	WAKMSK	Wakeup Interrupt Mask. The CANFD_CFG.WAKMSK bit enables the wakeup interrupt using the self wakeup mechanism.	
		0	Disable
		1	Enable
25 (R/W)	SOFTRST	Soft Reset. When the CANFD_CFG.SOFTRST bit is set, the internal state machines and some of the memory mapped registers of the CANFD module reset. For the processor to directly assert a soft reset, write to the CANFD_CFG.SOFTRST bit. A soft reset can also occur when a global soft reset is requested at the chip level. Soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, so it takes time to fully propagate the effect. The CANFD_CFG.SOFTRST bit remains set while reset is pending and is automatically negated when the reset completes. Poll the CANFD_CFG.SOFTRST bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low-power mode. Remove the CANFD module from low-power mode before applying soft reset. A soft reset does not affect the CANFD_CFG.SOFTRST bit.	

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	FRZACK	Freeze Mode Acknowledge. The CANFD_CFG.FRZACK bit is a read-only bit that indicates the CANFD module is in freeze mode and the prescaler is stopped. A freeze mode request is only granted after the current transmission or reception processes completes. Poll the CANFD_CFG.FRZACK bit in software to know when the CANFD module enters freeze mode. If a freeze mode request is negated, the CANFD_CFG.FRZACK bit is negated after the CANFD module prescaler is running again. If freeze mode is requested while the CANFD module is in a low-power mode, the CANFD_CFG.FRZACK bit sets only when the module exits the low-power mode. Soft reset does not affect the CANFD_CFG.FRZACK bit. Note that the CANFD_CFG.FRZACK bit is set within 178 CAN bits of the freeze mode request. It is negated within 2 CAN bits after a freeze mode request removal.	
		0	No Freeze Mode Prescaler is running.
		1	Freeze Mode The prescaler stopped.
22 (R/W)	SLFWAKEN	Self Wakeup Enable. The CANFD_CFG.SLFWAKEN bit enables the self wakeup feature when the CANFD module is in a low-power mode other than module disable. When the self wakeup feature is enabled, the CANFD module monitors the bus for a wakeup event (a recessive-to-dominant transition). If a wake up event is detected during doze mode, the CANFD module requests resumption of the clocks and when the CANFD_CFG.SLFWAKEN bit is enabled generates a wakeup interrupt to the processor. If a wakeup event is detected during stop mode, the CANFD module, when the CANFD_CFG.SLFWAKEN bit is enabled, generates a wake up interrupt to the processor so that it can exit stop mode globally and the CANFD module can request to resume the clocks. When the CANFD module is in a low-power mode other than module disable, the CANFD_CFG.SLFWAKEN bit cannot be written as it is blocked by hardware. Disable self wakeup when pretended networking mode is set.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	WRNEN	Warning Interrupt Enable. The CANFD_CFG.WRNEN bit enables the generation of the TWRNINT and RWRNINT flags in the CANFD_ESR1 register. If the CANFD_CFG.WRNEN bit is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters. No warning interrupt is generated. The CANFD_CFG.WRNEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
20 (R/NW)	LPMACK	Low Power Mode Acknowledge. The CANFD_CFG.LPMACK bit is a read-only bit that indicates that the CANFD module is in a low-power mode (module disable, doze, or stop). The module can only enter a low-power mode when all current transmission or reception processes have finished. Poll the CANFD_CFG.LPMACK bit to know when the CANFD module is in low-power mode. The CANFD_CFG.LPMACK bit is not affected by soft reset. Note that the CANFD_CFG.LPMACK bit is set within 180 CAN bits of the low-power mode request by the processor. It is negated within 2 CAN bits after removing the low-power mode request. When the CANFD module is in pretended networking mode, the CANFD_CFG.LPMACK bit is negated within 180 CAN bits from the removal of the low-power mode request.	
		0	Disable Not in low-power mode
		1	Enable In low-power mode
19 (R/W)	WSFLTREN	Wakeup Source. The CANFD_CFG.WSFLTREN bit controls whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wakeup. When enabled, the CANFD module uses the filtered Rx input to detect recessive-to-dominant edges on the CAN bus. When disabled the CANFD module uses the unfiltered Rx input to detect recessive-to-dominant edges on the CAN bus. The CANFD_CFG.WSFLTREN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	DOZEN	Doze Mode Enable. The CANFD_CFG.DOZEN bit controls whether the CANFD module can enter low-power mode when doze mode is requested at the chip level. When the CANFD_CFG.DOZEN bit is set the CANFD module can enter low-power mode and when it is cleared the module cannot. The CANFD_CFG.DOZEN bit is automatically reset when the CANFD module wakes up from doze mode upon detecting activity on the CAN bus (CANFD_CFG.SLFWAKEN enabled).	
		0	Disable
		1	Enable
17 (R/W)	SRXDIS	Self Reception Disable. The CANFD_CFG.SRXDIS bit controls whether the CANFD module can receive frames transmitted from itself. When the CANFD_CFG.SRXDIS bit is set, frames transmitted by the CANFD module are not stored in any MB, regardless of whether the MB is programmed with an ID that matches the transmitted frame. No interrupt flag or interrupt signal is generated due to the frame reception. The CANFD_CFG.SRXDIS bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Enable
		1	Disable
16 (R/W)	IRMQEN	Rx Masking and Queue Enable. The CANFD_CFG.IRMQEN bit controls whether the Rx matching process is based on individual masking and queue or on a masking scheme with the CANFD_RX_MB_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers. The CANFD_CFG.IRMQEN bit can only be written in freeze mode and is blocked by hardware in other modes. For backward compatibility, when CANFD_CFG.IRMQEN is disabled, reading a C/S word locks the MB even when it is empty.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	DMAEN	DMA Enable. The CANFD_CFG.DMAEN bit enables or disables the DMA feature. The DMA feature is only for use with the Rx FIFO and the CANFD_CFG.RFEN bit must be set. When both the CANFD_CFG.DMAEN and CANFD_CFG.RFEN bits are set, CANFD_IFLG1.MB05 generates the DMA request and no Rx FIFO interrupt is generated. The CANFD_CFG.DMAEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
14 (R/W)	PNETEN	Pretended Networking Enable. The CANFD_CFG.PNETEN bit enables the pretended networking mode. When pretended networking is enabled, the PE submodule is kept operational in doze mode and stop mode. It is able to process Rx message filtering as defined by the pretended networking configuration registers. The CANFD_CFG.PNETEN bit can be only be written in freeze mode.	
		0	Disable
		1	Enable
13 (R/W)	LPRIOEN	Local Priority Enable. The CANFD_CFG.LPRIOEN bit controls whether the local priority feature is enabled and is provided for backward compatibility with legacy applications. Local priority expands the arbitration ID. With this expanded ID concept, the arbitration process is based on the full 32-bit word, but the actual transmitted ID still has 11 bits for standard frames and 29 bits for extended frames. The CANFD_CFG.LPRIOEN bit can only be written only in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	ABORTEN	Abort Enable. The CANFD_CFG.ABORTEN bit enables the Tx abort mechanism. The Tx abort mechanism ensures a safe procedure for aborting a pending transmission, so that no frame is sent on the CAN bus without notification. The CANFD_CFG.ABORTEN bit can only be written only in freeze mode and is blocked by hardware in other modes. Note that when the CANFD_CFG.ABORTEN bit is set, the abort mechanism must only be used for updating mailboxes configured for transmission. Writing the abort code into Rx mailboxes can cause unpredictable results when the CANFD_CFG.ABORTEN bit is enabled.	
		0	Disable
		1	Enable
11 (R/W)	FDEN	CANFD Enable. The CANFD_CFG.FDEN bit enables CAN with Flexible Data rate (CAN FD) operation. When the CANFD_CFG.FDEN bit is enabled, the CANFD module can receive and transmit messages in both CAN FD and CAN 2.0 formats. When it is disabled, the CANFD module can only receive and transmit messages in CAN 2.0 format. The CANFD_CFG.FDEN bit can only be written in freeze mode only. Note that the CANFD_CFG.RFEN bit cannot be set if the CANFD_CFG.FDEN bit is set.	
		0	Disable
		1	Enable
9:8 (R/W)	IDAM	ID Acceptance Mode. The CANFD_CFG.IDAM bit field identifies the format of the Rx FIFO ID filter table elements (ID acceptance mode). All elements of the table are configured at the same time by the CANFD_CFG.IDAM bit field (they are all the same format). The CANFD_CFG.IDAM bit field can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Format A. There is one full ID (standard and extended) per ID filter table element.
		1	Format B. There are two full standard IDs or two partial 14-bit (standard and extended) IDs per ID filter table element.
		2	Format C. There are four partial 8-bit standard IDs per ID filter table element.
		3	Format D. All frames are rejected.

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	MAXMB	Number of the Last Message Buffer. The CANFD_CFG.MAXMB bit field defines the number of the last message buffer that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 message buffer configuration. The CANFD_CFG.MAXMB bit field can only be written in freeze mode and is blocked by hardware in other modes. Note that the CANFD_CFG.MAXMB bit field must be programmed with a value smaller than or equal to the number of available message buffers. Additionally, the definition of the CANFD_CFG.MAXMB bit value must take into account the region of MBs occupied by Rx FIFO and the ID filters table space defined by the CANFD_CTL2.RFFNUM bit. The CANFD_CFG.MAXMB bit field impacts the minimum number of peripheral clocks per CAN bit.

Memory Error Control Register

The CANFD_MEC register contains control bits for memory error detection and correction (ECC).

Note that when the CANFD_CTL2.ECRWREN bit is disabled, writes to the CANFD_MEC register are blocked, with the exception of the CANFD_MEC.ECRWRDIS bit.

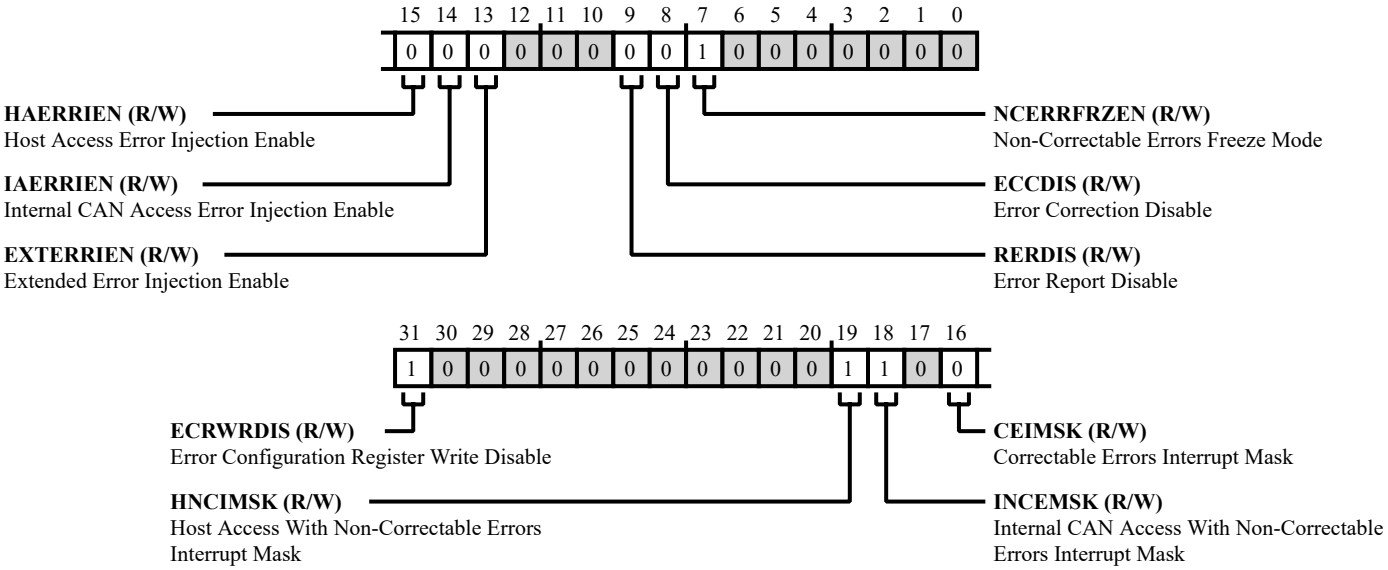


Figure 16-39: CANFD_MEC Register Diagram

Table 16-57: CANFD_MEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ECRWRDIS	Error Configuration Register Write Disable. The CANFD_MEC.ECRWRDIS bit disables writes to the CANFD_MEC register. The CANFD_MEC.ECRWRDIS bit is automatically set to 1 (disabled) when the CANFD_CTL2.ECRWREN bit is enabled. See the protocol described in the Detection and Correction of Memory Errors section of the CANFD chapter.	
		0	Enable
		1	Disable
19 (R/W)	HNCIMSK	Host Access With Non-Correctable Errors Interrupt Mask. The CANFD_MEC.HNCIMSK bit enables the interrupt in case of non-correctable errors detected in memory reads issued by the host.	
		0	Disable
		1	Enable

Table 16-57: CANFD_MEC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	INCEMSK	Internal CAN Access With Non-Correctable Errors Interrupt Mask. The `CANFD_MEC.INCEMSK` bit enables the interrupt in case of non-correctable errors detected in memory reads issued by the CAN internal processes.	
		0	Disable
		1	Enable
16 (R/W)	CEIMSK	Correctable Errors Interrupt Mask. The `CANFD_MEC.CEIMSK` bit enables the interrupt in case of correctable errors detected in memory reads issued by the host or CANFD module internal processes.	
		0	Disable
		1	Enable
15 (R/W)	HAERRIEN	Host Access Error Injection Enable. The `CANFD_MEC.HAERRIEN` bit enables the injection of errors for memory reads issued by the host.	
		0	Disable
		1	Enable
14 (R/W)	IAERRIEN	Internal CAN Access Error Injection Enable. The `CANFD_MEC.IAERRIEN` bit enables the injection of errors for memory reads issued by the CAN internal processes.	
		0	Disable
		1	Enable
13 (R/W)	EXTERRIEN	Extended Error Injection Enable. Memory accesses performed by internal CANFD processes are 64-bit accesses. The `CANFD_MEC.EXTERRIEN` bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. When the `CANFD_MEC.EXTERRIEN` bit is enabled, the error injection is applied to the 64-bit word. When the bit is disabled, error injection is only applied to the 32-bit word. See the descriptions of the `CANFD_ERR_IDP` and `CANFD_ERR_IPP` registers.	
		0	Disable
		1	Enable

Table 16-57: CANFD_MEC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	RERDIS	Error Report Disable. The `CANFD_MEC.RERDIS` bit disables the update of the error report registers. When the error report registers are disabled, the update of error-related flags and the generation of bus transfer errors are still active. Note that when reading the report registers, the `CANFD_MEC.RERDIS` bit must be set to ensure coherence on the consecutive register reads.	
		0	Enable
		1	Disable
8 (R/W)	ECCDIS	Error Correction Disable. The `CANFD_MEC.ECCDIS` bit disables the memory detection and correction mechanism. In addition to disabling the error report mechanism, when the `CANFD_MEC.ECCDIS` bit is set, the update of the error-related flags and generation of bus transfer errors also stops. The parity bits continue to be calculated and written into memory on write transactions.	
		0	Enable
		1	Disable
7 (R/W)	NCERRFRZEN	Non-Correctable Errors Freeze Mode. The `CANFD_MEC.NCERRFRZEN` bit determines the response when a non-correctable error is detected in a memory read performed by CANFD module internal processes. Enabling the `CANFD_MEC.NCERRFRZEN` bit puts the CANFD module in freeze mode, preventing corrupted data from being treated as valid by CANFD internal processes. Disabling the bit maintains normal operation.	
		0	Disable
		1	Enable

Pretended Networking Payload Low Filter2 Register

The CANFD_FLTR_DATA1_HI register contain Payload Filter 1 high order bits of the target value. This is for filtering the incoming message payload in pretended networking mode. It is for =, ≤, or ≥ to comparisons; or as the lower limit value in payload range detection.

The CANFD_FLTR_DATA1_HI register can only be written in freeze mode.

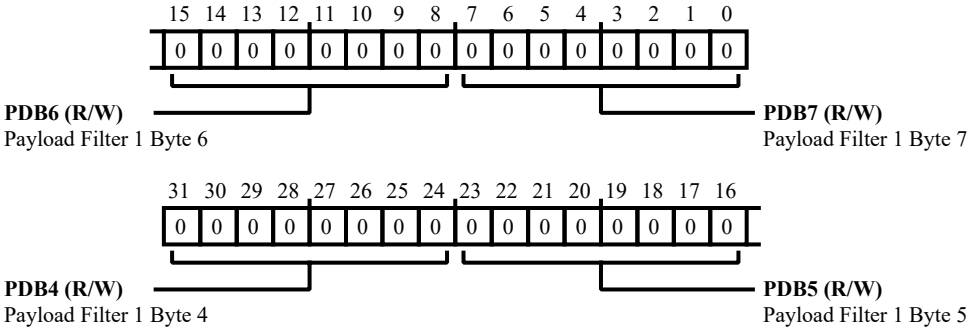


Figure 16-40: CANFD_FLTR_DATA1_HI Register Diagram

Table 16-58: CANFD_FLTR_DATA1_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB4	Payload Filter 1 Byte 4. The CANFD_FLTR_DATA1_HI.PDB4 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 4.
23:16 (R/W)	PDB5	Payload Filter 1 Byte 5. The CANFD_FLTR_DATA1_HI.PDB5 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 5.
15:8 (R/W)	PDB6	Payload Filter 1 Byte 6. The CANFD_FLTR_DATA1_HI.PDB6 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 6.
7:0 (R/W)	PDB7	Payload Filter 1 Byte 7. The CANFD_FLTR_DATA1_HI.PDB7 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 7.

Pretended Networking Payload Low Filter1 Register

The CANFD_FLTR_DATA1_LO register contain Payload Filter 1 low order bits of the target value. This is for filtering the incoming message payload in pretended networking mode. It is for =, ≤, or ≥ to comparisons; or as the lower limit value in payload range detection.

The CANFD_FLTR_DATA1_LO register can only be written in freeze mode.

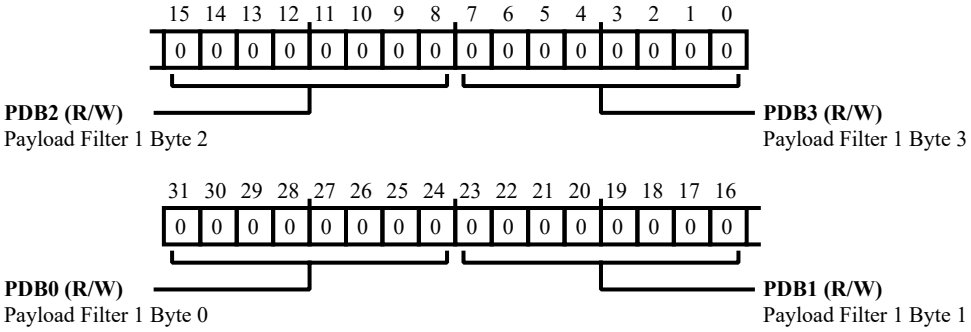


Figure 16-41: CANFD_FLTR_DATA1_LO Register Diagram

Table 16-59: CANFD_FLTR_DATA1_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB0	Payload Filter 1 Byte 0. The CANFD_FLTR_DATA1_LO.PDB0 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 0.
23:16 (R/W)	PDB1	Payload Filter 1 Byte 1. The CANFD_FLTR_DATA1_LO.PDB1 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 1.
15:8 (R/W)	PDB2	Payload Filter 1 Byte 2. The CANFD_FLTR_DATA1_LO.PDB2 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 2.
7:0 (R/W)	PDB3	Payload Filter 1 Byte 3. The CANFD_FLTR_DATA1_LO.PDB3 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 3.

Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register

The CANFD_FLTR_DATA2_DMSK_HI register has two functions:

It contains the high order bits for the Payload Filter 2 value for use as the upper limit in payload range detection.

When exact payload filtering criteria is selected, the CANFD_FLTR_DATA2_DMSK_HI register is the payload mask.

Otherwise, the CANFD_FLTR_DATA2_DMSK_HI register is not used. It can only be written in freeze mode.

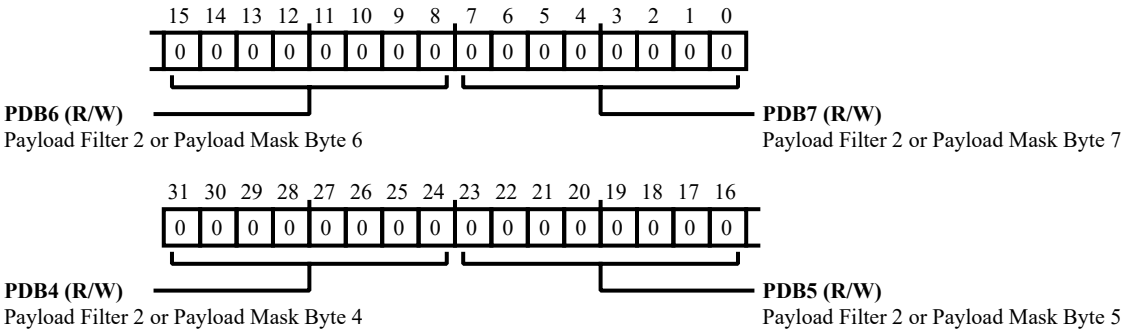


Figure 16-42: CANFD_FLTR_DATA2_DMSK_HI Register Diagram

Table 16-60: CANFD_FLTR_DATA2_DMSK_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB4	Payload Filter 2 or Payload Mask Byte 4. The CANFD_FLTR_DATA2_DMSK_HI.PDB4 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 4.
23:16 (R/W)	PDB5	Payload Filter 2 or Payload Mask Byte 5. The CANFD_FLTR_DATA2_DMSK_HI.PDB5 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 5.
15:8 (R/W)	PDB6	Payload Filter 2 or Payload Mask Byte 6. The CANFD_FLTR_DATA2_DMSK_HI.PDB6 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 6.
7:0 (R/W)	PDB7	Payload Filter 2 or Payload Mask Byte 7. The CANFD_FLTR_DATA2_DMSK_HI.PDB7 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 7.

Pretended Networking Payload Low Filter2 / Payload Low Mask Register

The CANFD_FLTR_DATA2_DMSK_LO register has two functions:

It contains the low order bits for the Payload Filter 2 value for use as the upper limit in payload range detection.

When exact payload filtering criteria is selected, the CANFD_FLTR_DATA2_DMSK_LO register is the payload mask.

Otherwise, the CANFD_FLTR_DATA2_DMSK_LO register is not used. It can only be written in freeze mode.

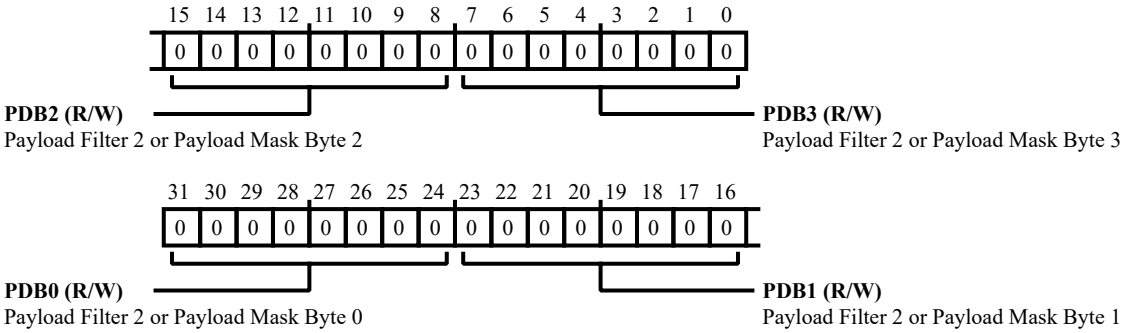


Figure 16-43: CANFD_FLTR_DATA2_DMSK_LO Register Diagram

Table 16-61: CANFD_FLTR_DATA2_DMSK_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB0	Payload Filter 2 or Payload Mask Byte 0. The CANFD_FLTR_DATA2_DMSK_LO.PDB0 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 0.
23:16 (R/W)	PDB1	Payload Filter 2 or Payload Mask Byte 1. The CANFD_FLTR_DATA2_DMSK_LO.PDB1 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 1.
15:8 (R/W)	PDB2	Payload Filter 2 or Payload Mask Byte 2. The CANFD_FLTR_DATA2_DMSK_LO.PDB2 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 2.
7:0 (R/W)	PDB3	Payload Filter 2 or Payload Mask Byte 3. The CANFD_FLTR_DATA2_DMSK_LO.PDB3 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 3.

Error Report Address Register

The CANFD_ERR_RADDR register reports the address of an access with an error (correctable or non-correctable) and reports the identification of the source of that access.

The address is 32-bit aligned. Non-aligned accesses, where the CANFD_ERRADDR[1:0] bits are non-zero, are reported with the address aligned and the data in the CANFD_ERR_RDAT register shifted accordingly. If there is an error in an access that is larger than 32 bits (as performed by CANFD internal processes), the address of the 32-bit word with the error is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.

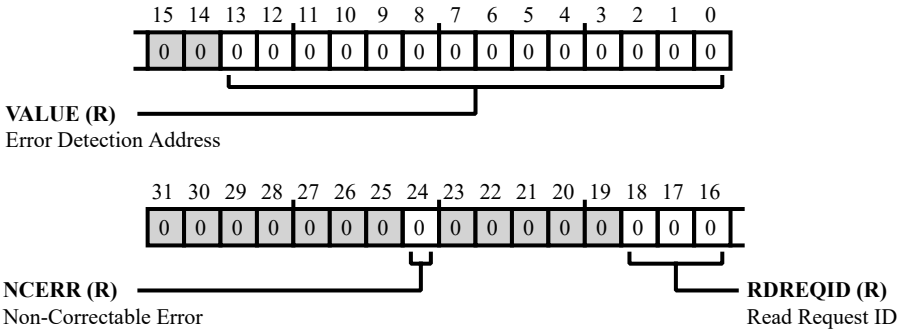


Figure 16-44: CANFD_ERR_RADDR Register Diagram

Table 16-62: CANFD_ERR_RADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	NCERR	Non-Correctable Error. The CANFD_ERR_RADDR.NCERR bit indicates that a report is due to an non-correctable error. When the CANFD_ERR_RADDR.NCERR bit is set, it is reporting a non-correctable error. When it is cleared, it is reporting a correctable-error.	
18:16 (R/NW)	RDREQID	Read Request ID. The CANFD_ERR_RADDR.RDREQID bit field identifies the details of a memory read request.	
		0	Move-Out CANFD Access
		1	Move-In CANFD Access
		2	TX Arbitration
		3	Rx Matching
		4	Move-Out Host Access
		5	Reserved
		6	Reserved
		7	Reserved

Table 16-62: CANFD_ERR_RADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
13:0 (R/NW)	VALUE	Error Detection Address. The `CANFD_ERR_RADDR.VALUE` bit field contains the address where the error was detected. See the description of the Error Injection Address Register (`CANFD_ERR_IADDR`).

Error Report Data Register

The CANFD_ERR_RDAT register reports the raw data (unmodified by the ECC logic correction) read from the memory with the error. The value reported does not represent the transient values of the BUSY bit when reading a message buffer.

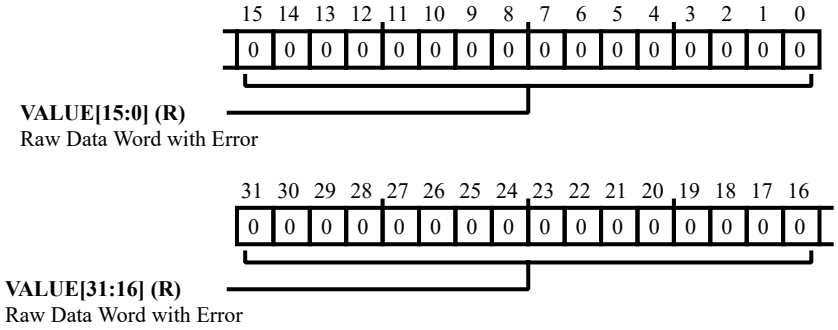


Figure 16-45: CANFD_ERR_RDAT Register Diagram

Table 16-63: CANFD_ERR_RDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Raw Data Word with Error. The CANFD_ERR_RDAT.VALUE bit field contains the raw data word read from memory with error. This data word is unmodified by the correction performed by ECC logic and does not represent the transient values of the BUSY bit when reading the message buffer.

Error Report Syndrome Register

The CANFD_ERR_RSYN register holds the syndrome detected in a memory read with an error. It reports the bytes that are read in the 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error. The CANFD_ERR_RSYN.SYND3 bit field corresponds to the most significant byte in the data word read from memory and the CANFD_ERR_RSYN.SYND0 bit field corresponds to the least significant.

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated and reported.

See the Detection and Correction of Memory Errors section of the CANFD chapter for more details.

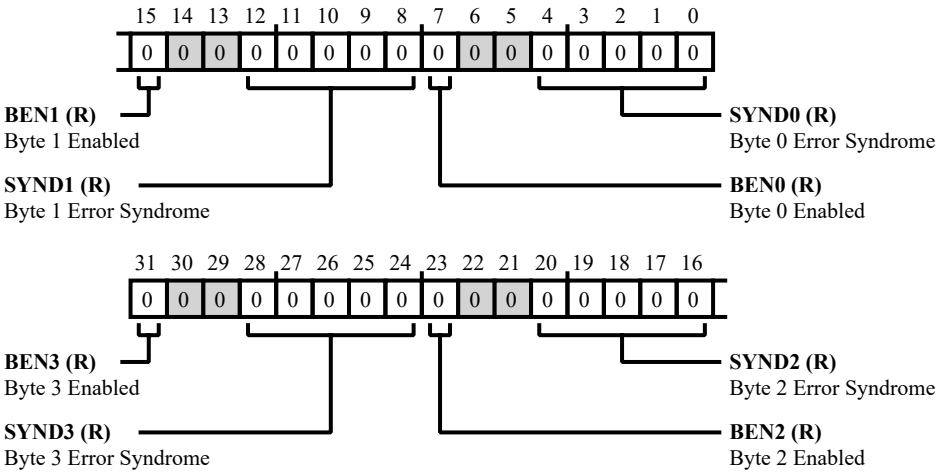


Figure 16-46: CANFD_ERR_RSYN Register Diagram

Table 16-64: CANFD_ERR_RSYN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	BEN3	Byte 3 Enabled. The CANFD_ERR_RSYN.BEN3 bit indicates if byte 3 (most significant) was read. If the CANFD_ERR_RSYN.BEN3 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
28:24 (R/NW)	SYND3	Byte 3 Error Syndrome. The CANFD_ERR_RSYN.SYND3 bit field contains the error syndrome for byte 3. The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND3 bit field.	

Table 16-64: CANFD_ERR_RSYN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/NW)	BEN2	Byte 2 Enabled. The CANFD_ERR_RSYN.BEN2 bit indicates if byte 2 was read. If the CANFD_ERR_RSYN.BEN2 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
20:16 (R/NW)	SYND2	Byte 2 Error Syndrome. The CANFD_ERR_RSYN.SYND2 bit field contains the error syndrome for byte 2. The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND2 bit field.	
15 (R/NW)	BEN1	Byte 1 Enabled. The CANFD_ERR_RSYN.BEN1 bit indicates if byte 1 was read. If the CANFD_ERR_RSYN.BEN1 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
12:8 (R/NW)	SYND1	Byte 1 Error Syndrome. The CANFD_ERR_RSYN.SYND1 bit field contains the error syndrome for byte 1. The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND1 bit field.	
7 (R/NW)	BEN0	Byte 0 Enabled. The CANFD_ERR_RSYN.BEN0 bit indicates if byte 2 (least significant) was read. If the CANFD_ERR_RSYN.BEN0 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
4:0 (R/NW)	SYND0	Byte 0 Error Syndrome. The CANFD_ERR_RSYN.SYND0 bit field contains the error syndrome for byte 0 (least significant). The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND0 bit field.	

Receive Mailbox14 Mask Register

The CANFD_RX_14_MSK register is located in RAM.

When the CANFD_CFG.IRMQEN bit is asserted, the CANFD_RX_14_MSK register has no effect.

The CANFD_RX_14_MSK register is used to mask the filter fields of message buffer 14.

This register can only be programmed while the CANFD module is in freeze mode as it is blocked by hardware in other modes.

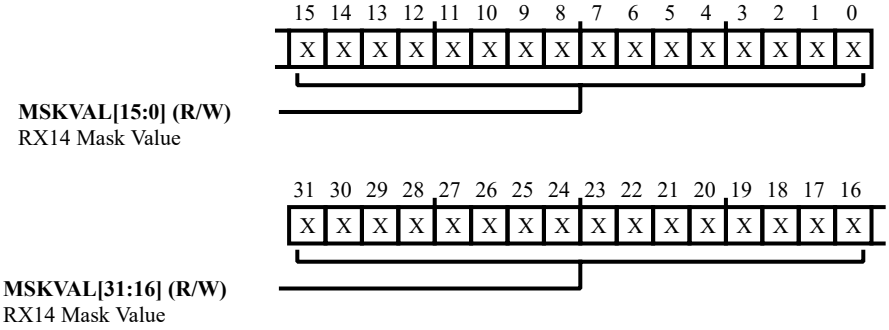


Figure 16-47: CANFD_RX_14_MSK Register Diagram

Table 16-65: CANFD_RX_14_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSKVAL	RX14 Mask Value. In the CANFD_RX_14_MSK.MSKVAL bit field, if a bit is a zero the corresponding bit in the filter is a don't care. If it is a one, the corresponding bit in the filter is checked. Each mask bit masks the corresponding mailbox 14 filter field in the same way that the CANFD_RX_MB_GMSK register masks other mailbox filters.

Receive Mailbox15 Mask Register

The CANFD_RX_15_MSK register is located in RAM.

When the CANFD_CFG.IRMQEN bit is asserted, the CANFD_RX_15_MSK register has no effect.

The CANFD_RX_15_MSK register is used to mask the filter fields of message buffer 15.

This register can only be programmed while the CANFD module is in freeze mode as it is blocked by hardware in other modes.

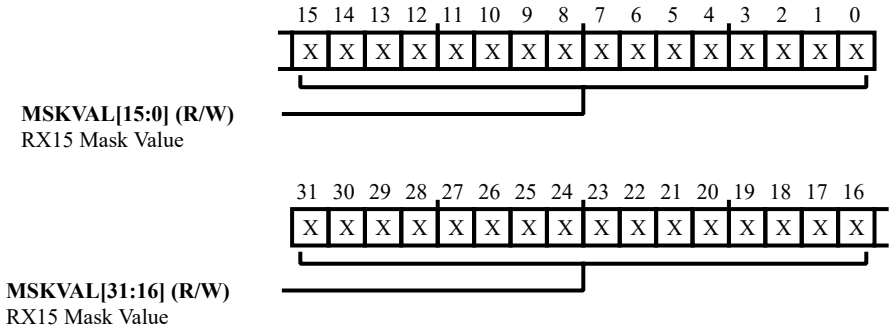


Figure 16-48: CANFD_RX_15_MSK Register Diagram

Table 16-66: CANFD_RX_15_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSKVAL	RX15 Mask Value. In the CANFD_RX_15_MSK.MSKVAL bit field, if a bit is a zero the corresponding bit in the filter is a don't care. If it is a one, the corresponding bit in the filter is checked. Each mask bit masks the corresponding mailbox 15 filter field in the same way that the CANFD_RX_MB_GMSK register masks other mailbox filters.

Receive FIFO Global Mask Register

The CANFD_RX_FIFO_GMSK register is located in RAM.

If the Rx FIFO is enabled, the CANFD_RX_FIFO_GMSK register is used to mask the Rx FIFO ID filter table elements that do not have a corresponding CANFD_RX_IMSK according to the CANFD_CTL2.RFFNUM bit field setting.

The CANFD_RX_FIFO_GMSK register can only be written in freeze mode as it is blocked by hardware in other modes.

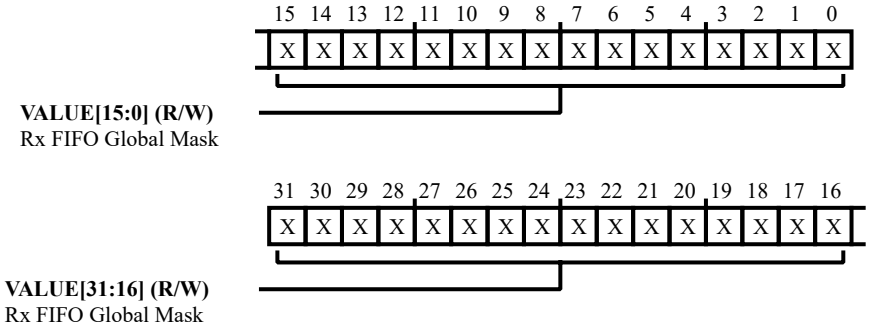


Figure 16-49: CANFD_RX_FIFO_GMSK Register Diagram

Table 16-67: CANFD_RX_FIFO_GMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	VALUE	Rx FIFO Global Mask. The bits in this field mask the ID filter table elements bits in a perfect alignment.	
		0	Corresponding bit in the filter is checked
		1	Corresponding bit in the filter is don't care

Receive FIFO Information Register

The CANFD_RX_FIFO register provides information on the Rx FIFO.

The CANFD_RX_FIFO register is the port through which the processor accesses the output of the RXFIR FIFO located in RAM. The CANFD_RX_FIFO register is written by the CANFD module whenever a new message is moved into the Rx FIFO and the register output is updated whenever the output of the Rx FIFO is updated with the next message.

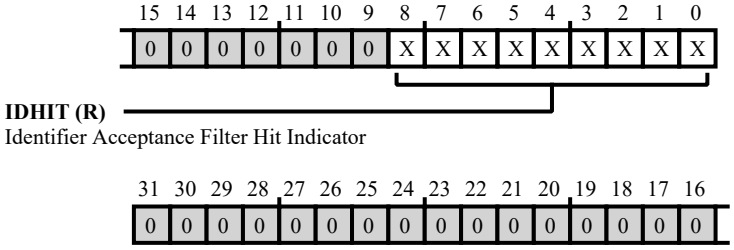


Figure 16-50: CANFD_RX_FIFO Register Diagram

Table 16-68: CANFD_RX_FIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	IDHIT	Identifier Acceptance Filter Hit Indicator. The CANFD_RX_FIFO.IDHIT bit field indicates which identifier acceptance filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID, the first matching IDAF found (lowest number) by the matching process is indicated. The CANFD_RX_FIFO.IDHIT bit field is valid only while the CANFD_IFLG1.MB05 bit is set.

Receive Individual Mask Register

The CANFD_RX_IMSK[n] registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

When the Rx FIFO is disabled (CANFD_CFG.RFEN is disabled), an individual mask is provided for each available Rx mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (CANFD_CFG.RFEN is enabled), an individual mask is provided for each Rx FIFO ID Filter table element on a one-to-one correspondence depending on the setting of the CANFD_CTL2.RFFNUM bit.

The CANFD_RX_IMSK0 register stores the individual mask associated with either MB0 or ID Filter Table Element 0, CANFD_RX_IMSK1 stores the individual mask associated with either MB1 or ID Filter Table Element 1, and so on.

The CANFD_RX_IMSK[n] registers are only be accessed by the processor while the CANFD is in freeze mode and are otherwise blocked by hardware.

The CANFD_RX_IMSK[n] registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

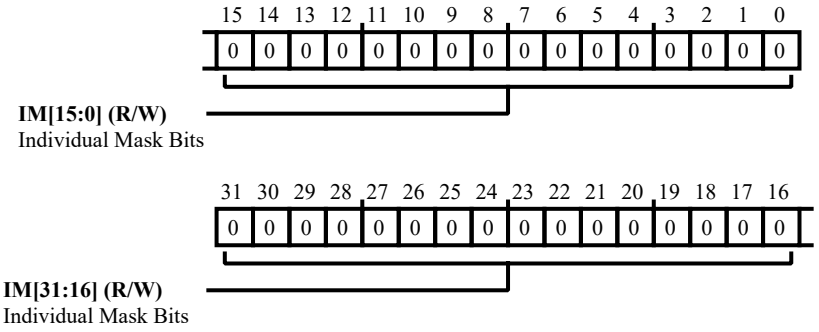


Figure 16-51: CANFD_RX_IMSK[n] Register Diagram

Table 16-69: CANFD_RX_IMSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	IM	Individual Mask Bits. The CANFD_RX_IMSK[n].IM bits masks the corresponding bit in both the mail-box filter and Rx FIFO ID filter table element in distinct ways. For mailbox filters, see the CANFD_RX_MB_GMSK register description. For Rx FIFO ID filter table elements, see the CANFD_RX_FIFO_GMSK register description. If a bit in the CANFD_RX_IMSK[n].IM field is set, the corresponding bit in the filter is checked. If it is cleared, the corresponding bit in the filter is a don't care.

Receive Mailbox Global Mask Register

The CANFD_RX_MB_GMSK register is located in RAM.

When the CANFD_CFG.IRMQEN bit is disabled, the CANFD_RX_MB_GMSK register is always in effect (the bits in the MG field will mask the mailbox filter bits).

When the CANFD_CFG.IRMQEN bit is enabled, the CANFD_RX_MB_GMSK register has no effect (the bits in the MG field will not mask the mailbox filter bits).

The CANFD_RX_MB_GMSK register is used to mask the filter fields of all Rx MBs, excluding MBs 14–15, which have individual mask registers.

The CANFD_RX_MB_GMSK register can only be written in freeze mode and is blocked by hardware in other modes.

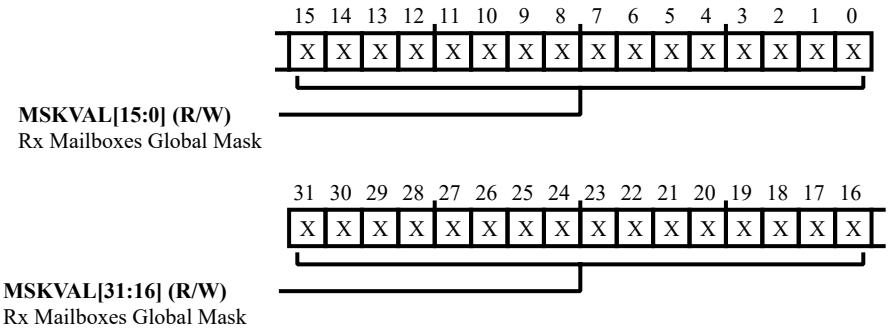


Figure 16-52: CANFD_RX_MB_GMSK Register Diagram

Table 16-70: CANFD_RX_MB_GMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	MSKVAL	Rx Mailboxes Global Mask. The CANFD_RX_MB_GMSK.MSKVAL bits mask the mailbox filter bits. Note that the alignment with the ID word of the mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status (C/S) word of the mailbox.	
		0	Don't care
		1	Filter is checked

Free Running Timer Register

The CANFD_TMR register is a 16-bit free running counter. The timer starts from 0x0 after reset, counts linearly to 0xFFFF, and wraps around.

The timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, the timer increments by one for each bit that is received or transmitted. When there is no message on the bus, the timer counts using the previously programmed baud rate. The timer is not incremented during module disable, doze, stop, pretended networking, and freeze modes.

The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the time stamp entry in a message buffer after the successful reception or transmission of a message.

If the CANFD_CTL1.TSYNEN bit is enabled, the timer is reset whenever a message is received in the first available mailbox, according to the CANFD_CTL2.RFFNUM setting.

The processor can write to this register anytime. However, if the write occurs at the same time that the timer is being reset by a reception in the first mailbox, then the write value is discarded.

Reading this register affects the mailbox unlocking procedure.

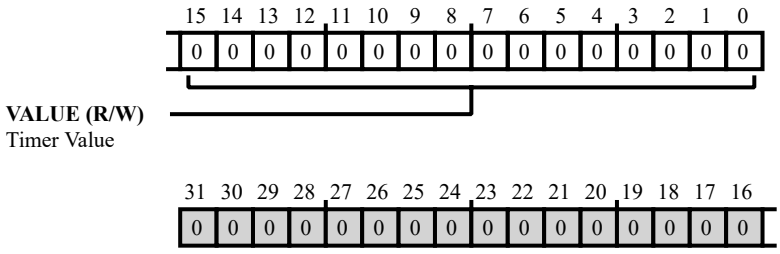


Figure 16-53: CANFD_TMR Register Diagram

Table 16-71: CANFD_TMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer Value. The CANFD_TMR.VALUE bit field contains the free-running counter value.

Pretended Networking Wake Up Message Buffer 0 Control Status Register

The CANFD_WMB0_CS register stores control and status information for an incoming Rx message.

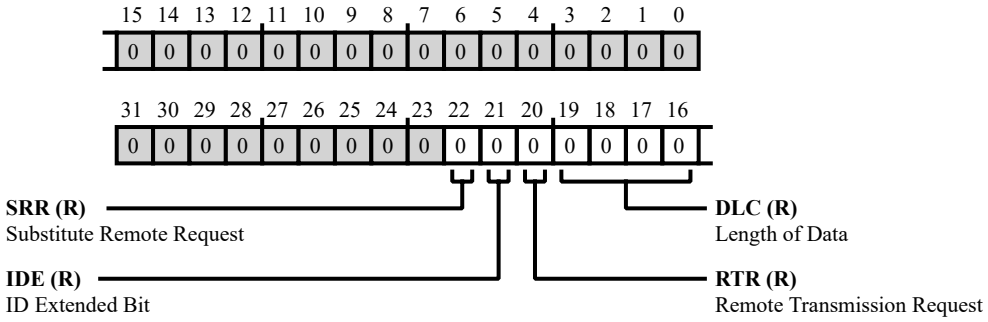


Figure 16-54: CANFD_WMB0_CS Register Diagram

Table 16-72: CANFD_WMB0_CS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/NW)	SRR	Substitute Remote Request. The CANFD_WMB0_CS.SRR bit is received as either recessive or dominant.	
21 (R/NW)	IDE	ID Extended Bit. The CANFD_WMB0_CS.IDE bit identifies whether the frame format is standard or extended.	
		0	Standard Frame Format
		1	Extended Frame Format
20 (R/NW)	RTR	Remote Transmission Request. The CANFD_WMB0_CS.RTR bit indicates whether the frame is remote.	
		0	Data Frame (Not Remote)
		1	Remote Frame
19:16 (R/NW)	DLC	Length of Data. The CANFD_WMB0_CS.DLC field represents the length (in bytes) of the Rx data received when the module is in pretended networking mode. The CANFD_WMB0_CS.DLC field is written by the CANFD module, copied from the DLC (data length code) field of the received frame. The field indicates which data bytes are valid.	

Pretended Networking Wake Up Message Buffer 0 Data 0-3 Register

The CANFD_WMB0_D03 register stores the data bytes 0 to 3 of the payload information of the incoming Rx message. This register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 0-3 registers are located at 0xB48 for WMB0.

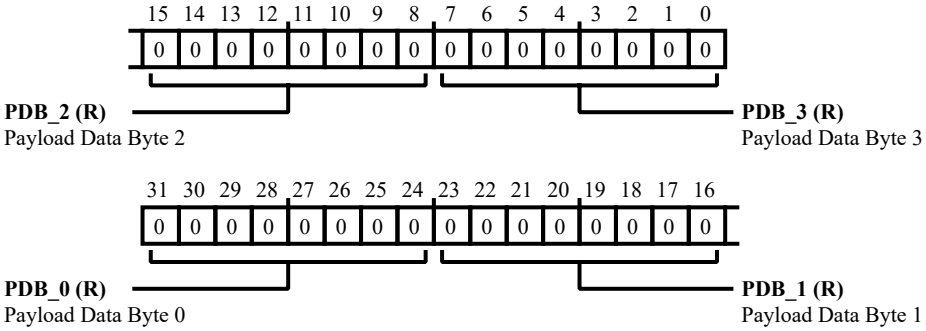


Figure 16-55: CANFD_WMB0_D03 Register Diagram

Table 16-73: CANFD_WMB0_D03 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_0	Payload Data Byte 0. The CANFD_WMB0_D03.PDB_0 bit field contains the received payload corresponding to data byte 0 in pretended networking mode.
23:16 (R/NW)	PDB_1	Payload Data Byte 1. The CANFD_WMB0_D03.PDB_1 bit field contains the received payload corresponding to data byte 1 in pretended networking mode.
15:8 (R/NW)	PDB_2	Payload Data Byte 2. The CANFD_WMB0_D03.PDB_2 bit field contains the received payload corresponding to data byte 2 in pretended networking mode.
7:0 (R/NW)	PDB_3	Payload Data Byte 3. The CANFD_WMB0_D03.PDB_3 bit field contains the received payload corresponding to data byte 3 in pretended networking mode.

Wakeup Message Buffer Data 4-7 Register

The CANFD_WMB0_D47 register stores the data bytes 4 to 7 of the payload information of an incoming Rx message.T his register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 4-7 registers are located at 0xB4C for WMB0.

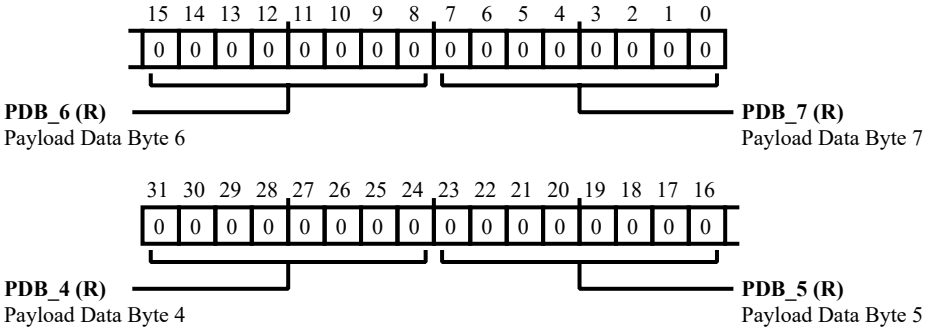


Figure 16-56: CANFD_WMB0_D47 Register Diagram

Table 16-74: CANFD_WMB0_D47 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_4	Payload Data Byte 4. The CANFD_WMB0_D47.PDB_4 bit field contains the received payload corresponding to data byte 4 in pretended networking mode.
23:16 (R/NW)	PDB_5	Payload Data Byte 5. The CANFD_WMB0_D47.PDB_5 bit field contains the received payload corresponding to data byte 5 in pretended networking mode.
15:8 (R/NW)	PDB_6	Payload Data Byte 6. The CANFD_WMB0_D47.PDB_6 bit field contains the received payload corresponding to data byte 6 in pretended networking mode.
7:0 (R/NW)	PDB_7	Payload Data Byte 7. The CANFD_WMB0_D47.PDB_7 bit field contains the received payload corresponding to data byte 7 in pretended networking mode.

Pretended Networking Wake Up Message Buffer ID 0 Register

The CANFD_WMB0_ID register stores the ID information of an incoming Rx message.

Note that the WMB0 ID registers is located at 0xB44.

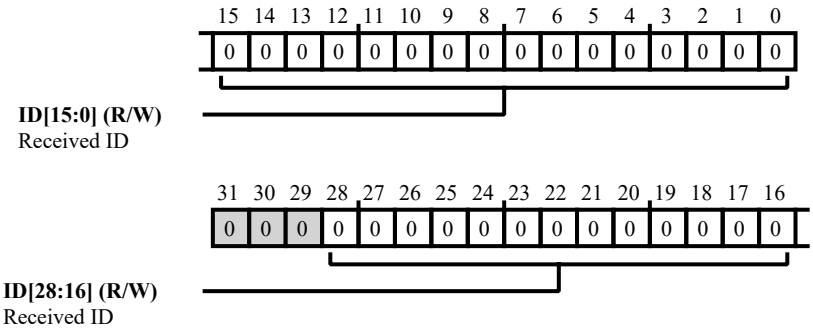


Figure 16-57: CANFD_WMB0_ID Register Diagram

Table 16-75: CANFD_WMB0_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	ID	Received ID. The CANFD_WMB0_ID.ID bit field stores the received ID as either, the 29 bits of the extended frame format (considering the ID[28:0] field) or the 11 bits of the standard frame format (considering the ID[28:18] field only). In the standard frame format, the remaining bits in the ID[17:0] range have no meaning.

Pretended Networking Wake Up Message Buffer 1 Control Status Register

The CANFD_WMB1_CS register stores control and status information for an incoming Rx message.

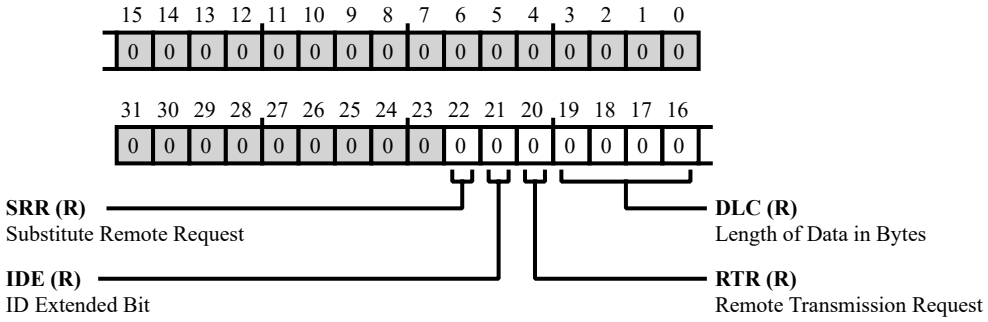


Figure 16-58: CANFD_WMB1_CS Register Diagram

Table 16-76: CANFD_WMB1_CS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/NW)	SRR	Substitute Remote Request. The CANFD_WMB1_CS.SRR bit is received as either recessive or dominant.	
21 (R/NW)	IDE	ID Extended Bit. The CANFD_WMB1_CS.IDE bit identifies whether the frame format is standard or extended.	
		0	Standard Frame Format
		1	Extended Frame Format
20 (R/NW)	RTR	Remote Transmission Request. The CANFD_WMB1_CS.RTR bit indicates whether the frame is remote.	
		0	Data Frame (Not Remote)
		1	Remote Frame
19:16 (R/NW)	DLC	Length of Data in Bytes. The CANFD_WMB1_CS.DLC field represents the length (in bytes) of the Rx data received when the module is in pretended networking mode. The CANFD_WMB1_CS.DLC field is written by the CANFD module, copied from the DLC (data length code) field of the received frame. The field indicates which data bytes are valid.	

Pretended Networking Wake Up Message Buffer 1 Data 0-3 Register

The CANFD_WMB1_D03 register stores the data bytes 0 to 3 of the payload information of the incoming Rx message. This register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 0-3 registers are located at 0xB58 for WMB1.

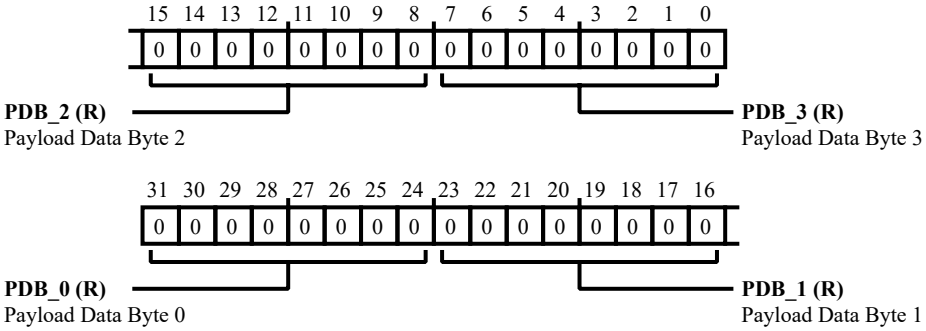


Figure 16-59: CANFD_WMB1_D03 Register Diagram

Table 16-77: CANFD_WMB1_D03 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_0	Payload Data Byte 0. The CANFD_WMB1_D03.PDB_0 bit field contains the received payload corresponding to data byte 0 in pretended networking mode.
23:16 (R/NW)	PDB_1	Payload Data Byte 1. The CANFD_WMB1_D03.PDB_1 bit field contains the received payload corresponding to data byte 1 in pretended networking mode.
15:8 (R/NW)	PDB_2	Payload Data Byte 2. The CANFD_WMB1_D03.PDB_2 bit field contains the received payload corresponding to data byte 2 in pretended networking mode.
7:0 (R/NW)	PDB_3	Payload Data Byte 3. The CANFD_WMB1_D03.PDB_3 bit field contains the received payload corresponding to data byte 3 in pretended networking mode.

Pretended Networking Wake Up Message Buffer 1 Data 4-7 Register

The CANFD_WMB1_D47 register stores the data bytes 4 to 7 of the payload information of an incoming Rx message.T his register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 4-7 registers are located at 0xB5C for WMB1.

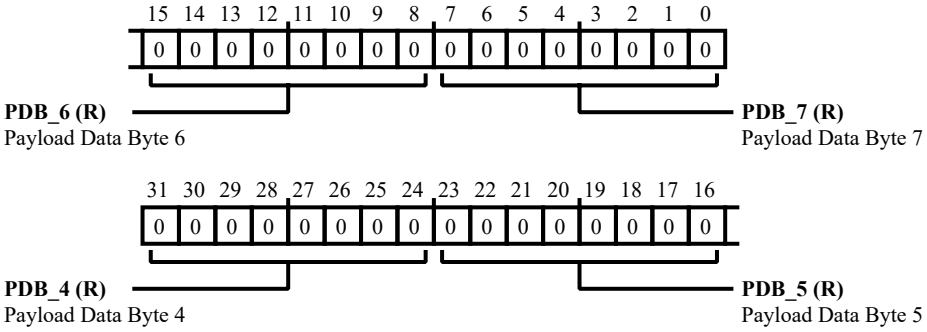


Figure 16-60: CANFD_WMB1_D47 Register Diagram

Table 16-78: CANFD_WMB1_D47 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_4	Payload Data Byte 4. The CANFD_WMB1_D47.PDB_4 bit field contains the received payload corresponding to data byte 4 in pretended networking mode.
23:16 (R/NW)	PDB_5	Payload Data Byte 5. The CANFD_WMB1_D47.PDB_5 bit field contains the received payload corresponding to data byte 5 in pretended networking mode.
15:8 (R/NW)	PDB_6	Payload Data Byte 6. The CANFD_WMB1_D47.PDB_6 bit field contains the received payload corresponding to data byte 6 in pretended networking mode.
7:0 (R/NW)	PDB_7	Payload Data Byte 7. The CANFD_WMB1_D47.PDB_7 bit field contains the received payload corresponding to data byte 7 in pretended networking mode.

Pretended Networking Wake Up Message Buffer ID 1 Register

The CANFD_WMB1_ID register stores the ID information of an incoming Rx message.

Note that the WMB1 ID registers is located at 0xB54.

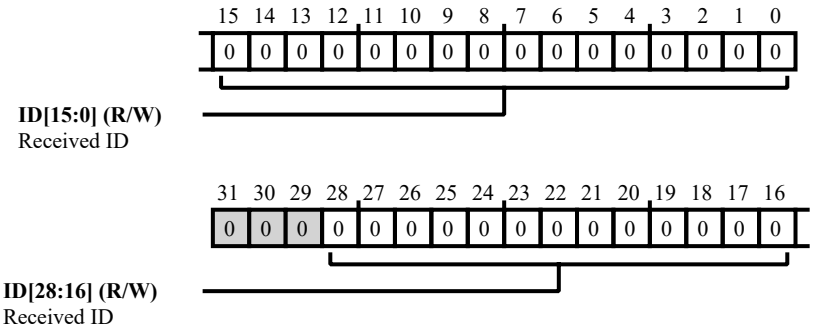


Figure 16-61: CANFD_WMB1_ID Register Diagram

Table 16-79: CANFD_WMB1_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	ID	Received ID. The CANFD_WMB1_ID.ID bit field stores the received ID as either, the 29 bits of the extended frame format (considering the ID[28:0] field) or the 11 bits of the standard frame format (considering the ID[28:18] field only). In the standard frame format, the remaining bits in the ID[17:0] range have no meaning.

Pretended Networking Wake Up Message Buffer 2 Control Status Register

The CANFD_WMB2_CS register stores control and status information for an incoming Rx message.

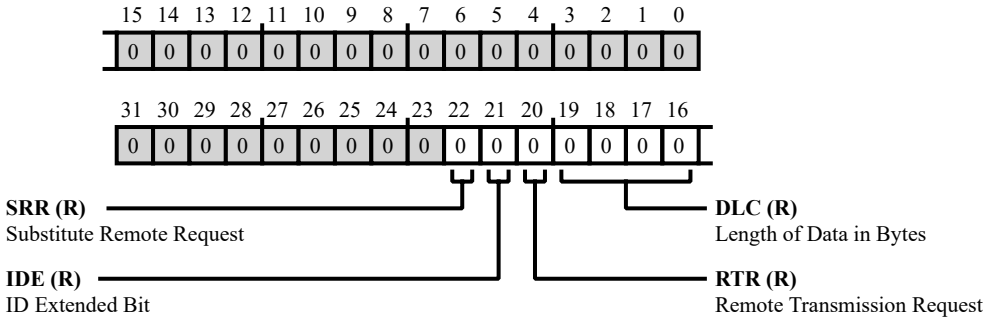


Figure 16-62: CANFD_WMB2_CS Register Diagram

Table 16-80: CANFD_WMB2_CS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/NW)	SRR	Substitute Remote Request. The CANFD_WMB2_CS.SRR bit is received as either recessive or dominant.	
21 (R/NW)	IDE	ID Extended Bit. The CANFD_WMB2_CS.IDE bit identifies whether the frame format is standard or extended.	
		0	Standard Frame Format
		1	Extended Frame Format
20 (R/NW)	RTR	Remote Transmission Request. The CANFD_WMB2_CS.RTR bit indicates whether the frame is remote.	
		0	Data Frame (Not Remote)
		1	Remote Frame
19:16 (R/NW)	DLC	Length of Data in Bytes. The CANFD_WMB2_CS.DLC field represents the length (in bytes) of the Rx data received when the module is in pretended networking mode. The CANFD_WMB2_CS.DLC field is written by the CANFD module, copied from the DLC (data length code) field of the received frame. The field indicates which data bytes are valid.	

Pretended Networking Wake Up Message Buffer 2 Data 0-3 Register

The CANFD_WMB2_D03 register stores the data bytes 0 to 3 of the payload information of the incoming Rx message. This register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 0-3 registers are located at 0xB68 for WMB2.

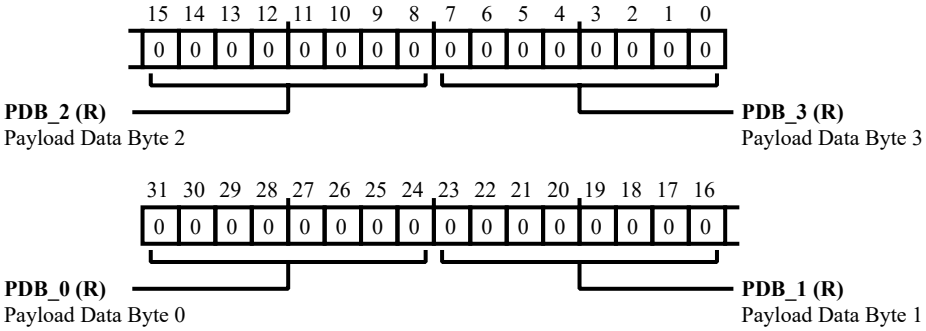


Figure 16-63: CANFD_WMB2_D03 Register Diagram

Table 16-81: CANFD_WMB2_D03 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_0	Payload Data Byte 0. The CANFD_WMB2_D03.PDB_0 bit field contains the received payload corresponding to data byte 0 in pretended networking mode.
23:16 (R/NW)	PDB_1	Payload Data Byte 1. The CANFD_WMB2_D03.PDB_1 bit field contains the received payload corresponding to data byte 1 in pretended networking mode.
15:8 (R/NW)	PDB_2	Payload Data Byte 2. The CANFD_WMB2_D03.PDB_2 bit field contains the received payload corresponding to data byte 2 in pretended networking mode.
7:0 (R/NW)	PDB_3	Payload Data Byte 3. The CANFD_WMB2_D03.PDB_3 bit field contains the received payload corresponding to data byte 3 in pretended networking mode.

Pretended Networking Wake Up Message Buffer 2 Data 4-7 Register

The CANFD_WMB2_D47 register stores the data bytes 4 to 7 of the payload information of an incoming Rx message.T his register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 4-7 registers are located at 0xB6C for WMB2.

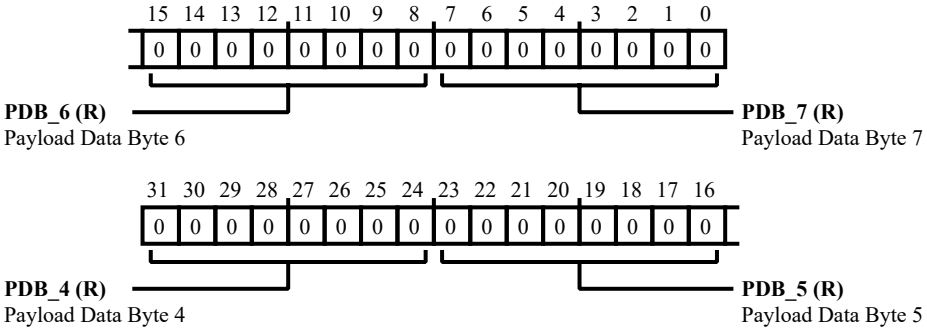


Figure 16-64: CANFD_WMB2_D47 Register Diagram

Table 16-82: CANFD_WMB2_D47 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_4	Payload Data Byte 4. The CANFD_WMB2_D47.PDB_4 bit field contains the received payload corresponding to data byte 4 in pretended networking mode.
23:16 (R/NW)	PDB_5	Payload Data Byte 5. The CANFD_WMB2_D47.PDB_5 bit field contains the received payload corresponding to data byte 5 in pretended networking mode.
15:8 (R/NW)	PDB_6	Payload Data Byte 6. The CANFD_WMB2_D47.PDB_6 bit field contains the received payload corresponding to data byte 6 in pretended networking mode.
7:0 (R/NW)	PDB_7	Payload Data Byte 7. The CANFD_WMB2_D47.PDB_7 bit field contains the received payload corresponding to data byte 7 in pretended networking mode.

Pretended Networking Wake Up Message Buffer ID 2 Register

The CANFD_WMB2_ID register stores the ID information of an incoming Rx message.

Note that the WMB2 ID registers is located at 0xB64.

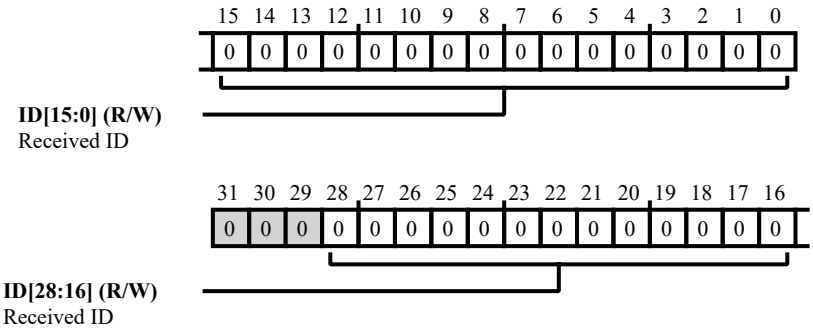


Figure 16-65: CANFD_WMB2_ID Register Diagram

Table 16-83: CANFD_WMB2_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	ID	Received ID. The CANFD_WMB2_ID.ID bit field stores the received ID as either, the 29 bits of the extended frame format (considering the ID[28:0] field) or the 11 bits of the standard frame format (considering the ID[28:18] field only). In the standard frame format, the remaining bits in the ID[17:0] range have no meaning.

Pretended Networking Wake Up Message Buffer 3 Control Status Register

The CANFD_WMB3_CS register stores control and status information for an incoming Rx message.

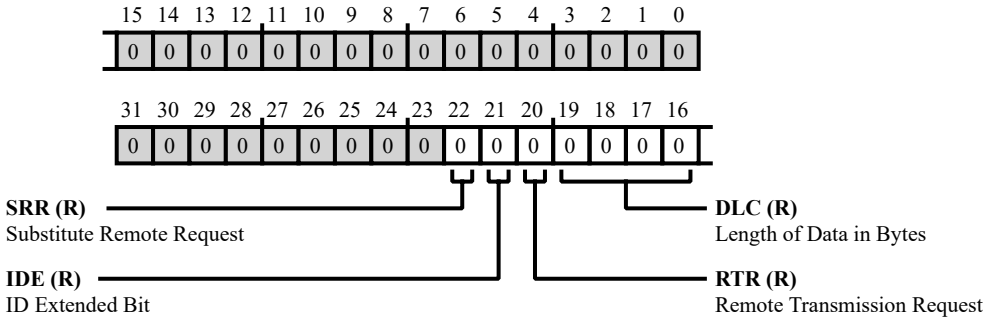


Figure 16-66: CANFD_WMB3_CS Register Diagram

Table 16-84: CANFD_WMB3_CS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/NW)	SRR	Substitute Remote Request. The CANFD_WMB3_CS.SRR bit is received as either recessive or dominant.	
21 (R/NW)	IDE	ID Extended Bit. The CANFD_WMB3_CS.IDE bit identifies whether the frame format is standard or extended.	
		0	Standard Frame Format
		1	Extended Frame Format
20 (R/NW)	RTR	Remote Transmission Request. The CANFD_WMB3_CS.RTR bit indicates whether the frame is remote.	
		0	Data Frame (Not Remote)
		1	Remote Frame
19:16 (R/NW)	DLC	Length of Data in Bytes. The CANFD_WMB3_CS.DLC field represents the length (in bytes) of the Rx data received when the module is in pretended networking mode. The CANFD_WMB3_CS.DLC field is written by the CANFD module, copied from the DLC (data length code) field of the received frame. The field indicates which data bytes are valid.	

Pretended Networking Wake Up Message Buffer 3 Data 0-3 Register

The CANFD_WMB3_D03 register stores the data bytes 0 to 3 of the payload information of the incoming Rx message. This register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 0-3 registers are located at 0xB78 for WMB3.

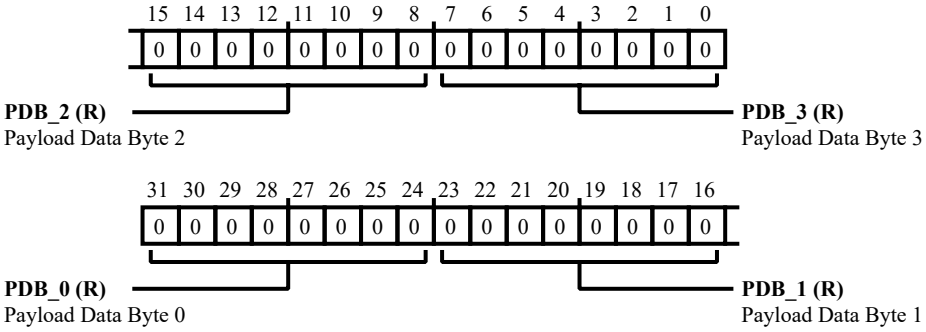


Figure 16-67: CANFD_WMB3_D03 Register Diagram

Table 16-85: CANFD_WMB3_D03 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_0	Payload Data Byte 0. The CANFD_WMB3_D03.PDB_0 bit field contains the received payload corresponding to data byte 0 in pretended networking mode.
23:16 (R/NW)	PDB_1	Payload Data Byte 1. The CANFD_WMB3_D03.PDB_1 bit field contains the received payload corresponding to data byte 1 in pretended networking mode.
15:8 (R/NW)	PDB_2	Payload Data Byte 2. The CANFD_WMB3_D03.PDB_2 bit field contains the received payload corresponding to data byte 2 in pretended networking mode.
7:0 (R/NW)	PDB_3	Payload Data Byte 3. The CANFD_WMB3_D03.PDB_3 bit field contains the received payload corresponding to data byte 3 in pretended networking mode.

Pretended Networking Wake Up Message Buffer 3 Data 4-7 Register

The CANFD_WMB3_D47 register stores the data bytes 4 to 7 of the payload information of an incoming Rx message.T his register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 4-7 registers are located at 0xB7C for WMB3.

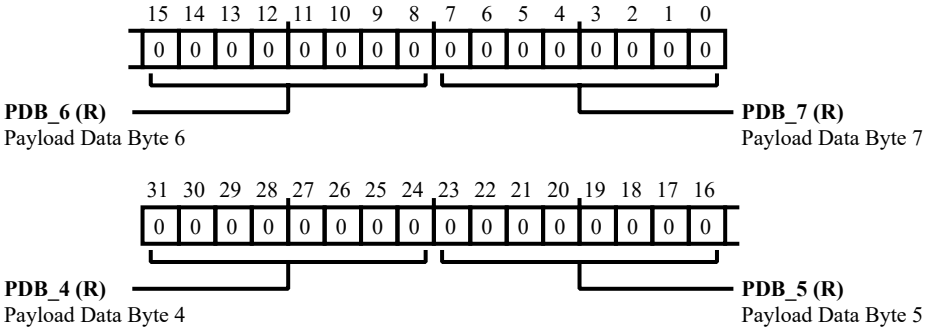


Figure 16-68: CANFD_WMB3_D47 Register Diagram

Table 16-86: CANFD_WMB3_D47 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB_4	Payload Data Byte 4. The CANFD_WMB3_D47.PDB_4 bit field contains the received payload corresponding to data byte 4 in pretended networking mode.
23:16 (R/NW)	PDB_5	Payload Data Byte 5. The CANFD_WMB3_D47.PDB_5 bit field contains the received payload corresponding to data byte 5 in pretended networking mode.
15:8 (R/NW)	PDB_6	Payload Data Byte 6. The CANFD_WMB3_D47.PDB_6 bit field contains the received payload corresponding to data byte 6 in pretended networking mode.
7:0 (R/NW)	PDB_7	Payload Data Byte 7. The CANFD_WMB3_D47.PDB_7 bit field contains the received payload corresponding to data byte 7 in pretended networking mode.

Pretended Networking Wake Up Message Buffer ID 3 Register

The CANFD_WMB3_ID register stores the ID information of an incoming Rx message.

Note that the WMB3 ID registers is located at 0xB74.

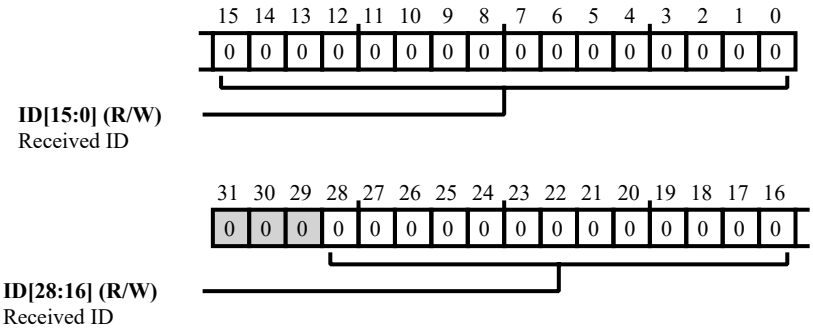


Figure 16-69: CANFD_WMB3_ID Register Diagram

Table 16-87: CANFD_WMB3_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	ID	Received ID. The CANFD_WMB3_ID.ID bit field stores the received ID as either, the 29 bits of the extended frame format (considering the ID[28:0] field) or the 11 bits of the standard frame format (considering the ID[28:18] field only). In the standard frame format, the remaining bits in the ID[17:0] range have no meaning.

Pretended Networking Wakeup Match Register

The CANFD_WUM register contains wake up information related to the matching processes performed while the CANFD module receives frames under pretended networking mode.

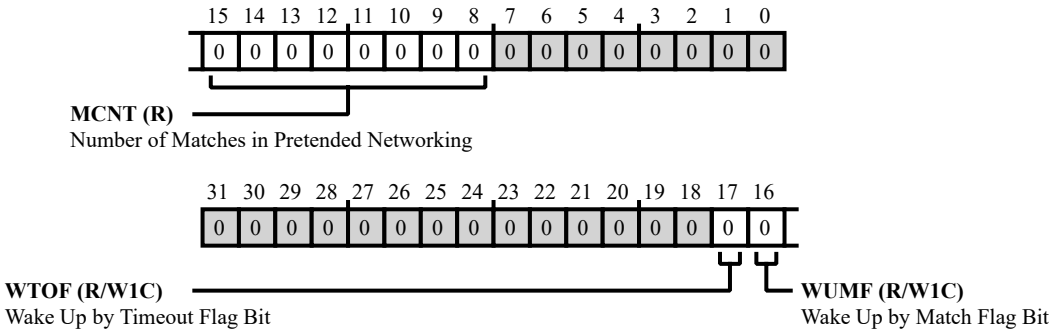


Figure 16-70: CANFD_WUM Register Diagram

Table 16-88: CANFD_WUM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	WTOF	Wake Up by Timeout Flag Bit. The CANFD_WUM.WTOF bit indicates if the CANFD module detects a timeout event during a time interval defined by the CANFD_PN_CTL2.MATCHTO bit field. WTOF generates a wakeup event if the CANFD_PN_CTL1.WTOFMSK enabled.	
		0	No event detected
		1	Wakeup event detected
16 (R/W1C)	WUMF	Wake Up by Match Flag Bit. The CANFD_WUM.WUMF bit indicates whether the CANFD module detects a matching Rx incoming message passing the filtering criteria specified in the CANFD_PN_CTL1 register. WUMF generates a wakeup event if the CANFD_PN_CTL1.WUMFMSK bit is enabled.	
		0	No event detected
		1	Wakeup event detected
15:8 (R/NW)	MCNT	Number of Matches in Pretended Networking. The CANFD_WUM.MCNT bit field reports the number of times a given message matches the predefined filtering criteria for ID and/or PL before a wake up event. The CANFD_WUM.MCNT bit field is reset by the CANFD module when it enters pretended networking mode. This bit field and is not affected by a soft reset.	

17 Watchdog Timer (WDOG)

The processor has a 32-bit watchdog timer that can be used to verify system reliability by generating an event to the processor core when the watchdog expires before software updates it.

The ADSP-2184x has four watchdog timers, providing more options to monitor different parts of the system separately.

WDOG Features

The watchdog timer has the following features:

- Programmable 32-bit watchdog count value
- 8-bit disable bit pattern
- General-purpose core event generation

The watchdog timer can supervise system software stability by periodically reloading it to prevent expiration of the downward-counting timer (such that the count never becomes 0). When used in this fashion, an expiring timer can indicate the system software is not running normally.

Expiration of the WDOG counter generates a general-purpose interrupt, which can be used in a variety of ways:

- as an interrupt vector sent through the System Event Controller (SEC) to the core to be serviced by a handler function, providing full software control of device resources (for example, GPIO management and reset control).
- as a fault condition through the SEC to provide hardware-automated:
    - signaling of the fault condition on external pins to the system,
    - system reset requests to the Reset Control Unit (RCU), and/or
    - trigger outputs (SEC0_FAULT trigger generator) through the Trigger Routing Unit (TRU) to initiate activities in a variety of potential trigger receiver (for example, GPIO control).

To help debugging, the watchdog timer does not decrement (even when enabled) if the processor is in emulation mode.

WDOG Functional Description

When enabled, the 32-bit watchdog timer counts downward every SCLK0_0 cycle. If it reaches zero, the watchdog expiration event is generated, which can be used in many ways.

To start the watchdog timer:

1. Program the watchdog timeout period (in SCLK0_0 cycles) in the WDOG_CNT register. With the watchdog disabled, this write also preloads the WDOG_STAT register.
2. Enable the watchdog timer by writing any value other than 0xAD to the WDOG_CTL.WDEN field.

Once the watchdog is enabled, writes to the WDOG_CNT register are ignored. The counter begins decrementing, and the current counter value can be read from the 32-bit WDOG_STAT register at any time.

To prevent the counter from expiring, software must "kick" the watchdog by writing any value to the WDOG_STAT register while the current count is non-zero. While the value written is irrelevant and ignored, this action resets the current counter in the WDOG_STAT register to the programmed WDOG_CNT value, and decrementing continues. The internal counter continues decrementing until it reaches zero, at which point the expiration event is generated, and the WDOG_CTL.WDRO rollover bit is set.

Watchdog operation continues in this manner unless disabled by explicitly writing 0xAD to the WDOG_CTL.WDEN field.

The watchdog expiration event can be used in a variety of ways, as listed below.

- The watchdog expiration event itself is one of numerous interrupt sources that is managed by the System Event Controller. Like other peripheral sources, this event can be used to cause a vector to a handler function that executes based on interrupt priority.
- The watchdog expiration event can be used to initiate automated hardware response through the SEC Fault Interface (SFI).

For this, the WDOG expiry must be configured as the fault source in the SEC. Then the response to the WDOG expiry can be set to any of the following actions:

- Signaling through the external fault pin.
- System reset.
- Trigger outputs to numerous potential trigger completers.

ADSP-2184x WDOG Register List

The Watchdog Timer unit (WDOG) provides a software-based watchdog timer that can improve system reliability by generating an event to the processor core if the watchdog expires before being updated by software. A set of registers governs WDOG operations. For more information on WDOG functionality, see the WDOG register descriptions.

Table 17-1: ADSP-2184x WDOG Register List

Name	Description
WDOG_CNT	Count Register
WDOG_CTL	Control Register
WDOG_STAT	Watchdog Timer Status Register
WDOG_WIN	Watchdog Timer Window Register

ADSP-2184x WDOG Interrupt List

Table 17-2: ADSP-2184x WDOG Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
382	WDOG0_EXP	WDOG0 Expiration	Level	
383	WDOG1_EXP	WDOG1 Expiration	Level	
384	WDOG2_EXP	WDOG2 Expiration	Level	
385	WDOG3_EXP	WDOG3 Expiration	Level	

ADSP-2184x WDOG Trigger List

Table 17-3: ADSP-2184x WDOG Trigger List Generators

Trigger ID	Name	Description	Sensitivity
197	WDOG0_EXP	WDOG0 Expiration	Level
198	WDOG1_EXP	WDOG1 Expiration	Level
199	WDOG2_EXP	WDOG2 Expiration	Level
200	WDOG3_EXP	WDOG3 Expiration	Level

Table 17-4: ADSP-2184x WDOG Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

WDOG Block Diagram

The *Watchdog Timer Block Diagram* figure shows the detailed block diagram for the watchdog timer.

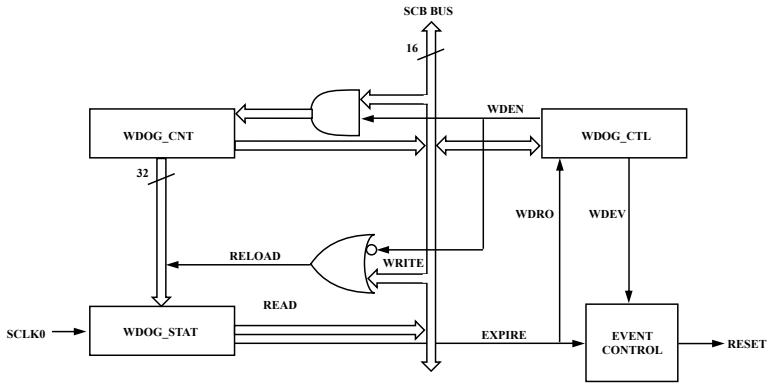


Figure 17-1: Watchdog Timer Block Diagram

Internal Interface

The system clock (SCLK) clocks the watchdog timer. The registers are accessed through the peripheral MMR access bus. 32-bit read/write operations always access the 32-bit `WDOG_CNT` and `WDOG_STAT` registers. Hardware ensures that those accesses are atomic. When the counter expires, the WDOG expiration event is generated.

External Interface

The watchdog timer does not directly interact with any external pins.

ADSP-2184x WDOG Register Descriptions

Watchdog Timer Unit (WDOG) contains the following registers.

Table 17-5: ADSP-2184x WDOG Register List

Name	Description
`WDOG_CNT`	Count Register
`WDOG_CTL`	Control Register
`WDOG_STAT`	Watchdog Timer Status Register
`WDOG_WIN`	Watchdog Timer Window Register

Count Register

The WDOG_CNT register holds the programmable, unsigned count value. A valid write to this register also pre-loads the WDOG counter. For added safety, the WDOG_CNT register can be updated only when the WDOG timer is disabled. A write to the WDOG_CNT register while the timer is enabled does not modify the contents of this register. This register must be accessed with 32-bit read/writes only.

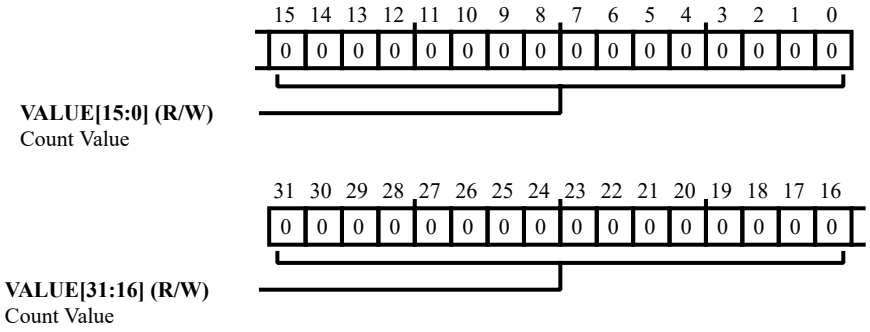


Figure 17-2: WDOG_CNT Register Diagram

Table 17-6: WDOG_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Count Value. The WDOG_CNT.VALUE bit field holds the programmable, unsigned count value.

Control Register

The WDOG_CTL register controls the watchdog timer. This register supports enabling/disabling the watchdog timer and supports checking the timer rollover status. Note that when the processor is in emulation mode, the watchdog timer counter will not decrement even if it is enabled.

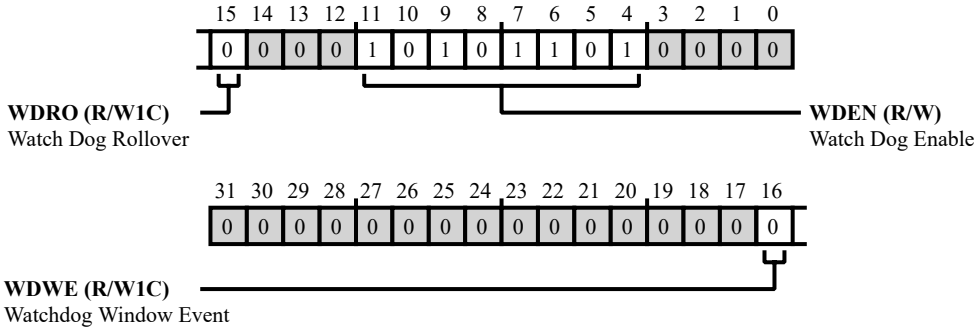


Figure 17-3: WDOG_CTL Register Diagram

Table 17-7: WDOG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	WDWE	Watchdog Window Event. Software can determine whether the timer has been serviced unexpectedly early by interrogating the WDOG_CTL.WDWE status bit. This is a sticky bit that is set whenever the watchdog is serviced and the WDOG_STAT value is greater than the WDOG_WIN value. It is cleared only by disabling the watchdog timer and by writing a 1 to the bit.	
		0	Window Event has not occurred
		1	Window Event has occurred
15 (R/W1C)	WDRO	Watch Dog Rollover. Software can determine whether the timer has rolled over by interrogating the WDOG_CTL.WDRO status bit. This is a sticky bit that is set whenever the watch dog timer count reaches 0 and cleared only by disabling the watch dog timer and then writing a 1 to the bit.	
		0	WDT Has Not Expired
		1	WDT Has Expired
11:4 (R/W)	WDEN	Watch Dog Enable. The WDOG_CTL.WDEN field is used to enable and disable the watchdog timer. Writing any value other than the disable value into this field enables the watchdog timer. This multi-bit disable key minimizes the chance of inadvertently disabling the watchdog timer.	
		173	Counter Disabled. All other values mean that the counter is enabled.

Watchdog Timer Status Register

The WDOG_STAT register contains the current count value of the watchdog timer. Reads of this register return the current count value. When the watchdog timer is enabled, the WDOG_STAT register is decremented by 1 on each SCLK cycle. When the count value reaches 0, the watchdog timer stops counting, and the expiry event is generated. The WDOG_STAT register is a 32-bit unsigned system MMR that must be accessed with 32-bit reads and writes.

Values cannot be stored directly in this register but are instead copied from the WDOG_CNT register. This copy process can happen in two ways:

- While the watchdog timer is disabled, writing the WDOG_CNT register pre-loads the WDOG_STAT register.
- While the watchdog timer is enabled, writing the WDOG_STAT register loads it with the value in the WDOG_CNT register.
- While the watchdog timer is disabled, writing to the WDOG_STAT register also reloads it with the value in the WDOG_CNT register.

When the processor executes a write (of an arbitrary value) to the WDOG_STAT register, the value in the WDOG_CNT register is copied into the WDOG_STAT register. Typically, software sets the value of WDOG_CNT at initialization, then periodically writes to the WDOG_STAT register before the watchdog timer expires. This reloads the watchdog timer with the value from WDOG_CNT and prevents generation of the expiry event.

If the program does not reload the counter before SCLK x count register cycles, an expiry event is generated, and the WDOG_CTL.WDRO bit is set. When this happens, the counter stops decrementing and remains at zero. If the counter is enabled with a zero loaded to the counter, the WDOG_CTL.WDRO bit is set immediately and the counter remains at zero and does not decrement.

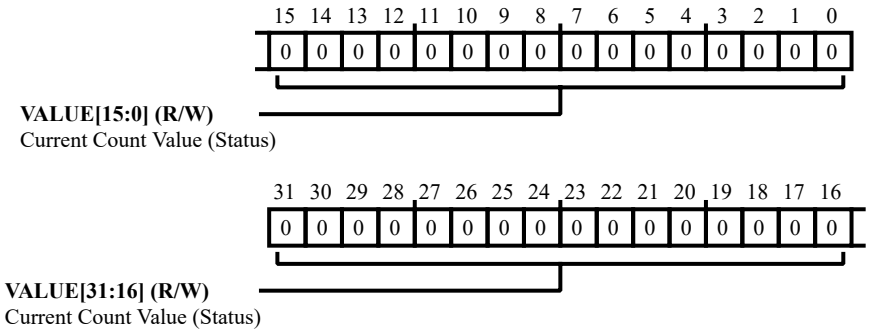


Figure 17-4: WDOG_STAT Register Diagram

Table 17-8: WDOG_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Current Count Value (Status). The `WDOG_STAT.VALUE` bit field contains the current count value of the watchdog timer.

Watchdog Timer Window Register

Watchdog window register holds the unsigned window value programmed. The window register can be programmed while the WDOG is disabled; any write to the register when WDOG is enabled doesn't alter the contents of the register. When the WDOG is enabled and the core services the WDOG by doing a write to `WDOG_STAT` register while the `WDOG_STAT` value is greater than `WDOG_WIN`, the WDWE event is generated and also the counter stops decrementing.

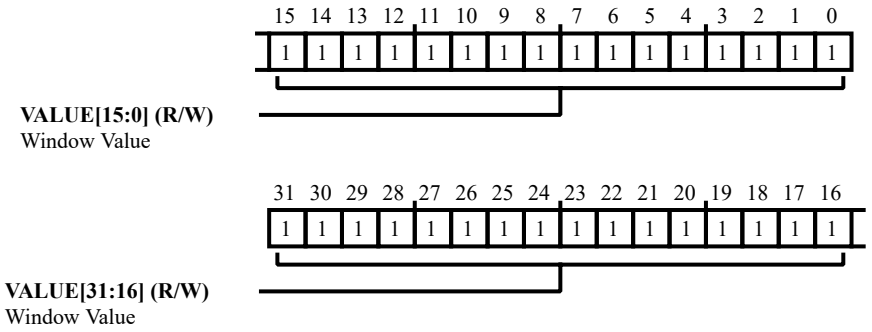


Figure 17-5: WDOG_WIN Register Diagram

Table 17-9: WDOG_WIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Window Value. The `WDOG_WIN.VALUE` bit field contains the unsigned window value.

18 Link Port (LP)

Link ports allow the processor to connect to other processors or peripheral link ports using a simple communication protocol for high-speed parallel data transfer. This peripheral allows various I/O peripheral interconnection schemes to I/O peripheral devices as well as co-processing and multiprocessing schemes.

The link ports of the processor support 2- and 4-bit wide data transfers DDR mode. The link port pins are multiplexed in the GPIO ports. For information on processor multiplexing, see the data sheet for the specific processor.

Link ports can operate independently and simultaneously, allowing glueless high-speed connectivity of up to four external processors.

NOTE: Reference to SCLK in this chapter can be considered as OCLK0_0.

LP Features

All link ports are identical in their design and have the following common features.

- Bidirectional ports with four data signals (`LP_D0` - `LP_D3`), an acknowledge signal (`LP_ACK`), and a clock signal (`LP_CLK`).
- Provide high-speed, point-to-point data transfers to other processors, allowing different types of interconnections between multiple processors.
- Pack data into 32-bit words. The processor can directly read this data or transfer it through DMA to or from on-chip memory.
- Support for data buffering through a 2-deep FIFO for transmit and a 4-deep FIFO for receive.
- Programmable clock and acknowledge based handshake mechanism for efficient communication.
- A dedicated DMA channel.
- Support for data transfer in 2-bit mode (DDR) and 4-bit mode (DDR)

LP Functional Description

This section provides a description of the link port, including a list of its registers and a functional block diagram.

ADSP-2184x LP Register List

The Link Port LP is an 8-bit wide parallel port that can connect to another processor's LP or another LP-compatible device. This port allows a variety of interconnection schemes to I/O peripheral devices as well as co-processing and multiprocessing schemes. A set of registers governs LP operations. For more information on LP functionality, see the LP register descriptions.

Table 18-1: ADSP-2184x LP Register List

Name	Description
LP_CTL	Control Register
LP_DIV	Clock Divider Value Register
LP_RX	Receive Buffer Register
LP_STAT	Status Register
LP_TX	Transmit Buffer Register
LP_TXIN_SHDW	Shadow Input Transmit Buffer Register
LP_TXOUT_SHDW	Shadow Output Transmit Buffer Register

ADSP-2184x LP Interrupt List

Table 18-2: ADSP-2184x LP Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
152	LP0_DMA	LP0 DMA Channel		30
153	LP0_STAT	LP0 Status		
154	LP1_DMA	LP1 DMA Channel		36
155	LP1_STAT	LP1 Status		
156	LP0_DMA_ERR	LP0 DMA Data Error		
157	LP1_DMA_ERR	LP1 DMA Data Error		

ADSP-2184x LP Trigger List

Table 18-3: ADSP-2184x LP Trigger List Generators

Trigger ID	Name	Description	Sensitivity
67	LP0_DMA	LP0 DMA Channel	
68	LP1_DMA	LP1 DMA Channel	

Table 18-4: ADSP-2184x LP Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
114	LP0_DMA	LP0 DMA Channel	Pulse
115	LP1_DMA	LP1 DMA Channel	Pulse

ADSP-2184x LP DMA Channel List

Table 18-5: ADSP-2184x LP DMA Channel List

DMA ID	DMA Channel Name	Description
DMA30	LP0_DMA	LP0 DMA Channel
DMA36	LP1_DMA	LP1 DMA Channel

Block Diagram

The ***Link Port Block Diagram*** shows the block diagram of a link port.

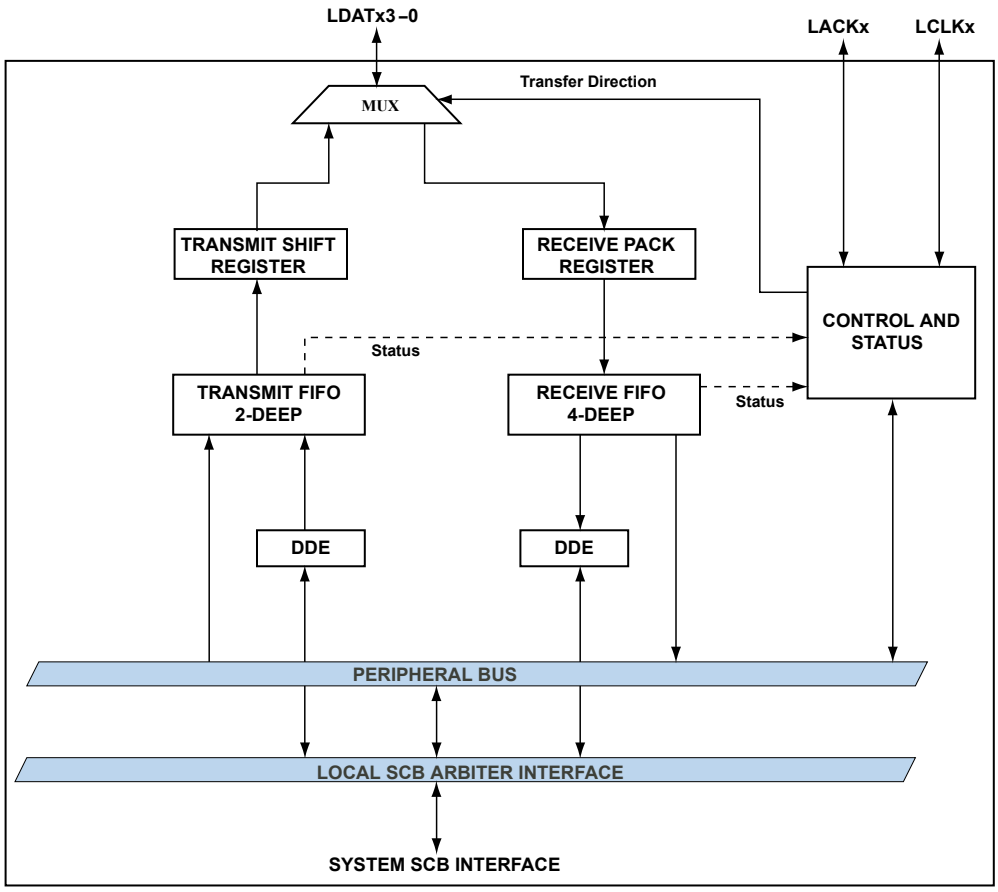


Figure 18-1: Link Port Block Diagram

External Connections

As shown in the ***Link Port Pin Connections*** figure, a link port has four data lines (LP_D0 – LP_D3), a clock line (LP_CLK), and an acknowledge line (LP_ACK). A link port can act as either a transmitter or a receiver but not both at the same time.

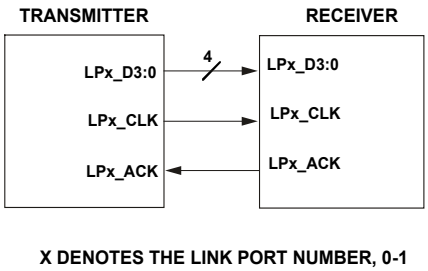


Figure 18-2: Link Port Pin Connections

Use external pull-downs for the LP_CLK and LP_ACK pins so that the link port can enable the transmitter and receiver, irrespective of the state of the other.

Internal Blocks

As shown in the block diagram, the link ports have independent modules for transmit and receive. If enabled as transmitter, the link port uses a 2 deep 32-bit FIFO. If enabled as a receiver, the port uses a 4 deep 32-bit FIFO. The core MMR access bus interfaces with these FIFOs. The distributed DMA engines (DDE) use the system cross bar (SCB) interface to access the FIFO. The link port uses the LP_CTL.TRAN bit to determine whether the module is enabled for transmit or receive operation.

Architectural Concepts

The following sections describe the architectural concepts of the link port.

- FIFO Buffers
- Handshake for Link Port Enable Process
- Multi-Processor Connectivity

FIFO Buffers

When a link port is configured as transmitter, the link port uses a 2-deep FIFO buffer. A shift register unpacks the single 32-bit word to either eight 4-bit data bytes or sixteen 2-bit data bytes, based on the selected transfer mode. As the FIFO has space for more data, the link port makes a new DMA request. If the FIFO becomes empty, the LP_CLK signal is deasserted. The core can access FIFO through the LP_TX register.

The core or DMA makes three writes (2-stage FIFO and 1 shift register) to the transmit buffer before it signals a full condition. The link port uses the LP_STAT.FFST bit field to reflect the status of the FIFO but not the shift register full or empty condition. However, the program can poll the LP_STAT.LPBS bit to discover whether the link port is driving data from the shift register to the pins. The LP_STAT.LPBS bit is also set when receiver has held off transmission by driving LP_ACK low.

NOTE: When the 2-deep FIFO and the output shift-register overflow, any further write to the link port buffer overwrites the input stage of the FIFO.

NOTE: The core can also read the transmit FIFO through the data LP_TX register.

NOTE: If the transmitter is disabled while performing writes to the transmit FIFO, a FIFO full condition is signaled after two writes.

The transmit buffer registers have shadow registers. Using these shadow registers, both stages of the 2-deep FIFO can be read without updating the status registers. The LP_TXIN_SHDW register corresponds to the input stage of the FIFO. The LP_TXOUT_SHDW register corresponds to the output stage of the FIFO as shown in the ***Transmit FIFO path*** figure.

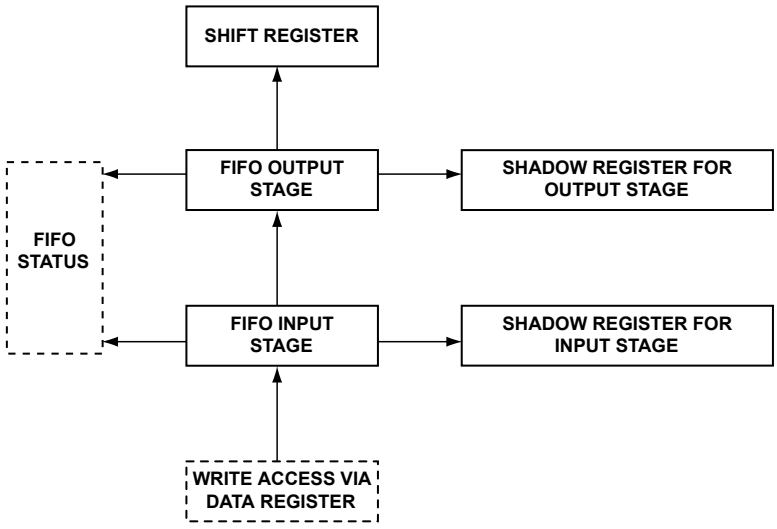


Figure 18-3: Transmit FIFO path

When a link port is configured as receiver, data transfers to the core or DMA from the full 4-deep receive FIFO. An internal packing register packs data to 32 bits. Four reads can occur from the receive buffer by the core or DMA before it signals an empty condition. The link port uses the LP_STAT.FFST bits to reflect the status of the 4-deep read buffer FIFO. The core can access this FIFO through the LP_RX register.

NOTE: When receive FIFO overflows (LP_STAT.ROVF bit=1), any further data from the transmitter is lost. Only the data retained in the receive FIFO can be retrieved further.

The receiver drives the LP_ACK output signal low, after the first byte of data for the next-to-last empty slot (in the 4-deep FIFO) is received. This functionality prevents data loss due to the transmitter starting transmission of the next word before the LP_ACK signal reaches the transmitter. (The timing is due to the larger delay in synchronization.) This functionality guarantees that even after allowing for the extra synchronization cycle in the transmitter and receiver, there is no overflow in the receive FIFO. The ***LACK Generation Based on Receive FIFO Status*** figure shows how FIFO slots influence the acknowledge signal generation. The grayed sections show received data. The white sections show empty locations where the decision to pull LP_ACK high is taken.

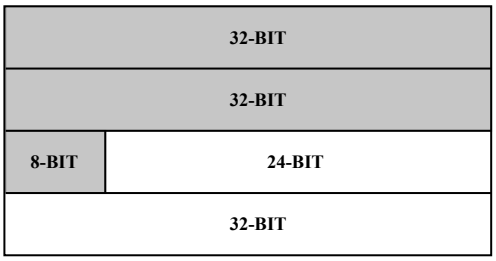


Figure 18-4: LACK Generation Based on Receive FIFO Status

NOTE: The link port uses a 4-deep receive FIFO only under a worst case situation, as mentioned. In all other cases, respond as if the FIFO has only a 3-deep stage. The LP_ACK signal is pulled high before the last stage of the FIFO.

The link port has memory-mapped buffers for both receive and transmit operations. A JTAG-based emulator can read the FIFO which can cause unexpected problems in data transfers. This activity can only happen during an emulation event (typically hitting a breakpoint or single-stepping). The emulator issues core reads through JTAG. To work around this issue, see the tools documentation for more information.

Handshake for Link Port Enable Process

In a link port-based system, the transmitter and the receiver can be enabled at different times. Use external pull-downs for the LP_CLK and LP_ACK signals.

If the receiver is enabled before the transmitter, the external pull-down holds the LP_CLK signal of the transmitter low. The receiver is held off. The receiver can wait for a rising edge on the LP_CLK signal to assert its receive service request interrupt. This rising edge occurs only when transmitter starts driving the first data on to the bus, after the application enables it.

If the transmitter is enabled before the receiver, the external pull-down holds the LP_ACK signal of the receiver low. Transmission is held off. Refer to the ***Enable the Transmitter Before the Receiver*** figure. The transmitter can wait for a rising edge on the LP_ACK signal to assert its transmit service request interrupt. This rising edge is asserted as soon as the receiver is enabled after the hardware drives the LP_ACK high.

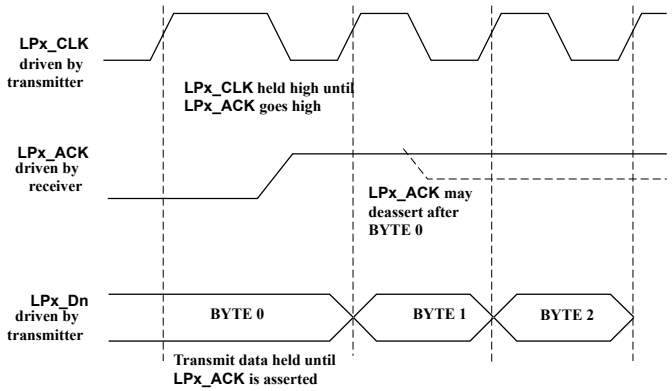


Figure 18-5: Enable the Transmitter Before the Receiver

NOTE: Service request interrupts or status are asserted only when the link port (receiver or transmitter) is disabled.

Multi-Processor Connectivity

Link ports can operate independently, allowing glueless connection with external processors. Link ports have dedicated DMA channels, allowing independent data transfers. The following group of figures shows some examples of different bus connection topology that can be used in multi-processor system design. The inter-connection methods are not limited to these examples.

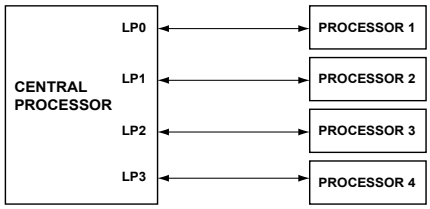


Figure 18-6: Central Processor-Based Model

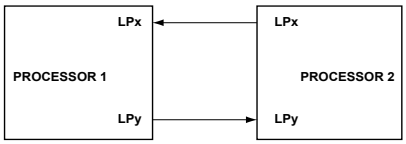


Figure 18-7: Link Port Full-Duplex Transfer Model

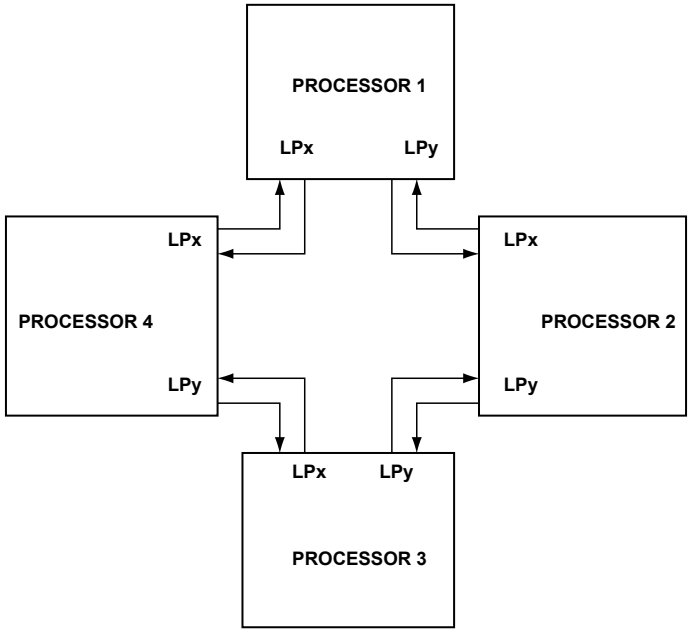


Figure 18-8: Link Port Ring Model

The link port protocol does not include built-in support for multiple requesters. However, there can be situations where multiple devices try to become the bus requester at the same time. Multi-requester conflicts can be resolved using token passing. In token passing, the token is a software flag that passes between processors.

At reset, the token is set to reside in the link port of one device, making it the requester and the transmitter. When a receiver (completer) wants to become the requester, it can assert its LP_ACK signal to get the attention of the requester. The requester knows, through the software protocol, whether to respond with actual data or whether the token is requested. If the requester wishes to give up the token, it can send back a user-defined token release word and thereafter clear its token flag.

Simultaneously, the completer sets its token and can thereafter transmit. The token release word can be any user-defined value. Because the transmitter and receiver expect a code word, this word does not need to be exclusive of normal data transmission. If the requester wishes to give up the token, it can send back a user-defined token release word and thereafter clear its token flag. Simultaneously, the completer examines the data sent back and if it is the token release word, the completer sets its token, and can thereafter transmit.

The link port protocol includes handshake mechanism to inform the other end of transfer (transmit or receive) of an enable instance. However, it does not support handshakes to inform a disabled instance, while a chunk of data transfers. The application must assume the disabled state of the other end and take appropriate action.

For example, in a multi-processing environment, a receiver did not read its full FIFO for an extended time due to internal bus arbitrations. The transmitter can require software or a peripheral timer-based timeout to inform the application that the LP_ACK signal is low for an extended period.

LP Operating Modes

The link port does not have particular modes of operation, as the peripheral is based on a simple protocol. The following sections explain the data transfer modes, using the core and using DMA.

- Core Data Transfers
- DMA Data Transfers

LP Data Transfer Modes

This section describes link port DMA and core data transfers.

Core Data Transfers

If DMA is disabled for a link port buffer, the processor core can write or read internal FIFO buffers as a memory-mapped register through the MMR access bus. To avoid FIFO overflow or underflow, the core can access the FIFO registers in one of the following ways.

- Access link port registers using an interrupt service routine (ISR) mapped to the data request interrupt of the link port. The interrupt request remains high only when the FIFO is accessible (when the FIFO is not full in transmit mode and not empty in receive mode).
- Poll the FIFO status bits of the LP_STAT register. Write to the transmit FIFO if not full or read from the receive FIFO if not empty.

DMA Data Transfers

Dedicated DMA channels are available for each link port. DMA-related activity is explained in the following steps.

1. Data Receive – Once the DMA channel and link port module are configured and enabled, the external device begins writing data to the FIFO through the data pins of the link port. The FIFO detects this activity and in turn sends a DMA request. After the request is granted, the DMA transfer progresses until the FIFO is empty.
2. Data Transmit – Once the DMA channel and link port module are configured and enabled, setting the `LP_CTL.EN` bit automatically asserts a DMA request when the transmit FIFO is empty. After the request is granted, DMA fills the FIFO. The external device begins reading data from the FIFO through the data pins of the link port. The FIFO detects that there is room in the buffer and asserts another DMA request, continuing the process.

LP Event Control

This section describes how the link port uses interrupts and status signals.

Interrupt Signals

Each link port has two dedicated interrupt lines—a data request interrupt and a status interrupt. Data request interrupts are asserted based on FIFO conditions for data transfer. Status interrupts are asserted when a service request status or an overflow status is set. The following list explains each of these interrupts.

- ***Data Request Interrupt.*** Asserted if the FIFO is not full in transmission mode and the FIFO is not empty in reception mode. This functionality serves as a core triggered interrupt in non-DMA mode and as the DMA interrupt request in DMA mode. Generation of this interrupt is based on the `LP_STAT.FFST` (status bit of the link port buffer).
- ***Link Port Transmit Service Request Interrupt (LTRQ)***. Allow a disabled link port to generate an interrupt when an external access is attempted. When a link port is configured as transmitter, the transmit service request interrupt is enabled by setting the `LP_CTL.TRQMSK` bit. When set, an external receiver can indicate to the disabled transmitter that it must receive data through the connected link port. The receiver does so by driving a high level on the `LP_ACK` line. When the `LP_ACK` of the disabled transmitter link port is detected high, a `LP_STAT.LTRQ` interrupt is generated. The transmitter can enable itself for data transfer with the receiver. The link port needs a pull-down on `LP_ACK` for this feature to function properly.
- ***Link Port Receive Service Request Interrupt (LRRQ)***. When a link port is configured as receiver, this interrupt is enabled by setting the `LP_CTL.RRQMSK` bit. When set, an external transmitter can indicate to the disabled receiver that it must receive data through the connected link port. The transmitter does so by driving out the first data. When the `LP_CLK` of the disabled receiver link port is detected high, a `LP_STAT.LRRQ` interrupt is generated. The receiver can further enable itself for data transfer with the transmitter. The link port needs a pull-down on the `LP_CLK` signal for this feature to function properly.
- ***Link Port Receive Overflow Interrupt (LPOVF)***. Generated when the receiver FIFO overflows and is enabled by setting the `LP_CTL.ROVFMSK` bit. This interrupt can happen if the transmitter continues to transmit data even though the receiver has deasserted `LP_ACK` signal causing the receive FIFO to overflow.

Enabling Link Port Interrupts

A data request interrupt is a direct interrupt and can be controlled separately from the application.

To mask the interrupt, set the mask bits in LP_CTL register corresponding to service interrupts and the overflow interrupt. These interrupts are ORed and fed to the SIC as a single LP_STAT interrupt. These interrupts are latched and stored in the associated bits of LP_STAT register. If an LP_STAT interrupt occurs, in the ISR, programs can read the LP_STAT register bits to determine the type of interrupt. These bits are write-one-to-clear (W1C); writing one to the bit resets the bit and disables the corresponding interrupt.

Status and Error Signals

This section explains the various status signals in the LP_STAT register.

- ***Transfer Status signals***. The link port uses the bus status bit (LP_STAT.LPBS) to give the status of the bus condition (busy or idle) when the link port is configured as transmitter. The LP_STAT.LPBS is high if the link port drives data into the link port pins. Programs can poll this bit after polling the LP_STAT.FFST bit to disable the link port safely.

  The link buffer status (LP_STAT.FFST) field directly indicates the status of the FIFO (including empty or full conditions) during data transfer. Software can poll this field in the LP_STAT register before writing to the FIFO (in case of transmission) or reading from the FIFO (in case of reception). The LP_STAT.FFST bit is automatically cleared when the link port is disabled.

- ***Transfer Request Status signals***. The link port uses the receive request status (LP_STAT.LRRQ) bit to indicate that an external receiver wants to receive data (in case the link port is a disabled transmitter). The link port uses the transmit request status (LP_STAT.LTRQ) bit to indicate that an external transmitter wants to send data (in case the link port is a disabled receiver). Software can poll these bits to enable the transmitter or receiver accordingly.

- ***Error Status signals***. In receive mode 32-bit data is received in eight chunks of 4-bit data. This data is then packed to a single 32-bit data before loading the FIFO. The link buffer error status (LP_STAT.LPACK) bit is high during this packing process and goes low after packing.

  The link port overflow status (LP_STAT.ROVF) bit is set when the receive FIFO overflows. This event can occur if the transmitter continues to transmit data even though the receiver has deasserted LP_ACK causing the receiver FIFO to overflow.

- *Link Port Clock Division signals*. In the SC84x family of processors, the Linkport peripheral is configured to use CLK_09 as its input clock source from the Clock Division Unit (CDU). The CDU0 input clocks can be one of the following: DCLK_0, CCLK2_0, DCLK_1, or CCLK2_1.

  The clock signal provided through CLK_09 undergoes an internal division by 4 before reaching the link port peripheral. This division occurs in two stages, with the clock being divided by 2 at each stage.

  After this division, the resulting clock serves as the input to the link port block. To further reduce the clock frequency, program the LP_DIV register using the following equation:

  LP_DIV = 0 →; LP_CLK = SCLK

LP_DIV not equal to 0; → LP_CLK = SCLK / (2 × DIV)

Here, SCLK refers to the clock entering the link port block after the initial division by 4 from CLK_09.

LP Programming Model

The following sections provide information on configuring the operating mode and enabling the link ports.

- Setting Up a DMA Transmit Operation
- Setting Up a DMA Receive Operation
- Setting Up a Core Transmit Operation
- Setting Up a Core Receive Operation

Setting Up a DMA Transmit Operation

This following procedure describes the typical steps for configuring the link ports in DMA transmit mode.

1. Enable the link port pins in the GPIO port mux using the appropriate `PORT_FER` and `PORT_MUX` registers.
2. Install interrupt handlers for DMA and for transfer status (service request interrupt).
3. Configure the link port to transmit by setting the `LP_CTL` bit and enable the transmit request interrupt mask by setting the `LP_CTL.TRQMSK` bit.
4. Program the link port clock divider by writing a value to the `LP_DIV` register.
5. If using DMA stop mode or auto buffer mode, program the appropriate DMA registers.

   ADDITIONAL INFORMATION: An example configuration is: `DMA_ADDRSTART`, `DMA_XCNT`, `DMA_XMOD`, and `DMA_CFG` registers (Stop/Auto, `DMA_CFG.PSIZE` =1, `DMA_CFG.MSIZE` =4, interrupt generation and memory read).
6. Wait for the link port receiver (connected externally) to be enabled. The application can wait for the transmit service request interrupt to assert.
7. Clear the transmit service request interrupt status by writing 1 to the `LP_STAT.LTRQ` bit.
8. Enable DMA by setting the `DMA_CFG.EN` bit.
9. Enable the link port by setting the `LP_CTL.EN` bit.
10. Wait for DMA to assert a transfer completion interrupt.
11. Clear the DMA interrupt source by writing 1 to the `DMA_STAT.IRQDONE` bit.

Setting Up a DMA Receive Operation

This section describes the typical steps for using the link ports in DMA receive mode.

1. Enable the link port pins in GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.
2. Install interrupt handlers for DMA and for transfer status (service request interrupt).
3. Configure the link port for reception (clear the LP_CTL.TRAN bit) and enable the receive request interrupt mask by setting the LP_CTL.RRQMSK bit.
4. If using DMA stop mode or auto buffer mode, program the DMA registers.

   ADDITIONAL INFORMATION: An example configuration is: DMA_ADDRSTART, DMA_XCNT, DMA_XMOD, and DMA_CFG registers (Stop/Auto, DMA_CFG.PSIZE=1, DMA_CFG.MSIZE=4, interrupt generation and memory write).
5. If using DMA array mode or list mode, create DMA configuration data structures filled with components.

   ADDITIONAL INFORMATION: An example configuration is: DMA_ADDRSTART, DMA_XCNT, DMA_XMOD, and DMA_CFG registers (Array/List, DMA_CFG.PSIZE =1, DMA_CFG.MSIZE =4, interrupt generation, memory write and fetch =4/5) and DMA_DSCPTR_NXT register (if list mode). Further, program DMA configuration register (Array/List, DMA_CFG.PSIZE =1, DMA_CFG.MSIZE =4, Memory Write and Fetch =4/5) and program the DMA_DSCPTR_NXT register (if list mode).
6. Wait for the link port transmitter (connected externally) to be enabled with subsequent data transmission. The application can wait for the receive service request interrupt to assert.
7. Clear the receive service request interrupt status by writing 1 to the LP_STAT.LRRQ bit.
8. Enable DMA by setting the DMA_CFG.EN bit.
9. Enable the link port by setting the LP_CTL.EN bit.
10. Wait for DMA to assert the transfer complete interrupt.
11. Clear the DMA interrupt source by writing 1 to the DMA_STAT.IRQDONE bit of the DMA status register.

Setting Up a Core Transmit Operation

This section describes the typical steps for using the link ports in processor core based transmission.

1. Enable the link port pins in the GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.
2. Install interrupt handlers for data transfer and for transfer status (service request interrupt). The interrupt handlers for data transfer are the same source or ID as the DMA interrupt line in the SEC.
3. Configure the link port for transmission by setting the LP_CTL.TRAN bit) and enable the transmit request interrupt mask by setting the LP_CTL.TRQMSK bit).
4. Program the link port clock divider by writing a value in to the LP_DIV register.
5. Wait for the link port receiver (connected externally) to be enabled. The application can wait for a transmit service request interrupt to assert.
6. Clear the transmit service request interrupt status by writing 1 to the LP_STAT.LTRQ bit.

7. Enable the link port by setting the LP_CTL.EN bit.
8. The data request interrupt is asserted whenever there is free space in the FIFO. The application can write to the LP_TX register based on the FIFO conditions (half or empty) reflected in the LP_STAT.FFST bit field.

Setting Up a Core Receive Operation

This section describes the typical steps for using the link ports in processor core-based reception.

1. Enable the link port pins in the GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.
2. Install interrupt handlers for data transfer and for transfer status (service request interrupt). The interrupt handlers for data transfer are the same source or ID as the DMA interrupt line in the SEC).
3. Configure link port for reception (clear LP_CTL.TRAN bit). Enable the receive request interrupt mask bit (set LP_CTL.RRQMSK).
4. Wait for the link port transmit (connected externally) to be enabled with subsequent transmission of data. The application can wait for receive service request interrupt to be asserted.
5. Clear the receive service request interrupt status by writing 1 to the LP_STAT.LRRQ bit.
6. Enable the link port by setting the LP_CTL.EN bit.
7. The data request interrupt is asserted whenever there is free space in the FIFO. The application can read from the LP_RX register based on the FIFO conditions (1 or 2 or 3 data available) which is reflected in the LP_STAT.FFST bit field.

ADSP-2184x LP Register Descriptions

Link Port (LP) contains the following registers.

Table 18-6: ADSP-2184x LP Register List

Name	Description
LP_CTL	Control Register
LP_DIV	Clock Divider Value Register
LP_RX	Receive Buffer Register
LP_STAT	Status Register
LP_TX	Transmit Buffer Register
LP_TXIN_SHDW	Shadow Input Transmit Buffer Register
LP_TXOUT_SHDW	Shadow Output Transmit Buffer Register

Control Register

The LP_CTL register provides LP interrupt masking, selection of transfer direction, and link port enable.

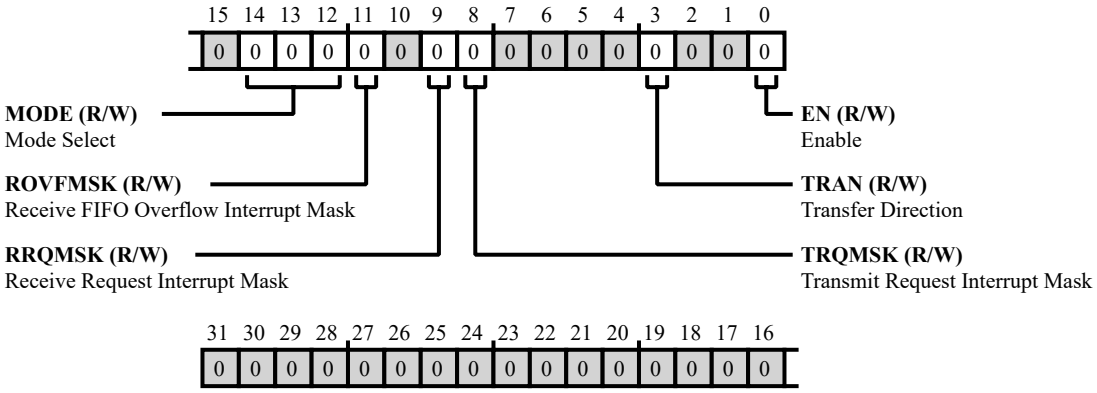


Figure 18-9: LP_CTL Register Diagram

Table 18-7: LP_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/W)	MODE	Mode Select. The LP_CTL.MODE field indicates the mode of operation for double data rates (DDR). Note that the LRRQ interrupt is not supported in DDR mode. The LP receiver must be enabled by the user before starting the LP transmitter. For LP DDR mode, the value of LP DDR clock should be twice that of LP clock. CDU CLKO9 is used for the LP DDR clock.	
		5	4 bit DDR. (D0, D1, D2 and D3 data pins are used)
		6	2 bit DDR. (D0 and D1 data pins are used)
11 (R/W)	ROVFMSK	Receive FIFO Overflow Interrupt Mask.	
		0	Mask Disable Receive FIFO Overflow Interrupt
		1	Unmask Enable Receive FIFO Overflow Interrupt
9 (R/W)	RRQMSK	Receive Request Interrupt Mask. Link Port Receive Request Mask	
		0	Mask Disable Receive Request interrupt.
		1	Unmask Enable Receive Request interrupt.
8 (R/W)	TRQMSK	Transmit Request Interrupt Mask. Link Port Transmit Request Mask	
		0	Mask Disable Transmit Request interrupt.
		1	Unmask Enable Transmit Request interrupt.

Table 18-7: LP_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TRAN	Transfer Direction. The `LP_CTL.TRAN` bit selects the transfer direction as transmit (if set) or receive (if cleared) for link buffer.	
		0	Receive Direction transfer is receive
		1	Transmit Direction transfer is transmit
0 (R/W)	EN	Enable. The `LP_CTL.EN` enables or disables the link port. When the processor disables the port (`LP_CTL.EN` transitions from high to low), the processor clears the corresponding `LP_STAT` bits.	
		0	Disable Disable linkport
		1	Enable linkport Enable linkport

Clock Divider Value Register

The LP_DIV register selects the divisor for ratio between the internal LP clock (LCLK) and system clock (SCLK). This programming is applicable only for the transmitter. The receiver can operate at any asynchronous frequency up to the maximum frequency independent of the ratio programmed.

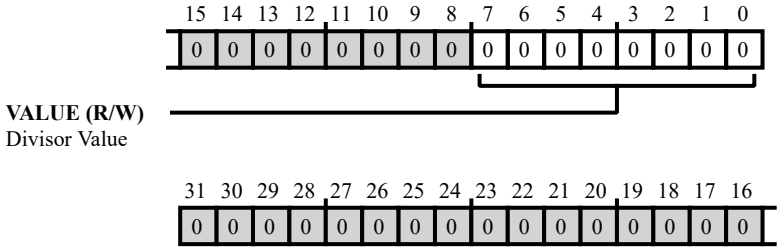


Figure 18-10: LP_DIV Register Diagram

Table 18-8: LP_DIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Divisor Value. The LP_DIV.VALUE bits select the clock divider (relating the LP' internally generated clock (LCLK) to the system clock (SCLK). The LP_DIV.VALUE should be programmed prior to LP enable. For LP_DIV.VALUE = 0, LCLK = SCLK For LP_DIV.VALUE = xxxxxxxx, LCLK = SCLK / (2 x DIV)

Receive Buffer Register

The LP_RX register buffers the receive data flow through the LP. The receive buffer is a four-location deep FIFO. In the receive buffer, data is transferred to the core or DMA from the receive FIFO where an internal register does the packing. This packing register is not software accessible. For more information on LP buffer features and operations, see the LP functional description.

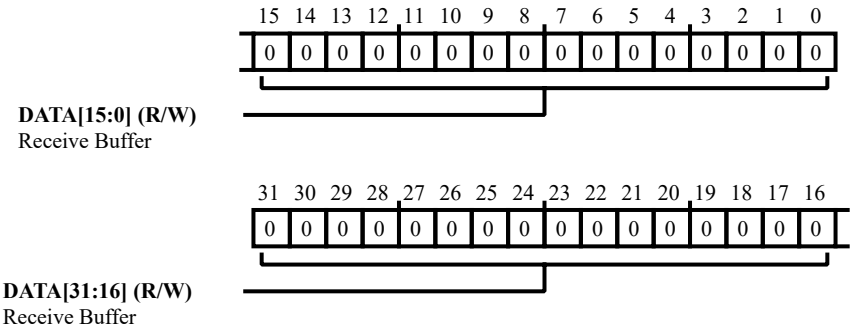


Figure 18-11: LP_RX Register Diagram

Table 18-9: LP_RX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Receive Buffer.

Status Register

The LP_STAT register provides status information on link port interrupts, FIFO, buses, and receive/transmit requests.

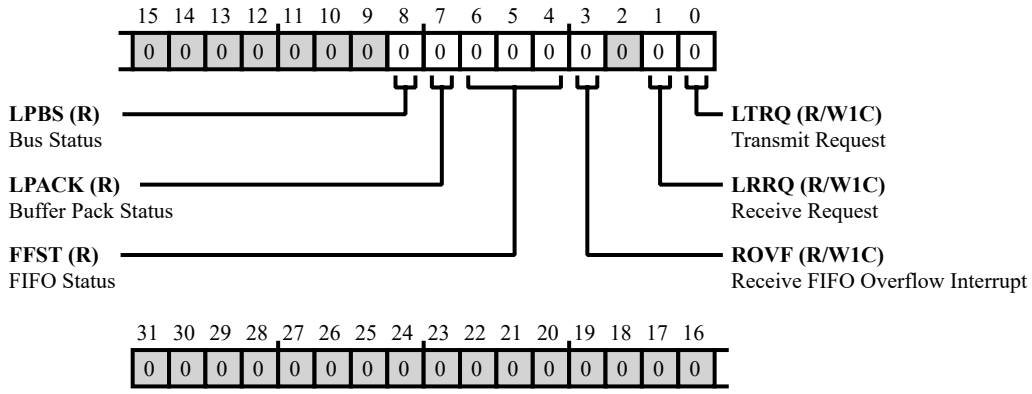


Figure 18-12: LP_STAT Register Diagram

Table 18-10: LP_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	LPBS	Bus Status. The LP_STAT.LPBS bit indicates the LPDAT bus status. LP_STAT.LPBS is kept high if data is being driven by the link port into the LP_D[n] pins.	
		0	Link port bus is idle
		1	Link port bus is busy
7 (R/NW)	LPACK	Buffer Pack Status. The LP_STAT.LPACK bit indicates packing status. In receive mode, 32-bit data is received in the block size configured in the LP_CTL.MODE bit field. Then, the data is packed to get a single 32-bit data before loading the FIFO. The LP_STAT.LPACK bit is high during this packing process and goes low after packing. In transmit mode, 32-bit data in the FIFO is unpacked to the appropriate data width before sending. The LP_STAT.LPACK is high during unpacking.	
		0	Packing Complete Packing done
		1	Packing Incomplete Packing is in progress
6:4 (R/NW)	FFST	FIFO Status. The LP_STAT.FFST bits indicate the FIFO status. These bits are cleared when the LP is disabled.	
		0	TX - Empty; RX - Empty Link buffer (TX OR RX) empty

Table 18-10: LP_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	TX - Reserved ; RX - Has 1 data word RX has 1 word of data. TX reserved
		2	TX - Reserved; RX - Has 2 data words RX has 2 word of data. TX reserved.
		3	TX - Reserved; RX - Has 3 data words RX has 3 word of data. TX reserved.
		4	TX - One Word; RX -Has 4 data words RX has 4 word of data. TX 1 word of data.
		5	Reserved
		6	TX - FIFO Full; RX - Reserved RX reserved. TX reserved.
		7	Reserved
3 (R/W1C)	ROVF	Receive FIFO Overflow Interrupt. This interrupt is generated when the receiver FIFO overflows. This overflow may happen if the transmitter continues to transmit data even though the receiver has de-asserted the LP_ACK pin.	
1 (R/W1C)	LRRQ	Receive Request. The LP generates this interrupt when the LP_CLK pin of a disabled link port (the receiver) is forced high by another link port (the transmitter). Note that the LRRQ interrupt is not supported in DDR mode.	
0 (R/W1C)	LTRQ	Transmit Request. The LP generates this interrupt when the LP_ACK pin of a disabled link port (the transmitter) is forced high by another link port (the receiver).	

Transmit Buffer Register

The LP_TX register buffers the transmit data flow through the LP. The transmit buffer is two words deep. In the transmit buffer, the input stage of the FIFO is used to accept core data or DMA data from internal memory, and the data is transferred to the link port interface from the output stage of the FIFO. The output stage performs the unpacking in the transmit buffer. The least significant byte is transmitted first. As each word is unpacked and transmitted, the next location in FIFO becomes available and a new DMA request is made if DMA is enabled. If the register becomes empty, the LP asserts the LP_CLK signal. For more information on LP buffer features and operations, see the LP functional description.

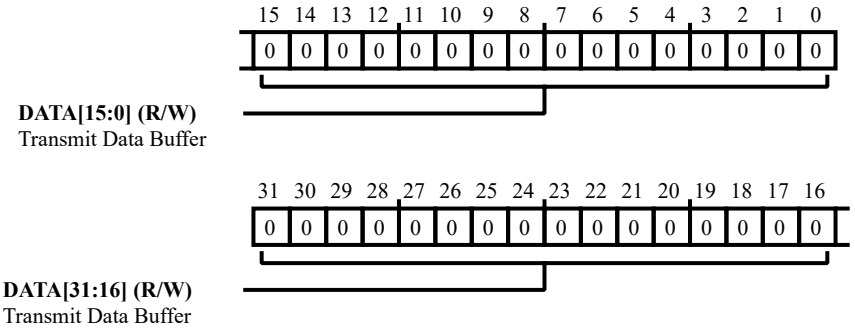


Figure 18-13: LP_TX Register Diagram

Table 18-11: LP_TX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Transmit Data Buffer.

Shadow Input Transmit Buffer Register

The LP_TXIN_SHDW register contains the same data as the input stage of the transmit buffer. Read of this shadow transmit buffer does not update the LP_STAT register.

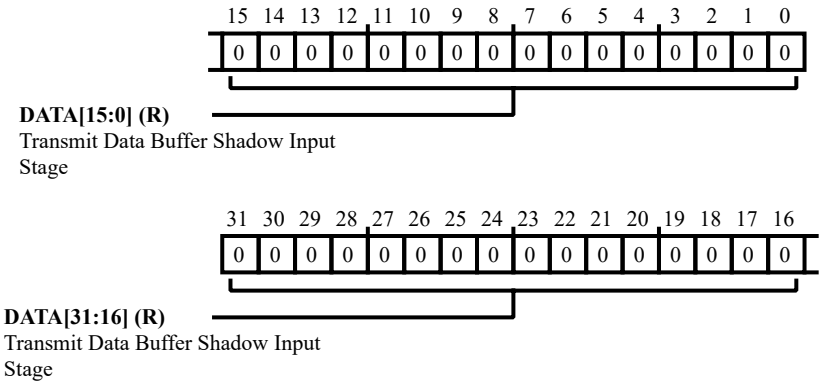


Figure 18-14: LP_TXIN_SHDW Register Diagram

Table 18-12: LP_TXIN_SHDW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Transmit Data Buffer Shadow Input Stage.

Shadow Output Transmit Buffer Register

The LP_TXOUT_SHDW register contains the same data as the output stage of the transmit buffer. Read of this shadow transmit buffer does not update the LP_STAT register.

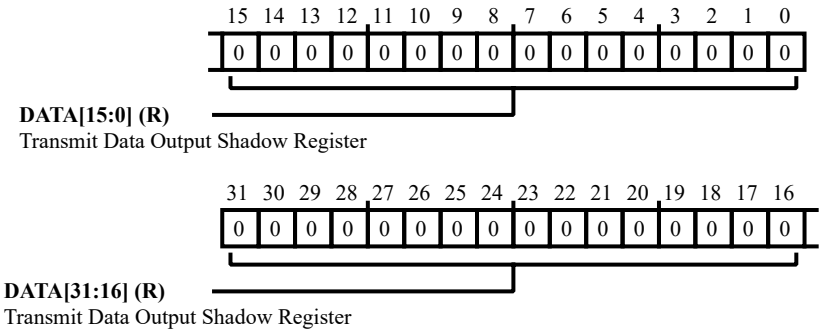


Figure 18-15: LP_TXOUT_SHDW Register Diagram

Table 18-13: LP_TXOUT_SHDW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Transmit Data Output Shadow Register.

19 Pulse Density Modulation (PDM) Microphone Interface

The pulse density modulation (PDM) microphone interface converts digital PDM microphone data to I^2S/TDM format. The microphone data in I^2S/TDM format is then routed internally to the SPORT/ASRC or externally via the DAI pins. Each DAI unit has one PDM interface.

PDM Features

The PDM interface includes the following features:

- Four channels of PDM audio inputs from digital microphones
- 16×, 32×, or 64× decimation ratio of PDM to pulse code modulation (PCM) audio data
- 24-bit resolution to support high sound pressure level (SPL) microphones
- 126 dB A-weighted SNR
- 4 kHz to 192 kHz output sampling rate
- Bit clock rates of 64×, 128×, 192×, 256×, 384×, or 512× the output sampling rate

  NOTE: Refer to the product data sheet for the maximum BCLK frequency.
- Automatic PDM clock generation
- Target I^2S or TDM output interface
- Supports up to 16 TDM slots
- Configurable TDM slot routing and sizes

PDM Functional Description

The following are the typical connections for the PDM module:

- PDM interface–PDM clock (PDM_CLK0) and PDM data (PDM_DAT0 and PDM_DAT1) are typically routed to DAI pins via the Signal Routing Unit (SRU) which is in turn connected to a PDM data source, for example, PDM microphones.
- Target I^2S/TDM interface –Bit clock (BCLK), frame sync (LRCLK), and serial data (SDATA) signals are typically routed internally to the SPORT using the SRU for the microphone data to reach memory in PCM format.
- Control registers–All PDM related control registers are included in the DAI MMR system.

ADSP-2184x PDM Register List

Pulse Density Modulation interface block

Table 19-1: ADSP-2184x PDM Register List

Name	Description
PDM_CTL0	PDM Control Register
PDM_HPF_CTL	High Pass Filter Control Register
PDM_RESET	Software Reset Register
PDM_SP_CTL0	Serial Port Control0 Register
PDM_SP_CTL1	Serial Port Control1 Register

PDM Block Diagram

The *PDM Interface Block Diagram* figure illustrates the interconnection of PDM block in the DAI subsystem.

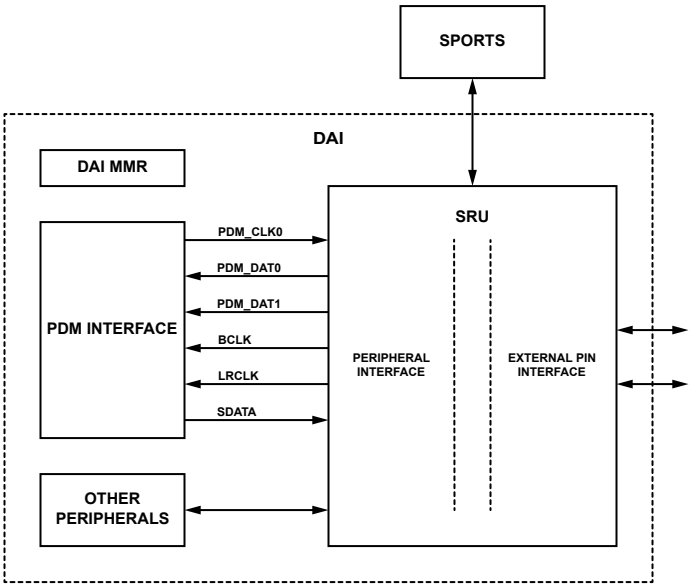


Figure 19-1: PDM Interface Block Diagram

PDM Architectural Concepts

The PDM interface provides up to four channels of decimation from a 1-bit PDM source to a 24-bit PCM audio stream. The down sampling ratio is typically 64 × f_S, where f_S is the PCM output sampling rate. The down sampling ratio can also be set at 32 × f_S or 16 × f_S, to facilitate higher output sampling rates. All channels decimate at the same ratio. The 24-bit down-sampled PCM audio data is output via standard stereo (I^2S, left justified, right justified) or TDM format. The input sources for the PDM interface can be any device that has a target PDM output, such as a digital microphone.

Internally, there are four channels. The *PDM Channel Mapping* table details the mapping of the PDM_DATx input pins to the internal channels. If a channel is not in use, the associated PDM_DATx pin can be disabled to save power.

Table 19-2: PDM Channel Mapping

Input Pin	PDM_CLK Edge	Internal Channel
PDM_DAT0	Falling	0
PDM_DAT0	Rising	1
PDM_DAT1	Falling	2
PDM_DAT1	Rising	3

PDM Initialization and Clocking

After I^2S clocks are applied, there are 16 full frame sync cycles of initialization time before the PDM clocks start. Once the PDM clocks start running, 48 additional frame sync cycles elapse before the PDM data is driven on the SDATA pin.

During normal operation, if the bit clock or frame sync is removed then the PDM clock output stops immediately and the PDM interface enters a lower power state. If the clocks resume, the PDM interface relocks to the bit clock and frame sync signals and adjusts the PDM clock output accordingly. The length of time before the PDM clock outputs resume is 4 frames ± 1 frame to lock to the incoming signal. If the format of the clock signals change, the PDM interface detects this change at the end of the frame and stops the PDM clock outputs. Then, the device reconfigures and resumes sending PDM clocks with no user intervention. Again, the PDM clock outputs resume after 4 frames ±1 frame to lock to the incoming signal.

The PDM interface requires a clock rate that is a minimum of 64× the frame sync sample rate. Bit clock rates of 128×, 192×, 256×, 384×, and 512× the frame sync rate are also supported. The PDM interface automatically detects the ratio between the bit clock and frame sync signals and generates a PDM clock output at 64 × the frame synce rate by default. If lower decimation ratios are selected in the `PDM_CTL0` register, the PDM output clock rate corresponds to the configuration set in the `PDM_CTL0.DEC_RATIO` bit field. The minimum sampling rate is 4 kHz and the maximum sampling rate is 192 kHz. The PDM clock range is 256 kHz to 6.144 MHz. Internally, all processing is done at the PDM clock rate.

SPORT Interface

The PDM interface supports I^{2}S and TDM serial output formats. It supports up to 16 TDM slots, with slot widths of 16, 24, or 32-bits. Internal channels are connected to output slots by configuring the appropriate channel slot bit fields in the PDM_SP_CTL1 register. By default, each channel is routed to the slot with the same number. For example, the reset value of the PDM_SP_CTL1.CH3_SLOT bit field is 0x0011, connecting channel 3 to slot 3.

Each channel has a drive select bit: PDM_SP_CTL1.CH0_DRV, PDM_SP_CTL1.CH1_DRV, PDM_SP_CTL1.CH2_DRV, or PDM_SP_CTL1.CH3_DRV. If the drive select bit is set to one, the channel is driven on the serial port in the appropriate output slot; and if it is cleared to zero, the channel does not drive (tristate high impedance mode). Note that this feature is useful only when the SPORT data is driven out of the processor via a DAI pin buffer and not when the data is routed internally to the SPORT or ASRC. To use this feature, tie the PDMx_SDATA_OE_O signal to the DAIx_PBENxx_I signal of the DAI pin buffer.

It is possible to erroneously configure more than one channel to drive on a single TDM slot. There is no cross-checking of register settings to prevent this configuration, but it will not damage the device. In this case, only the data from the lowest channel number is driven into the slot and the data from the higher channels will not be driven.

The PDM_SP_CTL0.SAI_MODE bit configures the SPORT interface mode. The two modes are stereo and TDM. The primary difference between these two modes is the format of the frame sync clock that is expected and the polarity of the active edge of the clock.

With the PDM_SP_CTL0.SAI_MODE bit and the PDM_SP_CTL0.LRCLK_POL bit set to zero, the serial port is in stereo mode with the clock polarity set to normal. In this mode, there must be only two channels of data sent. The frame starts with the falling edge of the frame sync, with an expected cycle of 50% high and 50% low. Channel 0 sends out data when the clock is low. As soon as the frame sync goes high the data from channel 0 stops and channel 1 begins sending data. Both edges of the frame sync clock are used. If the duty cycle is not 50/50, there may be errors in the resulting data. In this mode of operation, the PDM interface does not expect 32-bit clock transitions for each channel. All bit clock to frame sync ratios are supported.

With the PDM_SP_CTL0.SAI_MODE bit set to one and the PDM_SP_CTL0.LRCLK_POL bit set to zero, the serial port is in TDM mode with the clock polarity set to normal. In this mode, it can transmit anywhere from one channel to as many as four channels spread out across 16 data slots in TDM-16 format.

The PDM interface supports bit clock rates of 64×, 128×, 192×, 256×, 384×, or 512× the output sampling rate. The BCLK frequency however, must not exceed the maximum value specified in the data sheet. These bit clock rates are combined with the three different TDM slot sizes of 16-bit, 24-bit, or 32-bit slots, selected by the PDM_SP_CTL0.CH0_SLOT_WIDTH bit field. Note that as soon as the next frame sync edge is detected, the PDM interface restarts from slot 0 and any unreceived data from the previous frame is lost. This is how to achieve unusual TDM formats like TDM-5 or TDM-10. In addition, there is only support for TDM-16 or less for placing data into a TDM slot. Data cannot be placed into slots above 16, however it is possible to configure the PDM interface to tristate all unused TDM slots, which includes all the slots above the first 16 slots for modes that have more than 16 slots.

In TDM mode, the frame sync is expected to be a positive going pulse that is at least one-bit clock period wide. A falling edge is not necessary, it is only important that the signal is low long enough to meet the timing specification for a read before transitioning from low to high. The frame starts with the rising edge of this pulse. The data is clocked out according to the slot width and using the data format specified in the PDM_SP_CTL0 register. The PDM interface continues to send data until all active channels are sent and then the device waits for the next frame sync clock edge to start sending the next set of frame samples.

In TDM-8 format, when the PDM interface is set to output channel 0 through channel 3 into slot 0 to slot 3, the PDM interface can tristate for the remainder of the frame. This allows another PDM to output four channels onto slot 4 through slot 7. These slots do not have to be consecutive, and the two devices can be configured to interleave their respective data. The serial port can be set up to drive only when there is data to drive into a data slot. If all eight channels are not in use, configuring the drive select bit in the PDM_SP_CTL1 for the unused channels will assign those slots to drive or tristate in the TDM data stream. Note that this feature is useful only when the SPORT data is driven out of the chip via a DAI pin buffer and not when the data is routed internally to the SPORT or ASRC. To use this feature, tie the PDMx_SDATA_OE_O signal to the DAIx_PBENxx_I signal of the DAI pin buffer.

To invert the IRCLK polarity, set the PDM_SP_CTL0.LRCLK_POL bit to one. With inverted clock polarity in stereo mode, channel 0 is sent out when the frame sync is high, so the start of the frame is a low to high transition. With inverted clock polarity in TDM mode, the frame sync pulse is negative, so the frame starts with the high to low transition.

Configure the PDM_SP_CTL0.DATA_FORMAT bit field to align data within the 32-bit data slot. There is support for left justified mode, delayed by one-bit clock period; and right justified modes for 24-bit, 20-bit, and 16-bit data word sizes.

SPORT Timing

The *Stereo Mode* figure is a timing diagram for I^{2}S mode with 24-bit data.

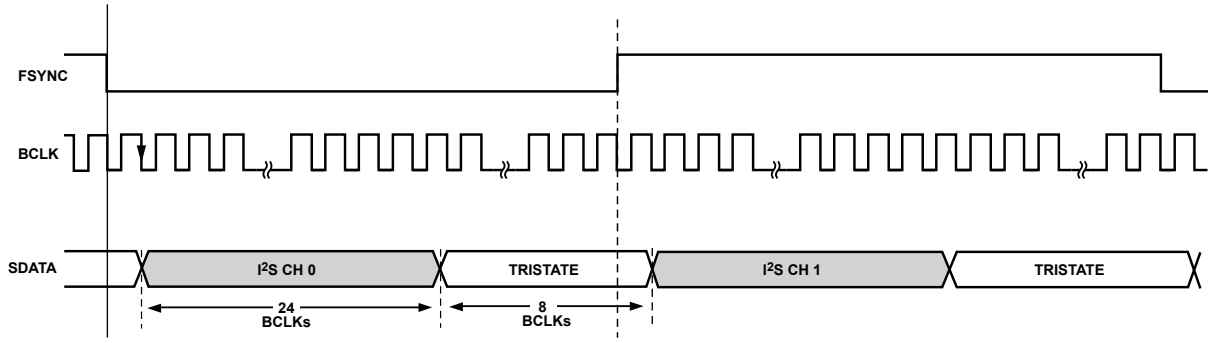


Figure 19-2: Stereo Mode

The *TDM4 Mode Four Channel* figure is a timing diagram for TDM4 mode with default channel assignments, 24-bit data, and 32-bit slots, and normal polarity clocks.

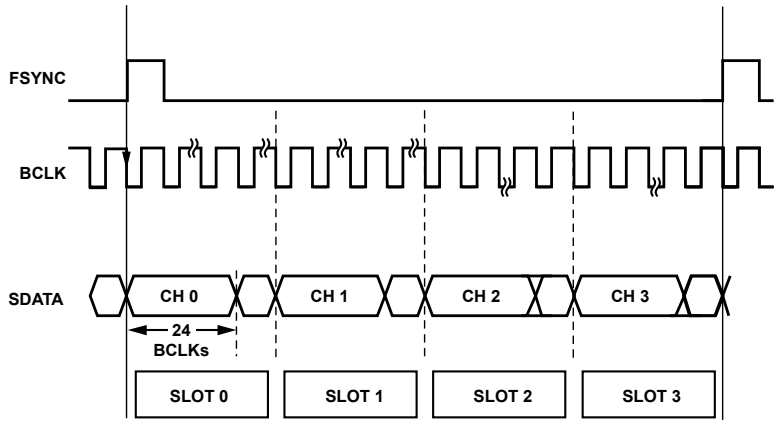


Figure 19-3: TDM4 Four Channel

The ***TDM4 Mode Two Channel*** figure is a timing diagram that shows TDM4 mode with only two channels enabled, 24-bit data delayed by zero BCLK, 32-bit slots, and normal polarity clocks.

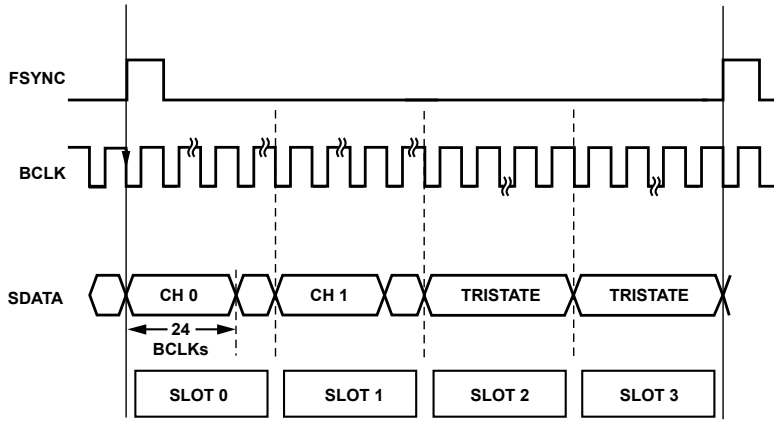


Figure 19-4: TDM4 Two Channel

The ***TDM8 Mode Four Channel*** figure is a timing diagram that shows TDM8 mode with all the four channels enabled, 24-bit data delayed by zero BCLK, 32-bit slots, and normal polarity clocks.

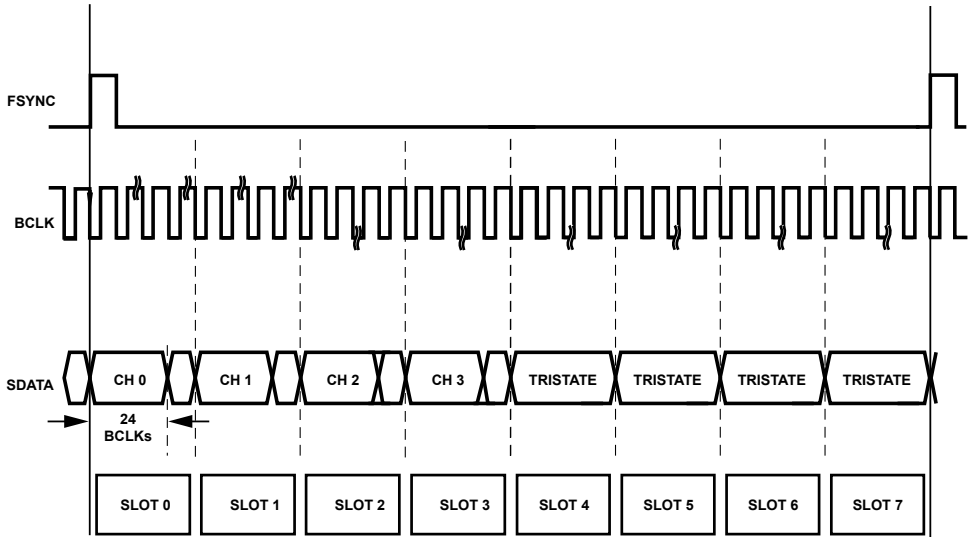


Figure 19-5: TDM8 Four Channels

The *TDM2 Mode Two Channel* figure is a timing diagram that shows TDM2 mode with two channels enabled, 24-bit data delayed by one BCLK, 32-bit slots, and normal polarity clocks.

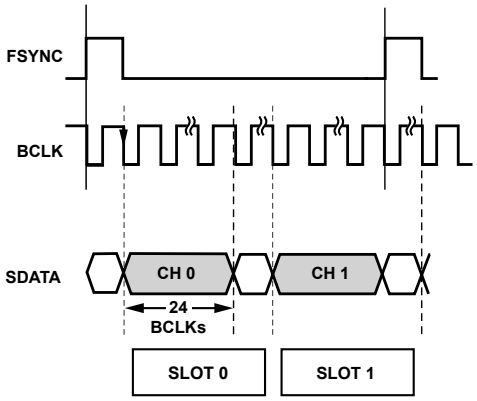


Figure 19-6: TDM2 Two Channels

High Pass Filter

There is a first order high-pass filter in the signal path. It is disabled by default and can be enabled by setting the `PDM_HPF_CTL.EN` bit. To adjust the cutoff frequency, configure the `PDM_HPF_CTL.FC` bit field. The settings are relative to the output sampling rate.

The *HPF Cutoff Frequency Selection* table shows the setting and the cutoff frequency for common sampling rates.

Table 19-3: HPF Cutoff Frequency Selections

PDM_HPF_CTL.HPF_FC	Multiplication Factor	48 kHz Sampling Rate Cutoff Frequency (Hz)	32 kHz Sampling Rate Cutoff Frequency (Hz)
0101	0.00505	242.4	161.6
1001	0.00251	120.48	80.32

Table 19-3: HPF Cutoff Frequency Selections (Continued)

PDM_HPF_CTL.HPF_FC	Multiplication Factor	48 kHz Sampling Rate Cutoff Frequency (Hz)	32 kHz Sampling Rate Cutoff Frequency (Hz)
0111	0.00125	60	40
1000	0.000623	29.904	19.936
1001	0.000311	14.928	9.952
1010	0.000155	7.44	4.96
1011	0.0000777	3.7296	2.4864
1100	0.0000389	1.8672	1.2448
1101	0.0000194	0.9312	0.6208
1110	0.00000971	0.46608	0.31072
1111	0.00000468	0.23328	0.15552

PDM Programming Model

The following section provides information on configuring the PDM interface.

Configuring the PDM Interface

Use the following steps to configure the PDM interface:

Optional high pass filter configuration:

- Configure the PDM_HPF_CTL register to enable the high pass filter using the PDM_HPF_CTL.EN bit and set the cutoff frequency by programming the PDM_HPF_CTL.FC bit field.

1. Configure PDM_SP_CTL0 register to set the following parameters:
    a. SPORT mode (Stereo/TDM) with the PDM_SP_CTL0.SAI_MODE bit
    b. SPORT data format (BCLK delay) with PDM_SP_CTL0.DATA_FORMAT bit field
    c. SPORT slot width (16, 24, or 32 bits) with PDM_SP_CTL0.CH0_SLOT_WIDTH bit field – applicable only for the TDM mode
    d. BCLK capture edge (rising/falling) with the PDM_SP_CTL0.BCLK_POL bit
    e. LRCLK polarity (normal/inverted) with the PDM_SP_CTL0.LRCLK_POL bit
2. Configure the PDM_SP_CTL1 register to set the following parameters:
    a. PDM to SPORT channel slot mapping with serial port channel slot bit fields (CHx_SLOT)
    b. To select if the PDM channel will get driven to the associated SPORT channel slot with the serial port channel drive select bit (CHx_DRV).

3. Configure the PDM_CTL0 register to configure the decimation ratio with the PDM_CTL0.DEC_RATIO bit field and finally enable the PDM clock with the PDM_CTL0.CLK0_EN bit.

Software Reset

The PDM_RESET.SOFT bit is set to perform a soft reset of the PDM block. Setting this bit resets internal bit clock counters and audio data, but it does not reset the PDM register settings. This bit is self-clearing.

ADSP-2184x PDM Register Descriptions

Pulse Density Modulation interface block (PDM) contains the following registers.

Table 19-4: ADSP-2184x PDM Register List

Name	Description
PDM_CTL0	PDM Control Register
PDM_HPF_CTL	High Pass Filter Control Register
PDM_RESET	Software Reset Register
PDM_SP_CTL0	Serial Port Control0 Register
PDM_SP_CTL1	Serial Port Control1 Register

PDM Control Register

The PDM_CTL0 register (read/write) controls the channel activation, clock enable and Decimation ratio.

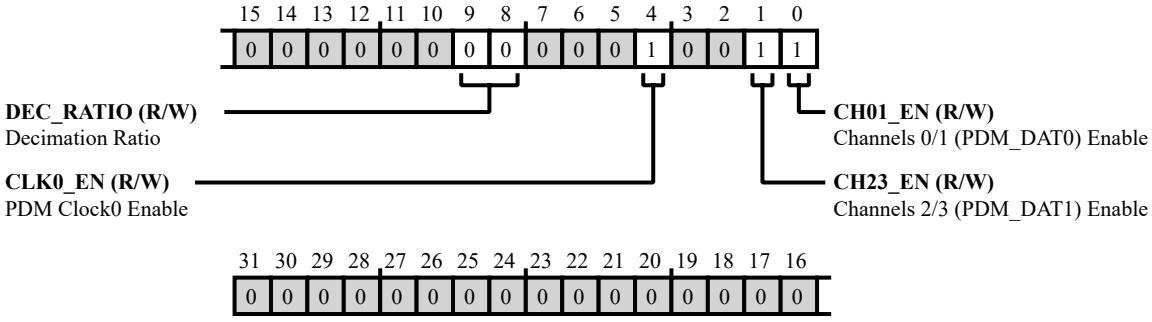


Figure 19-7: PDM_CTL0 Register Diagram

Table 19-5: PDM_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	DEC_RATIO	Decimation Ratio.	
		0	64 times decimation
		1	32 times decimation
		2	16 times decimation
4 (R/W)	CLK0_EN	PDM Clock0 Enable.	
		0	PDM_CLK0 output disabled
		1	PDM_CLK0 output enabled
1 (R/W)	CH23_EN	Channels 2/3 (PDM_DAT1) Enable.	
		0	PDM channels 2/3 disabled
		1	PDM channels 2/3 enabled
0 (R/W)	CH01_EN	Channels 0/1 (PDM_DAT0) Enable.	
		0	PDM channels 0/1 disabled
		1	PDM channels 0/1 enabled

High Pass Filter Control Register

The PDM_HPF_CTL register (read/write) controls the High Pass Filter.

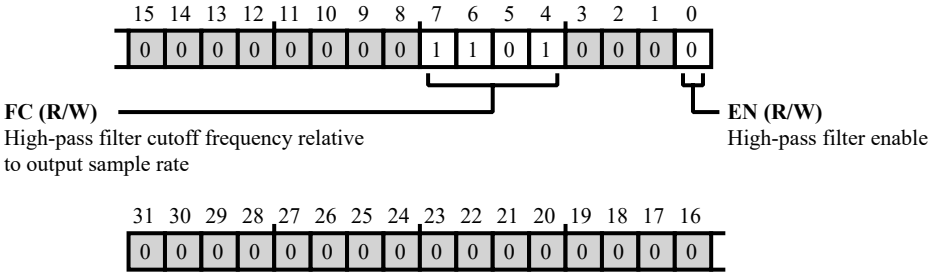


Figure 19-8: PDM_HPF_CTL Register Diagram

Table 19-6: PDM_HPF_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/W)	FC	High-pass filter cutoff frequency relative to output sample rate.	
		0	Reserved
		1	Reserved
		2	Reserved
		3	Reserved
		4	Reserved
		5	0.00505
		6	0.00251
		7	0.00125
		8	0.000623
		9	0.000311
		10	0.000155
		11	0.0000777
		12	0.0000389
		13	0.0000194
		14	0.00000971
		15	0.00000486
0 (R/W)	EN	High-pass filter enable.	
		0	High Pass Filter Off
		1	High Pass Filter On

Software Reset Register

The PDM_RESET register (read/write) controls the software reset operations.

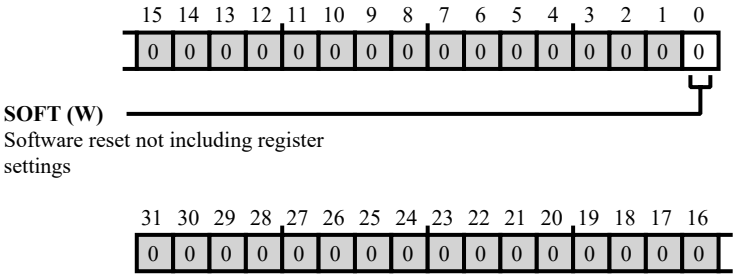


Figure 19-9: PDM_RESET Register Diagram

Table 19-7: PDM_RESET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (RX/W)	SOFT	Software reset not including register settings.	
		0	N/A
		1	Write once to soft reset

Serial Port Control0 Register

The PDM_SP_CTL0 register (read/write) controls serial port mode, data format and slot width.

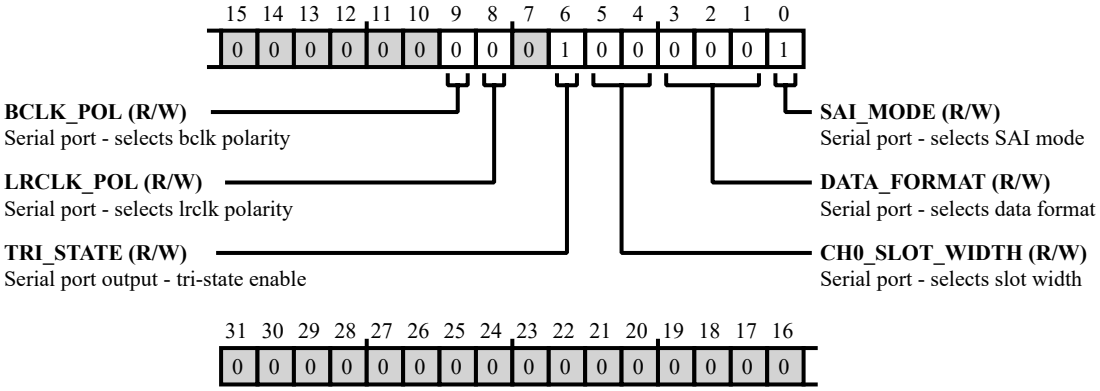


Figure 19-10: PDM_SP_CTL0 Register Diagram

Table 19-8: PDM_SP_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	BCLK_POL	Serial port - selects bclk polarity.	
		0	Capture on rising edge
		1	Capture on falling edge
8 (R/W)	LRCLK_POL	Serial port - selects lrclk polarity.	
		0	Normal Polarity
		1	Inverted Polarity
6 (R/W)	TRI_STATE	Serial port output - tri-state enable.	
		0	tri-state disabled
		1	tri-state enabled
5:4 (R/W)	CH0_SLOT_WIDTH	Serial port - selects slot width.	
		0	32 BCLK's per slot
		1	16 BCLK's per slot
		2	24 BCLK's per slot
3:1 (R/W)	DATA_FORMAT	Serial port - selects data format.	
		0	Typical I2S mode, delay by 1
		1	Left Justified,delay by 0
		2	Delay by 8
		3	Delay by 12

Table 19-8: PDM_SP_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	Delay by 16
0 (R/W)	SAI_MODE	Serial port - selects SAI mode.	
		0	STEREO (I2S, LJ or RJ)
		1	TDM

Serial Port Control1 Register

The PDM_SP_CTL1 register (read/write) controls the mapping of channel 0/1/2/3 to TDM slots.

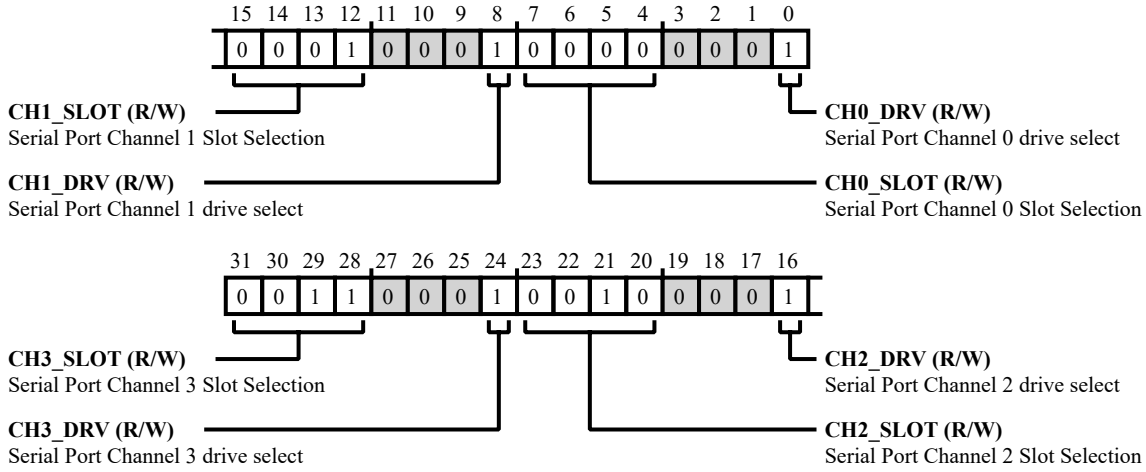


Figure 19-11: PDM_SP_CTL1 Register Diagram

Table 19-9: PDM_SP_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/W)	CH3_SLOT	Serial Port Channel 3 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14

Table 19-9: PDM_SP_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		15	Map channel to TDM slot 15
24 (R/W)	CH3_DRV	Serial Port Channel 3 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.
23:20 (R/W)	CH2_SLOT	Serial Port Channel 2 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14
		15	Map channel to TDM slot 15
16 (R/W)	CH2_DRV	Serial Port Channel 2 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.
15:12 (R/W)	CH1_SLOT	Serial Port Channel 1 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right

Table 19-9: PDM_SP_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14
		15	Map channel to TDM slot 15
8 (R/W)	CH1_DRV	Serial Port Channel 1 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.
7:4 (R/W)	CH0_SLOT	Serial Port Channel 0 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10

Table 19-9: PDM_SP_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14
		15	Map channel to TDM slot 15
0 (R/W)	CH0_DRV	Serial Port Channel 0 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.

20 Serial Peripheral Interface (SPI)

The serial peripheral interface is an industry-standard synchronous serial link that supports communication with multiple SPI-compatible devices. The baseline SPI peripheral is a synchronous, four-wire interface consisting of two data pins, one device select pin, and a gated clock pin. The two data pins allow full-duplex operation to other SPI-compatible devices. Two extra (optional) data pins are provided on specific SPIs to support quad SPI operation. Enhanced modes of operation such as flow control, fast mode, and dual-I/O mode (DIOM) are also supported. In addition, a direct memory access (DMA) mode allows for transferring several words with minimal CPU interaction.

With a range of configurable options, the SPI ports provide a glueless hardware interface with other SPI-compatible devices in controller mode, target mode, and multi-controller environments. The SPI peripheral includes programmable baud rates, clock phase, and clock polarity. The peripheral can operate in a multi-controller environment by interfacing with several other devices, acting as either a controller device or a target device. In a multi-controller environment, the SPI peripheral uses open-drain outputs to avoid data bus contention. The flow control features enable slow target devices to interface with fast controller devices by providing an SPI ready pin which flexibly controls the transfers.

NOTE: SPI peripherals on the processor operate in the CLKO6 domain. For more details on CLKO6 programming, refer the *Clock Generation Unit (CGU)* chapter.

SPI Features

The SPI module supports the following features:

- Full-duplex, synchronous serial interface
- 8, 16-bit and 32-bit word sizes
- Programmable baud rate, clock phase, and polarity
- Programmable interframe latency
- Flow control
- Support for Fast and DIOM modes
- SPI1 and SPI2 support quad mode and memory-mapped mode. Memory-mapped mode is only supported by SPI1/SPI2.

- Independent receive and transmit DMA channels
- Burst transfer mode for non-DMA write accesses

SPI Functional Description

This section provides information on the function of the SPI module.

Shift register functionality

The SPI is a shift register that serially transmits and receives data bits to or from other SPI devices. During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines.

Controller/target functionality

During a data transfer, one SPI system acts as the link controller which controls the data flow. The other system acts as the target, which has data shifted into and out of it by the controller. Different devices can take turn being controllers, and one controller can simultaneously shift data into multiple targets (broadcast mode). However, only one target can drive its output to write data back to the controller at any given time. This rule must be enforced in the broadcast mode. Several targets can be selected to receive data from the controller in this mode. But, only one target can be enabled to send data back to the controller.

Enhanced operating modes

SPI supports enhanced modes of operation like fast mode, DIOM, and Quad-SPI, and optional flow control. In fast mode, received data is sampled on the transmit edge instead of the standard receive edge, thus enabling a full-cycle path for the received data. In DIOM, both COTI and CITO are configured as input or output pins, and 2 bits are shifted in or out on each receive or transmit edge. In Quad-SPI mode, SPI_D3:0 are configured as input or output pins and 4 bits are shifted in or out on each receive or transmit edge. A slower target can use flow control to stall a faster controller device.

Single and multi-controller use

The SPI can be used in a single controller as well as multi-controller environment. The `SPI_COTI`, `SPI_CITO`, and the `SPI_CLK` signals are all tied together in both configurations. SPI transmission and reception can be enabled simultaneously or individually, depending on `SPI_RXCTL` and `SPI_TXCTL` settings. In broadcast mode, several target can be enabled to receive, but only one target must be in transmit mode and driving the `SPI_CITO` line.

ADSP-2184x SPI Register List

The Serial Peripheral Interface (SPI) provides a full-duplex, synchronous serial interface, which supports both controller/target modes and multi-controller environments. The SPI baud rate and clock phase/polarities are

programmable, and it has integrated DMA channels for both transmit and receive data streams. A set of registers governs SPI operations. For more information on SPI functionality, see the SPI register descriptions.

Table 20-1: ADSP-2184x SPI Register List

Name	Description
SPI_CLK	Clock Rate Register
SPI_CTL	Control Register
SPI_DLY	Delay Register
SPI_ILAT	Masked Interrupt Condition Register
SPI_ILAT_CLR	Masked Interrupt Clear Register
SPI_IMSK	Interrupt Mask Register
SPI_IMSK_CLR	Interrupt Mask Clear Register
SPI_IMSK_SET	Interrupt Mask Set Register
SPI_MMRDH	Memory Mapped Read Header
SPI_MMTOP	SPI Memory Top Address
SPI_RFIFO	Receive FIFO Data Register
SPI_RWC	Received Word Count Register
SPI_RWCR	Received Word Count Reload Register
SPI_RXCTL	Receive Control Register
SPI_TSEL	Target Select Register
SPI_STAT	Status Register
SPI_TFIFO	Transmit FIFO Data Register
SPI_TWC	Transmitted Word Count Register
SPI_TWCR	Transmitted Word Count Reload Register
SPI_TXCTL	Transmit Control Register

ADSP-2184x SPI Interrupt List

Table 20-2: ADSP-2184x SPI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
230	SPI0_TXDMA	SPI0 TX DMA Channel	Level	22
231	SPI0_RXDMA	SPI0 RX DMA Channel	Level	23
232	SPI0_STAT	SPI0 Status	Level	
233	SPI0_ERR	SPI0 Error	Level	

Table 20-2: ADSP-2184x SPI Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
234	SPI1_TXDMA	SPI1 TX DMA Channel	Level	24
235	SPI1_RXDMA	SPI1 RX DMA Channel	Level	25
236	SPI1_STAT	SPI1 Status	Level	
237	SPI1_ERR	SPI1 Error	Level	
238	SPI2_TXDMA	SPI2 TX DMA Channel	Level	26
239	SPI2_RXDMA	SPI2 RX DMA Channel	Level	27
240	SPI2_STAT	SPI2 Status	Level	
241	SPI2_ERR	SPI2 Error	Level	
242	SPI5_TXDMA	SPI5 TX DMA Channel	Level	59
243	SPI5_RXDMA	SPI5 RX DMA Channel	Level	60
244	SPI5_STAT	SPI5 Status	Level	
245	SPI5_ERR	SPI5 Error	Level	
246	SPI0_TXDMA_ERR	SPI0 TX DMA Channel Error	Level	
247	SPI0_RXDMA_ERR	SPI0 RX DMA Channel Error	Level	
248	SPI1_TXDMA_ERR	SPI1 TX DMA Channel Error	Level	
249	SPI1_RXDMA_ERR	SPI1 RX DMA Channel Error	Level	
250	SPI2_TXDMA_ERR	SPI2 TX DMA Channel Error	Level	
251	SPI2_RXDMA_ERR	SPI2 RX DMA Channel Error	Level	
252	SPI5_TXDMA_ERR	SPI5 TX DMA Error	Level	
253	SPI5_RXDMA_ERR	SPI5 RX DMA Error	Level	

ADSP-2184x SPI Trigger List

Table 20-3: ADSP-2184x SPI Trigger List Generators

Trigger ID	Name	Description	Sensitivity
117	SPI0_TXDMA	SPI0 TX DMA Channel	Edge
118	SPI0_RXDMA	SPI0 RX DMA Channel	Edge
119	SPI1_TXDMA	SPI1 TX DMA Channel	Edge
120	SPI1_RXDMA	SPI1 RX DMA Channel	Edge
121	SPI2_TXDMA	SPI2 TX DMA Channel	Edge
122	SPI2_RXDMA	SPI2 RX DMA Channel	Edge

Table 20-3: ADSP-2184x SPI Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
123	SPI5_TXDMA	SPI5 TX DMA Channel	Edge
124	SPI5_RXDMA	SPI5 RX DMA Channel	Edge

Table 20-4: ADSP-2184x SPI Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
149	SPI0_TXDMA	SPI0 TX DMA Channel	Pulse
150	SPI0_RXDMA	SPI0 RX DMA Channel	Pulse
151	SPI1_TXDMA	SPI1 TX DMA Channel	Pulse
152	SPI1_RXDMA	SPI1 RX DMA Channel	Pulse
153	SPI5_TXDMA	SPI5 TX DMA Channel	Pulse
154	SPI5_RXDMA	SPI5 RX DMA Channel	Pulse
155	SPI2_TXDMA	SPI2 TX DMA Channel	Pulse
156	SPI2_RXDMA	SPI2 RX DMA Channel	Pulse

ADSP-2184x SPI DMA Channel List

Table 20-5: ADSP-2184x SPI DMA Channel List

DMA ID	DMA Channel Name	Description
DMA22	SPI0_TXDMA	SPI0 TX DMA Channel
DMA23	SPI0_RXDMA	SPI0 RX DMA Channel
DMA24	SPI1_TXDMA	SPI1 TX DMA Channel
DMA25	SPI1_RXDMA	SPI1 RX DMA Channel
DMA26	SPI2_TXDMA	SPI2 TX DMA Channel
DMA27	SPI2_RXDMA	SPI2 RX DMA Channel
DMA59	SPI5_TXDMA	SPI5 TX DMA Channel
DMA60	SPI5_RXDMA	SPI5 RX DMA Channel

SPI Block Diagram

The ***SPI Controller Block Diagram*** illustrates the blocks of the SPI module. The module is comprised of three primary parts:

- SPI core contains the receive and transmit FIFOs and their associated shift registers
- Control blocks contain the synchronizer and logic to control the data flow through the data pipelines

- Register block

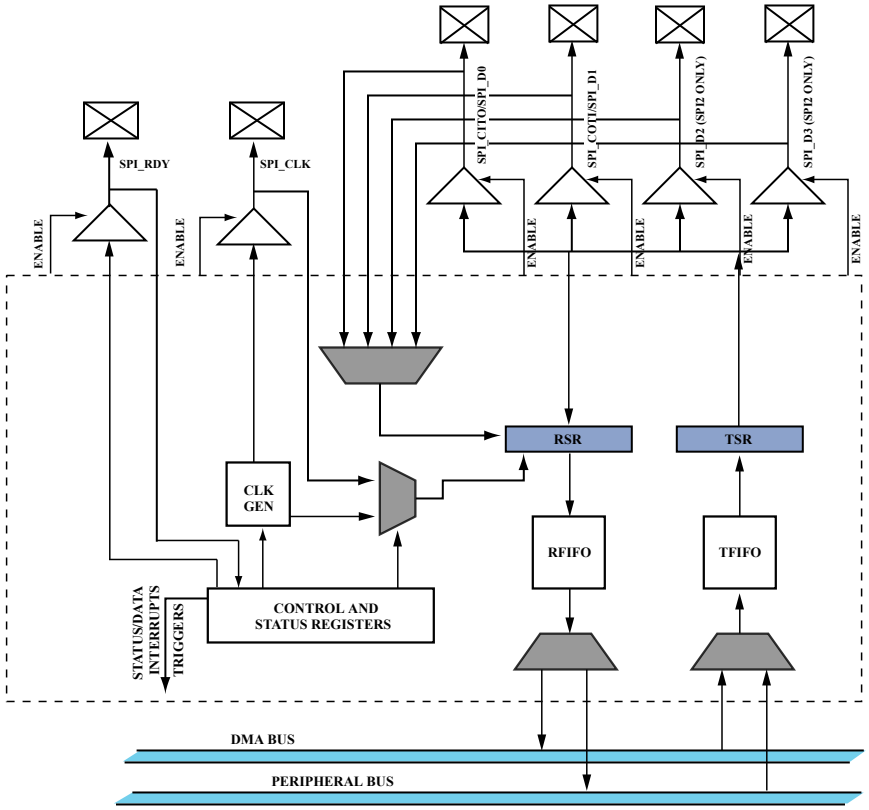


Figure 20-1: SPI Controller Block Diagram, Quad Mode

Transfer Protocol

The SPI module implements two channels that are independent of each other. The SPI module uses the SPI_RXCTL and SPI_TXCTL dedicated control registers to control these channels. Except in dual and quad modes, SPI can enable and use both channels simultaneously.

The SPI protocol supports four different combinations of serial clock phase and polarity. These combinations are selected through the SPI_CTL.CPOL and SPI_CTL.CPHA bits.

The ***SPI Transfer Protocol*** figures demonstrate the two basic transfer formats as defined by the CPHA bit. Two waveforms are shown for SPI_CLK; one for SPI_CTL.CPOL=0 and the other for SPI_CTL.CPOL=1. The diagrams can be interpreted as controller or target timing diagrams since the SPI_CLK, SPI_CITO, and SPI_COTI pins are directly connected between the controller and the target. The SPI_CITO signal is the output from the target (target transmission), and the SPI_COTI signal is the output from the controller (controller transmission). The controller generates the SPI_CLK signal. The SPI_TS signal is the target device select input to the target from the controller. The diagrams represent an 8-bit transfer (SPI_CTL.SIZE=0) with the MSB first (SPI_CTL.LSBF=0). Any combination of the SPI_CTL.SIZE and SPI_CTL.LSBF bits is permissible. For example, a 16-bit transfer with the LSB first is another possible configuration.

The clock polarity and the clock phase could be identical for the controller device and the target device involved in the communication link. The transfer format from the controller can be changed between transfers to adjust for various requirements of a target device.

The SPI module uses the SPI_CTL.ATSEL bit to determine when the SPI hardware or software control the SPI_SEL[n] line. When SPI_CTL.ATSEL=1, the target select line must be set to the polarity set in the SPI_CTL.POLSEL field between each serial transfer. The actual behavior of SPI_SEL[n] also depends on the parameters programmed into the SPI_DLY register. The SPI hardware logic automatically controls this functionality. When SPI_CTL.ATSEL=0, SPI_SEL[n] can either remain active between successive transfers or be inactive. The software must control this activity through manipulation of the SPI_TSEL register.

The ***SPI Transfer Protocol*** pair of figures illustrates the case when SPI_CTL.ATSEL = 1 and the SPI_SEL[n] line is inactive between frames. If ASSEL = 0, the SPI_SEL[n] line can remain active between frames; however, the first bit is only driven when an active transition of SPI_CLK occurs.

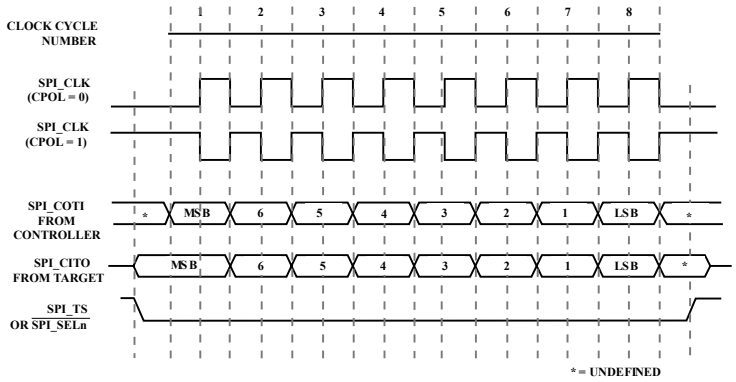


Figure 20-2: SPI Transfer Protocol for CPHA=0

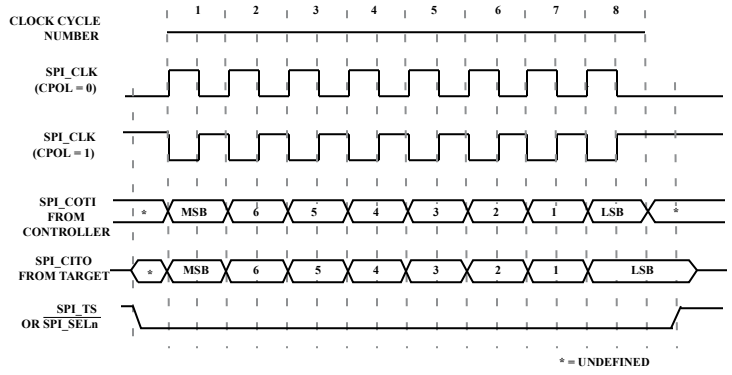


Figure 20-3: SPI Transfer Protocol for CPHA=1

Clock Considerations

The SPI_CLK signal is a gated clock that is only active during data transfers, for the time of the transferred word. In normal mode, the number of active edges is equal to the number of bits to be transmitted or received. In

dual-I/O mode, it is half of the number of bits to be transmitted or received, and in quad-SPI mode it is one-fourth of the number. The clock rate can be derived using both even and odd dividers from SCLK.

For controller devices, the SPI uses the SPI_CLK register value to determine the clock rate, whereas this value is ignored for target devices.

When the SPI controller is a controller, SPI_CLK is an output signal. Conversely, when the SPI controller is a target, SPI_CLK is an input signal. Peripheral devices ignore the SPI clock when the target select input is driven inactive. The SPI uses the SPI_CLK signal to shift out and shift in the data driven onto the SPI_CITO and SPI_COTI lines. The data is always shifted out on one edge of the clock (the active edge) and sampled on the opposite edge of the clock (the sampling edge). Clock polarity and clock phase relative to data are programmable through the SPI_CTL register and define the transfer format.

Controlling Delay Between Frames

The ***SPI Timing with Lead and Lag Programming (Independent of SPI_CTL.CPHA Setting)*** figure illustrates SPI timing using the SPI_DLY.LEADX and SPI_DLY.LAGX programming. The SPI uses the SPI_DLY.LAGX bits to control the timing between the chip select (SPI_TS) signal assertion and the first SPI_CLK edge. The SPI uses the SPI_DLY.LEADX bits to control the timing between the last SPI_CLK edge and deassertion of the SPI_TS signal. The lead and lag timing can be extended by a 1 SPI_CLK duration to ease timing restrictions on the target device.

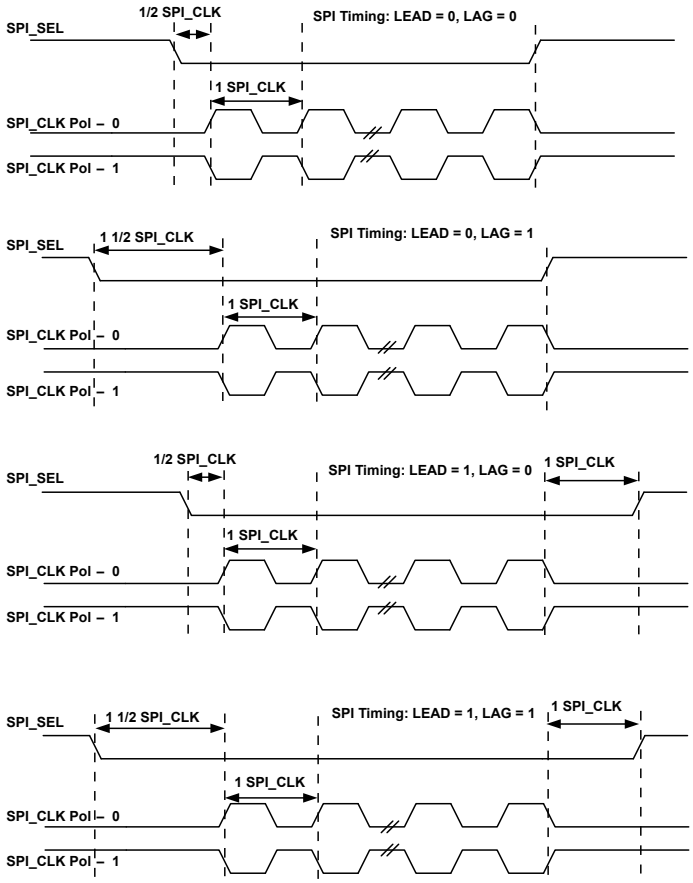


Figure 20-4: SPI Timing with Lead and Lag Programming (Independent of SPI_CTL.CPHA Setting)

The ***SPI Timing with SPI_DLY.STOP Programming (Independent of SPI_CTL.CPHA Setting)*** figure illustrates SPI timing with STOP programming. The SPI module uses this timing to insert multiples of `SPI_CLK` period delays between transfers. The `SPI_TS` line is deasserted for the duration specified in the `SPI_DLY.STOP` bit field, assuming the `SPI_CTL.POLSEL` bit is configured for deassertion between transfers.

If the `SPI_DLY.STOP` bit =0, the controller operates in a *continuous mode*. This mode causes an immediate start of the second word after the last bit is transferred from the first word. During this mode of operation, the chip select line is continuously asserted.

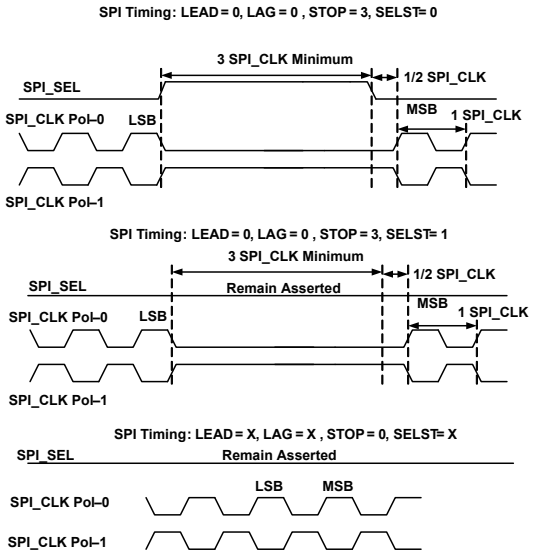


Figure 20-5: SPI Timing with SPI_DLY.STOP Programming (Independent of SPI_CTL.CPHA Setting)

When the SPI_DLY.STOP bit =0 and initial conditions for a transfer are not met, the interface pauses before the next transfer. During this pause, the SPI uses the SPI_CTL.POLSEL bit to determine the state of the chip select pin. The SPI uses the SPI_DLY.LEADX and SPI_DLY.LAGX bits to determine the timing between SPI_CLK edges and the chip select line.

Flow Control

In controller mode, the target device must drive the SPI_RDY pin. The pin acts as an input signal. The target can deassert the SPI_RDY pin to stop the controller from initiating any new transfer. If SPI_RDY is deasserted in the middle of a transfer, the current transfer continues, and the next transfer will not start unless the target asserts the SPI_RDY signal. Whenever the target deasserts SPI_RDY and stalls the controller, the SPI controller goes into a waiting state, and the SPI_STAT.FCS bit is set. When the target asserts SPI_RDY, the SPI controller resumes operation, and the SPI_STAT.FCS bit is cleared.

In target mode, the SPI_RDY pin acts as an output signal. Flow control can be configured on either the TX channel or the RX channel. The SPI uses the SPI_CTL.FCCH bit to control this configuration. If flow control is configured on the TX channel, as the SPI_TFIFO status nears the empty condition, the SPI_RDY pin is deasserted. If flow control is configured on the RX channel, as the SPI_RFIFO status nears the full condition, the SPI_RDY pin is deasserted. The SPI uses the SPI_CTL.FCWM bits to control the FIFO status at which SPI_RDY deassertion takes place. Flow control in target mode is purely based on the FIFO status and does not depend on the word counters.

The ***SPI Flow Control Timing in Controller Mode*** figure illustrates this timing.

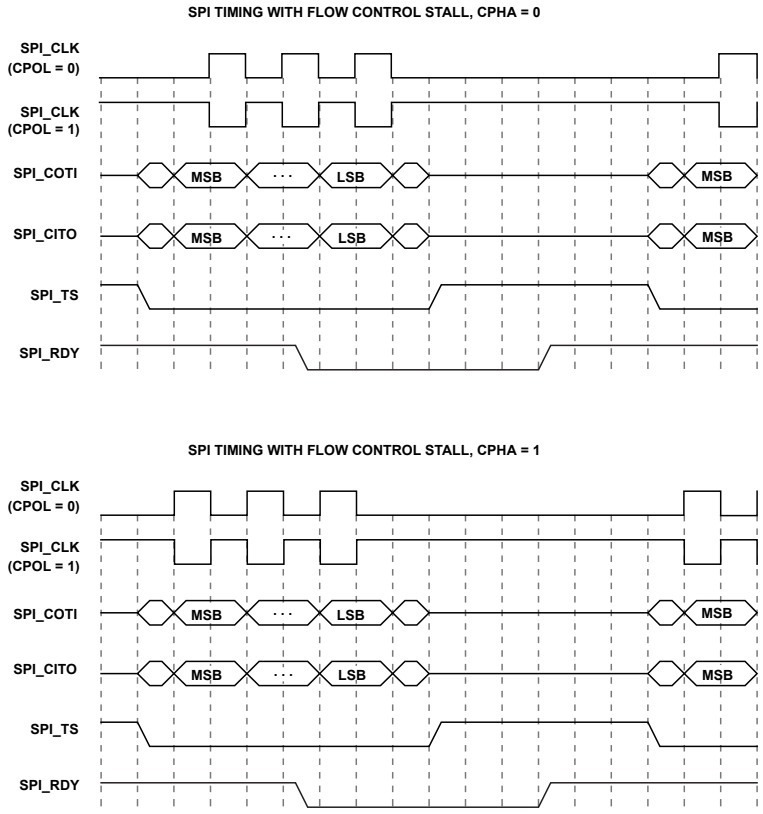


Figure 20-6: SPI Flow Control Timing in Controller Mode.

Target Select Operation

If the SPI is in target mode, $\overline{\texttt{SPI_TS}}$ acts as the target select input. When SPI is enabled as a controller, $\overline{\texttt{SPI_TS}}$ can serve as an error detection input for the SPI in a multi-controller environment. The `SPI_CTL.PTSE` bit enables this feature. When `SPI_CTL.PTSE`=1, the $\overline{\texttt{SPI_TS}}$ input is the controller mode error input. Otherwise, $\overline{\texttt{SPI_TS}}$ is ignored.

The $\overline{\texttt{SPI_TS}}$ signal is an active-low signal. The controller asserts the signal during the transfer. The signal can be deasserted or remain asserted between transfers. When $\overline{\texttt{SPI_TS}}$ is deasserted, `SPI_CLK` and inputs are ignored, and outputs are three-stated.

The target select bits (`SPI_TSEL.TSEL1` – `SPI_TSEL.TSEL7`) are used in a multiple-target SPI environment. For example, if there are eight SPI devices in the system including a processor controller, the controller processor can support the SPI mode transactions across the other seven devices. This configuration requires only one controller processor in this multi-target environment.

For example, assume that the SPI of the processor is the controller. The `SPI_TSEL.TSEL1` – `SPI_TSEL.TSEL7` bits on the processor can be connected to the target select pin of each target device. In this configuration, the target select bits can be used in three ways. In cases 1 and 2, the processor is the controller and the seven microcontrollers or targets with SPI interfaces are targets. The processor can do one of the following:

1. Transmit to all seven SPI devices at the same time in a broadcast mode. Here, all target select bits are set.

2. Receive and transmit from one SPI device by enabling only one target SPI device at a time.
3. If all the targets are also processors, then the requester can receive data from only one processor at a time. (The functionality is enabled by clearing the SPI_CTL.ECITO bit in the six other target processors.) The requester can transmit broadcast data to all seven at the same time. This CITO enabling feature is available in some other microcontrollers. Therefore, it is possible to use the CITO enabling feature with any other SPI device that includes this functionality.

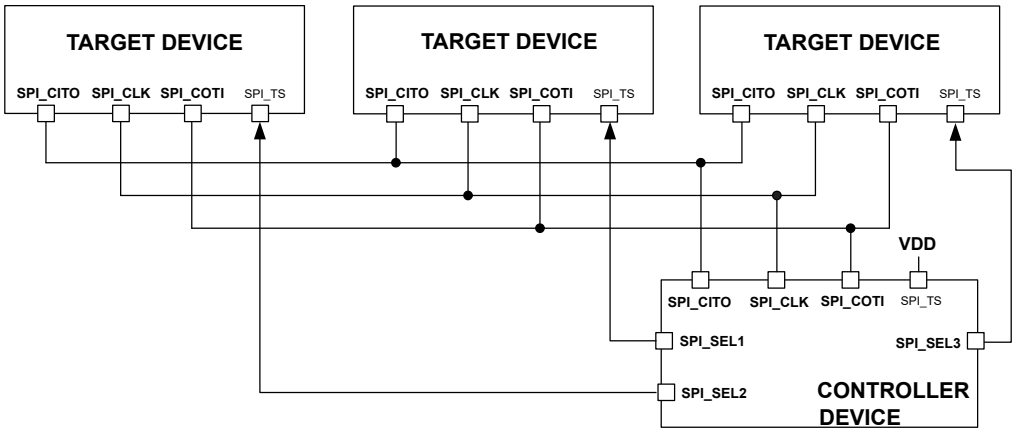


Figure 20-7: Single-Controller, Multiple-Target Configuration

Beginning and Ending a Non-DMA SPI Transfer

The start and finish of a non-DMA SPI transfer depend on the following settings.

1. Whether the device is configured as a controller or a target.
2. The state of the SPI_CTL.ATSEL bit, which selects between hardware and software control over SPI_TSEL.

When SPI_CTL.CPHA=0, the enabled target select outputs are driven active. However, the SPI_CLK signal remains inactive for the first half of the first cycle of SPI_CLK. For a target with SPI_CTL.CPHA=0, the transfer starts as soon as the SPI_TS input goes low.

When SPI_CTL.CPHA=1, a transfer starts with the first active edge of SPI_CLK for both target and controller devices. For a controller device, a transfer is complete after it sends the last data and simultaneously receives the last data bit. A transfer for a target device ends after the last sampling edge of SPI_CLK. If SPI_CTL.ATSEL=1, the hardware maintains responsibility for toggling SPI_TS between frames. If SPI_CTL.ATSEL=0, software controls the SPI_TS line and can keep it active between frames.

The SPI_STAT.RFE bit defines when the receive buffer can be read, indicating that SPI_RFIFO is not empty. The SPI_STAT.TFF bit defines when the transmit buffer can be written, indicating that the SPI_TFIFO is not full. The end of a single word transfer occurs when the SPI_STAT.RFE bit is cleared. The status indicates that a new word has been received and written into the receive FIFO. The SPI_STAT.RFE bit remains cleared when the receive FIFO has valid data.

To maintain software compatibility with other SPI devices, the SPI_STAT.SPIF bit is also available for polling. This bit can have a slightly different behavior from other commercially available devices.

In controller mode with the SPI_CTL.ATSEL bit cleared, software manually asserts the required target select signal before starting the transaction. After all data transfers, software typically releases the target select line.

When the receive or transmit word counters are enabled in the SPI_TXCTL or SPI_RXCTL registers, the SPI generates a finish interrupt at the end of the transfer. It signals the end of all transfers related to that transaction.

Transmit Operation in Non-DMA Mode

The transmit operation in non-DMA mode is enabled through the SPI_TXCTL.TEN bit. It can be enabled independently from the receive operation, and the transmit channel can become the initiating channel based on the SPI_TXCTL.TTI bit setting.

Transmit underrun is not possible in this mode, as no new transfer initiates unless the transmit FIFO is empty (in the case that SPI_TXCTL.TTI =1). A receive overflow is detected when data from a new frame transfer replaces older data in a full receive FIFO. This event can occur if SPI_TXCTL.TTI =1 and the receive channel is enabled in a non-initiating capacity.

An SPI transmit interrupt request is signaled once the transmit channel has been enabled and the transmit FIFO is not full. The SPI uses the SPI_TXCTL.TDR bit setting to control the frequency of the interrupt request.

Receive Operation in Non-DMA Mode

The receive operation in non-DMA mode is enabled through the SPI_RXCTL.REN bit. It can be enabled independently from the transmit operation, and the receive channel can become the initiating channel based on the SPI_RXCTL.RTI bit setting.

Receive overflow is not possible in this mode, as no new transfer initiates when the receive FIFO is full (in the case of SPI_RXCTL.RTI =1). A transmit underrun can occur (SPI_TXCTL.TDU bit) when no valid data is in the SPI_TFIFO register when a transfer is initiated. This event can occur if SPI_RXCTL.RTI =1 and the transmit channel is enabled in a non-initiating capacity.

An SPI receive interrupt request is signaled once the receive channel has been enabled and there is data waiting to be read. The SPI uses the SPI_RXCTL.RDR bit setting to control the frequency of the interrupt request.

Dual I/O Mode

In Dual I/O mode, the SPI_CITO and SPI_COTI pins are configured to operate in the same direction which doubles bandwidth. The SPI uses the SPI_CTL.SOTI bit to determine the order of bits on the pins. When set, the processor sends the first bit on the SPI_COTI pin and the second bit on the SPI_CITO pin. If the SPI_CTL.SOTI bit is cleared, the order is reversed. Since dual I/O mode uses both pins to transmit or receive data, only one channel can be enabled, either transmit or receive. Flow control through the SPI_RDY pin is supported. Interrupt request generation is unaffected by dual I/O mode. However, the interrupt service interval is reduced because the individual transfer latency is halved.

Changing to quad SPI mode must be done when the SPI is in a quiescent state.

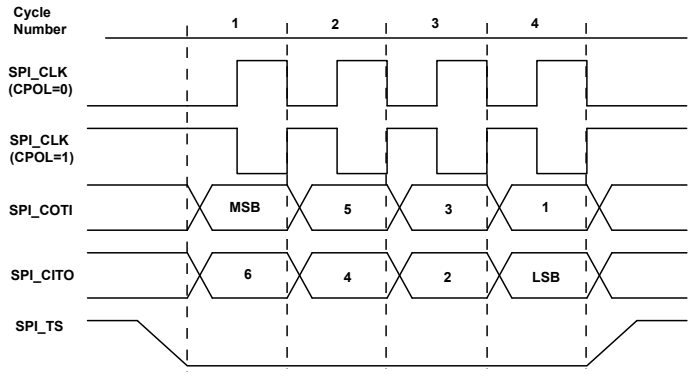


Figure 20-8: Dual I/O Mode Transfer Protocol for CPHA=0, SOSI=1, 8-Bit Transfer, LSBF=0.

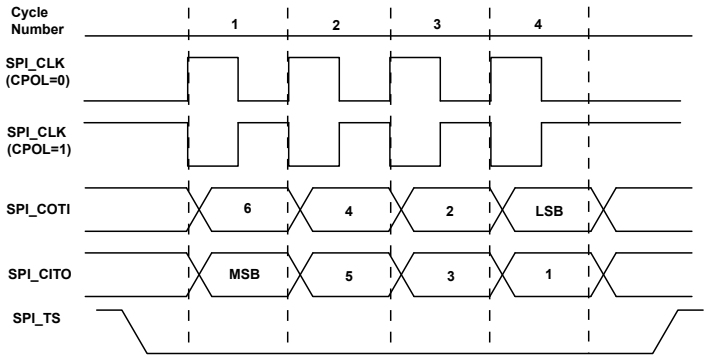


Figure 20-9: Dual I/O Mode Transfer Protocol for CPHA=1, SOSI=0, 8-Bit Transfer, LSBF=0.

Quad I/O Mode (SPI1/SPI2)

In quad SPI mode, the SPI_CITO and SPI_COTI pins, in tandem with the SPI_D2 and SPI_D3 pins, are configured to operate in the same direction. The SPI uses the SPI_CTL.SOTI bit to determine the order of bits on the pins. When set, the processor sends:

- The first bit on the SPI_COTI pin
- The second bit on the SPI_CITO pin
- The third bit on the SPI_D2 pin
- The fourth bit on the SPI_D3 pin

If the SPI_CTL.SOTI bit is cleared, the order is reversed. Since quad SPI mode uses all four pins to transmit or receive data, only one channel can be enabled as either transmit or receive. Flow control through the SPI_RDY pin is supported. Interrupt generation is unaffected by quad SPI mode.

Changing to quad SPI mode must be done when the SPI is in a quiescent state.

While using dual or quad I/O mode for communicating with flash devices, program the SPI_CTL.CPHA and the SPI_CTL.CPOL bits =1. This programming avoids bus contention during read operations, because the flash device starts driving out the bits immediately after dummy cycles in read header.

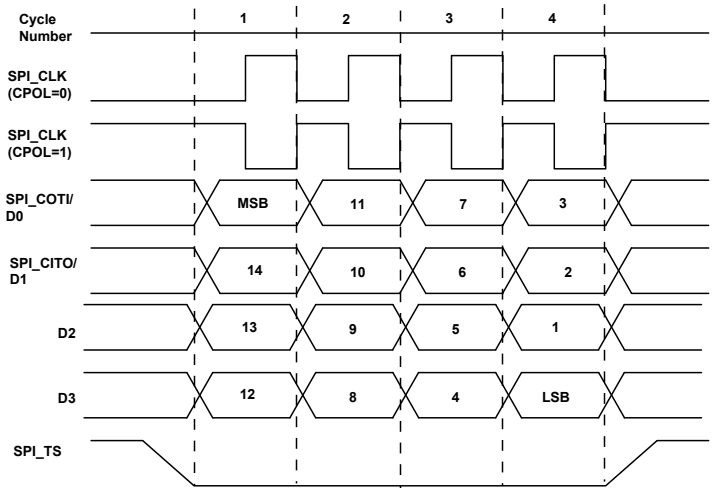


Figure 20-10: Quad Mode Timing for CPHA=0, SOSI=1, 16-Bit Transfer, LSBF=0.

NOTE: The SPI does support quad SPI 8-bit transfer in target continuous mode of operation with an SCLK:`SPI_CLK` ratio of less than 1:2. A minimum of 2 SCLK cycles is required between transfers in 8-bit quad SPI target mode with an SCLK:`SPI_CLK` ratio of less than 1:2.

Fast Mode

Fast mode is like the normal mode of operation when transmitting. When receiving, data is sampled at the next transmit edge allowing a full cycle of timing in the receive direction. This mode is valid in controller mode operation only. When the SPI operates in fast mode, the target drives the data for one full cycle.

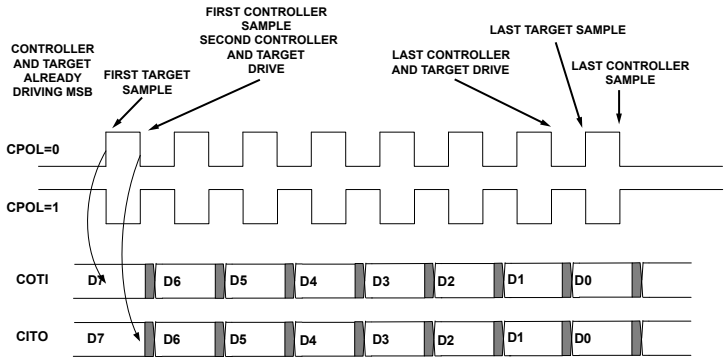


Figure 20-11: SPI Transfer Protocol in Fast Mode for SPI_CTL.CPHA = 0

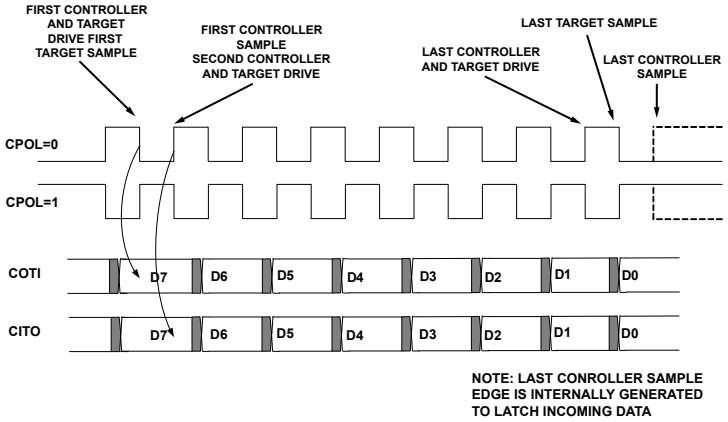


Figure 20-12: SPI Transfer Protocol in Fast Mode for SPI_CTL.CPHA = 1

Memory-Mapped Mode (SPI1/SPI2)

The SPI supports direct memory-mapped read accesses from a SPI memory device, enabled by setting the `SPI_CTL.MMSE` bit. This mode allows for direct execution of instructions from a SPI memory device without the need for a low-level software driver, as hardware handles all overhead tasks (for example, transmission of the read header, pin turnaround timing, and receive data sizing). The SPI features configurable options in the memory-mapped read header register (`SPI_MMRDH`) to provide compatibility with a wide range of SPI memory devices.

In non-memory-mapped mode, the software is responsible for providing the command and required dummy words for the read response, whereas this is all handled by hardware when the SPI is in memory-mapped mode. The memory of the SPI device is accessible directly through reads of the processor address space. The read accesses can be code or data accesses in core mode or when using memory DMA (MDMA). These accesses allow code to execute directly from SPI memory devices (true eXecute-In-Place operations), and the contents can be cached to improve performance. It is not necessary to access the SPI data buffer registers nor poll status bits; however, the hardware does not support peripheral DMA accesses nor write operations to the SPI memory space.

The ***Types of Operations*** table is a comparison of the permitted operations in the non-memory-mapped and memory-mapped modes supported by the SPI controller.

Table 20-6: Types of Operations

SPI Operation	Non-Memory-Mapped Mode	Memory-Mapped Mode
Core data write	Yes	No
Core data read	Yes	Yes
Code fetch: Execute-In-Place (XIP)	No	Yes
Read/Write accesses using SPI peripheral DMA	Yes	No
Read/Write accesses by other peripheral DMA channels	No	No

Table 20-6: Types of Operations (Continued)

SPI Operation	Non-Memory-Mapped Mode	Memory-Mapped Mode
MDMA read*	No	Yes
MDMA write	No	No

NOTE: MDMA read* - Only standard bandwidth MDMA (MDMA0 and MDMA1) and Enhanced Bandwidth MDMA (MDMA2) can be used for accessing flash address space in memory mapped mode. The maximum bandwidth MDMA (MDMA3) cannot access flash address space.

Memory-Mapped Description of Operation

Memory-mapped mode is enabled by setting the SPI_CTL.MMSE bit. When enabled, the SPI (if ready) accepts the read requests through a dedicated on-chip target interface. The memory subsystem controller drives this dedicated interface through the SCB fabric.

In a typical scenario, the memory subsystem controller issues read requests to the fabric, and the fabric routes these requests to the target port of the SPI peripheral. The controller describes the read access using a number of parameters such as starting address, transfer size, and burst type. The SPI responds to this read access request when it is ready for a new transfer. It loads the opcode, a specified number of address bytes, and an optional mode byte into the transmit FIFO. The SPI memory state machine begins when both the transmit and receive channels of the SPI are enabled with:

- the transmit transfer initiation bit is set (SPI_TXCTL.TTI=1), *and*
- the receive initiation bit is cleared (SPI_RXCTL.RTI=0)

The SPI memory read sequence starts with the assertion of $\overline{\text{SPI_SEL1}}$. If the SPI memory state machine is in the reset state, it looks for a command. The SPI hardware then sends the specific 8-bit read command (which can be optionally skipped), followed by the SPI memory read address. After this, a dummy period is inserted, in which a mode byte is optionally sent, and the pins are held or three-stated during the dummy clocking period.

NOTE: This read header is transmitted over the SPI standard protocol pins (SPI_CLK, SPI_COTI, SPI_CITO, $\overline{\text{SPI_SEL1}}$) or over the extended SPI protocol pins (SPI_CLK, SPI_COTI, SPI_CITO, SPI_D2, SPI_D3, $\overline{\text{SPI_SEL1}}$), based on the SPI_MMRDH.CMDPINS, SPI_MMRDH.ADRPINS, and SPI_CTL.MIOM bit settings. SPI memory devices usually support communication in MSB-first mode. In dual mode, the SPI typically uses SPI_CITO as IO1 and SPI_COTI as IO0. In quad mode, the SPI typically uses SPI_D3 pin as IO3, SPI_D2 as IO2, SPI_CITO as IO1, and SPI_COTI as IO0.

When all I/O data pins are three-stated, the SPI continues clocking the SPI memory device, which drives out the data bits at the addressed location, until all bytes are received. The SPI hardware reads the data as configured by the SPI_CTL.MIOM bit setting. Upon reception of the last byte, the SPI typically deasserts $\overline{\text{SPI_SEL1}}$ to prepare for the next requested read header.

Application code must ensure that the opcode sent is consistent with multiple I/O programming and that the parameters specified in the memory-mapped read header register are consistent with flash read access timing.

The *SPI Memory-Mapped Register Operations Flow* diagram shows how the fields of the SPI_MMRDH register determine the read header while initiating transfers in memory-mapped mode.

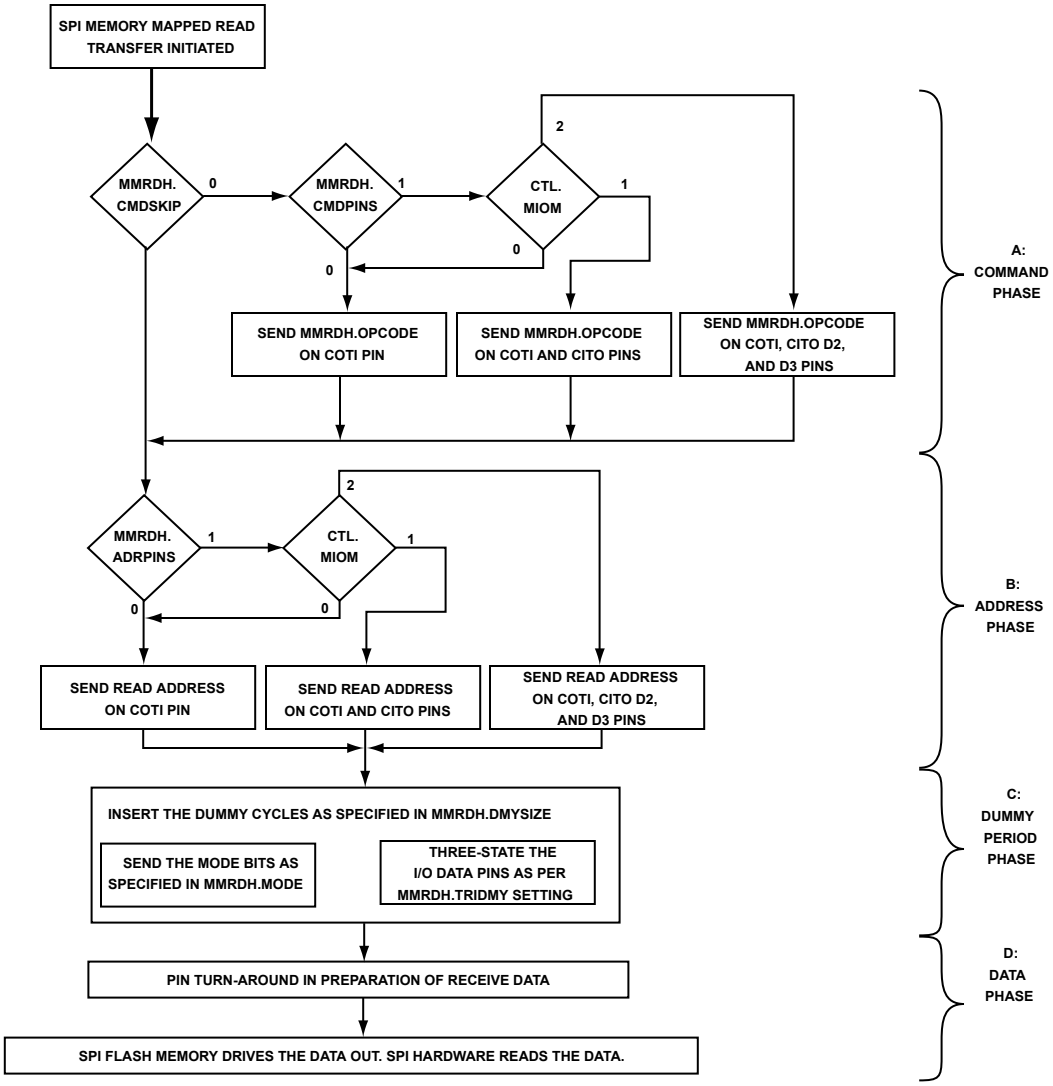


Figure 20-13: SPI Memory-Mapped Register Operations Flow

Memory-Mapped Architectural Concepts

In memory-mapped mode, the SPI accepts read requests through a dedicated on-chip target interface. The SPI (when ready) accepts these requests and begins the process of assembling the read header based on access attributes described in both the SPI_MMRDH register and the internal bus request. After the read header transmission completes, a pin turnaround period is timed, and the receiver is enabled. The SPI continues clocking the SPI memory device until all bytes are received.

The SPI memory-mapped hardware accommodates various memory devices with different read timing. The capabilities include extra mode bits, flexible dummy period timing, and three-state control, as configured in the SPI_MMRDH register.

The *Memory-Mapped Protocol* figure shows the protocol for the SPI controller in memory-mapped mode.

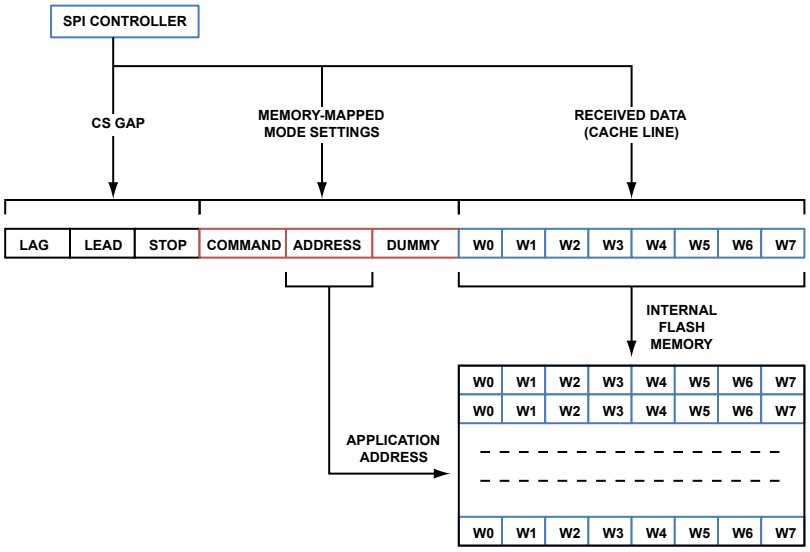


Figure 20-14: Memory-Mapped Protocol

As shown in the figure, the COMMAND field (`SPI_MMRDH.OPCODE`) is transmitted upon assertion of the `SPI_SEL[n]` signal. The SPI memory interprets this 8-bit value as a read command. Any 8-bit read opcode whose timing is compliant with the processor SPI and features provided by memory-mapped hardware is allowed, the most common being:

- Standard Read (0x03)
- Fast Read (0x0B)
- Fast Read Dual Output (0x3B)
- Fast Read Dual I/O (0x6B)
- Fast Read Quad Output (0xBB)
- Fast Read Quad I/O (0xEB)

NOTE: The SPI hardware does not validate the content of the `SPI_MMRDH.OPCODE` field prior to transmitting.

DMYSIZE (Number of Dummy Bytes)

When operating at a high clock frequency in multi-IO modes, most flash devices require some dummy clocks after the address bits. These dummy clock cycles allow the internal circuits of the device extra time for setting up the initial address. These bits specify the number of bytes separating address transmission and read data return.

The number of dummy cycles required varies per manufacturer, the read command used, and the SPI access time. The SPI hardware allows dummy cycles to be programmed in bytes in the `SPI_MMRDH.DMYSIZE` field, the value of which is a function of the number of pins used to transmit the address (`SPI_MMRDH.ADRPINS`), as shown in the *Pins Used to Transmit the Address (ADRPINS)* table.

Table 20-7: Pins Used to Transmit the Address (ADRPINS)

	Dummy clock cycles		
SPI_MMRDH.DMYSIZE	(**SPI_MMRDH.ADRPINS**=0, **SPI_CTL.MIOM**=x) Dummy bytes elapse over 1-pin	(**SPI_MMRDH.ADRPINS**=1, **SPI_CTL.MIOM**=1) Dummy bytes elapse over 2-pins	(**SPI_MMRDH.ADRPINS**=1, MIOM=2) Dummy bytes elapse over 4-pins
000	0	0	0
001	8	4	2
010	16	8	4
011	24	12	6
100	32	16	8
101	40	20	10
110	48	24	12
111	56	28	14

This dummy clocking period allows the mode bits to be sent, the pins to be three-stated, and the pins to be turned around in preparation for the receive data.

Memory-Mapped Read Accesses

The SPI hardware supports the most used read operations.

- Two standard SPI reads (read and read fast), which use the unidirectional `SPI_COTI` and `SPI_CITO` pins in addition to `SPI_SEL[n]` and `SPI_CLK`
- Four extended SPI multiple I/O reads: dual output, quad output, dual I/O, and quad I/O reads

The ***SPI Read Operations*** table and ***SPI Flash Fast Read Sequence*** figures summarize the types of read operations. Program each read operation in a way that is compatible with the description given in the SPI flash data sheet.

Table 20-8: SPI Read Operations

Operation	Read Command (Opcode)	CMDPIN	ADRPIN	DMYSIZE	Three-state	Multiple I/O Mode	Data Pins
Read	0x03	1	1	Zero	No	No	1
Fast Read	0x0B	1	1	Non-Zero	Yes	No	1
Dual Output Read	0x3B	1	1	Non-Zero	Yes	Yes(IO0-1)	2
Quad Output Read	0x6B	1	1	Non-Zero	Yes	Yes(IO0-3)	4
Dual I/O Read	0xBB	1, 2	2	Non-Zero	Yes	Yes (IO0-1)	2
Quad I/O Read	0xEB	1, 4	4	Non-Zero	Yes	Yes (IO0-3)	4

Some memory devices also support word quad I/O read (0xE7) and octal quad I/O read (0xE3) operations. These operations require fewer dummy cycles than normal quad I/O read operations.

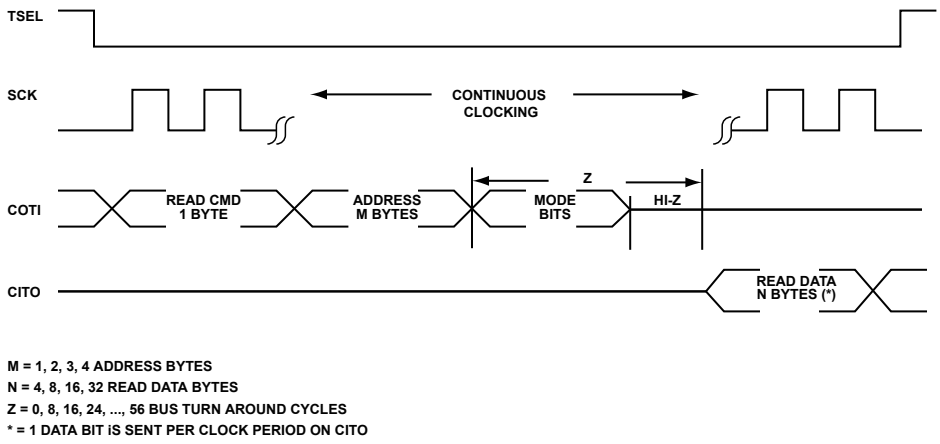


Figure 20-15: SPI Flash Fast Read Sequence

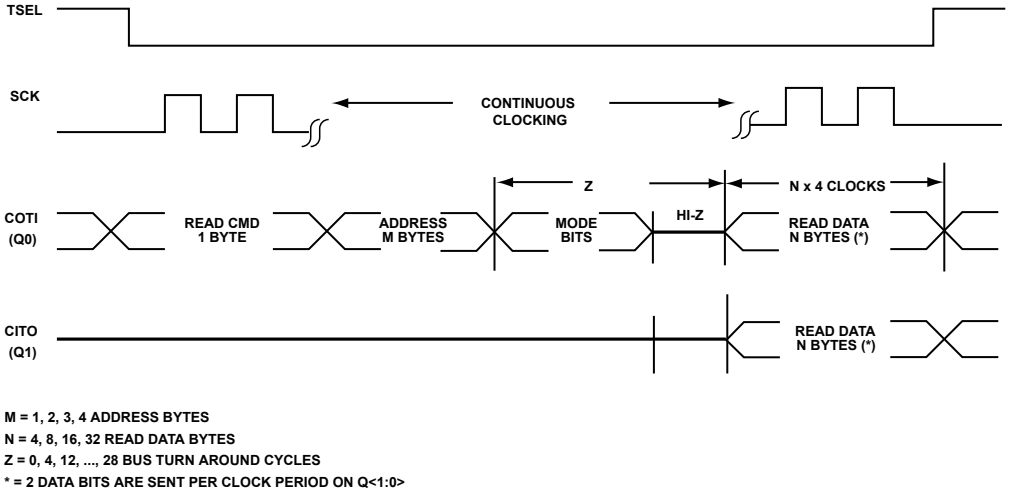


Figure 20-16: SPI Flash Fast Read (Dual Output) Sequence

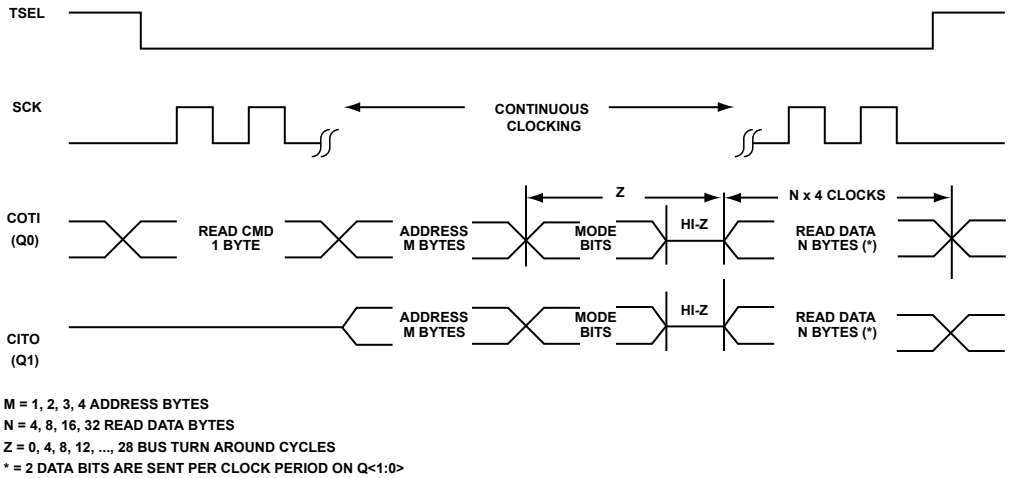


Figure 20-17: SPI Flash Fast Read (Dual I/O) Sequence

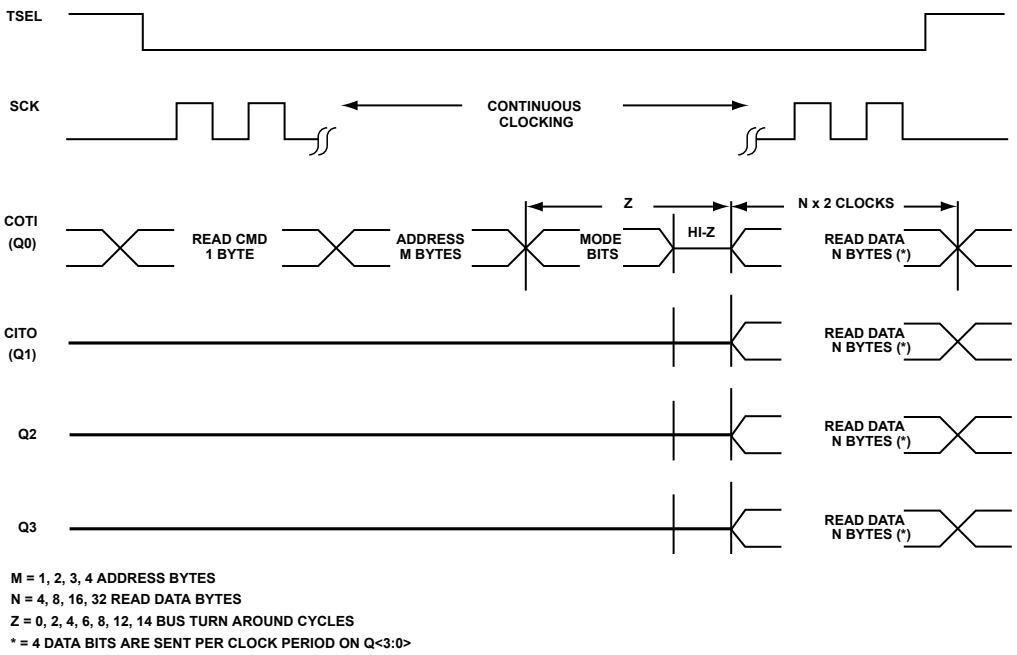


Figure 20-18: SPI Flash Quad Output Read Sequence

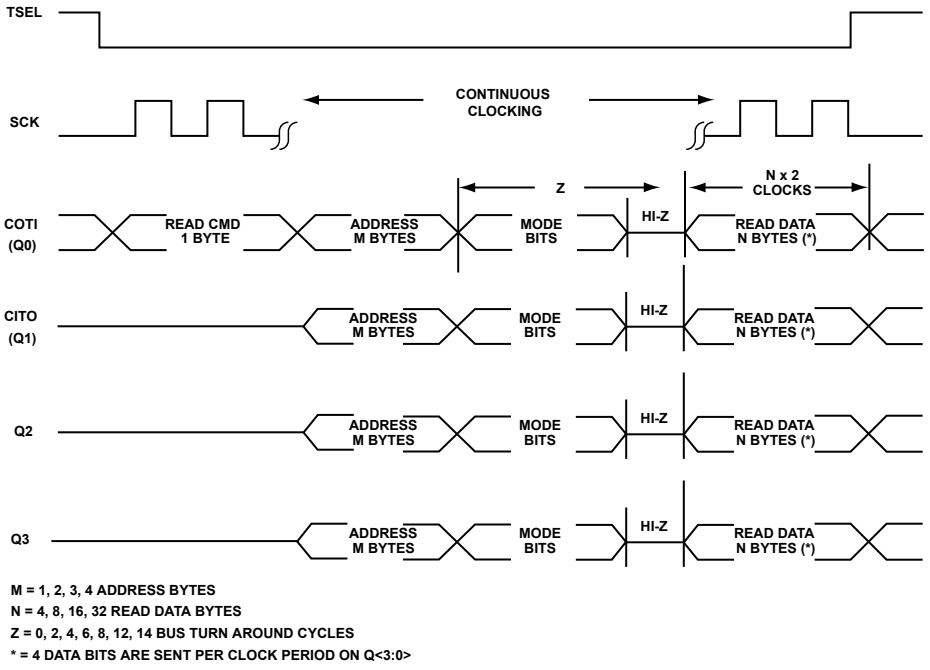


Figure 20-19: SPI Flash Quad I/O Read Sequence

SPI memory-mapped reads can be made cacheable in the core's internal memory by properly configuring the region as cacheable memory without bypass (see the related core's cache configuration documentation for details). In the figures, the number of read data bytes (N) is based on the following:

- For an instruction fetch by core (when in XIP mode); the number of instruction bytes to be fetched depends on the cache line size of the cache.

- For a data fetch by the core (data read), the number of data bytes to be fetched depends on the cache line size of the cache.

Although the minimum size of a memory-mapped data read transfer is 4 bytes, applications can fetch a single byte or a 2-byte data. (For example, it can fetch an unsigned char or short access in C code). In this case, only the required bytes are provided to the core and the other bytes are cached.

The on-chip memory subsystem controller provides a starting address for the burst and the SPI hardware issues this address as part of the read header. The address provided is N-byte aligned. For example, to read the 30th byte from SPI memory, then the typical address to provide is:

- 28 (0x0000_001C) for a 32-bit cache line
- 24 (0x0000_0018) for a 64-bit cache line
- 16 (0x0000_0010) for a 128-bit cache line
- 0 (0x0000_0000) for a 256-bit cache line

The read data is returned to the memory subsystem in the order provided by the SPI memory. There can be considerable delay for the expected data provided to the controller.

To minimize this delay, the wrap feature can be used where the memory subsystem provides the address of the critical word.

- For MDMA reads, the number of read data bytes (N) is always equal to 4 bytes. The MDMA read does not depend on the cache setting. For MDMA reads, limit the `DMA_CFG.MSIZE` field to one, two or four bytes. The address provided by the memory subsystem controller to the SPI hardware is always 4-bytealigned.

Memory-Mapped High-Performance Features

In addition to automating the SPI memory read accesses, the memory-mapped hardware also provides some features to improve SPI memory fetches and increase the system performance. The following sections describe these features.

Merged Read Accesses

It is common for the memory subsystem to fetch two or more cache lines from consecutive addresses (the address sequencing is linear without any jumps). To take advantage of this situation, the SPI memory-mapped hardware provides a feature called merging. Enable merging by setting the `SPI_MMRDH.MERGE` bit.

When enabled, the hardware compares the address of an incoming read request to the address of a request the SPI memory is actively servicing. It can decide to merge two accesses when the address for the second access is incremental. For example, if the first address of a 32-byte cache line fetch is 0x0000_0000 and the second fetch is to address 0x0000_0020, then these two accesses can be merged. Merging increases efficiency and overall fetch bandwidth by eliminating the read header for those accesses which only require continuation of the SPI clock.

Wrap Around Accesses

Many SPI flash memory devices support wrapping which is used to enhance critical word fetching of cache lines. In this mode, the SPI device automatically wraps the read address to the base of a cache line once the end of the cache line is reached.

Wrap around accesses are enabled by setting the SPI_MMRDH.WRAP bit.

Some flash devices require programs to send a ***Set Wrap*** command to place the device in wrap mode. Other flash devices provide a configuration register which must be programmed to set the flash in wrap mode. Since the SPI memory-mapped hardware does not support any write operations to flash, perform this step in non-memory-mapped mode (SPI_CTL.MMSE=0) by accessing the SPI registers.

Data access is limited to 8-byte, 16-byte, or 32-byte sections of flash page in wrap mode. The Arm core uses the Wrap 4 access (64-bit data) for L1 cache. The Arm core uses Wrap 4 and Wrap 8 accesses (64-bit data) for L2 cache. The cores use the Wrap 8 accesses for unaligned accesses. During the read request to the SPI memory-mapped hardware, the memory subsystem controller of the processor provides the address of a critical word instead of the line base. The read-data starts at the address specified in the instruction. Once it reaches the end boundary of the 8, 16, or 32-byte section, the output automatically wraps around to the beginning boundary to the line base address. The data fetch continues. It is not necessary to deassert the SPI SPI_SEL[n] signal or resend the read header to wrap to the cache line base when servicing misaligned cache fill requests.

The ***Byte Sequence in Wrap Modes*** table shows byte sequences in various wrap modes.

Table 20-9: Byte Sequence in Wrap Modes

Starting Address	8-Byte Wrap (cache_line = 8 byte)
0	0-1-2- . . . -6-7
1	1-2- 3-. . . -7-0
7	7-0-1- . . . -5-6
15	15-8-9- . . . -13-14
31	31-24-25-. . . .-29-30

The burst with wrap feature allows applications to fetch a critical address quickly. Applications then fill the cache afterwards within a fixed length (8/16/32-byte) of data without issuing multiple read commands. Certain applications can benefit from this feature to improve cache fill efficiency and overall performance of system code execution.

NOTE: Do not use the merge and wrap feature together. Using wrap bursts can unintentionally disable merging (merging cannot occur for unaligned wrapping bursts). A wrap burst can start fetching data words in the middle of the cache line and cannot be merged with the next access.

Execute-In-Place (XIP, SPI2 only)

Execute-In-Place, known as XIP, permits software code to execute directly from an SPI flash device rather than downloading the code and executing it out of RAM. XIP, also known as Command Skip mode, is a general term and can be applied to fetching data as well.

There is a difference between XIP mode and standard mode. In XIP mode, after the SPI memory device is selected (CS# =LOW), the memory device does not decode the first input byte as command code. Instead, it expects the read header to directly start with address bytes. In standard mode, the memory decodes the first input byte it receives as a command code.

The XIP mode dramatically reduces random access time for applications that require fast code execution without shadowing the memory content on a RAM. The SPI memory-mapped hardware provides a control bit, SPI_MMRDH.CMDSKIP to skip the command from read header.

Some SPI memory devices require configuration of their control register to enable the XIP mode of operation, using the non-memory-mapped mode of the processor SPI. Typically, during the dummy cycle period, the mode bits are used to confirm the XIP operation and the SPI_MMRDH.MODE field must be set appropriately. A dummy memory-mapped access may be needed before setting the SPI_MMRDH.CMDSKIP bit to set the SPI memory device in Command Skip mode.

For more details about how to configure SPI memories into XIP mode, refer to the device data sheet.

NOTE: When configuring the flash to XIP mode from the SHARC+ core, ensure that the routine that configures flash to XIP is not routed through the L2CC. This is accomplished by first configuring the flash to XIP mode, then enabling the L2CC from the core.

Memory-Mapped Mode Error Status Bits

The SPI memory-mapped hardware provides bits in the SPI_STAT register to report errors. It provides these bits for notification only and their state has no effect on SPI operations. The status register bits are sticky. A W1C (write-1-to-clear) operation clears the bits.

- Memory-Mapped Write Error (SPI_STAT.MMWE). This bit is set (=1) if an attempt is made to write to address space that is reserved for memory-mapped SPI memory. The SPI memory-mapped hardware does not support automated write access to SPI memory space.
- Memory-Mapped Read Error (SPI_STAT.MMRE). This bit is set (=1) if an attempt is made to read address space reserved for memory-mapped SPI memory while memory mapping is disabled (SPI_CTL.MMSE =0).
- Memory-Mapped Access Error (SPI_STAT.MMAE). This bit is set (=1) if an attempt is made to access either the TX or RX FIFO while memory-mapped access of SPI memory is enabled. In this case, attempts to communicate with the SPI device using legacy methods are blocked and receive fabric reports an error. Legacy methods include any direct access made to the TX and RX FIFOs, whether by DMA or processor MMR.
- Memory-Mapped Write Error Mask (SPI_CTL.MMWEM) bit specifies whether an error response is returned to the fabric on write attempts to address space that is reserved for memory-mapped SPI memory reads. Regardless of whether a write error response is masked using this bit, the memory-mapped write error (SPI_STAT.MMWE) sticky notification bit is still set.

  NOTE: Unlike other bits in the SPI_STAT register, these memory-mapped mode error bits do not have associated bits in the SPI interrupt mask (SPI_IMSK) and SPI interrupt condition (SPI_ILAT) registers.

The memory-mapped top register (SPI_MMTOP) is used to specify the upper limit of the SPI memory address. The memory-mapped accesses to SPI memory addresses equal to or above this range are considered illegal. The accesses are blocked, and a bus error response is generated.

This register is useful to block the invalid SPI memory address accesses. Some SPI memory vendors do not clearly specify (guarantee) that overrange address bits are ignored (address spaces can be wrapped).

Memory-Mapped Programming Guidelines

Setting the SPI_CTL.MMSE bit enables SPI memory-mapped mode. When enabled, the SPI interface is forced to be consistent with SPI memory requirements regardless the settings of certain control bits. The following tables specify typical settings for configuring the SPI in memory-mapped mode:

Table 20-10: SPI Control (SPI_CTL) Register

Bits	Typical values to set	Description	Comments
SPI_CTL.MODE	1	Controller mode enable	
SPI_CTL.PTSE	0	Protected target select enable	
SPI_CTL.ODM	0	Open-drain mode enable	
SPI_CTL.CPHASPI_CTL.CPOL	0–0 or 1–1	SPI mode of communication	Flash dependent, usually SPI flash supports mode-0 (CPHA=CPOL=0) and mode-3 (CPHA=CPOL=1)
SPI_CTL.ATSEL	1	Hardware target select pin control	
SPI_CTL.POLSEL	1	Target select asserted between transfers	
SPI_CTL.ECITO	1	CITO pin enable	
SPI_CTL.SIZE	2	32-bit transfer size	
SPI_CTL.LSBF	0	MSB bit first mode	Flash dependent, usually SPI flash communicates in MSB bit first mode
SPI_CTL.FCEN SPI_CTL.FCCH SPI_CTL.FCPL SPI_CTL.FCWM	0	Hardware flow control related bits	
SPI_CTL.FMODE	1	Fast mode enable	Typically set to 1 for full cycle timing, 0 only works at low speed
SPI_CTL.SOTI	0	Treat SPI_COTI pin as IO0 pin.	

Table 20-11: SPI Receive Control Register

Bits	Typical values to set	Description
SPI_RXCTL.REN	1	Receive channel enable
SPI_RXCTL.RTI	0	Receive transfer initiation disable
SPI_RXCTL.RWCEN	0	Receive word counter disable

Table 20-11: SPI Receive Control Register (Continued)

Bits	Typical values to set	Description
SPI_RXCTL.RDR	0	Receive data request disable
SPI_RXCTL.RDO	0	Discard incoming data if RFIFO is full
SPI_RXCTL.RRWM	0	Receive FIFO regular watermark
SPI_RXCTL.RUWM	0	Receive FIFO urgent watermark disable

Table 20-12: SPI Transmit Control Register

Bits	Typical values to set	Description
SPI_TXCTL.TEN	1	Transmit channel enable
SPI_TXCTL.TTI	1	Transmit transfer initiation disable
SPI_TXCTL.TWCEN	0	Transmit word counter disable
SPI_TXCTL.TDR	0	Transmit data request disable
SPI_TXCTL.TDU	0	Send last word when TFIFO is empty
SPI_TXCTL.TRWM	0	Transmit FIFO regular watermark
SPI_TXCTL.TUWM	0	Transmit FIFO urgent watermark disable

Table 20-13: SPI DLY Control Register

Bits	Typical Values to Set	Description	Comments See Flash data sheet for CS (for example, SSEL) Timing Specs
SPI_DLY.LAGX	1	Extended lag timing	
SPI_DLY.LEADX	1	Extended lead timing	
SPI_DLY.STOP	3	Stop bit between the transfers	Can be set to 1 at lower SPI clock frequencies.

The multiple I/O mode (SPI_CTL.MIOM) bits are partially ignored:

- The command (opcode) is transmitted using either just one or the number of pins specified by the SPI_CTL.MIOM bits, depending on SPI_MMRDH.CMDPINS bit setting.
- The address is then transmitted using either just one or the number of pins specified by the SPI_CTL.MIOM bits, depending on SPI_MMRDH.ADRPINS bit setting.
- The data is always read with the number of pins specified by the SPI_CTL.MIOM bits.

NOTE: Set the SPI module enable bits SPI_CTL.EN last after configuring all registers.

Use the following programming guidelines for memory-mapped mode:

- The SPI memory-mapped hardware does not check the flash status before initiating the access. It assumes that SPI memory is always able to respond to a read access. Before enabling memory-mapped mode (for example, setting the SPI_CTL.MMSE bit) ensure that SPI flash is ready for a read access. When using non-memory-mapped mode, a write-complete status can be examined prior to enabling the SPI in memory-mapped mode. (See the write in progress bit in the SPI flash memory status register.) Also, immediately after initial power-up, SPI memory devices can be inaccessible for a vendor-specified period.
- When SPI is enabled in memory-mapped mode, attempts to communicate with the SPI device using legacy methods are blocked. Legacy methods include any direct access made to the transmit or receive FIFOs, whether initiated by DMA or by a processor MMR access.
- To use some of the features offered by SPI memory devices, programs can first configure the SPI memory device by setting its control word or sending some commands. Because SPI memory-mapped hardware does not allow any type of SPI write operations, configure the SPI in non-memory-mapped mode prior to enabling memory-mapped mode.
- The memory-mapped hardware does not interpret the opcode. It does not check the validity of the timing that is specified in the SPI_MMRDH register for a particular opcode. Programs must set the fields of the SPI_MMRDH register to be consistent with the read-type selected.
- When the core requests the data or code fetch, the memory-mapped transfer depends on cache settings. The cache configuration register in the SPI memory device must be appropriately configured before enabling memory-mapped mode. Some of the high performance modes like merge, wrap, and transfer size depend on cache parameters.
- SPI memory-mapped MDMA reads do not support wrapping. For MDMA reads, limit the DMA_CFG.MSIZE field to 1 byte, 2 bytes or 4 bytes.
- There is not always tool support to change the SPI memory-mapped hardware setting or cache settings on-the-fly. Changing these settings can optimize the performance of code that accesses SPI memory in memory-mapped mode. It is expected that the SPI memory, SPI peripheral, and cache are programmed to one specific set of control settings for the whole application. Profiling or benchmarking of the actual application can be done to find the setting that works best.

SPI Interrupt Signals

The SPI controller supports three types of interrupt request signals that correspond to data, status, and error conditions.

Data Interrupts

The SPI target supports two data interrupt channels – receive and transmit. These interrupt signals are multiplexed into the DMA request lines. Since the target interfaces with separate read and write interfaces with DMA, the read and write data interrupts are independent. When the DMA channels are not used, the interrupts are routed directly to the system event controller. The interrupts occupy the same vector locations as the corresponding DMA channels.

Each of the data interrupt requests can be individually controlled. Program the SPI_RXCTL.RDR and SPI_TXCTL.TDR bit fields for receive and transmit, respectively. When receive is enabled, the RX interrupt request is issued whenever there is data available in the receive data path for reading. (The event occurs according to the SPI_RXCTL.RDR bit setting.) When transmit is enabled, the TX interrupt request is issued whenever the transmit data path can be written. (The event occurs according to the SPI_TXCTL.TDR setting.) DMA data interrupts are compatible with second-generation DMA to incorporate urgent data requests and transfer finish interrupt requests apart from the usual data request interrupts. Transmit interrupt requests operate independently from the word counter-value in the SPI_TWC register.

Status Interrupts

The SPI controller supports several status interrupt requests to indicate different conditions of the receiver and transmitter. All status interrupt requests can be masked. Status interrupt requests are signaled directly through a single SPI status IRQ line. The line cannot be combined with the SPI error IRQ line for some processors. The ***SPI Status Interrupts*** table describes the status interrupt requests that are available for the SPI controller.

Table 20-14: SPI Status Interrupts

SPI_STAT Bit	Description
SPI_STAT.RUWM	Receive FIFO urgent watermark interrupt request. Issued when the level of the RFIFO breaches the watermark set in the SPI_RXCTL.RUWM field. It is cleared when the level of the RFIFO reaches the watermark set in the SPI_RXCTL.RRWM field. If the RX channel is configured in DMA mode, SPI_RXCTL.RUWM is multiplexed with the data request.
SPI_STAT.TUWM	Transmit FIFO urgent watermark interrupt request. Issued when the level of the TFIFO breaches the watermark set using the SPI_TXCTL.TUWM bit. It is cleared when the level of the TFIFO reaches the watermark set in the SPI_TXCTL.TRWM field. If the TX channel is configured in DMA mode, SPI_STAT.TUWM is multiplexed with the data request.
SPI_STAT.TS	Transmit start interrupt request. Issued when the start of a transmit burst is detected by loading of the SPI_TWC register with the contents of the SPI_TWCR register.
SPI_STAT.RS	Receive start interrupt request. Issued when the start of a receive burst is detected by the loading of SPI_RWC with the contents of SPI_RWCR.
SPI_STAT.TF	Transmit finish interrupt request. Issued when a transmit burst completes (SPI_TWC decrements to zero).
SPI_STAT.RF	Receive finish interrupt request. Issued when a receive burst completes (SPI_RWC decrements to zero).

Error Conditions

The SPI controller supports interrupt requests upon several different error conditions. All interrupt requests are maskable. The individual error indications combine into a single SPI error IRQ signal, which can be multiplexed on some processors with the aggregated SPI status IRQ signal. The ***SPI Error Interrupts*** table details the possible error indications.

Error conditions arise depending on which of the channels (transmit or receive) are enabled. If a channel is disabled, all errors related to it are ignored. When both channels are enabled, errors from both channels are enabled.

Table 20-15: SPI Error Interrupts

Bit	Description
SPI_STAT.MF	Mode fault. Signaled when another device also tries to be a controller in a multi-controller system and drives the SPI_TS input low. This error is signaled in controller mode operation.
SPI_STAT.TUR	Transmission error. Signaled when an underflow condition occurs on the transmit channel. This event occurs when a new transfer starts but SPI_TFIFO is empty. This error does not occur in controller transmit initiating mode since SPI_TFIFO *Not Empty* is one of the conditions for transfer initiation.
SPI_STAT.ROR	Reception error. Signaled when an overflow condition occurs on the receive channel. This event occurs when a new data word is received, but the SPI_RFIFO is full. This error condition does not occur in controller receive initiating mode since SPI_RFIFO *Not Full* is one of the conditions for transfer initiation.
SPI_STAT.TC	Transmit collision error. Signaled when loading data to the transmit shift register happens near the first transmitting edge of SPI_CLK. In target mode of operation, the SPI controller is unaware of when the next transfer starts. Loading of data to the transmit shift register can happen just after the transmitting edge. This event results in the setup time not being met for the first bit transmitted. The transmitted data is corrupt. In SPI_CTL.CPHA 1 mode, the first SPI_CLK edge is taken as the first transmitting edge. If SPI_CTL.CPHA =0, then the last SPI_CLK edge of the last transmission (SPI_CTL.POLSEL =1) or target select deassertion (SPI_CTL.POLSEL =0) is taken as the first transmitting edge. This error is signaled only in the target mode of operation. In controller mode of operation, loading of data happens before the first transmitting edge of SPI_CLK.

SPI Programming Concepts

The following sections provide general programming guidelines and procedures.

Programming Guidelines

It is acceptable to program SPI_RXCTL and SPI_TXCTL registers after programming the SPI_CTL register. However, program the initiating mode register and its counter-register, if enabled, after the non-initiating mode register. For example, if transmit is the initiating mode and receive is the non-initiating mode, then program the SPI_RXCTL and SPI_RWC registers before the SPI_TXCTL and SPI_TWC registers. If enabling both transmit and receive in initiating mode, enable the SPI_CTL register after programming both the SPI_RXCTL and SPI_TXCTL registers.

These programming guidelines prevent SPI from starting a transfer when SPI registers are not fully programmed. Other ways of programming are also allowed when the initiating conditions prevent the start of communication until after programming of SPI registers is complete.

Avoid data corruption when changing the SPI module configuration. Do not change the configuration during a data transfer. Additionally, change the clock polarity only when no target is selected. However, an exception to this rule exists. When an SPI communication link consists of a single controller and target, SPI_CTL.ATSEL = 0. The target select input of the target is permanently tied low. In this case, the target is always selected. Avoid data corruption by enabling the target only after both the controller and target devices are configured.

The module supports 8, 16-bit and 32-bit word sizes. To ensure correct operation, configure both the controller and target with the same word size.

Controller Operation in Non-DMA Modes

This section describes the operation of the SPI as a controller in non-DMA mode.

1. Write to the SPI_TSEL register, setting one or more of the SPI select enable bits. This operation ensures that the desired targets are properly deselected while the controller is configured.
2. The SPI_RXCTL.RTI and SPI_TXCTL.TTI bits determine the SPI initiating mode. The initiating mode defines the primary transfer channel, and also the initiating condition for the transfer.
3. Write to the SPI_CLK, SPI_CTL, SPI_RXCTL, and SPI_TXCTL registers. This operation enables the device as a controller and configures the SPI system. It specifies the transfer modes and channels, appropriate word length, transfer format, baud rate, and other control information.

   ADDITIONAL INFORMATION: If SPI_RXCTL.RTI is enabled and SPI_TXCTL.TTI is not, write to the SPI_RXCTL register after writing into SPI_CTL, SPI_TXCTL, and SPI_TFIFO registers to prevent a transmit underrun for the first transfer.
4. If SPI_CTL.ATSEL=0, activate the desired targets by clearing one or more of the SPI_TSEL flag bits. Otherwise, the SPI hardware performs target activation.
5. The SPI controller then generates the programmed clock pulses on SPI_CLK and simultaneously shifts data out of SPI_COTI while shifting data in from SPI_CITO. Before a shift, the shift register is loaded with the contents of the SPI_TFIFO register. At the end of the transfer, the contents of the shift register are loaded into SPI_RFIFO.
6. Whenever the initiating conditions are satisfied, the SPI continues to send and receive words. If the transmit buffer remains empty or the receive buffer remains full, the device operates according to the states of the SPI_TXCTL.TDU and SPI_RXCTL.RDO bits.
7. It is possible to program a secondary channel in addition to the initiating channel. This feature allows usage of available channel resources for receives or transmits simultaneously with the initiating channel.

Target Operation in Non-DMA Modes

When a device is enabled as a target in a non-DMA mode, a transition of the $\overline{\text{SPI_TS}}$ select signal to the active state (low) triggers the start of a transfer. Or the first active edge of SPI_CLK triggers the start, depending on the state of SPI_CTL.CPHA bit. The interface operates in the following manner.

1. The core writes to the SPI_CTL, SPI_RXCTL, and SPI_TXCTL registers. The operation defines the mode of the serial link to be the same as the mode setup in the SPI controller.
2. To prepare for the data transfer, the core writes data to be transmitted into SPI_TFIFO.
3. Once the $\overline{\text{SPI_TS}}$ falling edge is detected, the target starts sending data on active SPI_CLK edges and sampling data on inactive SPI_CLK edges.
4. Reception or transmission continues until $\overline{\text{SPI_TS}}$ is released or until the target has received the proper number of clock cycles.

5. The target device continues to receive or transmit with each new falling edge transition on $\overline{\texttt{SPI_TS}}$ or active SPI_CLK edge. If the transmit buffer remains empty or the receive buffer remains full, the device operates according to the states of the SPI_TXCTL.TDU and SPI_RXCTL.RDO bits.

Configuring DMA Controller Mode

The SPI interface supports a write DMA channel and a read DMA channel. It can use these functions individually or in a lock-step manner in duplex mode (SPI_TXCTL.TTI= SPI_RXCTL.RTI=1).

1. Write to the appropriate DMA registers to enable the SPI DMA channel and to configure the necessary work units, access direction, word count, and so on.
2. Write to the SPI_TSEL register, setting one or more of the SPI flag select bits.
3. Write to the SPI_CLK and SPI_CTL registers, enabling the device as a controller and configuring the SPI system by specifying the appropriate word length, transfer format, baud rate, and so forth.
4. Write to SPI_RXCTL to configure SPI controller receive mode, or write to SPI_TXCTL to configure SPI controller transmit mode.
5. Finally, write to the SPI_RXCTL.REN bit to enable the receive channel, or write to SPI_TXCTL.TEN to enable the transmit channel.
6. When the SPI_RXCTL.RTI bit is enabled, a receive transfer is initiated upon enabling SPI_CTL.EN bit. If the receive word counter is enabled (SPI_RXCTL.RWCEN), then the SPI_RWC register must be non-zero for a transfer to initiate.

   ADDITIONAL INFORMATION: If enabling both receive and transmit DMA channels, but not enabling SPI_TXCTL.TTI, write to the SPI_RXCTL register after writing the SPI_CTL and SPI_TXCTL registers. In this way, a transmit underrun can be prevented for the first transfer. Subsequent transfers are initiated as the SPI reads data from the receive shift register and writes to the SPI receive FIFO. The SPI then requests a write from DMA to memory. Upon a DMA grant, the DMA engine reads a word from the SPI receive FIFO and writes to memory. New requests continue to be initiated as long as the receive FIFO does not fill up, when SPI_RWC does not become zero while SPI_RXCTL.RWCEN=1.
7. When SPI_TXCTL.TTI is enabled, the SPI controller requests DMA reads from memory when there is space for more data in the transmit pipe. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO. As long as transmit data is available in the FIFO, and the SPI_TWC register is non-zero when SPI_TXCTL.TWCEN=1, the SPI continues to initiate transfers until disabled.
8. When both the SPI_TXCTL.TTI and SPI_RXCTL.RTI bits are enabled, the SPI controller requests a DMA read from memory. However, there must be space for more data in the transmit pipe and the number of words written into the SPI must be less than SPI_TWC if SPI_TXCTL.TWCEN=1. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO.

   ADDITIONAL INFORMATION: As the SPI writes data from the transmit FIFO into the transmit shift register, it initiates a transfer on the SPI link. Data received from the transfer is moved from the SPI receive shift register to the receive FIFO. The SPI controller requests a write from DMA to memory. Upon a DMA

grant, the DMA engine reads a word from the receive FIFO and writes to memory. Transfer continues to be initiated when both receives and transmits can accommodate new data.

9. When the receive pipe fills up due to unavailability of DMA grants, the transmit pipe stalls until the pipe is drained. If the transmit pipe fills up, the SPI stops requesting for DMA writes. When the value in SPI_RWC expires, further write-requests to DMA stop. However, data already written into the transmit FIFO is sent, and read requests to DMA continue until the receive data is read from the receive FIFO.
10. The SPI then generates the programmed clock pulses on SPI_CLK and simultaneously shifts data out of SPI_COTI while shifting data in from SPI_CITO. For receive transfers, the value in the shift register is loaded into the SPI_RFIFO register at the end of the transfer. For transmit transfers, the value in the SPI_TFIFO register is loaded into the shift register at the start of the transfer.

Configuring DMA Target Mode Operation

This mode occurs when the SPI is enabled as a target and the DMA engine is configured to transmit or receive data. A transition of the SPI_TS signal to the active-low state triggers the start of a transfer. Or, the first active edge of SPI_CLK triggers the start of a transfer, depending on the state of the SPI_CTL.CPHA bit. The following steps illustrate the SPI receive or transmit DMA sequence in an SPI target (in response to a controller command). The SPI supports a receive DMA channel and a transmit DMA channel.

1. Write to the appropriate DMA registers to enable the SPI DMA channel and configure the necessary work units, access direction, word count, and so on.
2. Write to the SPI_CTL, SPI_RXCTL, and SPI_TXCTL registers to define the mode of the serial link to be the same as the mode configured in the SPI controller.
3. When the receive channel is enabled (SPI_RXCTL.REN is asserted), the following actions occur:
    a. Once the target select input is active, the target starts receiving and transmitting data on active SPI_CLK edges.
    b. The value in the shift register is loaded into the SPI_RFIFO register at the end of the transfer.
    c. Once SPI_RFIFO has valid data, it requests a write from DMA to memory.
    d. Upon a DMA grant, the DMA engine reads a word from the receive FIFO and writes to memory.
    e. When there is data in the receive FIFO, the SPI target continues to request a DMA write to memory. The DMA engine continues to read a word from the FIFO and writes to memory. The SPI target continues receiving words on active SPI_CLK edges when the SPI_TS input is active.
    f. When the data collected in the receive pipe breaches the set level, and the DMA engine cannot keep up with the receive rate, the target can deassert the SPI_RDY signal. This signaling throttles the controller. The receive pipe level is set according to the SPI_CTL.FCWM field. The signal is deasserted as the DMA drains the receive FIFO. Alternatively, the SPI can use the SPI_RXCTL.RDO bit to decide when the incoming data is discarded or overwritten into the receive FIFO (when SPI_CTL.FCEN is inactive).
4. When the transmit channel is enabled (SPI_TXCTL.TEN is asserted), the following actions occur:

a. The SPI requests a DMA read from memory.

b. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO.

c. The SPI then reads DMA data from the transmit FIFO and writes to the transmit shift register, awaiting the start of the next transfer.

d. Once the target select input is active, the target starts receiving and transmitting data on active SPI_CLK edges.

e. When there is room in the transmit FIFO, the SPI target continues to request a DMA read from memory. The DMA engine continues to read a word from memory and write to the transmit FIFO. The SPI target continues transmitting words on active SPI_CLK edges as long as the SPI_TS input is active.

f. When the number of outstanding data entries in the transmit pipe breaches the level set and the DMA cannot keep up with the transmit rate, the target deasserts the SPI_RDY signal. This signaling throttles the controller. The transmit pipe level is set according to the SPI_CTL.FCWM field. The signal is deasserted as the DMA fills the transmit FIFO. Alternately, the SPI_TXCTL.TDU bit decides the state of the transmit data (when SPI_CTL.FCEN is deasserted).

5. When both receive and transmit channels are enabled, the following actions occur after the actions for each channel. Transfers continue when both receive and transmit channels can accommodate new data.

a. When the receive pipe fills up due to the unavailability of DMA grant, the SPI interface stalls the controller by asserting the SPI_RDY pin. This signal is deasserted as the DMA drains the receive FIFO. Alternately, the SPI uses the SPI_RXCTL.RDO bit to decide when the incoming data is discarded or overwritten in the receive FIFO (when SPI_CTL.FCEN is deasserted).

b. When the transmit pipe fills up, the SPI stops requesting DMA writes until the pipe clears.

c. When there is an underflow problem in the transmit pipe, the target stalls the controller by deasserting SPI_RDY while the DMA fills the transmit FIFO. Alternately, the SPI uses the SPI_TXCTL.TDU bit to decide the state of the transmit data (when SPI_CTL.FCEN is deasserted).

ADSP-2184x SPI Register Descriptions

Serial Peripheral Interface (SPI) contains the following registers.

Table 20-16: ADSP-2184x SPI Register List

Name	Description
SPI_CLK	Clock Rate Register
SPI_CTL	Control Register
SPI_DLY	Delay Register
SPI_ILAT	Masked Interrupt Condition Register
SPI_ILAT_CLR	Masked Interrupt Clear Register

Table 20-16: ADSP-2184x SPI Register List (Continued)

Name	Description
SPI_IMSK	Interrupt Mask Register
SPI_IMSK_CLR	Interrupt Mask Clear Register
SPI_IMSK_SET	Interrupt Mask Set Register
SPI_MMRDH	Memory Mapped Read Header
SPI_MMTOP	SPI Memory Top Address
SPI_RFIFO	Receive FIFO Data Register
SPI_RWC	Received Word Count Register
SPI_RWCR	Received Word Count Reload Register
SPI_RXCTL	Receive Control Register
SPI_TSEL	Target Select Register
SPI_STAT	Status Register
SPI_TFIFO	Transmit FIFO Data Register
SPI_TWC	Transmitted Word Count Register
SPI_TWCR	Transmitted Word Count Reload Register
SPI_TXCTL	Transmit Control Register

Clock Rate Register

The SPI_CLK register selects the baud rate for SPI data transfers, relating this rate to the SPI serial clock (SPI clock) and the system clock (SCLK).

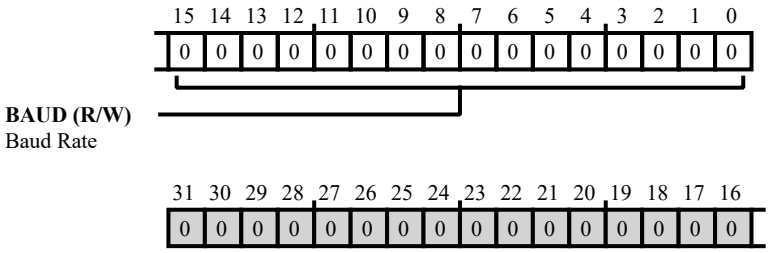


Figure 20-20: SPI_CLK Register Diagram

Table 20-17: SPI_CLK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	BAUD	Baud Rate. The SPI_CLK.BAUD bits set the SPI baud rate according to the formula: BAUD = (SCLK / SPI Clock) - 1

Control Register

The SPI_CTL register enables the SPI and configures settings for operating modes, communication protocols, and buffer operations.

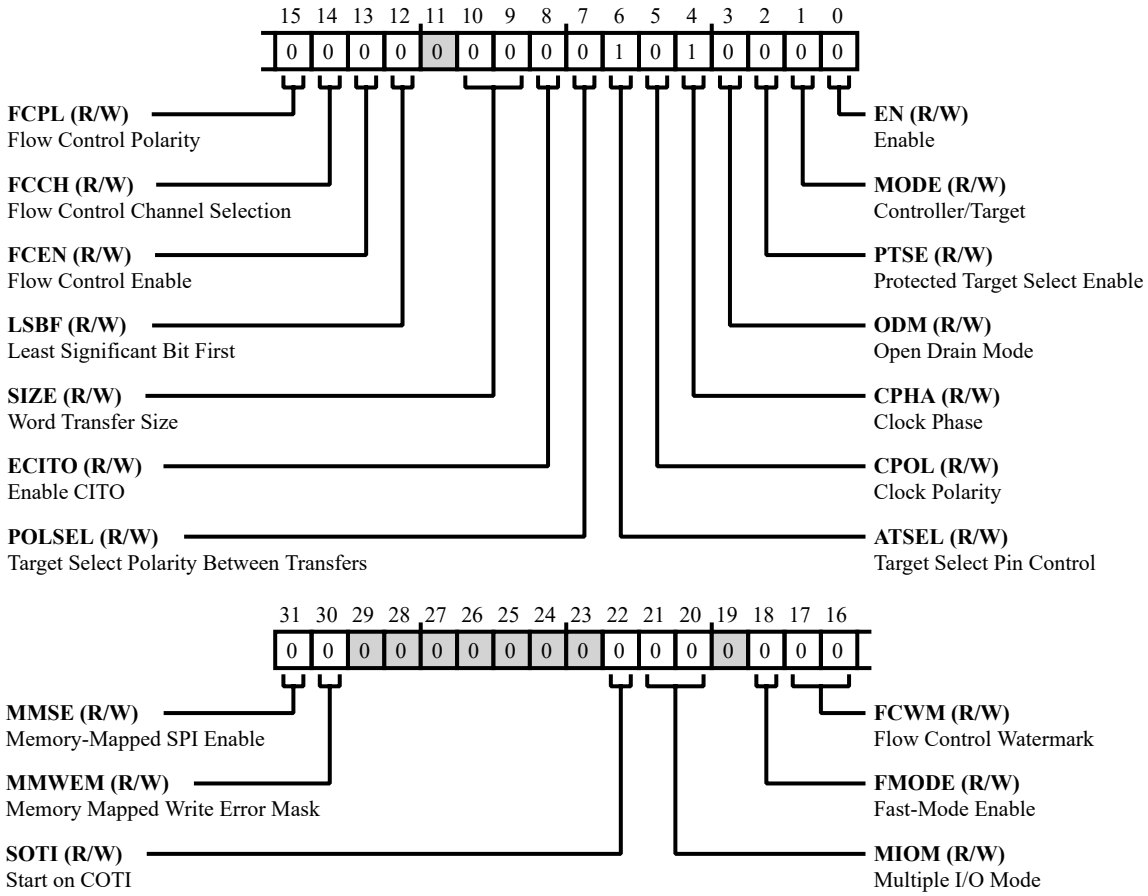


Figure 20-21: SPI_CTL Register Diagram

Table 20-18: SPI_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	MMSE	Memory-Mapped SPI Enable. When the SPI_CTL.MMSE bit is asserted, communication to an SPI memory device is automated such that the memory it contains is accessible directly through the read of processor address space assigned to it. (As far as the SPI peripheral is concerned, this includes all read accesses received by the SPI peripherals system crossbar port.) Note that when memory-mapped access of SPI memory is enabled, attempts to communicate with the SPI device using legacy methods are blocked and receive fabric error responses are generated. Legacy methods include any direct access made to the Tx and Rx FIFOs, whether initiated by DMA or processor MMR access.	
		0	Hardware automated access of memory-mapped SPI memory disabled.
		1	Hardware-automated access of memory-mapped SPI memory enabled.
30 (R/W)	MMWEM	Memory Mapped Write Error Mask. The SPI_CTL.MMWEM bit specifies whether an error response is returned to the fabric upon write attempts to address space reserved for memory-mapped reads of SPI memory.	
		0	Write error response returned upon write attempts to memory-mapped SPI memory
		1	Write error response masked (not returned) upon write attempts to memory-mapped SPI memory
22 (R/W)	SOTI	Start on COTI. The SPI_CTL.SOTI bit is valid only when SPI_CTL.MIOM is enabled for either DIOM or QIOM, and this bit selects the starting pin and the bit placement on pins for these modes. In DIOM, by default, (SPI_CTL.SOTI =0) SPI sends the first bit on the SPI_CITO pin and the second bit on the SPI_COTI pin. In QIOM, by default, the SPI sends the first bit on the SPI_D3 pin, the second bit on the SPI_D2 pin, the third bit on the SPI_CITO pin and the fourth bit on the SPI_COTI pin. This order can be reversed by setting the SPI_CTL.SOTI bit. When this bit is set, the SPI sends the first bit on the SPI_COTI pin. The first bit referred to here depends on the SPI_CTL.LSBF bit setting (MSB bit or LSB bit).	
		0	Start on CITO (DIOM) or Start on SPI_D3
		1	Start on COTI

Table 20-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:20 (R/W)	MIOM	Multiple I/O Mode. The SPI_CTL.MIOM bits enable SPI operation in dual I/O mode (DIOM) or quad I/O mode (QIOM). These bits can only be changed when the SPI is disabled (SPI_CTL.EN =0).	
		0	No MIOM (disabled)
		1	DIOM operation
		2	QIOM operation
		3	Reserved
18 (R/W)	FMODE	Fast-Mode Enable. The SPI_CTL.FMODE bit enables fast mode operation for SPI receive transfers. SPI transmit operations in fast mode are the same as normal mode.	
		0	Disable
		1	Enable
17:16 (R/W)	FCWM	Flow Control Watermark. The SPI_CTL.FCWM bits select the watermark level of the transmit channel (SPI_TFIFO buffer) or receive channel (SPI_RFIFO buffer) that triggers flow control operation. These bits are applicable only when the SPI is a target (SPI_CTL.MODE = 0) and flow control is enabled (SPI_CTL.FCEN =1). When the watermark condition is met, the SPI controller deasserts the SPI_RDY pin.	
		0	TFIFO empty or RFIFO full
		1	TFIFO 75% or more empty, or RFIFO 75% or more full
		2	TFIFO 50% or more empty, or RFIFO 50% or more full
		3	Reserved
15 (R/W)	FCPL	Flow Control Polarity. The SPI_CTL.FCPL bit selects flow control polarity for the SPI_RDY pin when flow control is enabled. When the SPI_RDY pin is active, the SPI is indicating it is ready for data transfer.	
		0	Active-low RDY
		1	Active-high RDY

Table 20-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	FCCH	Flow Control Channel Selection. The SPI_CTL.FCCH bit selects whether the SPI applies flow control to the transmit channel (SPI_TFIFO buffer) or receive channel (SPI_RFIFO buffer). This bit is applicable only when the SPI is a target and flow control is enabled.	
		0	Flow Control on RX Buffer
		1	Flow Control on TX Buffer
13 (R/W)	FCEN	Flow Control Enable. The SPI_CTL.FCEN bit enables SPI flow control operation, which permits slow target devices to interface with fast controller devices. This bit controls the operation of the SPI_RDY pin. Note that options for flow control operation are available using the SPI_CTL.FCCH, SPI_CTL.FCPL, and SPI_CTL.FCWM bits.	
		0	Disable
		1	Enable
12 (R/W)	LSBF	Least Significant Bit First. The SPI_CTL.LSBF bit selects whether the SPI transmits/receives data as LSB first (little endian) or MSB first (big endian). This bit can only be changed when the SPI is disabled.	
		0	MSB sent/received first (big endian)
		1	LSB sent/received first (little endian)
10:9 (R/W)	SIZE	Word Transfer Size. The SPI_CTL.SIZE bits select the SPI transfer word size as 8, 16 or 32 bits. To ensure correct operation, both the controller and target must be configured with the same word size. This bit can only be changed when the SPI is disabled (SPI_CTL.EN =0).	
		0	8-bit word
		1	16-bit word
		2	32-bit word
		3	Reserved
8 (R/W)	ECITO	Enable CITO. The SPI_CTL.ECITO bit enables controller-in-target-out (CITO) mode. This SPI mode is applicable only when the SPI is a target.	
		0	Disable
		1	Enable

Table 20-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	POLSEL	Target Select Polarity Between Transfers. The SPI_CTL.POLSEL bit selects the state (polarity) for the SPI_SEL[n] pin between SPI transfers when the SPI is a controller and hardware target select assertion is enabled (SPI_CTL.ATSEL =1). In target mode, this bit affects the detection of both transmit collision (SPI_STAT.TC and underrun (SPI_STAT.TUR) errors.	
		0	Deassert Target Select (high)
		1	Assert Target Select (low)
6 (R/W)	ATSEL	Target Select Pin Control. The SPI_CTL.ATSEL bit selects whether the SPI hardware sets the SPI_SEL[n] pin output value (ignoring the target select SPI_TSEL.TSEL1 - SPI_TSEL.TSEL7 bits) or whether software control of the target select bits set the SPI_SEL[n] pin output value. This feature is applicable only when the SPI is a controller. When hardware control is enabled, the SPI_SEL[n] pin output is asserted during the transfers, and the pin polarity between transfers is selected by the SPI_CTL.POLSEL bit. When software control is enabled, the SPI_SEL[n] pin output value is set through software control of the target select bits, and as such, the pin may either remain asserted (low) or be deasserted between transfers.	
		0	Software Target Select Control
		1	Hardware Target Select Control
5 (R/W)	CPOL	Clock Polarity. The SPI_CTL.CPOL bit selects whether the SPI uses an active-low or active-high signal for the SPI clock (SPI_CLK). This bit works with the SPI_CTL.CPHA bit to select combinations of clock phase and polarity for the SPI_CLK pin. This bit can only be changed when the SPI is disabled.	
		0	Active-high SPI CLK
		1	Active-low SPI CLK
4 (R/W)	CPHA	Clock Phase. The SPI_CTL.CPHA bit selects whether the SPI starts toggling the signal for the SPI clock (SPI_CLK) from the start of the first data bit or from the middle of the first data bit. The SPI_CTL.CPHA bit works with the SPI_CTL.CPOL bit to select combinations of clock phase and polarity for the SPI_CLK pin. This bit can only be changed when the SPI is disabled.	
		0	SPI CLK toggles from middle
		1	SPI CLK toggles from start

Table 20-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	ODM	Open Drain Mode. The SPI_CTL.ODM bit configures the data output pins (SPI_COTI and SPI_CITO) to behave as open drain outputs, which prevents contention and possible damage to pin drivers in multi-controller or multi-target SPI systems. When SPI_CTL.ODM is enabled and the SPI is a controller, the SPI three-states the SPI_COTI pin when the data driven out on COTI is a logic-high. The SPI does not three-state the SPI_COTI pin when the driven data is a logic-low. When SPI_CTL.ODM is enabled and the SPI is a target, the SPI three-states the SPI_CITO pin when the data driven out on SPI_CITO is a logic-high. Note that an external pull-up resistor is required on both the SPI_COTI and SPI_CITO pins when SPI_CTL.ODM is enabled.	
		0	Disable
		1	Enable
2 (R/W)	PTSE	Protected Target Select Enable. The SPI_CTL.PTSE bit enables the $\overline{\text{SPI_TS}}$ pin to provide error detection input in a multi-controller environment when the SPI is in controller mode. If some other device in the system asserts the $\overline{\text{SPI_TS}}$ pin while SPI is enabled as a controller (and SPI_CTL.PTSE is enabled), this condition causes a mode fault error.	
		0	Disable
		1	Enable
1 (R/W)	MODE	Controller/Target. The SPI_CTL.MODE bit toggles the SPI between controller mode and target mode. This bit can only be changed when the SPI is disabled.	
		0	Target
		1	Controller
0 (R/W)	EN	Enable. The SPI_CTL.EN bit enables SPI operation.	
		0	Disable SPI module
		1	Enable

Delay Register

The SPI_DLY register selects a transfer delay and the lead/lag timing between target select signals and SPI clock edge assertion/deassertion.

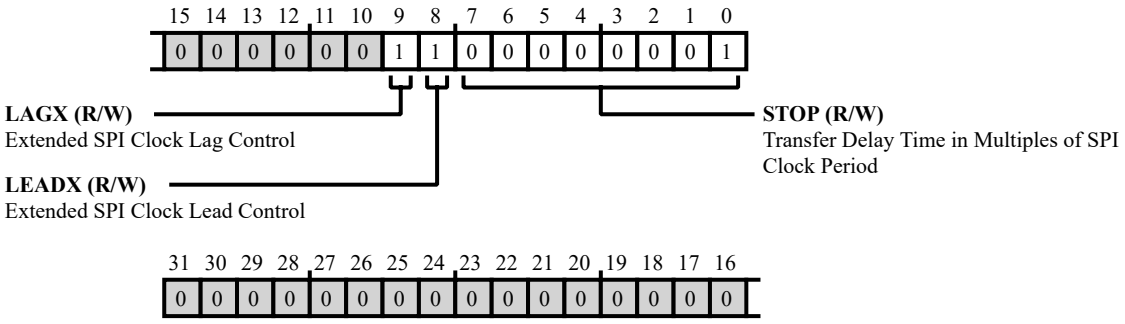


Figure 20-22: SPI_DLY Register Diagram

Table 20-19: SPI_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	LAGX	Extended SPI Clock Lag Control. The SPI_DLY.LAGX bit enables insertion of a 1-SPI_CLK cycle lag (extend lag) in the timing between the target select (SPI_SEL[n]) assertion and first SPI clock edge.	
		0	Disable
		1	Enable
8 (R/W)	LEADX	Extended SPI Clock Lead Control. The SPI_DLY.LEADX bit enables insertion of a 1-SPI_CLK cycle lead (extend lead) in the timing between the target select (SPI_SEL[n]) deassertion and last SPI clock edge.	
		0	Disable
		1	Enable
7:0 (R/W)	STOP	Transfer Delay Time in Multiples of SPI Clock Period. The SPI_DLY.STOP bits select a delay (number of stop bits in multiples of SPI clock duration) at the end of each SPI transfer. The default delay is the minimum value required to comply with the SPI protocol (1-bit duration). The SPI_DLY.STOP bits can be programmed with smaller delay values, resulting in continuous operation (for example, stop bits =0).	

Masked Interrupt Condition Register

The SPI_ILAT register latches interrupts, queuing the interrupt requests for service. When a condition is indicated by a bit in the SPI_STAT register and the corresponding interrupt request is unmasked in SPI_IMSK, the SPI latches the interrupt request bit in SPI_ILAT.

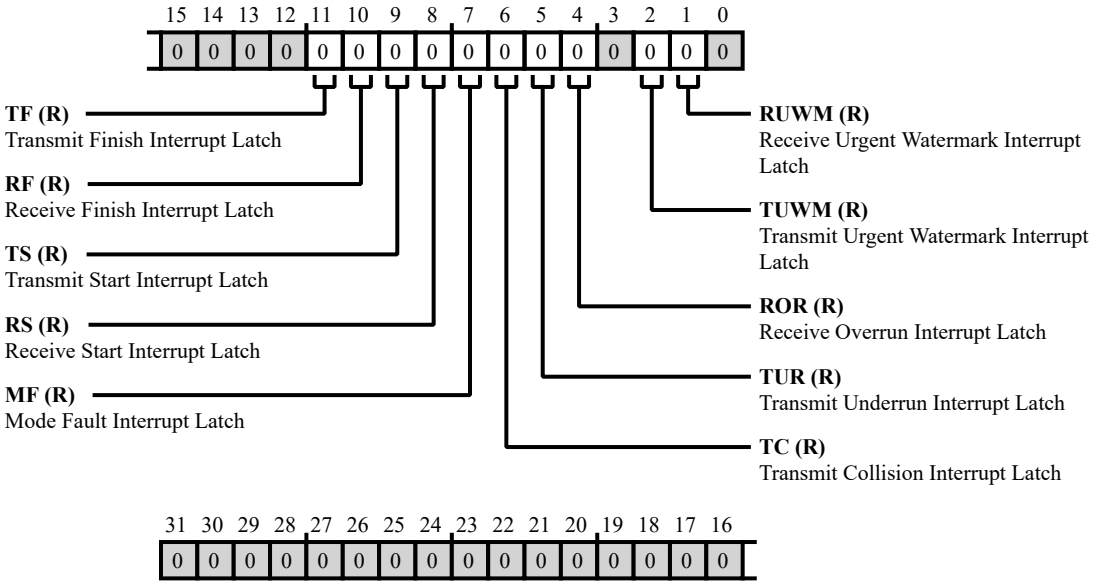


Figure 20-23: SPI_ILAT Register Diagram

Table 20-20: SPI_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	TF	Transmit Finish Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
10 (R/NW)	RF	Receive Finish Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
9 (R/NW)	TS	Transmit Start Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
8 (R/NW)	RS	Receive Start Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request

Table 20-20: SPI_ILAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	MF	Mode Fault Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
6 (R/NW)	TC	Transmit Collision Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
5 (R/NW)	TUR	Transmit Underrun Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
4 (R/NW)	ROR	Receive Overrun Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
2 (R/NW)	TUWM	Transmit Urgent Watermark Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
1 (R/NW)	RUWM	Receive Urgent Watermark Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request

Masked Interrupt Clear Register

The SPI_ILAT_CLR register permits clearing individual mask bits in the SPI_ILAT register without affecting other bits in the register. Use write-1-to-clear on a bit in the SPI_ILAT_CLR register to clear the corresponding bit in the SPI_ILAT register.

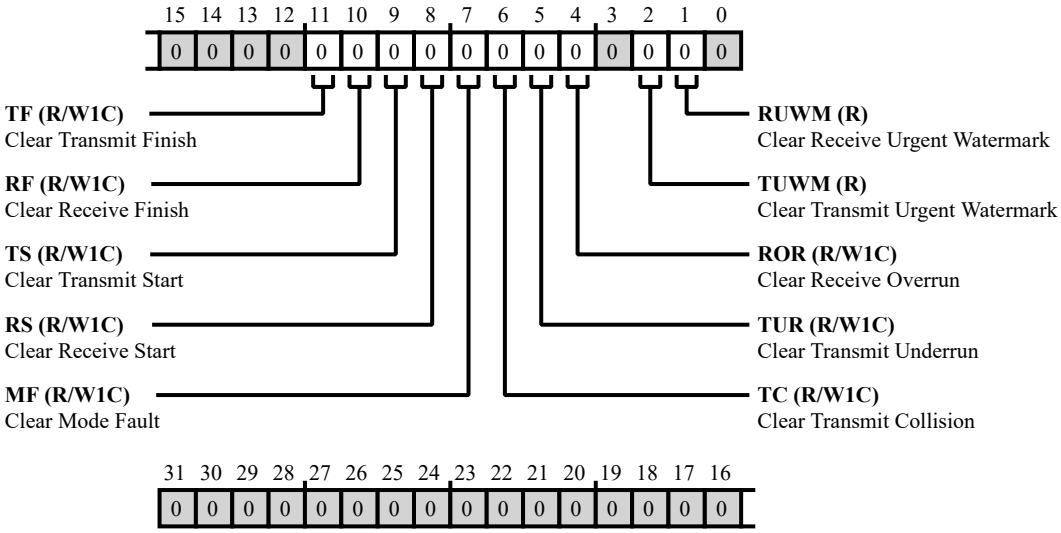


Figure 20-24: SPI_ILAT_CLR Register Diagram

Table 20-21: SPI_ILAT_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W1C)	TF	Clear Transmit Finish. The SPI_ILAT_CLR.TF bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
10 (R/W1C)	RF	Clear Receive Finish. The SPI_ILAT_CLR.RF bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
9 (R/W1C)	TS	Clear Transmit Start. The SPI_ILAT_CLR.TS bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit

Table 20-21: SPI_ILAT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	RS	Clear Receive Start. The SPI_ILAT_CLR.RS bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
7 (R/W1C)	MF	Clear Mode Fault. The SPI_ILAT_CLR.MF bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
6 (R/W1C)	TC	Clear Transmit Collision. The SPI_ILAT_CLR.TC bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
5 (R/W1C)	TUR	Clear Transmit Underrun. The SPI_ILAT_CLR.TUR bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
4 (R/W1C)	ROR	Clear Receive Overrun. The SPI_ILAT_CLR.ROR bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
2 (R/NW)	TUWM	Clear Transmit Urgent Watermark. The SPI_ILAT_CLR.TUWM bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit

Table 20-21: SPI_ILAT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	RUWM	Clear Receive Urgent Watermark. The `SPI_ILAT_CLR.RUWM` bit clears the corresponding mask bit in the `SPI_ILAT` register.	
		0	No effect
		1	Clear mask bit

Interrupt Mask Register

The SPI_IMSK register unmasks (enables) or masks (disables) SPI interrupt requests. When a condition is indicated by a bit in the SPI_STAT register and the corresponding interrupt request is unmasked in SPI_IMSK, the SPI latches the interrupt request bit in the SPI_ILAT register, queuing the interrupt request for service.

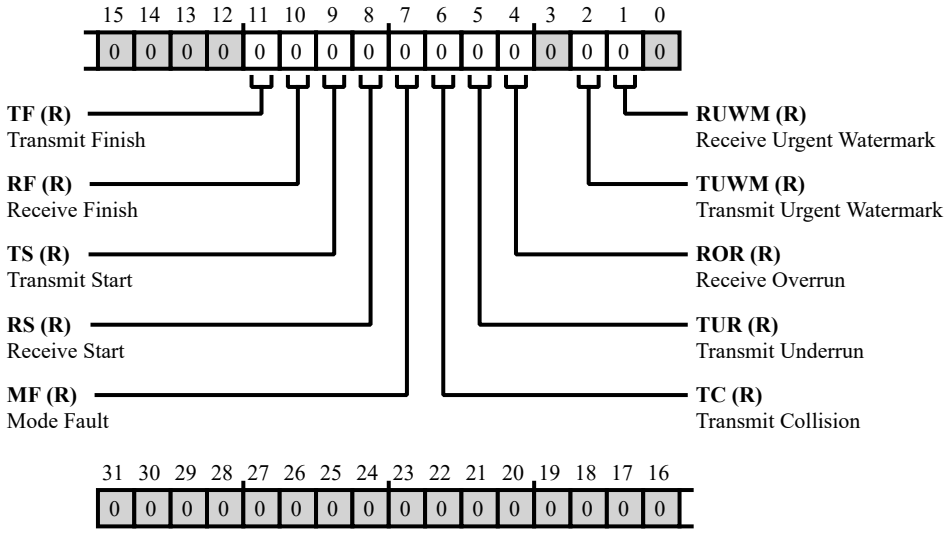


Figure 20-25: SPI_IMSK Register Diagram

Table 20-22: SPI_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	TF	Transmit Finish. The SPI_IMSK.TF bit unmasks (enables) or masks (disables) the TF interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
10 (R/NW)	RF	Receive Finish. The SPI_IMSK.RF bit unmasks (enables) or masks (disables) the RF interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
9 (R/NW)	TS	Transmit Start. The SPI_IMSK.TS bit unmasks (enables) or masks (disables) the TS interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request

Table 20-22: SPI_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	RS	Receive Start. The SPI_IMSK.RS bit unmasks (enables) or masks (disables) the RS interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
7 (R/NW)	MF	Mode Fault. The SPI_IMSK.MF bit unmasks (enables) or masks (disables) the MF interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
6 (R/NW)	TC	Transmit Collision. The SPI_IMSK.TC bit unmasks (enables) or masks (disables) the TC interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
5 (R/NW)	TUR	Transmit Underrun. The SPI_IMSK.TUR bit unmasks (enables) or masks (disables) the TUR interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
4 (R/NW)	ROR	Receive Overrun. The SPI_IMSK.ROR bit unmasks (enables) or masks (disables) the ROR interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
2 (R/NW)	TUWM	Transmit Urgent Watermark. The SPI_IMSK.TUWM bit unmasks (enables) or masks (disables) the TUWM interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
1 (R/NW)	RUWM	Receive Urgent Watermark. The SPI_IMSK.RUWM bit unmasks (enables) or masks (disables) the RUWM interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request

Interrupt Mask Clear Register

The SPI_IMSK_CLR register permits clearing individual mask bits in the SPI_IMSK register without affecting other bits in the register. Use write-1-to-clear on a bit in the SPI_IMSK_CLR register to clear the corresponding bit in the SPI_IMSK register.

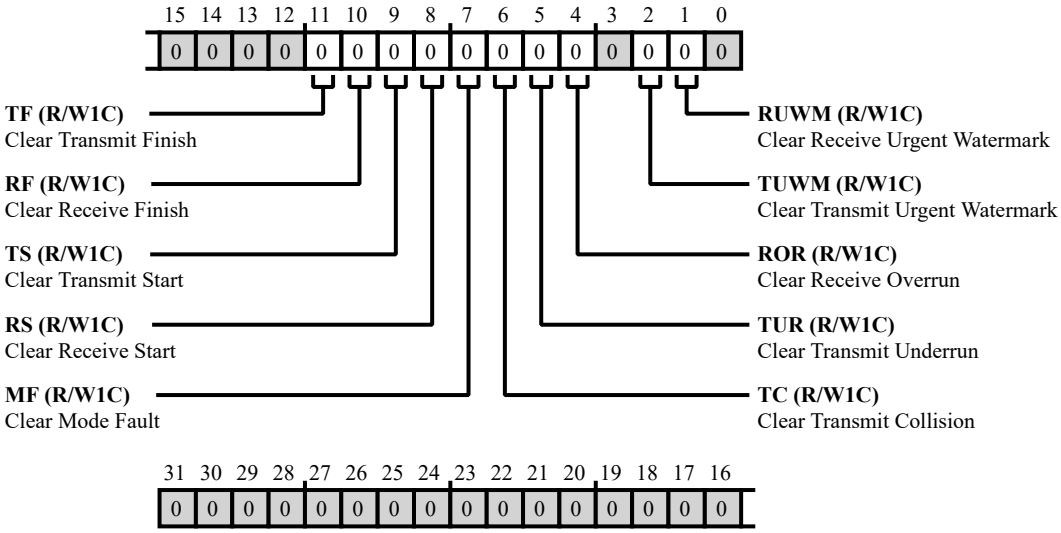


Figure 20-26: SPI_IMSK_CLR Register Diagram

Table 20-23: SPI_IMSK_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W1C)	TF	Clear Transmit Finish. The SPI_IMSK_CLR.TF bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
10 (R/W1C)	RF	Clear Receive Finish. The SPI_IMSK_CLR.RF bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
9 (R/W1C)	TS	Clear Transmit Start. The SPI_IMSK_CLR.TS bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit

Table 20-23: SPI_IMSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	RS	Clear Receive Start. The SPI_IMSK_CLR.RS bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
7 (R/W1C)	MF	Clear Mode Fault. The SPI_IMSK_CLR.MF bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
6 (R/W1C)	TC	Clear Transmit Collision. The SPI_IMSK_CLR.TC bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
5 (R/W1C)	TUR	Clear Transmit Underrun. The SPI_IMSK_CLR.TUR bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
4 (R/W1C)	ROR	Clear Receive Overrun. The SPI_IMSK_CLR.ROR bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
2 (R/W1C)	TUWM	Clear Transmit Urgent Watermark. The SPI_IMSK_CLR.TUWM bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit

Table 20-23: SPI_IMSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	RUWM	Clear Receive Urgent Watermark. The `SPI_IMSK_CLR.RUWM` bit clears the corresponding mask bit in the `SPI_IMSK` register.	
		0	No effect
		1	Clear mask bit

Interrupt Mask Set Register

The SPI_IMSK_SET register permits setting individual mask bits in the SPI_IMSK register without affecting other bits in the register. Use write-1-to-set on a bit in the SPI_IMSK_SET register to set the corresponding bit in the SPI_IMSK register.

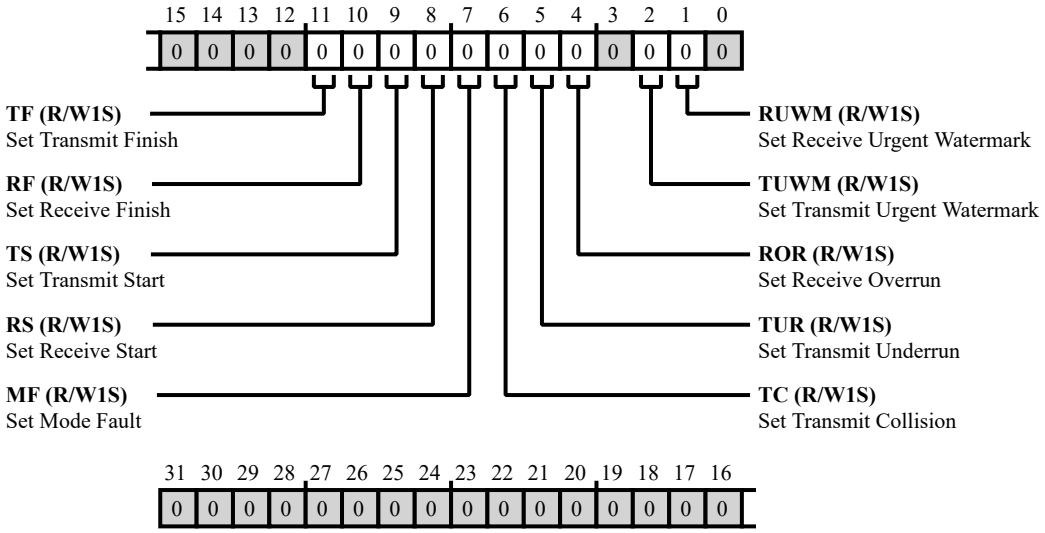


Figure 20-27: SPI_IMSK_SET Register Diagram

Table 20-24: SPI_IMSK_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W1S)	TF	Set Transmit Finish. The SPI_IMSK_SET.TF bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
10 (R/W1S)	RF	Set Receive Finish. The SPI_IMSK_SET.RF bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
9 (R/W1S)	TS	Set Transmit Start. The SPI_IMSK_SET.TS bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit

Table 20-24: SPI_IMSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1S)	RS	Set Receive Start. The SPI_IMSK_SET.RS bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
7 (R/W1S)	MF	Set Mode Fault. The SPI_IMSK_SET.MF bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
6 (R/W1S)	TC	Set Transmit Collision. The SPI_IMSK_SET.TC bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
5 (R/W1S)	TUR	Set Transmit Underrun. The SPI_IMSK_SET.TUR bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
4 (R/W1S)	ROR	Set Receive Overrun. The SPI_IMSK_SET.ROR bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
2 (R/W1S)	TUWM	Set Transmit Urgent Watermark. The SPI_IMSK_SET.TUWM bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit

Table 20-24: SPI_IMSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1S)	RUWM	Set Receive Urgent Watermark. The `SPI_IMSK_SET.RUWM` bit sets the corresponding mask bit in the `SPI_IMSK` register.	
		0	No effect
		1	Set mask bit

Memory Mapped Read Header

The SPI_MMRDH register enables the use of memory-mapped mode. This mode allows direct memory-mapped read accesses of an SPI memory device and is primarily used to directly execute instructions from an SPI FLASH memory without using a low-level software driver. All overhead tasks such as transmission of the read header, pin turnaround timing and receive data sizing are handled in hardware.

The memory-mapped access mode is enabled by setting the SPI_CTL.MMSE bit. The features within the SPI_MMRDH register include a command skip mode, variable length byte addressing, and independent multi-pin support for command transmission, address transmission and data reception. In addition, the command opcode and mode bytes are fully programmable.

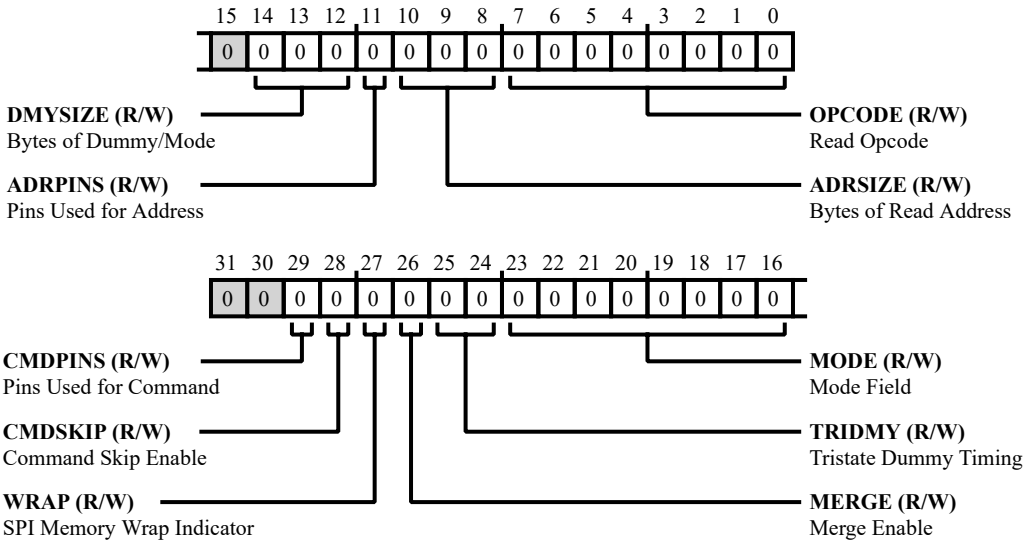


Figure 20-28: SPI_MMRDH Register Diagram

Table 20-25: SPI_MMRDH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	CMDPINS	Pins Used for Command. The SPI_MMRDH.CMDPINS bit specifies the number of pins to be used for command transmission. This bit must be set consistent with the expectations established by the read opcode. Hardware does not interpret SPI_MMRDH.OPCODE, but rather relies on this bit to specify behavior. When cleared, it overrides the SPI_CTL.MIOM bits. When set, it uses bits specified by the SPI_CTL.MIOM bit setting.	
		0	Use Only One Pin: COTI (overrides SPI_CTL.MIOM bits)
		1	Use Pins Specified by SPI_CTL.MIOM Bits

Table 20-25: SPI_MMRDH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	CMDSKIP	Command Skip Enable. The SPI_MMRDH.CMDSKIP bit enables command skip mode where the address is sent first and the OPCODE field is not sent (SPI_MMRDH.CMDSKIP bit =1). This mode is useful for supporting XIP (Execute-In-Place) operation where only the address is sent and the same read command is assumed. The SPI flash device must be primed with an initial read command, before the SPI_MMRDH.CMDSKIP bit is set.	
		0	OPCODE field is sent first followed by address
		1	OPCODE field is not sent; address is sent first
27 (R/W)	WRAP	SPI Memory Wrap Indicator. The SPI_MMRDH.WRAP bit must be set by software if software places a connected SPI memory device into a 8-byte, 16-byte or 32-byte wrap mode based on the ILINE and DLINE field setting of the cache configuration register address wrap mode. Software achieves this by transmitting a vendor specified command to the SPI memory device while the SPI_CTL.MMSE bit =0. If the SPI_MMRDH.WRAP bit =1, the SPI does not need to deassert the SPI target select signal and resend the read header in order to wrap to the cache line base when servicing misaligned cache fill requests. Although this improves cache fill efficiency, it requires that the SPI deassert the SPI target select pin and resend the read header whenever a DMA burst requests crosses 32 byte alignments. Setting this bit improves cache throughput but decreases DMA throughput.	
		0	SPI Memory auto increments address purely sequentially
		1	SPI Memory auto increments address but wraps within 32 Byte lines
26 (R/W)	MERGE	Merge Enable. When the SPI_MMRDH.MERGE bit is set, SPI hardware combines the two successive transfers. This increases the throughput rate when accessing a large number of sequential memory locations. For more information refer to the Merged Read Accesses section.	

Table 20-25: SPI_MMRDH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/W)	TRIDMY	Tristate Dummy Timing. The SPI_MMRDH.TRIDMY bits specify whether and when output pins are three-stated during the interval of time specified by the SPI_MMRDH.DMYSIZE bits. Output pins potentially three-stated include all pins which were used to transmit the address	
		0	Tristate outputs immediately
		1	Tristate outputs after 4 bits of dummy/mode are transmitted
		2	Tristate outputs after 8 bits of dummy/mode are transmitted
		3	Never tristate outputs (previously specified output state is held)
23:16 (R/W)	MODE	Mode Field. These bits specify up to a leading byte to be transmitted during the interval of time specified by the SPI_MMRDH.DMYSIZE bit field. This first byte, or a portion of it, is interpreted as mode bits when certain opcodes are used in conjunction with certain SPI memory devices. Mode bits are sent using the same number of pins which were used to transmit the address. Once sent, output pins will be held in their final resultant state until the conclusion of all dummy byte periods, unless three-stating the outputs is specified first by the SPI_MMRDH.TRIDMY bits.	
14:12 (R/W)	DMYSIZE	Bytes of Dummy/Mode. The SPI_MMRDH.DMYSIZE bit field specifies the number of bytes separating address transmission and read data return. Dummy bytes elapse assuming dummy bits are transmitted using the same number of pins which were used to transmit address.	
		0	0 Bytes
		1	1 Bytes
		2	2 Bytes
		3	3 Bytes
		4	4 Bytes
		5	5 Bytes
		6	6 Bytes
		7	7 Bytes

Table 20-25: SPI_MMRDH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	ADRPINS	Pins Used for Address. The SPI_MMRDH.ADRPINS bit specifies the number of pins to be used for address transmission. This bit must be set consistent with expectations established by read opcode. Hardware does not interpret the SPI_MMRDH.OPCODE, but rather relies on this bit to specify behavior.	
		0	Use Only One Pin: COTI (overrides SPI_CTL.MIOM bits)
		1	Use Pins Specified by SPI_CTL.MIOM Bits
10:8 (R/W)	ADRSIZE	Bytes of Read Address. The SPI_MMRDH.ADRSIZE bit field defines the number of bytes used to specify the read address. The read address is sent immediately following the transmission of opcode. Unlike opcode bits, address bits may be sent using either one or multiple pins. The number of pins is selected using the SPI_MMRDH.ADRPINS bit. The address sent to a connected SPI memory device is an echo of the read address received by the SPI peripheral target port. The least significant bytes of address are sent when the entire address is not sent.	
		0	1 Byte
		1	1 Byte
		2	2 Bytes
		3	3 Bytes
		4	4 Bytes
7:0 (R/W)	OPCODE	Read Opcode. The SPI_MMRDH.OPCODE bit field specifies the initial bits transmitted in response to a read request of SPI memory. Although any opcode may be sent, values 0x03, 0x0B, 0x3B, 0x6B, 0xBB, and 0xEB are likely to be the most commonly used. SPI_MMRDH.OPCODE is sent by the SPI without interpretation; the states of these bits have no effect beyond specifying what is initially shifted across the SPI interface.	

SPI Memory Top Address

The SPI_MMTOP register specifies the top populated address of a connected SPI memory device.

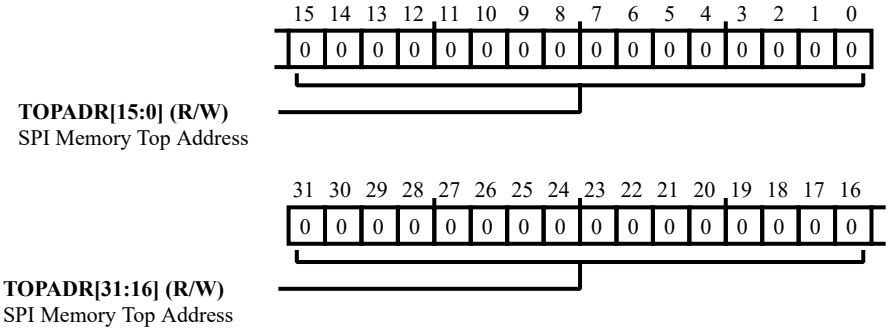


Figure 20-29: SPI_MMTOP Register Diagram

Table 20-26: SPI_MMTOP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TOPADR	SPI Memory Top Address. The SPI_MMTOP.TOPADR bit field specifies the top populated address of a connected SPI memory device. Attempts to access SPI memory are not blocked if this address is exceeded and an error is generated as part of the read response.

Receive FIFO Data Register

The SPI_RFIFO register has an interface to the receive shift register in the SPI and has an interface to the processor data buses. The top level of the buffer is visible to programs as the 32-bit SPI_RFIFO register, but the size (number of word locations) of the receive FIFO is actually flexible with transfer word size. The size of the receive FIFO is 8 if the word size is 8-bit, or the size is 4 if the word size is 16-bit, or the size is 2 if the word size is 32-bit.

Both controllers and targets may stop or stall receive transfers based on FIFO status. When the receive FIFO is full, the SPI controller stops initiating new transfers on the SPI if SPI_RXCTL.RTI is enabled. A target may stall the SPI interface when the content of the FIFO crosses the selected watermark. If data reception continues after SPI_RFIFO is full, the data in the receive FIFO is invalid. The SPI indicates this condition with receive overrun (SPI_STAT.ROR) error. This condition is possible when SPI_RXCTL.RTI =0 and SPI_RXCTL.REN =1 for a controller, or for a target that does not exercise flow control.

Note that the receive FIFO is reset (cleared) when the SPI is disabled after being enabled.

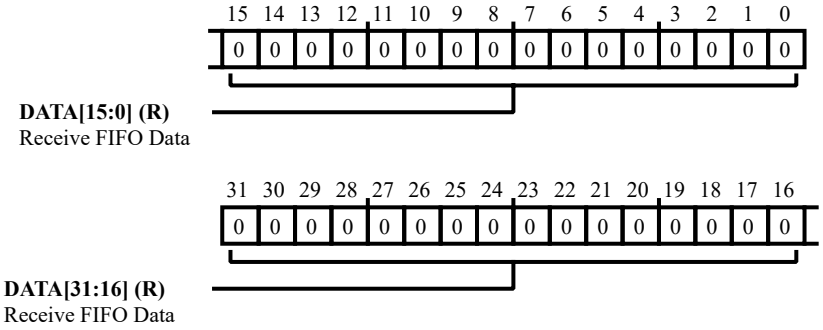


Figure 20-30: SPI_RFIFO Register Diagram

Table 20-27: SPI_RFIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Receive FIFO Data. The SPI_RFIFO.DATA bit field contains the FIFO receive data.

Received Word Count Register

The SPI_RWC register holds a count of the number of words remaining to be received by the SPI. To start the decrement of the word count in SPI_RWC, enable the receive word counter (SPI_RXCTL.RWCEN =1). The SPI uses the word count to control the duration of transfers and to signal the completion of a burst of transfers with the receive finish interrupt (SPI_ILAT.RF). In DMA mode, the SPI uses the SPI_RWC register to ensure that the number of frames received during a DMA transfer is equal to the number of words programmed in the DMA channel controller. The values programmed into the SPI_RWC registers should match the word count in the DMA configuration. The SPI_RWC register maintains the number of frames to be received in a transfer. The SPI_RWC should only be changed when the counter is disabled.

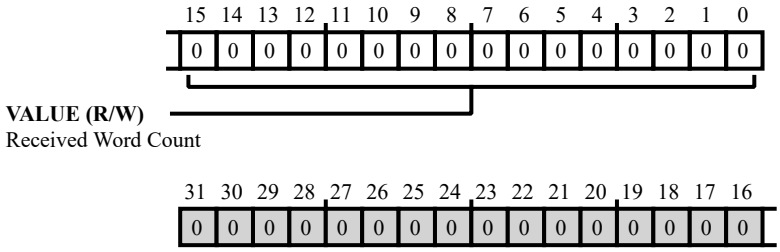


Figure 20-31: SPI_RWC Register Diagram

Table 20-28: SPI_RWC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Received Word Count. The SPI_RWC.VALUE bits hold the receive transfer word count.

Received Word Count Reload Register

The SPI_RWCR register holds the receive word count value that the SPI loads into the SPI_RWC register when the transfer count decrements to zero. To prevent the SPI from reloading the counter, use zero for the reload count value. The SPI_RWCR register should only be changed when the counter is disabled.

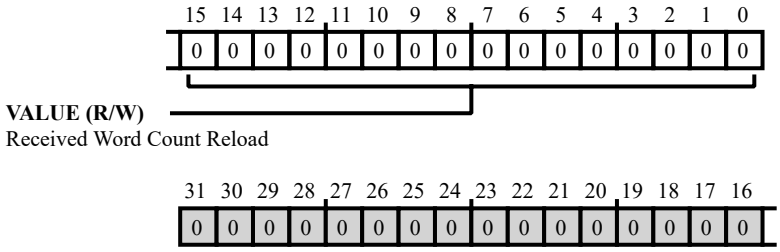


Figure 20-32: SPI_RWCR Register Diagram

Table 20-29: SPI_RWCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Received Word Count Reload. The SPI_RWCR.VALUE bits hold the receive transfer word count reload value.

Receive Control Register

The SPI_RXCTL register enables the SPI receive channel, initiates receive transfers, and configures SPI_RFIFO buffer watermark settings.

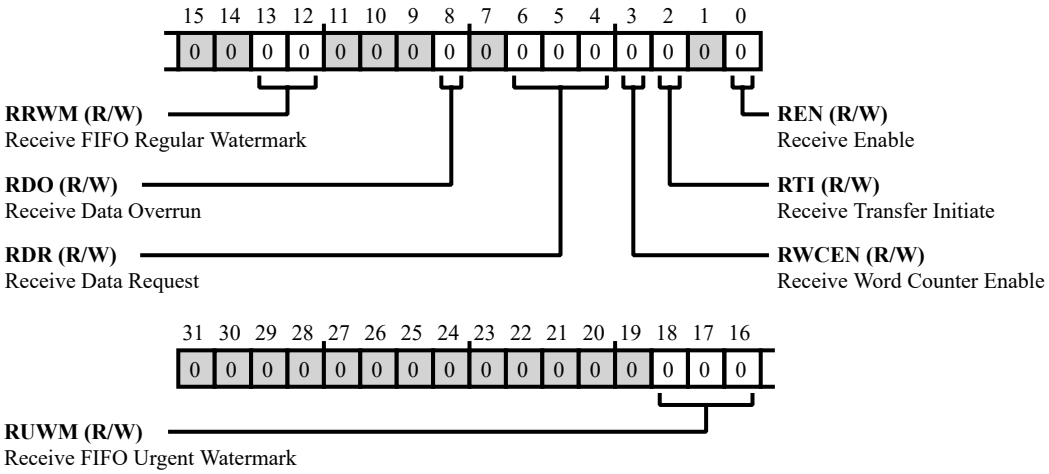


Figure 20-33: SPI_RXCTL Register Diagram

Table 20-30: SPI_RXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	RUWM	Receive FIFO Urgent Watermark. The SPI_RXCTL.RUWM bits select the receive FIFO (SPI_RFIFO) watermark level for urgent data bus requests. The SPI also uses this watermark level for generation of the SPI_ILAT.RUWM interrupt. When an urgent SPI_RFIFO watermark is enabled with SPI_RXCTL.RUWM, the SPI_RXCTL.RRWM selection is used as the deassertion condition for any SPI_ILAT.RUWM interrupts that are latched.	
		0	Disabled
		1	25% full RFIFO
		2	50% full RFIFO
		3	75% full RFIFO
		4	Full RFIFO
		5	Reserved
		6	Reserved
		7	Reserved

Table 20-30: SPI_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:12 (R/W)	RRWM	Receive FIFO Regular Watermark. The SPI_RXCTL.RRWM bits select the receive FIFO (SPI_RFIFO) watermark level for regular data bus requests. When an urgent SPI_RFIFO watermark is enabled with SPI_RXCTL.RUWM, the SPI_RXCTL.RRWM selection is used as the deassertion condition for any SPI_ILAT.RUWM interrupts that are latched.	
		0	Empty RFIFO
		1	RFIFO less than 25% full
		2	RFIFO less than 50% full
		3	RFIFO less than 75% full
8 (R/W)	RDO	Receive Data Overrun. The SPI_RXCTL.RDO bit selects handling for receive data requests when the receive buffer (SPI_RFIFO) is full. If enabled and SPI_RFIFO is full, the SPI overwrites old data in the buffer with incoming data. If disabled and SPI_RFIFO is full, the SPI keeps old data in the buffer and discards incoming data.	
		0	Discard incoming data if SPI_RFIFO is full
		1	Overwrite old data if SPI_RFIFO is full
6:4 (R/W)	RDR	Receive Data Request. The SPI_RXCTL.RDR bits select receive FIFO (SPI_RFIFO) watermark conditions that direct the SPI to generate a receive data request.	
		0	Disabled
		1	Not empty RFIFO
		2	25% full RFIFO
		3	50% full RFIFO
		4	75% full RFIFO
		5	Full RFIFO
		6	Reserved
		7	Reserved
3 (R/W)	RWCEN	Receive Word Counter Enable. The SPI_RXCTL.RWCEN bit enables the decrement of the SPI_RWC register when the count is not zero and SPI_RXCTL.RTI is enabled. Enabling SPI_RXCTL.RWCEN prevents receive overrun errors from occurring. The SPI_RXCTL.RWCEN bit is valid only when the SPI is a controller.	
		0	Disable
		1	Enable

Table 20-30: SPI_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	RTI	Receive Transfer Initiate. The SPI_RXCTL.RTI bit enables initiation of receive transfers if the receive FIFO (SPI_RFIFO) is not full. The bit also enables this initiation if SPI_RWC is not zero when SPI_RXCTL.RWCEN is enabled. Enabling SPI_RXCTL.RTI prevents receive overrun errors from occurring. The SPI_RXCTL.RTI bit is valid only when the SPI is a controller.	
		0	Disable
		1	Enable
0 (R/W)	REN	Receive Enable. The SPI_RXCTL.REN bit enables SPI receive channel operation.	
		0	Disable
		1	Enable

Target Select Register

The SPI_TSEL register enables the pins for output and indicates the state (high or low) of these pins when enabled.

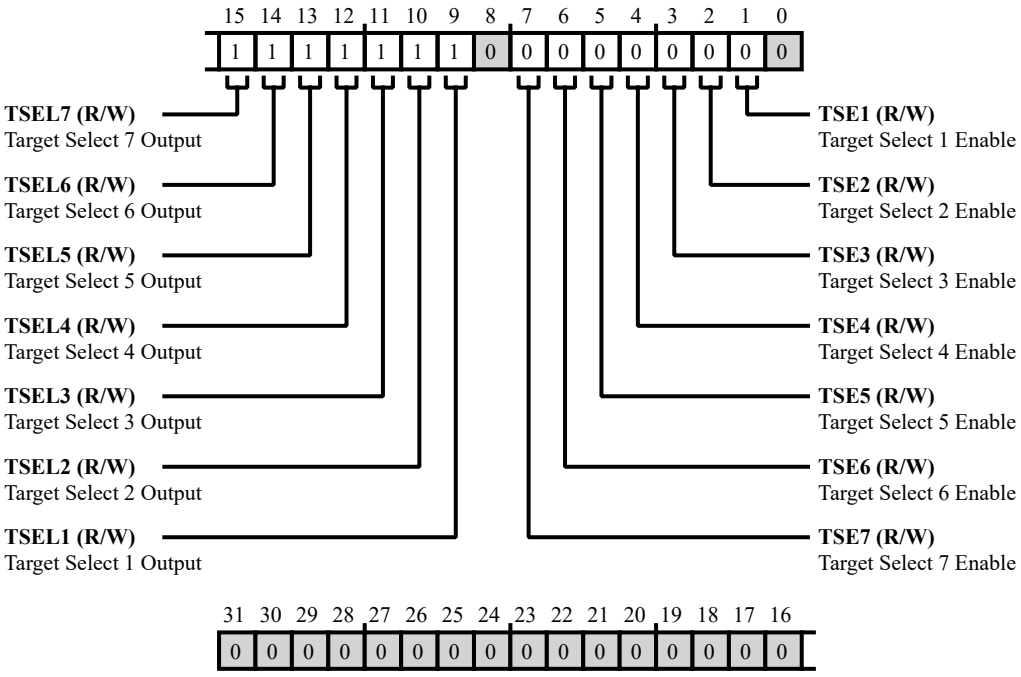


Figure 20-34: SPI_TSEL Register Diagram

Table 20-31: SPI_TSEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	TSEL7	Target Select 7 Output. The SPI_TSEL.TSEL7 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
14 (R/W)	TSEL6	Target Select 6 Output. The SPI_TSEL.TSEL6 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High

Table 20-31: SPI_TSEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TSEL5	Target Select 5 Output. The SPI_TSEL.TSEL5 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
12 (R/W)	TSEL4	Target Select 4 Output. The SPI_TSEL.TSEL4 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
11 (R/W)	TSEL3	Target Select 3 Output. The SPI_TSEL.TSEL3 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
10 (R/W)	TSEL2	Target Select 2 Output. The SPI_TSEL.TSEL2 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
9 (R/W)	TSEL1	Target Select 1 Output. The SPI_TSEL.TSEL1 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
7 (R/W)	TSE7	Target Select 7 Enable. The SPI_TSEL.TSE7 bit enables the related $\overline{\text{SPI_SEL[n]}}$ pin for output. If disabled, the SPI three-states the related $\overline{\text{SPI_SEL[n]}}$ pin.	
		0	Disable
		1	Enable

Table 20-31: SPI_TSEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	TSE6	Target Select 6 Enable. The SPI_TSEL.TSE6 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_TSEL.TSE7 bit description for more information.	
		0	Disable
		1	Enable
5 (R/W)	TSE5	Target Select 5 Enable. The SPI_TSEL.TSE5 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_TSEL.TSE7 bit description for more information.	
		0	Disable
		1	Enable
4 (R/W)	TSE4	Target Select 4 Enable. The SPI_TSEL.TSE4 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_TSEL.TSE7 bit description for more information.	
		0	Disable
		1	Enable
3 (R/W)	TSE3	Target Select 3 Enable. The SPI_TSEL.TSE3 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_TSEL.TSE7 bit description for more information.	
		0	Disable
		1	Enable
2 (R/W)	TSE2	Target Select 2 Enable. The SPI_TSEL.TSE2 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_TSEL.TSE7 bit description for more information.	
		0	Disable
		1	Enable
1 (R/W)	TSE1	Target Select 1 Enable. The SPI_TSEL.TSE1 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_TSEL.TSE7 bit description for more information.	
		0	Disable
		1	Enable

Status Register

The SPI_STAT register indicates SPI status including FIFO status, error conditions, and interrupt conditions. When an interrupt condition from this register is unmasked (enabled) by the corresponding bit in the SPI_IMSK register, the interrupt request is latched into the corresponding bit in the SPI_ILAT register.

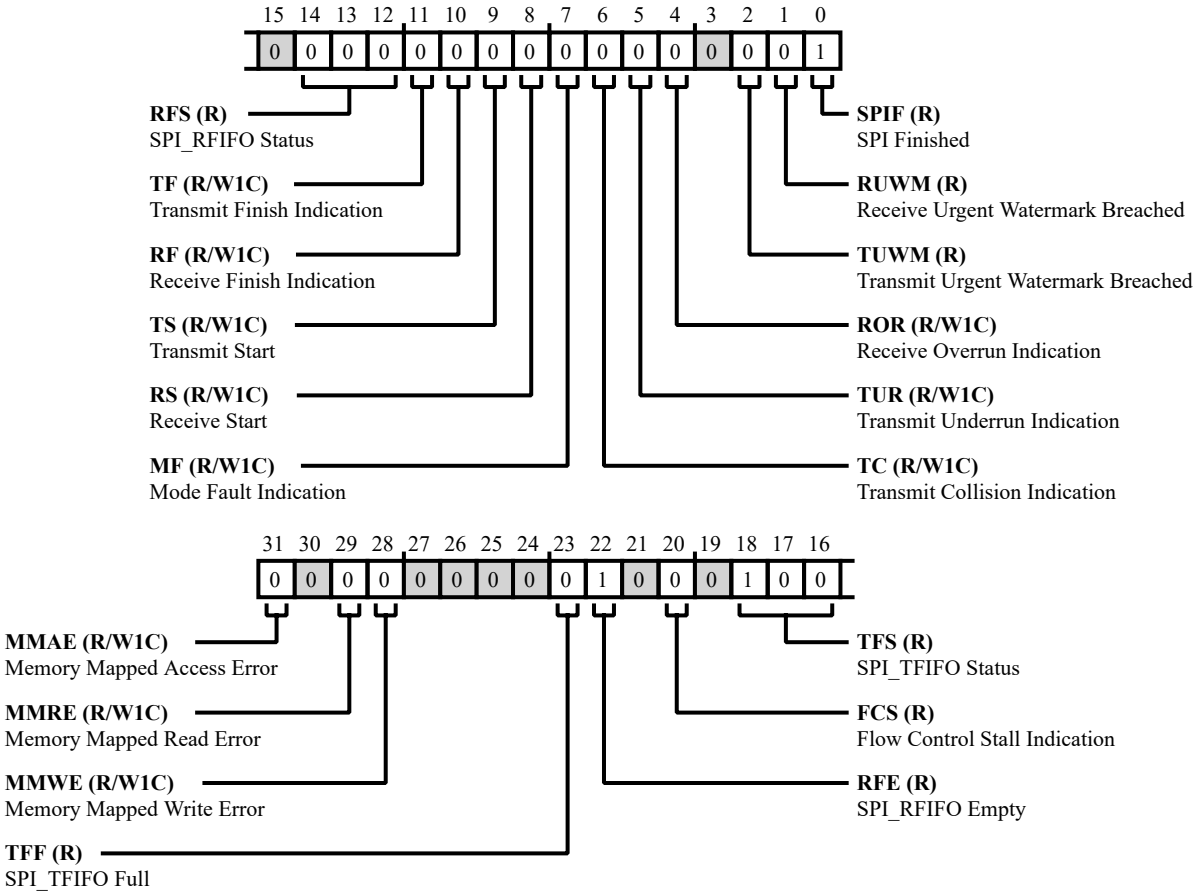


Figure 20-35: SPI_STAT Register Diagram

Table 20-32: SPI_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	MMAE	Memory Mapped Access Error. The SPI_STAT.MMAE bit =1 if an attempt is made to access either the Tx or Rx FIFO while memory-mapped access of SPI memory is enabled (see the SPI_CTL.MMSE bit). The SPI_STAT.MMAE bit =0 when a 1 is written to it. The SPI_STAT.MMAE bit is provided for software notification only. Its state has no further effect.

Table 20-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W1C)	MMRE	Memory Mapped Read Error. The SPI_STAT.MMRE bit =1 if an attempt is made to read address space reserved for memory-mapped SPI memory while memory mapping is disabled (see the SPI_CTL.MMSE bit). The SPI_STAT.MMRE bit =0 when a 1 is written to it. This bit is provided for software notification only. Its state has no further effect.	
28 (R/W1C)	MMWE	Memory Mapped Write Error. The SPI_STAT.MMWE bit =1 if an attempt is made to write address space reserved for memory-mapped SPI memory. The SPI_STAT.MMWE bit =0 when a 1 is written to it. This bit is provided for software notification only. Its state has no further effect.	
23 (R/NW)	TFF	SPI_TFIFO Full. The SPI_STAT.TFF bit indicates whether the SPI_TFIFO is full or not full.	
		0	Not full Tx FIFO
		1	Full Tx FIFO
22 (R/NW)	RFE	SPI_RFIFO Empty. The SPI_STAT.RFE bit indicates whether the SPI_RFIFO is empty or not empty.	
		0	Rx FIFO not empty
		1	Rx FIFO empty
20 (R/NW)	FCS	Flow Control Stall Indication. The SPI_STAT.FCS bit indicates whether a target has deasserted the SPI_RDY pin to stall the SPI controller while the target is unable to service the SPI controller request. This bit is valid only when the SPI is a controller (SPI_CTL.MODE =1) and flow control is enabled (SPI_CTL.FCEN =1).	
		0	No Stall (RDY pin asserted)
		1	Stall (RDY pin deasserted)
18:16 (R/NW)	TFS	SPI_TFIFO Status. The SPI_STAT.TFS bits indicate the status of the SPI_TFIFO. The SPI uses this status when evaluating transmit watermark conditions.	
		0	Full TFIFO
		1	25% empty TFIFO
		2	50% empty TFIFO
		3	75% empty TFIFO
		4	Empty TFIFO

Table 20-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/NW)	RFS	SPI_RFIFO Status. The SPI_STAT.RFS bits indicate the status of the SPI_RFIFO. The SPI uses this status when evaluating receive watermark conditions.	
		0	Empty RFIFO
		1	25% full RFIFO
		2	50% full RFIFO
		3	75% full RFIFO
		4	Full RFIFO
11 (R/W1C)	TF	Transmit Finish Indication. The SPI_STAT.TF bit indicates that the SPI has detected the finish of a transmit burst transfer (the SPI_TWC count decrements to zero). This condition can only occur when SPI_TXCTL.TTI and SPI_TXCTL.TWCEN are enabled.	
		0	No status
		1	Transmit finish detected
10 (R/W1C)	RF	Receive Finish Indication. The SPI_STAT.RF bit indicates that the SPI has detected the finish of a receive burst transfer (the SPI_RWC count decrements to zero). This condition can only occur when SPI_RXCTL.RTI and SPI_RXCTL.RWCEN are enabled.	
		0	No status
		1	Receive finish detected
9 (R/W1C)	TS	Transmit Start. The SPI_STAT.TS bit indicates that the SPI has detected the start of a transmit burst transfer. A transmit bursts starts with the load of SPI_TWC from the SPI_TWCR. This condition can only occur when SPI_TXCTL.TTI and SPI_TXCTL.TWCEN are enabled.	
		0	No status
		1	Transmit start detected
8 (R/W1C)	RS	Receive Start. The SPI_STAT.RS bit indicates that the SPI has detected the start of a receive burst transfer. A receive bursts starts with the load of SPI_RWC from the SPI_RWCR. This condition can only occur when SPI_RXCTL.RTI and SPI_RXCTL.RWCEN are enabled.	
		0	No status
		1	Receive start detected

Table 20-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	MF	Mode Fault Indication. The SPI_STAT.MF bit, when SPI is a controller and SPI_CTL.PTSE is enabled, indicates that multiple controllers have asserted target select inputs.	
		0	No status
		1	Mode fault occurred
6 (R/W1C)	TC	Transmit Collision Indication. The SPI_STAT.TC bit, when SPI is a target, indicates that the load of data into the shift register has occurred too close to the first transmitting edge of the SPI clock.	
		0	No status
		1	Transmit collision occurred
5 (R/W1C)	TUR	Transmit Underrun Indication. The SPI_STAT.TUR bit, when the transmit FIFO (SPI_TFIFO) is empty, indicates that the last word in the transmit FIFO has been re-sent as transmit data. Alternately, it indicates that zero has been sent as transmit data.	
		0	No status
		1	Transmit underrun occurred
4 (R/W1C)	ROR	Receive Overrun Indication. The SPI_STAT.ROR bit, when the receive FIFO (SPI_RFIFO) is full, indicates that a word in the receive FIFO has been overwritten with incoming receive data. Alternately, it indicates that incoming receive data has been discarded.	
		0	No status
		1	Receive overrun occurred
2 (R/NW)	TUWM	Transmit Urgent Watermark Breached. The SPI_STAT.TUWM bit indicates that the transmit urgent watermark (SPI_TXCTL.TUWM) has been reached. This condition is cleared when the transmit FIFO fills enough to reach the transmit regular watermark (SPI_TXCTL.TRWM).	
		0	Tx regular watermark reached
		1	Tx urgent watermark breached
1 (R/NW)	RUWM	Receive Urgent Watermark Breached. The SPI_STAT.RUWM bit indicates that the receive urgent watermark (SPI_RXCTL.RUWM) has been reached. This condition is cleared when the receive FIFO empties enough to reach the receive regular watermark (SPI_RXCTL.RRWM).	
		0	Rx regular watermark reached
		1	Rx urgent watermark breached

Table 20-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SPIF	SPI Finished. The `SPI_STAT.SPIF` bit indicates that a single word transfer is complete.	
		0	No status
		1	Completed single word transfer

Transmit FIFO Data Register

The SPI_TFIFO register has an interface to the transmit shift register in the SPI and has an interface to the processor data buses. The top level of the buffer is visible to programs as the 32-bit SPI_TFIFO register, but the size (number of word locations) of the transmit FIFO is actually flexible with transfer word size. The size of the transmit FIFO is 8 if the word size is 8-bit, or the size is 4 if the word size is 16-bit, or the size is 2 if the word size is 32-bit.

Both controllers and targets may stop or stall transmit transfers based on FIFO status. When the transmit FIFO is empty, the SPI controller stops initiating new transfers on the SPI if SPI_TXCTL.TTI is enabled. A target may stall the SPI interface when the content of the FIFO crosses the selected watermark. If data transmit requests continue after SPI_TFIFO is empty, the data sent from the transmit FIFO is invalid, and the SPI indicates this condition with transmit underrun (SPI_STAT.TUR). This condition is possible when SPI_TXCTL.TTI =0 and SPI_TXCTL.TEN =1 for a controller, or for a target that does not exercise flow control.

Note that the transmit FIFO is reset (cleared) when the SPI is disabled after being enabled.

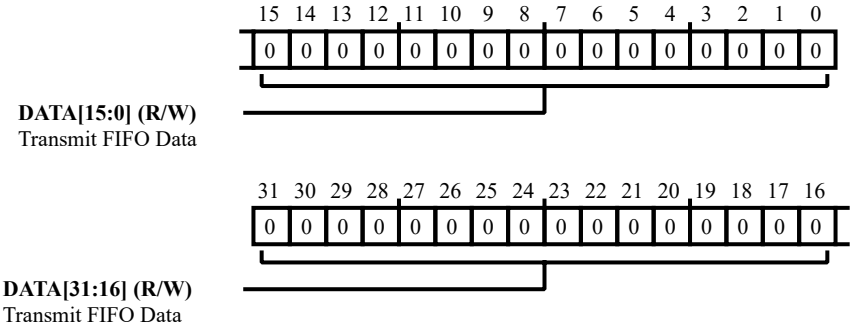


Figure 20-36: SPI_TFIFO Register Diagram

Table 20-33: SPI_TFIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Transmit FIFO Data. The SPI_TFIFO.DATA bit field contains the FIFO transmit data.

Transmitted Word Count Register

The SPI_TWC register holds a count of the number of words remaining to be transmitted by the SPI. To start the decrement of the word count in SPI_TWC, enable the transmit word counter (SPI_TXCTL.TWCEN =1). The SPI uses the word count to control the duration of transfers and to signal the completion of a burst of transfers with the transmit finish interrupt request. In DMA mode, the SPI uses the SPI_TWC to ensure that the number of frames transmitted during a DMA transfer is equal to the number of words programmed in the DMA channel controller. The values programmed into the SPI_TWC registers should match the word count in the DMA configuration. The SPI_TWC maintains the number of frames to be transmitted in a transfer. The SPI_TWC should only be changed when the counter is disabled.

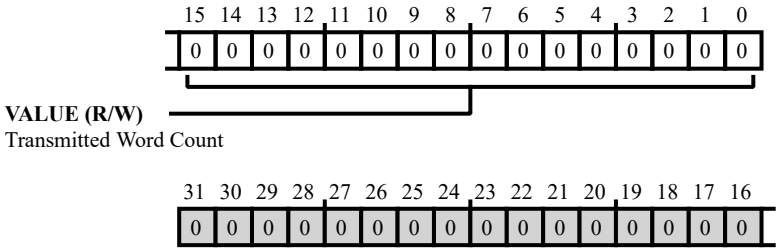


Figure 20-37: SPI_TWC Register Diagram

Table 20-34: SPI_TWC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Transmitted Word Count. The SPI_TWC.VALUE bits hold the transmit transfer word count.

Transmitted Word Count Reload Register

The SPI_TWCR register holds the transmit word count value that the SPI loads into the SPI_TWC register when the transfer count decrements to zero. To prevent the SPI from reloading the counter, use zero for the reload count value. The SPI_TWCR should only be changed when the counter is disabled.

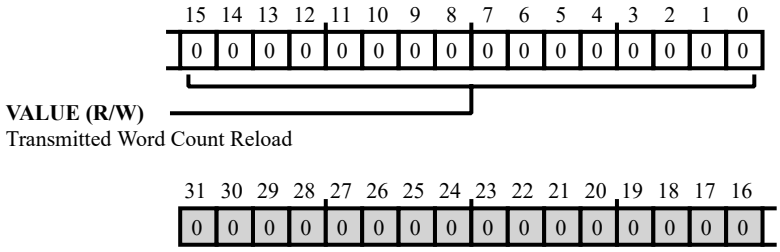


Figure 20-38: SPI_TWCR Register Diagram

Table 20-35: SPI_TWCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Transmitted Word Count Reload. The SPI_TWCR.VALUE bits hold the transmit transfer word count reload value.

Transmit Control Register

The SPI_TXCTL register enables the SPI transmit channel, initiates transmit transfers, and configures SPI_TFIFO buffer watermark settings.

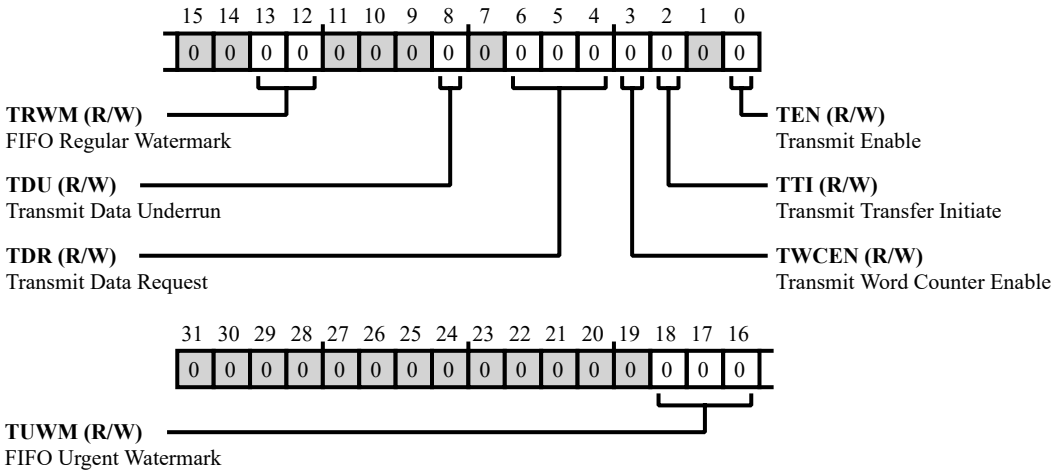


Figure 20-39: SPI_TXCTL Register Diagram

Table 20-36: SPI_TXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	TUWM	FIFO Urgent Watermark. The SPI_TXCTL.TUWM bits select the transmit FIFO (SPI_TFIFO) watermark level for urgent data bus requests. The SPI also uses this watermark level for generation of the SPI_ILAT.TUWM interrupt request. When an urgent SPI_TFIFO watermark is enabled with SPI_TXCTL.TUWM, the SPI_TXCTL.TRWM selection is used as the deassertion condition for any SPI_ILAT.TUWM interrupt requests that are latched.	
		0	Disabled
		1	25% empty TFIFO
		2	50% empty TFIFO
		3	75% empty TFIFO
		4	Empty TFIFO

Table 20-36: SPI_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:12 (R/W)	TRWM	FIFO Regular Watermark. The SPI_TXCTL.TRWM bits select the transmit FIFO (SPI_TFIFO) watermark level for regular data bus requests. When an urgent SPI_TFIFO watermark is enabled with SPI_TXCTL.TUWM, the SPI_TXCTL.TRWM selection is used as the deassertion condition for any SPI_ILAT.TUWM interrupt requests that are latched.	
		0	Full TFIFO
		1	TFIFO less than 25% empty
		2	TFIFO less than 50% empty
		3	TFIFO less than 75% empty
8 (R/W)	TDU	Transmit Data Underrun. The SPI_TXCTL.TDU bit selects handling for transmit data requests when the transmit buffer (SPI_TFIFO) is empty. If enabled and SPI_TFIFO is empty, the SPI transmits zero as data. If disabled and SPI_TFIFO is empty, the SPI transmits the last word in the buffer as data.	
		0	Send last word when SPI_TFIFO is empty
		1	Send zeros when SPI_TFIFO is empty
6:4 (R/W)	TDR	Transmit Data Request. The SPI_TXCTL.TDR bits select transmit FIFO (SPI_TFIFO) watermark conditions that direct the SPI to generate a transmit status interrupt request.	
		0	Disabled
		1	Not full TFIFO
		2	25% empty TFIFO
		3	50% empty TFIFO
		4	75% empty TFIFO
		5	Empty TFIFO
3 (R/W)	TWCEN	Transmit Word Counter Enable. The SPI_TXCTL.TWCEN bit enables the decrement of the transmit word count (SPI_TWC) register when the count is not zero and SPI_TXCTL.TTI is enabled. Enabling SPI_TXCTL.TWCEN prevents transmit underrun errors from occurring. The SPI_TXCTL.TWCEN bit is valid only when the SPI is a controller.	
		0	Disable
		1	Enable

Table 20-36: SPI_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	TTI	Transmit Transfer Initiate. The SPI_TXCTL.TTI bit enables initiation of transmit transfers if the transmit FIFO (SPI_TFIFO) is not empty. The bit also enables this initiation if SPI_TWC is not zero when SPI_TXCTL.TWCEN is enabled. Enabling SPI_TXCTL.TTI prevents transmit underrun errors from occurring. The SPI_TXCTL.TTI bit is valid only when the SPI is a controller.	
		0	Disable
		1	Enable
0 (R/W)	TEN	Transmit Enable. The SPI_TXCTL.TEN bit enables SPI transmit channel operation.	
		0	Disable
		1	Enable

21 Extended Serial Peripheral Interface (xSPI)

The xSPI flash controller provides access to serial flash devices that support JESD216 and JESD251 standards. The controller accesses flash devices in modes that permit efficient communication with minimal software overhead. The supported HyperBus™ protocol enables seamless communication with HyperFlash™/HyperRAM™devices. The standard SPI protocol is supported along with high performance dual, quad, and octal SPI modes, where data can be transferred on up to 8 data pins. In addition to the normal mode of operation in single transfer (STR) mode, the xSPI controller supports a dual transfer rate (DTR) mode. In DTR mode, data, address, and commands are transferred on both edges of the serial clock. With direct access mode support, any read/write to the xSPI memory-mapped space triggers a read/write to flash memory.

An integrated PHY handles the low-level timings of the data, address and control signals between the device and controller.

The xSPI controller also provides support for RWDS/DQS when sampling data. This feature improves read data capture, which, along with the PHY, enables high-speed data transfers.

xSPI Features

The xSPI controller supports the following features:

- Memory-mapped direct mode operation for performing flash data transfers
- Auto Command Mode (ACMD) to perform various memory operations with minimum software overhead
- An ACMD DMA engine that triggers transactions using descriptors fetched from the main memory. It automatically moves data between the device memory and system host memory.
- STIG mode operation to issue flash commands
- Automatic flash status polling support for flash program
- Execute in Place (XIP) support
- Device Discovery (DD) to identify the memory device
- Support for single, dual, quad, or octal modes of operation
- Support for DDR mode and DTR protocol (including octal DDR protocol)

- Data integrity features (CRC or ECC) for flash memories
- Programmable baud rate, clock phase, and polarity
- Programmable interrupt generation
- Programmable chip select control timing
- Programmable dummy cycles for read and write operations
- Support for up to 80 MHz in non-DQS mode
- Support for up to 125 MHz in DQS mode

Supported Memory Devices

The xSPI controller supports any device that is compatible with the JESD216 specification standard. In this standard, these devices are defined as profile 1 or profile 2 type devices. Traditional serial NOR flash devices tend to be compatible with the profile 1 type as defined in that standard. Devices supporting HyperBus (traditional HyperFlash and most HyperRAM or pSRAM devices) tend to be compatible with the profile 2 type. Sometimes, a pSRAM or HyperRAM device may support a profile 1 style interface; the device manufacturer selects the interface.

The supported devices include:

- *Traditional serial SPI (single/dual/quad/octal)* - Memory devices defined as profile 1 in the JEDEC JESD216 specification. The device may have been manufactured before the specification was released. Any device that specifies compliance with an xSPI profile 1, where profile 1 is described in the xSPI JEDEC JESD216 specification.
- *Traditional HyperFlash* - Memory devices defined as profile 2 in the JEDEC JESD216 specification. The device may have been manufactured before the specification was released. Any devices that specify specifies compliance with an xSPI profile 2, where profile 2 is described in the xSPI JEDEC JESD216 specification.
- *Serial RAM and HyperRAM* - Serial SRAM devices (for example, OctaRAM™/OctalRAM, HyperRAM, pSRAM) devices. These volatile RAM devices tend to be interfaced similar to profile 1 or profile 2. However, they must be occasionally self-refreshed to maintain state and may not require some elements of the sequences used to read or program traditional flash devices.
- *Serial NAND* - also referred to as SPI NAND

xSPI Functional Description

This section provides information on the function of the xSPI module.

ADSP-2184x XSPI Register List

The XSPI is an expanded SPI controller. It contains the following registers.

Table 21-1: ADSP-2184x XSPI Register List

Name	Description
XSPI_BOOT_STAT	Boot Status Register
XSPI_CMD0	Command Register 0
XSPI_CMD1	Command Register 1
XSPI_CMD2	Command Register 2
XSPI_CMD3	Command Register 3
XSPI_CMD4	Command Register 4
XSPI_CMD5	Command Register 5
XSPI_CMD_STAT	Command Status Register
XSPI_CMD_STAT_PTR	Command Status Pointer Register
XSPI_DAC_CFG	DAC Configuration Register
XSPI_DAC_REMAPADDR0	Address Remapping Register 0
XSPI_DAC_REMAPADDR1	Address Remapping Register 1
XSPI_DISCOVERY_GCTL	Device Discovery Control Register
XSPI_DMA_CTL	DMA Interface Control Register
XSPI_DMA_ERRADDR_HI	DMA Error Address High Register
XSPI_DMA_ERRADDR_LO	DMA Error Address Low Register
XSPI_ERS_SEQ_CFG0	Erase Sequence Configuration Register 0
XSPI_ERS_SEQ_CFG1	Erase Sequence Configuration Register 1
XSPI_ERS_SEQ_CFG2	Erase Sequence Configuration Register 2
XSPI_FEATURES	Controller Features Register
XSPI_GSTAT	General Controller Status Register
XSPI_INT_EN	Interrupt Enable Register
XSPI_ISTAT	Interrupt Status Register
XSPI_LONGPOL_GCTL	Long Polling Count Register
XSPI_MINICTL_DEV_ACTIVE_MAX	Device Active Maximum Clock Cycle Register
XSPI_MINICTL_CLKMODE	Clock Mode Control Register
XSPI_MINICTL_DEV_DLY	Device Delay Register
XSPI_MINICTL_HF_OFFSET	HyperFlash Offset Register
XSPI_MINICTL_JEDEC_RST_TR	JEDEC Reset Delay Register
XSPI_MINICTL_RST_PIN_CTL	Hardware Reset Control Register
XSPI_MINICTL_RST_RECOV	Reset Recovery Delay Register

Table 21-1: ADSP-2184x XSPI Register List (Continued)

Name	Description
XSPI_MINICTL_WP_CTL	Write Protect Register
XSPI_PHY_DLLOB0	PHY DLL Observable Points 0 Register
XSPI_PHY_DLLOB1	PHY DLL Observable Points 1 Register
XSPI_PHY_DLL_CTL	PHY DLL Control Register
XSPI_PHY_DLL_REQ_CTL	Bus Requester DLL Control Register
XSPI_PHY_DLL_COMP_CTL	Bus Completer DLL Control Register
XSPI_PHY_DLL_UPDT_CNT	PHY DLL Resynchronization Register
XSPI_PHY_DQS_TR	PHY DQS Timing Register
XSPI_PHY_DQ_TR	PHY DQ Timing Register
XSPI_PHY_FEATURES	PHY Features Register
XSPI_PHY_GATE_LPBK_CTL	PHY Gate Loopback Control Register
XSPI_PHY_GCTL	PHY Global Control Register
XSPI_PHY_GPIO_CTL0	PHY GPIO Control Register 0
XSPI_PHY_GPIO_CTL1	PHY GPIO Control Register 1
XSPI_PHY_GPIO_STAT0	PHY GPIO Status Register 0
XSPI_PHY_GPIO_STAT1	PHY GPIO Status Register 1
XSPI_PHY_GTSEL	PHY Global Termination Control Register
XSPI_PHY_IE_TR	PHY DQS Input Enable Timing Register
XSPI_PHY_OB0	PHY Observable Points Register
XSPI_PHY_REVID	PHY Revision ID Register
XSPI_PHY_STATIC_TGL	PHY Static Aging Register
XSPI_PHY_WR_DESKEW_PAD_CTL0	PHY Deskew Write Register
XSPI_PROG_SEQ_CFG0	Program Sequence Configuration Register 0
XSPI_PROG_SEQ_CFG1	Program Sequence Configuration Register 1
XSPI_PROG_SEQ_CFG2	Program Sequence Configuration Register 2
XSPI_READ_SEQ_CFG0	Read Sequence Configuration Register 0
XSPI_READ_SEQ_CFG1	Read Sequence Configuration Register 1
XSPI_READ_SEQ_CFG2	Read Sequence Configuration Register 2
XSPI_REVID	Revision ID Register
XSPI_RST_SEQ_CFG0	Reset Sequence Configuration Register 0
XSPI_RST_SEQ_CFG1	Reset Sequence Configuration Register 1

Table 21-1: ADSP-2184x XSPI Register List (Continued)

Name	Description
XSPI_SDMA_ADDR0	Host DMA Buffer Address Register 0
XSPI_SDMA_ADDR1	Host DMA Buffer Address Register 1
XSPI_SDMA_SIZ	Host DMA Block Size Register
XSPI_SDMA_TRD_STAT	Host DMA Thread Status Register
XSPI_SEQ_GCTL0	Sequence Configuration Register 0
XSPI_SEQ_GCTL1	Sequence Configuration Register 1
XSPI_SHORTPOL_GCTL	Short Polling Count Register
XSPI_STAT_SEQ_CFG0	Status Checking Sequence Configuration Register 0
XSPI_STAT_SEQ_CFG1	Status Checking Sequence Configuration Register 1
XSPI_STAT_SEQ_CFG10	Status Checking Sequence Configuration Register 10
XSPI_STAT_SEQ_CFG2	Status Checking Sequence Configuration Register 2
XSPI_STAT_SEQ_CFG3	Status Checking Sequence Configuration Register 3
XSPI_STAT_SEQ_CFG4	Status Checking Sequence Configuration Register 4
XSPI_STAT_SEQ_CFG5	Status Checking Sequence Configuration Register 5
XSPI_STAT_SEQ_CFG7	Status Checking Sequence Configuration Register 7
XSPI_STAT_SEQ_CFG8	Status Checking Sequence Configuration Register 8
XSPI_STAT_SEQ_CFG9	Status Checking Sequence Configuration Register 9
XSPI_TRD_COMP_ISTAT	Auto Command Engine Interrupt Status Thread Register
XSPI_TRD_ERR_INT_EN	Thread Error Interrupt Enable Register
XSPI_TRD_ERR_ISTAT	Thread Error Interrupt Status Register
XSPI_TRD_STAT	Auto Command Engine Thread Status Register
XSPI_WE_SEQ_CFG0	WEL Sequence Configuration Register
XSPI_WORKMODE_CTL	Device Control Register
XSPI_XIP_GCTL	XIP Configuration Register

ADSP-2184x xSPI Interrupt List

Table 21-2: ADSP-2184x SPI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
386	XSPI0_IRQ	XSPI0 Interrupt Request	Level	
387	XSPI1_IRQ	XSPI1 Interrupt Request	Level	

xSPI Block Diagram

The xSPI module is comprised of:

- Main command sequencer
- PHY block
- Internal transmit/receive FIFOs
- DAC/STIG controller
- Register interface

The ***xSPI Controller Block Diagram*** shows the xSPI controller functional blocks.

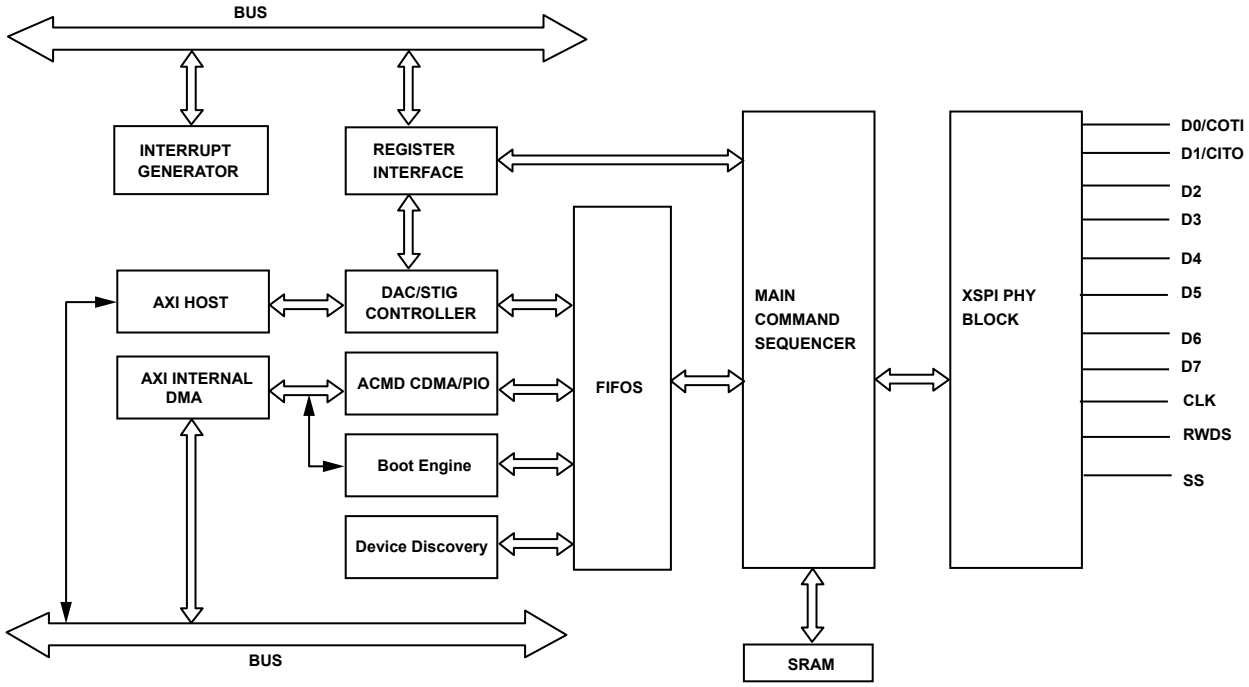


Figure 21-1: xSPI Controller Block Diagram

Architectural Concepts

There are three clock sources for xSPI controller : XSPI_CLK, SCLK0_0, SYSCLK_0. SYSCLK_0 is used to transfer data over the bus between the controller (core/DMA) and the xSPI controller. XSPI_CLK is driven on the external XSPI_CLK pin to transfer the data to/from external devices. And SCLK0_0 is used to access the xSPI controller registers to perform basic configuration of the controller and interrupt handling.

NOTE: For XSPI0, XSPI_CLK is sourced from CLKO10 (output clock of CDU) and for XSPI1, XSPI_CLK is sourced from CLKO8 (output clock of CDU). A user can select one of the options for CLKO8 and CLKO10 to source xSPI controllers. Rrefer to the CDU chapter for the clocking diagram and more details.

The xSPI controller supports independent modes/controllers for different memory devices:

- Direct mode
- STIG mode
- Auto Command Mode (ACMD, CDMA or PIO)

The bus interface provides access to controller status and configuration registers. The AXI host interface is used in direct mode, STIG mode (when necessary), and optionally in ACMD mode. The AXI internal DMA interface is only used in ACMD mode, where the internal DMA module of the xSPI controller owns and manages system transfers, and handles DMA descriptor fetches and writes and associated data buffer transfers. The DMA module used in ACMD mode implements the bus requester capability. It can automatically transfer data from/to system memory and from/to the connected xSPI device (using internal SRAM).

The boot engine and device discovery blocks in the xSPI controller use the internal DMA interface and FIFOs.

The main command sequencer holds the main sequencing logic. It is responsible for building the command structures and meeting the required device timings. It also interfaces with the PHY module and regulates data flow between the physical layer and the system interface. PHY is responsible for handling the low-level timings of the data, address, and control signals between the device and controller.

Operating Modes

There are three main usage modes for the xSPI controller: direct mode, STIG mode, and ACMD mode. These modes can perform standard read/write transactions to a device memory. However, only one mode can be active at a time.

One active work mode is selected at a time to communicate with an attached device. ACMD or direct modes are the primary operational modes. STIG mode is used as an accompanying mode to send commands or a sequences of commands to the device that may not be directly supported in ACMD or direct mode. STIG mode is a highly flexible mode that can also be used to write or read data to the device and perform other operations. However, STIG mode relies more on software to construct the necessary command structures.

ACMD mode requires that software either manage a descriptor table with high-level sequences defined within, or manage some command registers within the controller to issue these same high-level sequences. The controller can automatically translate high-level sequences (reset, erase, erase_all, program, read) into subsequences that are tied to the device type.

Direct mode, as the name suggests, directly translates read and write operations detected on the AXI host bus completer interface to equivalent low-level read and write subsequences at the device. As such, direct mode can only support read or program operations. It relies on STIG mode to perform other operations such as erase. It can also be used to directly boot from the device, when required.

Direct Mode

In direct mode, reads and writes that are issued by the processor (either by the core or MDMA controller) enter the controller using the AXI bus completer interface. The read/write operations are directly translated to read/write at the memory device. The shaded portion of the ***Direct Mode Block Diagram*** shows the flow. Direct mode allows the

processor to execute code directly, or have direct access from/to the attached memory device. This flow is sometimes referred to as eXecute-In-Place or XIP. The controller support devices operating in XIP mode (in all work modes, not just direct). However, there are steps the host must take to trigger entry to and exit from XIP mode.

See the topic Operating in Direct Mode for details on operating in direct mode, programming the mode, and usage restrictions.

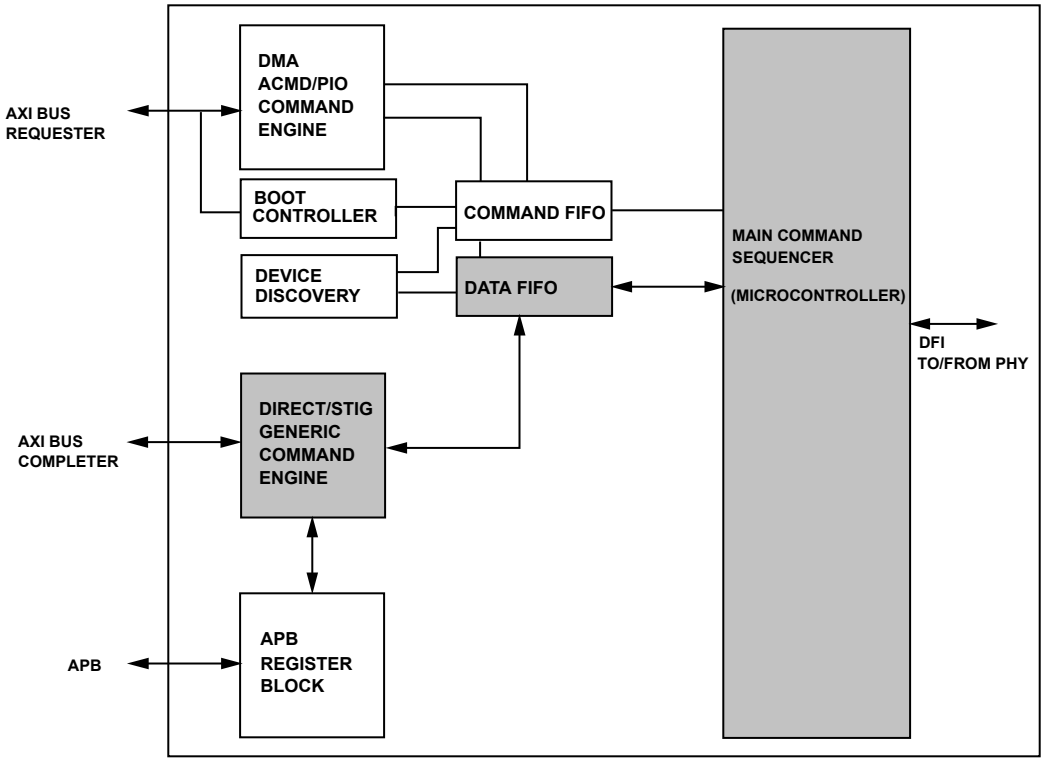


Figure 21-2: Direct Mode Block Diagram

STIG Mode

Software Triggered Instruction Generator (STIG) mode is configured and triggered using the programming interface. This mode is very flexible and configurable. In addition to performing standard reads and writes, it can also be used to access memory device registers and/or trigger functions in the memory device such as erase, read JEDEC ID, etc. STIG mode uses data paths that are similar to direct mode. However, the commands are configured and triggered through access to registers using the APB interface. Data can optionally be passed from the AXI bus completer interface, or through other registers in the APB register block. The shaded portion of the ***STIG Mode Block Diagram*** shows the operational flow.

See the topic Operating in STIG Mode for details on operating in STIG mode, programming the mode, and usage restrictions.

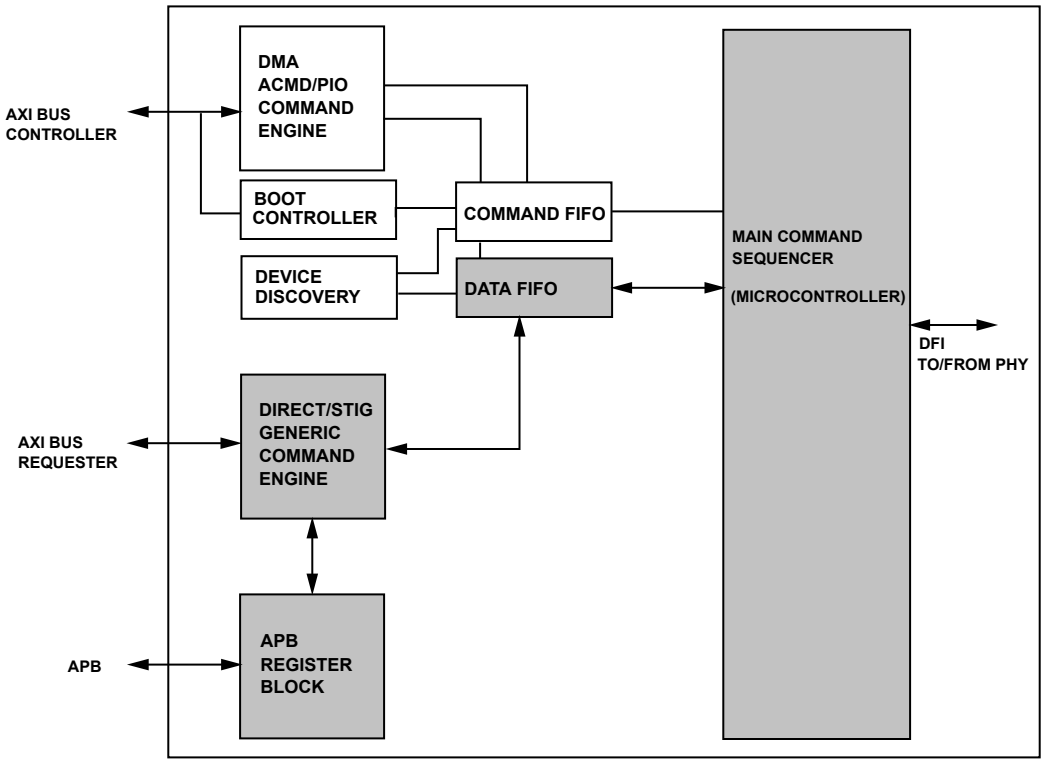


Figure 21-3: STIG Mode Block Diagram

ACMD Mode

Auto Command Mode (ACMD) mode, the user configures an advanced DMA engine capable of operating multiple parallel threads. The configuration uses controller registers and/or descriptors held in the main memory. The shaded portion of the ***ACMD Mode Block Diagram*** shows the operational flow. The ACMD DMA engine triggers transactions using the descriptors fetch from main memory to automatically move data between the device memory and system memory. ACMD mode is configured to use the AXI internal DMA bus requester interface to transfer data and to collect and process DMA descriptors.

ACMD mode can also be configured to use the AXI host interface instead. In this case, the controller signals to the processor using an interrupt or status register when it requires the host to send write data on the bus completer AXI write ports, or when it needs the processor to issue an AXI read to fetch read data.

See the topic Operating in ACMD Mode for details on operating in ACMD mode, programming the mode, and usage restrictions.

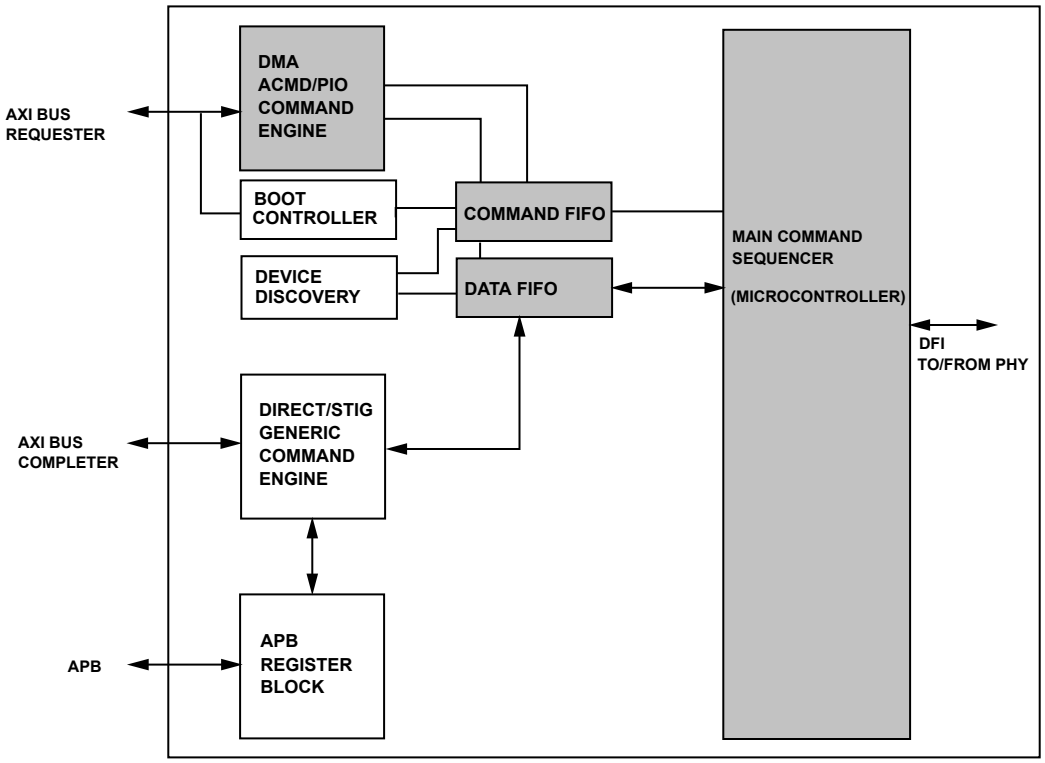


Figure 21-4: ACMD Block Diagram

Integrated Soft PHY

The xSPI PHY controller is responsible for handling the low-level timings of the data, address, and control signals between the device and controller. It provides additional logic, which operates as a bridge between formulated xSPI transactions and the timing-aware off-chip xSPI transfer to the external device. PHY has its own programmable registers. Accessing these registers occurs through the standard controller APB address space. Before the processor can communicate with the memory device, the PHY must be initialized and locked.

NOTE: In ADSP-2156x processors, the internal DMA of the xSPI controller is not supported.

See the section xSPI PHY Controller for details on how to use PHY, programming the PHY, and usage restrictions.

XIP Mode

The eXecute-In-Place (XIP) mode is also referred to as continuous mode. When the controller is in XIP mode, the processor executes code directly from the attached memory device or from where the processor has direct access to the attached memory device. The controller supports this function using the direct work mode. If executing code directly is needed, enable direct work mode.

The device uses XIP mode to improve read performance by removing the need for the controller to send the read opcode for any read operation. If the read opcode is typically sent over eight device clock cycles (SDR, serial transmission of the opcode), the read operation latency is reduced by eight cycles. This difference can significantly improve performance when executing code directly from the device. On the downside, when XIP mode is entered, the device can only perform read operations. If a different operation is required, XIP mode must be exited. Although

the controller supports XIP mode, there are steps the host must take to trigger entry to and exit from XIP mode. These steps depend on the selected work mode. See the section Entering and Exiting XIP Mode.

Configuring the Controller

To use the controller in direct, ACMD, or STIG mode, there are global configuration registers that must be configured. These are defined in the common sequence registers.

Use the following steps to configure the controller.

1. Configure the XSPI_SEQ_GCTL0 register to select the correct xSPI profile as per the memory device, to enable or disable the data swap feature, to select single or dual bytes per address, device page program size, device page read size, and other desired features.
2. Configure the XSPI_SEQ_GCTL1 register. This register is used only for SPI NAND profile to configure plane count, pages per block, extended page size, etc.
3. Configure the erase/read/program sequence. Configure the read or program sequence that the controller must send to the memory device. This sequence depends on the attached memory device. Refer to the device data sheet for details.

Some devices require a write enable sub sequence to be issued prior to a program operation (for example, to configure the write enable latch or WEL in the device). By default, it is assumed to be memory device requirement. However, the sequence can be disabled using the XSPI_WE_SEQ_CFG0.P1_EN bit. By default, the command extension of the write enable sequence is disabled. It is sent in a serial SDR format with an opcode of 0x6. If any of the parameters need changes, configure the XSPI_WE_SEQ_CFG0 register accordingly.

Some devices require the device to be polled for correctness and readiness after issuing an operation. Due to the differences between how device types interpret and respond to status requests, and different types of status requests, the controller requires several programmable registers that must be tuned to the device requirements. For direct mode, the controller can optionally check the device fail or busy status following a program or erase operation (or additionally for soft_reset/read for SPI NAND devices). For SPI NAND devices, the controller can also check the ECC status following a read page command. All of these requirements are configured using the XSPI_STAT_SEQ_CFGn registers (XSPI_STAT_SEQ_CFG0 to XSPI_STAT_SEQ_CFG10).

NOTE: When using the device discovery engine, the registers are updated automatically after power on reset. Refer to section Device Discovery for details.

The ***Global Configurations Registers*** table shows the registers used to configure the controller for various operations and types of flash devices. These registers must be initialized and configured when operating in direct or ACMD mode.

Table 21-3: Global Configurations Registers

	Profile 1 Devices	Profile 2 – HyperFlash Devices	Profile 2 – HyperRAM Devices	SPI NAND Devices
Program	XSPI_PROG_SEQ_CFG0 XSPI_PROG_SEQ_CFG1	XSPI_PROG_SEQ_CFG2	XSPI_PROG_SEQ_CFG2	XSPI_PROG_SEQ_CFG0 XSPI_PROG_SEQ_CFG1
Read	XSPI_READ_SEQ_CFG0 XSPI_READ_SEQ_CFG1	XSPI_READ_SEQ_CFG2	XSPI_READ_SEQ_CFG2	XSPI_READ_SEQ_CFG0 XSPI_READ_SEQ_CFG1
Write	XSPI_WE_SEQ_CFG0	N/A	N/A	XSPI_WE_SEQ_CFG0
Device Fail Status Check Device Busy Status Check	XSPI_STAT_SEQ_CFG0 XSPI_STAT_SEQ_CFG1 XSPI_STAT_SEQ_CFG2 XSPI_STAT_SEQ_CFG3 XSPI_STAT_SEQ_CFG5 XSPI_STAT_SEQ_CFG7 XSPI_STAT_SEQ_CFG8 XSPI_STAT_SEQ_CFG9	XSPI_STAT_SEQ_CFG4 XSPI_STAT_SEQ_CFG5 XSPI_STAT_SEQ_CFG7 XSPI_STAT_SEQ_CFG8 XSPI_STAT_SEQ_CFG9	N/A	XSPI_STAT_SEQ_CFG0 XSPI_STAT_SEQ_CFG1 XSPI_STAT_SEQ_CFG2 XSPI_STAT_SEQ_CFG3 XSPI_STAT_SEQ_CFG5 XSPI_STAT_SEQ_CFG7 XSPI_STAT_SEQ_CFG8 XSPI_STAT_SEQ_CFG9
ECC Status Check	N/A	N/A	N/A	XSPI_STAT_SEQ_CFG10

Operating in Direct Mode

In direct mode, reads and writes that are issued by the processor enter the controller through the AXI bus completer interface. The read/write operations are directly translated to reads and writes at the memory device. Along with standard code execution, this mode can also be used immediately after power on reset to boot data from the attached memory device.

NOTE: This mode is different from the dedicated boot feature. The boot engine is specifically used to transfer boot code from the memory device to an area of host memory following power on reset (from which the host can then boot). There is a dedicated xSPI boot mode for the processor to support this feature. See the xSPI Controller Boot Mode topic in the Booting chapter for details.

Direct work mode is enabled by default (XSPI_WORKMODE_CTL.WORKMODE). The XSPI_WORKMODE_CTL register can only be changed when the controller is in an idle state. This state can be determined by polling the CRTL_BUSY pin of the xSPI flash controller or by polling the XSPI_GSTAT.CTL_BUSY bit. When direct mode is enabled, access to the XSPI_CMDn registers (which are only required for other work modes) are ignored and should be avoided.

Read and program/write transactions to the memory device occur only as a direct consequence of the processor applying read or write AXI transactions to the AXI bus completer interface through memory-mapped xSPI addresses.

NOTE: In direct mode, the configured page size for write or read transfers must be configured to a value wider than the selected word size of the AXI transaction (sA*SIZE). Also, the xSPI page boundaries must be aligned to the AXI transaction word size (sA*SIZE).

In addition to configuring the global registers described in Configuring the Controller, configure the direct mode specific registers such as XSPI_DAC_CFG and XSPI_DAC_REMAPADDR0/XSPI_DAC_REMAPADDR1 :

1. Write to the XSPI_DAC_CFG register to select the appropriate bank number in the attached memory device, to enable or disable AXI bus completer address remap, XIP mode, and other desired features.
2. Configure the remap address offset in the XSPI_DAC_REMAPADDR0 register.

Write Limitations

By default, the xSPI controller does not support the following functions:

- Byte writes (for example, an AXI write with an AWSIZE value of 0) to a device that is 16-bit addressed, or to an octal DDR configured device
- Issuing a write burst to an odd address to a device that is 16-bit addressed, or to an octal DDR configured device
- Using AXI write strobes to mask bytes on the write bus

These write operations can optionally be supported by setting the bit XSPI_DAC_CFG.RWDS_CAP_EN bit if the connected device supports byte masking using the DQS signal. DQS is traditionally used by a memory device to transfer a strobe with read data to help read data sampling. However, some devices also use the same signal as a byte mask for writes (in which case the limitations can be avoided).

AXI Wrapping Bursts

The following wrapping bursts restrictions are taken from the AXI standard:

- The start address must be aligned to the size of each transfer
- The length of the burst must be 2, 4, 8, or 16 transfers

Wrapping bursts are used for cache line accesses issued after a cache miss, and are, therefore, more relevant to the direct mode of operation than ACMD/STIG work modes. Internally, the xSPI controller handles AXI wrapping bursts by splitting them into separate incrementing bursts, automatically. Wrapping bursts on the AXI bus completer interface are supported even when the attached memory device does not support wrapping.

There is no restriction on using wrapping read bursts. However, there is one known scenario where wrapping write bursts received by the controller operating in direct mode through the AXI bus completer interface are not be supported. This scenario is described below:

1. The wrapping write burst restriction applies only to attached memory devices that require time to execute any program (write) operation. Flash devices, typically, operate in this manner.
2. While any program (write) operation is ongoing within the memory device, the controller has a function (which, by default, is enabled) to automatically check the status of the memory device. It does not allow any other read or write operation from the controller to proceed until the memory device has indicated that it is ready for a new operation. This function is programmable and can be disabled.
3. If the user has chosen to disable the function above (through the XSPI_STAT_SEQ_CFG1 register), the splitting of the AXI wrapping write transaction into separate write incrementing transactions results in the second incrementing write being discarded by the memory device.

Therefore, wrapping write bursts *cannot* be supported when the attached memory device requires time to process write operations and the controller function to check ready/busy status of the device has been disabled (using the XSPI_STAT_SEQ_CFG1 register). Otherwise, wrapping write bursts in direct mode are fully supported.

Entering and Exiting XIP Mode

The XSPI_DAC_CFG.XIP_EN_MB_VAL and XSPI_DAC_CFG.XIP_DIS_MB_VAL bit fields control how and when XIP mode in the device is entered or exited when using direct mode. By default, XIP mode is disabled. The ***XIP Mode Register Configurations*** table shows the registers used to configure XIP mode operations.

Most memory device use either of the one or both of the following steps to enter the XIP mode.

- Configure a specific bit in the memory configuration/status register (this step is not mandatory for all flash devices, but it which can be achieved easily using STIG mode).
- Send a read command with the correct value applied in the mode byte; the info is located immediately after the address phase of the read transaction

Set the XSPI_DAC_CFG.XIP_EN_MB_VAL bit to enter XIP mode while operating in direct mode. Follow up with a normal direct mode read transaction. As the read transaction is executed and the read sequence is sent to the device, the xSPI controller sends the value encoded in the XSPI_XIP_GCTL.XIP_EN_MB_VALUE register bit, immediately following the address phase of the transaction. Once the read transaction completes, the XSPI_XIP_GCTL.XIP_EN bit is set (=1) to indicate that both the controller and the memory device are now configured to work in XIP mode.

IMPORTANT: A direct read transfer to enable or disable XIP mode in the controller and memory device must not have more than 64B of data and must not cause crossing between pages.

To enable XIP mode during power on reset (assuming that the memory device is already configured to work in XIP mode in a non-volatile manner from reset), the XSPI_XIP_GCTL.XIP_EN bit should be set (=1). Then, each read transaction is performed in XIP mode. For example, the controller issues the read operation without sending any opcode and additionally setting the enable XIP mode bit following the address phase (when

XSPI_DAC_CFG.XIP_EN_MB_VAL = 1). The value of the XSPI_DAC_CFG.XIP_EN_MB_VAL field is sent following the address phase of the transaction.

To exit XIP mode in both the xSPI controller and the memory device, a new direct read transaction must be issued to the device with the disable XIP mode bit following address phase. Set (=1) the XSPI_DAC_CFG.XIP_DIS_MB_VAL bit to disable XIP mode. Issue a new direct read. As the read is executed and the read sequence is sent to the device, the controller sends the value encoded in the XSPI_XIP_GCTL.XIP_DIS_MB_VALUE field immediately following the address phase of the transaction. Once completed, the XSPI_XIP_GCTL.XIP_EN bit is cleared (=0) to indicate that the controller and memory device have exited XIP mode and are configured to work in normal mode.

Table 21-4: XIP Mode Register Configurations

XSPI_XIP_GCTL.XIP_EN	**XSPI_DAC_CFG.XIP_EN_MB_VAL**	**XSPI_DAC_CFG.XIP_DIS_MB_VAL**	**XSPI_READ_SEQ_CFG1.P1_MB_EN**	**Description**
0	0	0	0	Normal read mode without mode byte. (Number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)
0	0	1	0	Normal read mode with the additional disable mode byte following the address phase. (Number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)
0	0	0	1	Normal read mode with the additional disable mode byte following the address phase (number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)

Table 21-4: XIP Mode Register Configurations (Continued)

XSPI_XIP_GCTL.XIP_EN	XSPI_DAC_CFG.XIP_EN_MB_VAL	XSPI_DAC_CFG.XIP_DIS_MB_VAL	XSPI_READ_SEQ_CFG1.P1_MB_EN	Description
0	1	N/A	N/A	Enables XIP mode in the controller and memory device on the next read transfer – the enable mode byte is sent after the opcode and address phases. Then, XSPI_XIP_GCTL.XIP_EN =1. (Number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)
1	0	0	N/A	Controller reports DIR_CMD_ERR error. Mode byte must be sent when controller is configured to work in XIP mode.
1	0	1	N/A	Disables XIP mode in the controller and memory device on the next read transfer - the disable mode byte is sent after the address phase. Then, XSPI_XIP_GCTL.XIP_EN =0. (Number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)
1	1	N/A	N/A	Controller works in XIP mode - the enable mode byte is sent after the address phase. (Number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)

Table 21-4: XIP Mode Register Configurations (Continued)

XSPI_XIP_GCTL.XIP_EN	XSPI_DAC_CFG.XIP_EN_MB_VAL	XSPI_DAC_CFG.XIP_DIS_MB_VAL	XSPI_READ_SEQ_CFG1.P1_MB_EN	Description
0	0	1	1	Normal read mode with the additional disable mode byte following the address phase. (Number of dummy cycles = XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT)

Operating in ACMD Mode

The ACMD work mode relieves the host from issuing read and write commands directly. Instead, the host uses a dedicated engine to translate high-level commands into a series of low-level commands on the xSPI interface. It selects the optimal command sequence to send and to follow all transfer restrictions defined for the targeted memory device.

The auto command engine receives high-level commands either through a DMA descriptor format or directly through programmable registers. It is capable of operating multiple parallel threads that are configured using the controller registers and descriptors held in the main memory. It can operate in one of two modes – CDMA mode or PIO mode. Data transfers can be performed using either xSPI internal DMA mode or host DMA mode.

CDMA mode uses an embedded DMA to trigger various memory transactions using descriptors it fetches from main memory, automatically moving data between device memory and the host memory. In PIO work mode, dedicated controller registers are used in place of the descriptors to trigger the operation in the selected execution thread. The normal use case is to configure ACMD mode to use the internal DMA interface. The controller issues the relevant bus transactions directly to transfer data and to collect and process DMA descriptors. It is, however, possible to enable the ACMD engine to use the host DMA interface instead. In this case, the xSPI controller signals the host using an interrupt or status register when it requires the host to send write data on the host DMA write ports, or when it needs the host to issue a read command to fetch read data.

The auto command engine operates in the system (mACLK) clock domain. This mode uses system SRAM to buffer descriptor content locally; it is synchronous to mACLK.

NOTE: mACLK — This clock is SCLK0. It drives the front end of the controller. Both the AXI internal DMA and host DMA interfaces are synchronous to this clock.

The auto command engine can handle the following high-level sequences:

- Reset — sequence which allows the reset command to be sent to the selected device
- Erase — sequence to erase part of device or a whole device at once
- Program — sequence to transfer a specified amount of data from the host to the xSPI device
- Read — sequence to transfer a specified amount of data from the xSPI device to the Host

When operating in CDMA mode, the high-level sequence selection is made using a command descriptor. The DMA uses a standard structure where the head of the descriptor list is stored in APB register space. Each queued descriptor is chained together (one descriptor points to the next descriptor). For extra flexibility, up to eight descriptor chains can be managed simultaneously by the controller, and this is referred to as execution threads. Refer to section, Operating in ACMD Mode on for further details.

In PIO mode, the sequence is defined directly in specific command registers. Refer to section, Using the Auto Command in PIO Mode for further details.

NOTE: The high-level sequence the controller sends often must be broken up into multiple sub-sequences applied at the low-level xSPI interface. For example, it is common that prior to a program operation, a write enable sequence must be issued to instruct the memory device that a write operation is imminent, and, therefore, to configure the write enable latch of the device. This, obviously, depends on the attached memory type and controller configuration. The auto command engine selects the most appropriate low-level interface.

Using the Auto Command in CDMA Mode

To enable the auto command engine in CDMA mode, configure `XSPI_WORKMODE_CTL.WORKMODE` = 3. This operation can be done when the controller is in an idle state (poll the `XSPI_GSTAT.CTL_BUSY` bit). If software tries to access the `XSPI_WORKMODE_CTL` register when the controller is in the busy state, the access is ignored. To trigger the auto command engine in CDMA work mode, the `XSPI_CMD0`, `XSPI_CMD2` and `XSPI_CMD3` registers must be configured to pass the descriptor chain start address and the thread number associated with the descriptor chain. The command format for these registers are shown in the following tables.

When the ACMD mode is enabled, each write to the `XSPI_CMD0` register starts execution on a new command. Therefore, the `XSPI_CMD0` register must be written last; a write access to this register triggers the execution of the programmed operation.

Table 21-5: `XSPI_CMD0` Format for ACMD mode

`XSPI_CMD0` (offset 0x0)			
31:30	**29:27**	**26:24**	**23:0**
CDMA/PIO =00	Reserved	TRD_NUM (thread number)	Reserved

Table 21-6: `XSPI_CMD2` Format for ACMD mode

`XSPI_CMD2` **(offset 0x2) [31:0]**
Lower 32 bits of the descriptor head address

Table 21-7: `XSPI_CMD3` Format for ACMD mode

`XSPI_CMD3` (offset 0x3) [31:0]
Upper 32 bits of the descriptor head address

The host must write zero to the CDMA/PIO field of the XSPI_CMD0 register and provide the address of the head of the linked descriptor list using the XSPI_CMD3 and XSPI_CMD2 registers.

The controller issues a bus transaction to fetch the complete descriptor from main memory. The ***Command Descriptor Structure*** and ***Command Descriptor Layout*** tables present the details of the descriptor format.

Table 21-8: Command Descriptor Structure

No	63:48	47:32	31:16	15:0
0	Next pointer			
1	System memory pointer			
2	xSPI pointer			
3	Reserved			
4	Command counter		Command flags	Command type
5	Reserved		Status	

Table 21-9: Command Descriptor Layout

Field	Description
Next pointer	Next descriptor address. Address needs to follow system bus restrictions.
System memory pointer	System memory address required for data DMA commands. For read operations, this field is the address where the start of the read data is placed. For write operations, this field is the location of the start of the write data.
xSPI pointer	The field indicates the xSPI address for sequences that require addressing of the xSPI address space.
Command type	This field identifies the operation that the controller must perform. The encoding is: • 0x1000 – Erase the number of sequential sectors specified in the command counter field • 0x1001 – Full chip erase • 0x1100 – Device reset • 0x1101 – Device JEDEC reset • 0x2100 – Command to program the number of data bytes specified in the command counter field • 0x2200 – Command to read the number of data bytes specified in the command counter field • Other values are reserved
Command flags	This field contains different control flags for the operation of the command (See the ***Valid Command Flag*** table).

Table 21-9: Command Descriptor Layout (Continued)

Field	Description
Command counter	For read/program operations, the field indicates the number of data bytes to transfer minus one. For example, to transfer 1024 bytes, the value encoded is 1023. For an erase operation, this field represents the number of sectors to erase. For other operations, this field is ignored.
Status	The controller updates this field with the command status once the command operation is complete, and writes the status back to the descriptor. Refer to the Table 21-11 Descriptor Status Field table for bit descriptions.

On receipt of a complete descriptor, the controller constructs a low-level command or a series of low-level commands.

NOTE: If XIP mode is enabled in the controller (that is, `XSPI_XIP_GCTL.XIP_EN` = 1, which is supported by some devices to improve read performance), the connected device can only be read. Therefore, any other command will fail. As such, invoking the CDMA descriptor with a non-read command type results in the controller ignoring it and raising a DSC_ERROR fail status.

The ***Valid Command Flag*** table describes the allowed values in the command flags field of the descriptor.

Table 21-10: Valid Command Flag

Bits	Name	Description
15:11	Reserved	Reserved
10	DMA_SEL	This bit indicates which DMA interface is used to transfer data for read or program operations. If the selected command does not require a data transfer, this bit is ignored. 0: Selects the host DMA interface 1 : Selects the internal DMA interface Refer to section,Selecting the DMA Interface in ACMD mode for details on how to configure this bit.
9	CONT	Set (=1) the bit to indicate that another valid descriptor after this has been queued and is ready for the controller to fetch and process. Clear (=0) the bit when this descriptor is last in the descriptor chain.
8	INT	Set (=1) the bit to instruct the controller to issue an interrupt after the completion of descriptor processing. The triggered interrupt is identified in the

Table 21-10: Valid Command Flag (Continued)

Bits	Name	Description
		XSPI_TRD_COMP_ISTAT register. The bit of the interrupt status register that is set (=1) maps to the thread of the descriptor (as indicated originally in the descriptor by the TRD[n] field).
7	MB_XIP_DIS	When XIP mode is enabled in the controller and the attached memory device, setting this bit (=1) indicates to the controller that it must send mode bits to disable and exit XIP mode. When set, the controller sends mode bits as specified in the XSPI_XIP_GCTL.XIP_DIS_MB_VALUE field. Additionally, the XSPI_XIP_GCTL.XIP_EN_MB_VALUE bit is cleared to disable XIP mode in the controller. This bit is valid only for read commands. If XSPI_READ_SEQ_CFG1.P1_MB_EN =1 and XSPI_XIP_GCTL.XIP_EN = 0 the controller ignores the value of this field and it is treated as set (=1) (for example, mode bits are sent regardless of the value of this field).
6	MB_XIP_EN	This bit triggers entry to XIP mode. Setting this bit indicates to the controller that it must send mode bits to the device to enable XIP mode. The mode bits that are sent depend on the XSPI_XIP_GCTL.XIP_EN_MB_VALUE field. Additionally, the XSPI_XIP_GCTL.XIP_EN bit is set to enable XIP mode in the controller. This bit is valid only for read commands.

Table 21-10: Valid Command Flag (Continued)

Bits	Name	Description
5	xSPI_PTR_CONT	If this bit is set, the xSPI pointer field of current descriptor is ignored, and command engine uses the existing xSPI pointer value. Essentially, setting this bit allows the host to continue a data transfer from the address on the flash interface where a previously completed descriptor finished. This feature can only be triggered if a previous descriptor in the chain had this bit cleared and the xSPI pointer was initialized. It must also only be used for a command of the same type (for example, a series of read descriptors or series of write descriptors). For SPI NAND devices, a transfer is always continued from the beginning of the next page even if the current page was not filled completely (column address = 0, row address incremented).
4	SYS_PTR_CONT	If this bit is set (=1), the system memory pointer field of the current descriptor is ignored. The command engine uses the existing memory pointer value. This bit allows the data transfer to continue on the host interface where the previously completed descriptor finished. This feature can only be triggered if a previous descriptor in the chain had this bit cleared and the system memory pointer was initialized. It must also only be used for commands of the same type (for example, a series of read descriptors or series of write descriptors).
3	Reserved	Reserved
2:0	BANK	The field indicates the device number (bank/CS) that is selected for sequence execution.

NOTE: If the INT bit is not set in the current descriptor and the following descriptor chain is dropped from execution due to an error, the controller does not set the TRD[n] interrupt flag in the `XSPI_TRD_COMP_ISTAT` register. The error condition is, however, always reported by setting the TRD[n] bit in the `XSPI_TRD_ERR_ISTAT` register. The bit can be used to detect that the descriptor chain execution was interrupted due to an error. The n in the field names represent the thread number for which the error or complete condition occurred.

Opon completion of a command, the controller writes back the status into the descriptor status field. This field value is ignored during descriptor read operation. The status word is described in the ***Descriptor Status Field*** table.

Table 21-11: Descriptor Status Field

Bits	Name	Description
31:24	Reserved	Reserved
23:16	ECC_STAT	Value of the ECC status returned by the SPI NAND device. The value of this field applies to last read operation. It is updated only when fail checking is enabled.
15	COMPLETE	When set, the bit denotes that the controller has updated status information and the operation is complete. This bit is set even when the operation ended as a failure. This bit should be in a cleared state while descriptor is constructed.
14	FAIL	When set, the bit denotes that the operation failed to complete successfully.
13:6	Reserved	Reserved
5	ECC_CORR_ERROR	The bit indicates that an ECC correctable error was detected during page read. This bit is valid only for SPI NAND devices when `XSPI_STAT_SEQ_CFG10.ECC_FAIL_EN=1`.
4	DEVICE_ERROR	The bit indicates that a device error was detected during a read status operation for erase/program/read.
3	DQS_ERROR	This bit is set when an incorrect number of DQS pulses were detected during a direct mode read or status check. Essentially, this status is passed from the soft PHY and could mean: The PHY is configured badly such that it expects DQS strobes but never received them, or, The value (`XSPI_PHY_GATE_LPBK_CTL.RD_DEL_SEL` field) is incorrect and has caused data corruption and pointer misalignment in the PHY. To resolve this issue, the PHY must be reset to clear the DQS_OVF and DQS_UR flags and reset the read data pointers.

Table 21-11: Descriptor Status Field (Continued)

Bits	Name	Description
2	CRC_ERROR	When set, the bit indicates that the controller detected a CRC error.
1	BUS_ERROR	When set, the bit indicates that the controller got an error response on the system DMA interface.
0	DSC_ERROR	This bit indicates that an invalid descriptor sequence has been detected.

The auto command engine accept new commands only when the thread selected is in am idle state. The host software should check the thread status before it queues any new commands. The threads status is available in the XSPI_TRD_STAT register. Commands sent to a busy thread are ignored and XSPI_ISTAT.CMD_IGNORED = 1.

Using the Auto Command in PIO Mode

In PIO work mode, dedicated command registers are used to trigger operation in a selected execution thread of the auto command engine. In this mode it is only possible to trigger a single command in each of the execution threads. This mode is useful when host software wants to trigger a single operation. In this case it is much simpler and faster to program a few controller registers than prepare a descriptor list in system memory. Functionality of this work mode is similar to the CDMA work mode for cases where the descriptors chain contains only a single descriptor. The auto command engine executes the programmed command and returns a status word separately for each execution thread. Status words are mapped into controller status registers, where it remains valid until a new command is triggered on this thread.

Depending on the selected operation, the number of XSPI_CMD* required to be programmed. The registers described as unused for given operation do not need to be set. XSPI_CMD1 to XSPI_CMD5registers retain the contents from previous writes meaning that if the current register configuration matches what is needed for the next operation, it can be left unchanged. The only exception is the XSPI_CMD0 register. This register must always be written on each operation. Write access to this register triggers operation execution. Host software needs to initialize all required command registers before it writes to command register 0.

- The reset and chip erase sequences only requires XSPI_CMD0 to be written to. XSPI_CMD1 through XSPI_CMD5 registers are unused.
- The sector erase sequence requires XSPI_CMD0, XSPI_CMD1, XSPI_CMD4 and XSPI_CMD5 to be written, XSPI_CMD2 and XSPI_CMD3 are unused.
- The program and read sequences require all XSPI_CMD* registers to be written.

The XSPI_CMD0 - XSPI_CMD5 tables below show the layout for the available high-level sequences that may be triggered in PIO mode. The operation is always triggered when XSPI_CMD0 is written to, so configure XSPI_CMD1-XSPI_CMD5 registers first.

Table 21-12: XSPI_CMD0 Layout for PIO Sequences

Bits	Name	Description
31:30	CDMA/PIO	Set (=1) for PIO work mode.
29:27	-	Reserved
26:24	TRD_NUM	This field selects the destination thread number for command. Software can select any available thread. Commands can be issued in parallel to all threads.
23	-	Reserved
22:20	BANK/CS	Selects device targeted by the sequence
19	DMA_SEL	This bit selects which DMA interface should be used for transferring data. It is only relevant for commands that transfer data (for example, read or program operations). Clear (=0) to use the host DMA interface. Set (=1) to use the internal DMA interface.
18	INT	If this bit is set, an interrupt is issued after the operation has finished. The triggered interrupt is indicated in the XSPI_TRD_COMP_ISTAT register, where interrupt bit is selected by the thread number in the TRD[n] field.
17	MB_XIP_DIS	When XIP mode is enabled in the controller and the attached memory device, setting this bit indicates to the controller that it must send mode bits to disable and exit XIP mode. When set, the controller sends mode bits as specified in the XSPI_XIP_GCTL.XIP_DIS_MB_VALUE field . Additionally, the XSPI_XIP_GCTL.XIP_EN_MB_VALUE bit is cleared to disable XIP mode in the controller. This bit is valid only for read operations. If XSPI_READ_SEQ_CFG1.P1_MB_EN= 1 and XSPI_XIP_GCTL.XIP_EN_MB_VALUE = 0 the controller ignores the value of this field. It is treated as set (= 1) (for example, mode bits are sent regardless of the value of this field).

Table 21-12: XSPI_CMD0 Layout for PIO Sequences (Continued)

Bits	Name	Description
16	MB_XIP_EN	This bit triggers entry to XIP mode. Setting this bit indicates to the controller that it must send mode bits to the device to enable XIP mode. The mode bits sent depends on the XSPI_XIP_GCTL.XIP_EN_MB_VALUE field. Additionally, the XSPI_XIP_GCTL.XIP_EN_MB_VALUE bit is set to enable XIP mode in the con- troller. This bit is valid only for read operations.
15:0	CMD_TYPE	This field identifies the kind of operation that is executed by the con- troller. It is con- figured as follows: 0x1100 : Reset sequence (soft reset) 0x1101 : Reset sequence (JEDEC reset) 0x1000 : Sector erase sequence 0x1001 : Chip erase sequence 0x2100 : Program sequence 0x2200 : Read sequence

The XSPI_CMD1 register is required for all operations other than reset and chip erase. It holds the lower half of the memory device starting address where the operation takes place.

Table 21-13: XSPI_CMD1 Layout for PIO Sequences

Bits	Name	Description
31:0	xSPIAddress (Lower)	This field contains the lower half of the starting address of the operation in the xSPI device.

The XSPI_CMD2 is only required for read and program operations. It holds the lower half of the host side address where the operation fetches data for writes, or places data for reads.

Table 21-14: XSPI_CMD2 Layout for PIO Sequences

Bits	Name	Description
31:0	SYS_ADDR_PTR_L	Host memory address required for DMA transfers. It is lower half of the address.

The XSPI_CMD3 is only required for read and program operations. It holds the upper half of the host side address where the operation should fetch data for writes, or place data for reads.

Table 21-15: XSPI_CMD3 Layout for PIO Sequences

Bits	Name	Description
31:0	SYS_ADDR_PTR_H	Host memory address required for DMA transfers. It is upper half of the address.

The XSPI_CMD4 is required for sector erase operations as well as read and program operations. For sector erase, it holds the number of sectors that must be erased. For read/program operations, it holds the number of data bytes that should be transferred.

Table 21-16: XSPI_CMD4 Layout for PIO SECTOR ERASE Sequences

Bits	Name	Description
31:0	SECT_CNT	Sector counter. Determines how many sectors shouldbe erased. Number of sectors is decoded as a value of this field plus 1 (that is, to erase 1 sector this field should have value 0, and so on).

Table 21-17: XSPI_CMD4 Layout for PIO READ/PROGRAM Sequences

Bits	Name	Description
31:0	DATA_CNT	Data counter. Determines how many data bytes should be transferred minus 1. For example, if 1024 bytes should be transferred, the value to be programmed should be 1023.

The XSPI_CMD5 is required for all operations other than reset and chip erase. It holds the upper half of the memory device starting address where the operation should take place.

Table 21-18: XSPI_CMD5 Layout for PIO Sequences

Bits	Name	Description
31:0	xSPI Address (Upper)	This field contains additional address space for the xSPI address. It is utilized for SPI NAND devices to hold both the row and column address as a single xSPI address. For sector erase operations, bits that belong to column address must be cleared (=0).

Status Checking

The host can check the operating status of each active thread. After the thread ID is written to the XSPI_CMD_STAT_PTR register, the status can be read from the XSPI_CMD_STAT register. The COMPLETE bit is cleared when a new operation is triggered, and all other fields within the XSPI_CMD_STAT register should at this point be ignored. When the selected operation is complete, the controller sets the COMPLETE bit and all other register fields reflects the status of that operation.

The ***Status Checking*** table describes the status fields.

Table 21-19: Status Checking

Bits	Name	Description
31:24	Reserved	Reserved
23:16	ECC_STAT	Value of the ECC status returned by the SPI NAND device. Value of this field applies to last read operation and is updated only when fail checking is enabled.
15	COMPLETE	When set, denotes that the controller has updated status information and the operation is complete. This bit shall be set even if the operation ended as a failure.
14	FAIL	When set, denotes that operation failed to complete successfully.
13:6	Reserved	Reserved
5	ECC_CORR_ERROR	ECC correctable error was detected during page read. This field is valid only for SPI NAND devices when the `XSPI_STAT_SEQ_CFG10.ECC_FAIL_EN` =1.
4	DEVICE_ERROR	Device error was detected during read status operation for erase/program/read operation.
3	DQS_ERROR	Incorrect DQS pulses number was detected during the data read operation. This is information from the PHY that either the DQS strobe did not appear during read (for example, device is not connected to the controller) or RD_DEL_SEL signal value is incorrect (`XSPI_PHY_GATE_LPBK_CTL.RD_DEL_SEL`) and data read from the flash device are corrupted and read FIFO pointers in the PHY are misaligned. The $\overline{\text{DLL_RST}}$ signals to clear DQS_OVF and DQS_UR flags in the PHY and clear all PHY read data pointers. One of these is required by the PHY before continuing to work after the DQS_ERROR signal assertion.
2	CRC_ERROR	When set, it indicates that the controller detected a CRC error.

Table 21-19: Status Checking (Continued)

Bits	Name	Description
1	BUS_ERROR	When set, it indicates that the controller got an error response on the system DMA interface.
0	CMD_ERROR	This bit indicates that an invalid command sequence has been detected.

Selecting the DMA Interface

ACMD mode can be configured to use an internal DMA interface to have the controller issue the relevant bus transactions to transfer data and to collect and process DMA descriptors. This mode requires the least amount of software overhead. It is, however, also possible to enable the ACMD engine to use the host DMA interface instead.

- To use the ACMD in CDMA mode with data transfers passing via the host DMA interface, follow the process as described in the section, Using the Auto Command in CDMA Mode, clearing the DMA_SEL bit in the command flags.
- To use the ACMD in CDMA mode with data transfers passing via the internal DMA interface, follow the process as described in the section, Using the Auto Command in CDMA Mode, setting the DMA_SEL bit in the command flags.
- To use the ACMD in PIO mode with data transfers passing via the host DMA interface, follow the process as described in the section, Using the Auto Command in PIO Mode, clearing the DMA_SEL bit of the `XSPI_CMD0` register.
- To use the ACMD in PIO mode with data transfers passing via the internal DMA interface, follow the process as described in the section, Using the Auto Command in PIO Mode, setting the DMA_SEL bit of the `XSPI_CMD0` register.

When transfers using the bus completer interface are enabled, the following sequence also needs to be followed by the external host to transfer the required data:

1. Poll the `XSPI_ISTAT.SDMA_TRIGG` bit. Or, if an interrupt is preferable, unmask the interrupt associated with SDMA_TRIGG by setting the `XSPI_INT_EN.SDMA_TRIGG_EN` bit. When the bit is asserted, the host can proceed.
2. The external host reads the `XSPI_SDMA_SIZ` register, the `XSPI_SDMA_TRD_STAT` register and the `XSPI_SDMA_ADDR0`/`XSPI_SDMA_ADDR1` registers. The `XSPI_SDMA_SIZ` register reports the size of the data transfer that must be transferred. The `XSPI_SDMA_TRD_STAT` register reports the thread ID of the operation that triggered SDMA data transfer and also reports the data transfer direction. The `XSPI_SDMA_ADDR0`/`XSPI_SDMA_ADDR1` registers indicate the address (awaddr/araddr) the external host must use on the write or read transaction when it issues it.
3. Before starting data transmission, the external host must clear the `XSPI_ISTAT.SDMA_TRIGG` register by writing 1.

4. Host can now transfer data. For program operations, it is expected that the host will issue one or more write requests with a total size matching the data transfer indicated in step 2. For read operations, it is expected that the host will issue one or more read requests with a total size matching the data transfer indicated in step 2. The address issued by the host should match that being reported in the XSPI_SDMA_ADDR0/ XSPI_SDMA_ADDR1 registers.

If the host ignores any of the requirements and starts the data transfer when the host DMA is not ready, theXSPI_ISTAT.SDMA_ERR flag is set. If the XSPI_INT_EN.SDMA_ERR_EN bit is set, an interrupt is also triggered. When the XSPI_DMA_CTL.SDMA_ERR_RSP bit is set, an error response is returned; if it is cleared, the OK response is returned.

If the host sends an unsupported transaction to a bus completer interface, the host DMA ignores this access and the XSPI_ISTAT.SDMA_ERR flag is set. If an error response is detected on the system bus during host DMA transfer, the fail and bus error bits are set in the last operation status. The last operation status can check in:

- The XSPI_CMD_STAT register. After selecting thread in which command was executed (for ACMD PIO work mode). Thread is selected by value of the XSPI_CMD_STAT_PTR register. This method is valid for the ACMD PIO and STIG work modes.
- The status field of the command descriptor. This method is valid for the CDMA work mode.

Performing transactions in wrong direction (that is, the host issuing a write transaction to the bus completer port in case of sending read operation command to an xSPI device) is forbidden and causes the XSPI_ISTAT.SDMA_ERR flag to be set. If the XSPI_INT_EN.SDMA_ERR_EN bit is set, an interrupt is also triggered.

Rules and Limitations

The host must follow the following rules when using ACMD mode:

- For SPI NAND devices, the initial column address must be specified within the range of the page size.
- For profile 2 or profile 1, octal DDR devices, the number of bytes to be transferred must be even.
- For profile 2 or profile 1, octal DDR devices, the start address of the transfer in the xSPI device must be even.
- For profile 1 and octal DDR devices, the command extension must be enabled.
- For DATA_PER_ADDR (when the device is 16-bit addressed), the start address must be even and the number of bytes to transfer must be even.
- If XIP mode is enabled, only read operations are permitted.
- Only the supported command types as illustrated in this section should be attempted.

Violating any of the above rules results in the controller returning the DSC_ERROR/CMD_ERROR status.

NOTE: DSC_ERROR and CMD_ERROR status bits are only intended to ease debug and integration and should never be triggered for correct controller operation. Using the controller outside of the descriptor and command rules may lead to undefined behavior.

In automatic work modes, the controller writes/reads data consecutively from the specified address. If a page boundary is crossed, the transaction continues from the beginning of the next page.

Operating in STIG Mode

The Software-Triggered Instruction Generator (STIG) mode allows software to define the low-level commands and sequences sent to the memory device. Relative to direct or ACMD mode, it is more flexible because software can control the low-level characteristics of each operation carried out on the device. STIG mode is normally used with the direct or ACMD modes of operation.

To enable the STIG work mode, the `XSPI_WORKMODE_CTL.WORKMODE` field must be set (= 1) . This can be done only when the controller is in an idle state. The ***STIG Mode Registers*** table shows the registers used to configure STIG mode operations.

Table 21-20: STIG Mode Registers

Bits	Name	Description
31:0	`XSPI_CMD0`	A write to this register triggers the STIG operation and has no other purpose. Reading this register is ignored.
31:0	`XSPI_CMD1`	Holds bits [31:0] of the STIG instruction
31:0	`XSPI_CMD2`	Holds bits [63:32] of the STIG instruction
31:0	`XSPI_CMD3`	Holds bits [95:64] of the STIG instruction
31:0	`XSPI_CMD4`	Holds bits [127:96] of the STIG instruction

Instruction Variants

The main instruction variants for STIG mode include the generic command instruction, legacy HyperFlash or profile 2 instructions, and legacy SPI xSPI profile 1 or SPI NAND instructions. The following tables show the structure of the STIG command instruction (`XSPI_CMDn`) for each variant.

Table 21-21: STIG Command Instruction - Legacy SPI, xSPI Profile 1 or SPI NAND

Name	Bits	Description
Reserved	127:126	N/A
t_{CMS}/t_{CEM} Enable (`XSPI_SEQ_GCTL0.TCMS_EN`)	125	Setting this bit enables the feature that automatically forces the controller to pause activity to the memory device to allow it to perform a periodic refresh operation. Some volatile memory devices require this functionality (for example, HyperRAM). This functionality is referred to as t_{CMS} timing. When set, the controller pauses activity as specified in the `XSPI_MINICTL_DEV_ACTIVE_MAX` register.

Table 21-21: STIG Command Instruction - Legacy SPI, xSPI Profile 1 or SPI NAND (Continued)

Name	Bits	Description
Instruction Termination (INSTR_LINK)	124	This bit is only relevant when the instruction type field of the instruction = 0, 32, 36 or 96. See the section Instruction Type Variants for details on instruction type configuration. The field assigned to this bit indicates whether the instruction is to be terminated (CS is deactivated after the instruction finishes) or the instruction is glued with the next instruction (providing it is ready to be fetched from the command FIFO). 0 – Current sequence is terminated and not glued with another instruction of the same type 1 – Current sequence is glued with the next STIG instruction
Reserved	123:119	N/A
CRC Variant (CRC_VARIANT)	118	Determines when the CRC is calculated on an outbound command. When cleared (=0), calculate for the address bytes only. When set (=1), calculate for all bytes in the sequence.
CRC Enable (CRC_EN)	117	Set to enable dynamic CRC calculation and output the value on xSPI flash interface in the transfer modes as specified by ADDR_EDGE_MODE and ADDR_NO_OF_IOS. If the octal DDR (8D) variant is configured, the calculated CRC byte is maintained for a full flash clock cycle.
XIP Enable (XIP_EN)	116	Setting this bit instructs the controller to block the sending of the opcode, which is needed when the device is operating in XIP mode 0 – Opcode transfer phase is sent 1 – Opcode transfer phase is not sent (XIP enabled)
Address Shift (ADDR_SHIFT)	115	When the ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. Therefore, the host must configure an even number of bytes in the quoted case. However, there are specific exceptions such as the Adesto octal DTR SFDP command which requires a {A2,A1,A0,do not care} flash interface address alignment rather than a common {A3,A2,A1,A0} alignment. The ADDR_SHIFT bit shifts the address byte chain by one byte towards the left and fills the least-significant byte with zeroes (when an octal DDR address structure is required).
Reserved	114:111	N/A
Bank Number (BANK_NUM)	110:108	This field informs the controller on which memory device the sequences should be executed. This information directly maps to the chip select (CS) pin.

Table 21-21: STIG Command Instruction - Legacy SPI, xSPI Profile 1 or SPI NAND (Continued)

Name	Bits	Description
Opcode Edge Mode (OPCODE_EDGE_MODE)	107	This bit selects SDR or DDR operation at the memory device 0 – SDR 1 – DDR
Reserved	106	N/A
Opcode IO Mode (OPCODE_NO_OF_IOS)	105:104	Determines number of active IOs for opcode phase of the instruction 0 – Single 1 – Dual 2 – Quad 3 – Octal
Reserved	103:101	N/A
Opcode Extension Enable (OPCODE_EXT_EN)	100	Enables dual byte opcode for the instruction 0 – Only CMD[87:80] is transmitted in the opcode phase of the instruction. 1 – CMD[87:80] followed by CMD_EXT[79:72] is transmitted in the opcode phase of the instruction. If the OPCODE_EDGE_MODE and OPCODE_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. The host must also set OPCODE_EXT_EN =1. Failure to do this results in an error being reported.
Address Edge Mode (ADDR_EDGE_MODE)	99	Determines edge mode for the address phase of the instruction. 0 – SDR 1 – DDR
Reserved	98	N/A
Address IO Mode (ADDR_NO_OF_IOS)	97:96	Determines the number of active IOs for the address phase of the instruction 0 – Single 1 – Dual 2 – Quad 3 – Octal
Reserved	95	N/A

Table 21-21: STIG Command Instruction - Legacy SPI, xSPI Profile 1 or SPI NAND (Continued)

Name	Bits	Description
Address Number of bytes (ADDR_NO_OF_BYTES)	94:92	Determines the number of address bytes to be sent in the instruction. Up to six address bytes can be sent in one sequence. A value of 7 encoded here is not permitted. A value of 0 encoded implies that no address bytes are sent (for example, address phase is disabled). If ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. The host must configure an even number of address. If the host configures ADDR_NO_OF_BYTES = 1 or 3 or 5, the controller generates an error.
Reserved	91:90	N/A
Number of data bytes (DATA_NO_OF_BYTES) or Number of mode bytes (MODE_NO_OF_BYTES)	89:88	For cases where the number of data (or mode) bytes is ≤ 2, this field allows that data to be included within this instruction, directly avoiding the need to glue a separate data sequence instruction. If there are more than two data bytes required to be transferred, the instruction must be glued with a data sequence. Note that if up to two data (or mode) bytes are transferred directly from this instruction, the format in which they are presented to the device depends on the ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration. If the host needs the data to be sent in a different format, this field must be cleared (=0). Instead, transfer the data using a glued data sequence instruction, or generic instruction where the format of those data bytes can be individually configured. 0 – zero data/mode bytes are sent to device (data phase disabled) 1 – One byte is transferred to the device 2 – Two bytes are transferred to the device 3 – N/A If ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. The host must configure an even number of bytes. Failure to do this results in an error being reported.
CMD	87:80	Opcode mnemonic
CMD_EXT	79:72	Opcode extension mnemonic
ADDR5	71:64	Byte 5 of the address phase
ADDR4	63:56	Byte 4 of the address phase

Table 21-21: STIG Command Instruction - Legacy SPI, xSPI Profile 1 or SPI NAND (Continued)

Name	Bits	Description
ADDR3	55:48	Byte 3 of the address phase
ADDR2	47:40	Byte 2 of the address phase
ADDR1	39:32	Byte 1 of the address phase
ADDR0	31:24	Byte 0 of the address phase
DATA1/MODE1	23:16	Byte 1 of the data/mode phase
DATA0/MODE0	15:8	Byte 0 of the data/mode phase
Reserved	7	N/A
Instruction Type	6:0	This field defines the type (index) of the instruction. A description of each value and its corresponding instruction can be found in the section, Instruction Type Variants.

Table 21-22: STIG Instruction - Legacy HyperFlash or xSPI Profile 2

Name	Bits	Description
Reserved	127:126	N/A
t_{CMS}/t_{CEM} Enable (TCMS_EN)	125	Setting this bit enables the feature that automatically forces the controller to pause activity to the memory device to allow it to perform a periodic refresh operation. Some volatile memory devices require this functionality (for example, HyperRAM) and is referred to as t_{CMS} timing. When set, the controller pauses activity as specified in the XSPI_MINICTL_DEV_ACTIVE_MAX register.
Instruction Termination (INSTR_LINK)	124	This bit is only relevant when the instruction_type field of the instruction = 0, 32, 36 or 96. Refer to the Instruction Type Variants section for details on instruction type configuration. The field assigned to this bit indicates whether the instruction is to be terminated (CS is to be deactivated after the instruction finishes) or the instruction is to be glued with the next instruction providing it is ready to be fetched from the command FIFO. 0 – Current sequence is terminated and not glued with another instruction of the same type 1 – Current sequence is glued with the next STIG instruction
Reserved	123:119	N/A
CRC Variant (CRC_VARIANT)	118	When the CRC is calculated on the outbound command, this bit indicates when cleared (=0) to calculate for the address bytes only, or when set (=1) to calculate for all bytes in the sequence.

Table 21-22: STIG Instruction - Legacy HyperFlash or xSPI Profile 2 (Continued)

Name	Bits	Description
CRC Enable (CRC_EN)	117	Set to enable dynamic CRC calculation and output this value on xSPI flash interface in the transfer modes specified by ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration. If an octal DDR (8D) variant is configured, the calculated CRC byte is maintained for full flash clock cycle.
XIP Enable (XIP_EN)	116	Setting this bit instructs the controller to block the sending of the opcode, which is needed when the device is operating in XIP mode. 0 – Opcode transfer phase is sent 1 – Opcode transfer phase is not sent (XIP enabled)
Address Shift (ADDR_SHIFT)	115	If ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration determine the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. The host must configure an even number of bytes in the quoted case. However, there are specific exceptions. For example, the Ades-to octal DTR SFDP command requires a {A2,A1,A0,do not care} flash interface address alignment rather than the common {A3,A2,A1,A0} alignment. The ADDR_SHIFT bit shifts the address byte chain by one byte towards the left and fills the least-significant byte with zeroes providing the octal DDR address is required.
Reserved	114:111	N/A
Bank Number (BANK_NUM)	110:108	This field informs the controller on which memory device the sequence should be executed. This information directly maps to the chip select (CS) pin.
Opcode Edge Mode (OPCODE_EDGE_MODE)	107	This bit selects SDR or DDR operation at the memory device 0 – SDR 1 – DDR
Reserved	106	NA
Opcode IO Mode (OPCODE_NO_OF_IOS)	105:104	Indicates the number of active IOs for opcode phase of the instruction 0 – Single 1 – Dual 2 – Quad 3 – Octal
Reserved	103:101	N/A

Table 21-22: STIG Instruction - Legacy HyperFlash or xSPI Profile 2 (Continued)

Name	Bits	Description
Opcode Extension Enable (OPCODE_EXT_EN)	100	Enables dual byte opcode for the instruction 0 – Only CMD[87:80] is transmitted in the opcode phase of the instruction 1 – CMD[87:80] followed by CMD_EXT[79:72] is transmitted in the opcode phase of the instruction If OPCODE_EDGE_MODE and OPCODE_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. It is therefore required that the host also sets OPCODE_EXT_EN. Failure to do this results in an error being reported.
Address Edge Mode (ADDR_EDGE_MODE)	99	Determines edge mode for address phase of the instruction 0 – SDR 1 – DDR
Reserved	98	N/A
Address IO Mode (ADDR_NO_OF_IOS)	97:96	Determines number of active IOs for the address phase of the instruction 0 – Single 1 – Dual 2 – Quad 3 – Octal
Reserved	95	N/A
Address Number of bytes (ADDR_NO_OF_BYTES)	94:92	Determines number of address bytes to be sent in the instruction. Up to 6 address bytes can be sent in one sequence. A value of 7 encoded here is not permitted. A value of 0 encoded implies that no address bytes is sent (for example, the address phase is disabled). If ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. It is, therefore, required for the host to configure an even number of address. If the host configures ADDR_NO_OF_BYTES = 1, 3 or 5, the controller generates an error.
Reserved	91:90	N/A

Table 21-22: STIG Instruction - Legacy HyperFlash or xSPI Profile 2 (Continued)

Name	Bits	Description
Number of data bytes (DATA_NO_OF_BYTES) or Number of mode bytes (MODE_NO_OF_BYTES)	89:88	For cases where the number of data (or mode) bytes is ≤ 2, this field allows that data be included within this instruction, directly avoiding the need to glue a separate data sequence instruction. If there are more than two data bytes required to be transferred, the instruction must be glued with a data sequence. Note that if up to two data (or mode) bytes are transferred directly from this instruction, the format in which they are presented to the device depends on the ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration. If the host needs the data to be sent in a different format, the field must be cleared (=0) . Instead, transfer the data using a glued data sequence instruction, or generic instruction, where the format of the data bytes can be individually configured. 0 – Zero data/mode bytes are sent to device (data phase disabled) 1 – One byte is transferred to the device 2 – Two bytes are transferred to the device 3 – N/A If ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration determines the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. The host must configure an even number of bytes. Failure to do this results in an error being reported.
CMD	87:80	Opcode mnemonic
CMD_EXT	79:72	Opcode extension mnemonic
ADDR5	71:64	Byte 5 of the address phase
ADDR4	63:56	Byte 4 of the address phase
ADDR3	55:48	Byte 3 of the address phase
ADDR2	47:40	Byte 2 of the address phase
ADDR1	39:32	Byte 1 of the address phase
ADDR0	31:24	Byte 0 of the address phase
DATA1/MODE1	23:16	Byte 1 of the data/mode phase
DATA0/MODE0	15:8	Byte 0 of the data/mode phase
Reserved	7	N/A
Instruction Type	6:0	This field defines the type (index) of the instruction. A description of each value and its corresponding instruction can be found in the section, Instruction Type Variants.

Table 21-23: Generic Command STIG Instruction

Name	Bits	Description
Reserved	127:125	N/A
Instruction Termination (INSTR_LINK)	124	This bit is only relevant when the instruction_type field of the instruction=0, 32, 36 or 96. Refer to section Instruction Type Variants for details on the instruction type setting. The field assigned to this bit indicates whether the instruction is to be terminated (CS must be deactivated after the instruction finishes) or glued with the next instruction (providing it is ready to be fetched from the command FIFO). 0 – Current sequence is terminated and not glued with another instruction of the same type 1 – Current sequence is glued with the next STIG instruction
Reserved	123:118	N/A
CRC Enable (CRC_EN)	117	Set to enable dynamic CRC calculation and output this value on the xSPI flash interface. The transfer modes are specified by ADDR_EDGE_MODE and ADDR_NO_OF_IOS configuration. If the octal DDR (8D) variant is configured, the calculated CRC byte is maintained for a full flash clock cycle.
Reserved	116:111	N/A
Bank Number (BANK_NUM)	110:108	This field informs the controller on which memory device the sequence should be executed. This information directly maps to the chip select (CS) pin.
Edge Mode (EDGE_MODE)	107	This bit selects SDR or DDR operation at the memory device. 0 – SDR 1 – DDR
Reserved	106	N/A
IO Mode (NO_OF_IOS)	105:104	Determines number of active IOs for the instruction 0 – Single 1 – Dual 2 – Quad 3 – Octal
Reserved	103:92	N/A

Table 21-23: Generic Command STIG Instruction (Continued)

Name	Bits	Description
Number of bytes (NO_OF_BYTES)	91:88	Determines the number of bytes to be sent in the instruction At least one byte must be sent, but no more than 10 bytes in one sequence. If EDGE_MODE and NO_OF_IOS configuration determines the octal DDR variant, two bytes must be sent in a single flash clock cycle. The host must configure an even number of bytes in the quoted case (allowable NO_OF_BYTES = 0 or 2 or 4 or 6 or 8 or 10). If the host configures NO_OF_BYTES = 1 or 3 or 5 or 7 or 9, the controller does not accept the transfer request and generates an error.
BYTE9	87:80	The 10th byte sent when NO_OF_BYTES = 10
BYTE8	79:72	The 9th byte sent when NO_OF_BYTES ≥9
BYTE7	71:64	The 8th byte sent when NO_OF_BYTES ≥8
BYTE6	63:56	The 7th byte sent when NO_OF_BYTES ≥7
BYTE5	55:48	The 6th byte sent when NO_OF_BYTES ≥6
BYTE4	47:40	The 5th byte sent when NO_OF_BYTES ≥5
BYTE3	39:32	The 4th byte sent when NO_OF_BYTES ≥4
BYTE2	31:24	The 3rd byte sent when NO_OF_BYTES ≥3
BYTE1	23:16	The 2nd byte sent when NO_OF_BYTES ≥2
BYTE0	15:8	The 1st byte sent
Reserved	7	N/A
Instruction Type	6:0	This field defines the type (index) of the instruction. A description of each value and its corresponding instruction can be found in the section Instruction Type Variants.

Glued Data Instruction

The ***Data Instruction Structure*** table outlines the structure of the main data instruction variant that the host issues as the final glued instruction of a sequence of instructions. This variant cannot be used by itself and must be glued to one of the other variants. It is typically used to define how data is transferred in accordance with the details specified in the other three main command variants. As such, this variant must never be the first instruction issued by the host; it is always the last instruction.

Table 21-24: Data Instruction Structure

Name	Bits	Description
Reserved	127:123	N/A

Table 21-24: Data Instruction Structure (Continued)

Name	Bits	Description
CRC Chunk Size (CRC_CHUNK_SIZE)	122:120	This field presents the size of the CRC chunk as configured in the memory device. It is the number of bytes after which the CRC byte is placed onto the bus. 1 – N/A 2 – 8B chunk size 3 – 16B chunk size 4 – 32B chunk size 5 – 64B chunk size 6 – 128B chunk size 7 – 256B chunk size 8 – 512B chunk size
CRC Read Status Inversion Enable (CRC_INV)	119	This bit controls whether the controller requests one more status slice to check its data integrity. The expectation is that the redundant data chunk will be inverted (as applicable for Macronix MX25 octal devices). If redundant, data from flash is not inverted. A CRC error is generated. This bit must only be set when CRC_EN = 1, DATA_EDGE_MODE = 1 (DDR), DATA_NO_OF_IOS = 3 (octal) and DATA_NO_OF_BYTES = 2 because this set of parameters matches the implementation of Macronix MX25 octal devices. If CRC_INV = 1, the CRC_CHUNK_SIZE configuration is ignored because these two sequence parameters are mutually exclusive.
CRC Output Enable (CRC_OE)	118	This bit specifies whether the CDC is an output on both clock edges, with the CRC being inverted on the second clock edge. 0 – Inverted CRC is not output 1 – Inverted CRC is output
CRC Enable (CRC_EN)	117	This bit is set to enable dynamic CRC calculation and output this value on the xSPI flash interface. The transfer modes are specified by ADDR_EDGE_MODE and ADDR_NO_OF_IOS configurations. If an octal DDR (8D) variant is configured, the calculated CRC byte is maintained for a full flash clock cycle.

Table 21-24: Data Instruction Structure (Continued)

Name	Bits	Description
CRC Unaligned Chunk Enable (CRC_UAL_CHUNK_EN)	116	Setting this bit instructs the controller to take into consideration the command address in determining where the CRC bytes are expected to be returned from the memory device. This bit can be set (=1) only when the sequence (DATA_SEQ) is glued with the READ_PROFILE_1 sequence. It must be cleared (=0), otherwise. For example, if the original flash address is 0x4 and CRC_CHUNK_SIZE is 16B, the first CRC from the device is expected to occur after 12 bytes (CRC_CHUNK_SIZE - (device address)). Only even addresses are supported (when CRC_UAL_CHUNK_EN = 1) as required by Macronix MX25 octal devices.
CRC Unaligned Chunk Check (CRC_UAL_CHUNK_CHK)	115	Set this bit to have the controller check the correctness of a CRC unaligned chunk from a flash device. It can be set (=1) only when CRC_UAL_CHUNK_EN = 1. It must be cleared (=0), otherwise. Because Macronix MX25 octal devices do not guarantee the correctness of CRC calculated for unaligned chunk, the controller provides control of this check. When configured, DATA_NO_OF_BYTES accommodates many CRC chunks; the CRC correctness is checked for all chunk-aligned CRC slices regardless of the CRC_UAL_CHUNK_CHK value.
Reserved	114:113	N/A
Data Swap	112	When set (=1), this bit swaps the order of bytes transferred in a single octal DDR clock cycle. When DATA_EDGE_MODE and DATA_NO_OF_IOS do not reflect the octal DDR mode, this bit should be cleared (=0).
Reserved	111	N/A
Bank Number (BANK_NUM)	110:108	This field informs the controller on which memory device the sequence should be executed. This value directly maps to the chip select (CS) pin.
Data Edge Mode (DATA_EDGE_MODE)	107	This bit selects SDR or DDR operation at the memory device 0 – SDR 1 – DDR
Reserved	106	N/A

Table 21-24: Data Instruction Structure (Continued)

Name	Bits	Description
Data IO Mode (DATA_NO_OF_IOS)	105:104	Determines number of active IOs for all data bytes of the instruction Data bytes associated with the instruction are taken from an internal data FIFO (for write operations) and put into the internal data FIFO (for read operations) 0 – Single 1 – Dual 2 – Quad 3 – Octal
Reserved	103:101	N/A
Transfer Direction (DIR)	100	Determines the direction of the data stream 0 – Read from memory device 1 – Write to memory device
Reserved	99:93	N/A
Data Size per Flash Address Location (DATA_PER_ADDR)	92	Indicates how many bytes of data are assigned to a single flash memory address. The controller uses this information to internally calculate consecutive addresses of all data bytes in a sequence. The address incrementation step is assumed to be 1. 0 – The memory device is byte addressable 1 – The memory device is two byte addressable
Reserved	91:90	N/A
Number of dummy (latency) cycles (NO_OF_DUMMY)	89:84	Indicates the number of dummy cycles before the data phase Note that the definition of latency cycles for legacy HyperFlash, and xSPI profile 2 devices overlaps with the address transfer phase by one clock cycle. The number of required latency cycles is one less than programmed. If an external board delay incurs a few more clock cycle long additional latency (for read commands), compensation is handled by the PHY module.
HyperFlash Read Crossing Boundary Enable (HF_READ_BOUND_EN)	83	The controller uses this bit to calculate a read transaction crossing a page boundary phase in which data is not valid (based on the number of dummy cycles (NO_OF_DUMMY) and the start address of the associated instruction). During the latency period, the device does not generate DQS.
Reserved	82:80	N/A

Table 21-24: Data Instruction Structure (Continued)

Name	Bits	Description
Number of data bytes (DATA_NO_OF_BYTES)	79:48	Indicates the number of data bytes to be sent in the instruction. All data bytes are taken from the data FIFO of the controller (for write operations) and put into the data FIFO (for read operations). When DATA_EDGE_MODE and DATA_NO_OF_IOS configuration indicates the octal DDR variant, two bytes are required to be sent in a single flash clock cycle. Therefore, the host must configure an even number of data bytes. Failure to do this causes the controller to generate an error.
Reserved	47:25	N/A
Status Data Source (STATUS_SOURCE)	24	The bit is valid only when DIR = 0 (read transaction) and there are ≤2 bytes to be read. When there are >2 bytes to be read, this bit should be zero. This bit determines how the host eventually receives the data. If there are ≤ 2 bytes to be read, the host can choose to have the two bytes reported in the STIG status register at offset 0x44. Otherwise, the read data must be fetched using the bus completer AXI interface. 0 – Read data is sent to the AXI bus completer interface 1 – Read data is sent to the STIG status register at offset 0x44
Reserved	23:7	N/A
Instruction Type	6:0	This value is always fixed at 127 for the glued data instruction.

Instruction Type Variants

The *Typical Device Operations* table shows typical operations and the associated value in the instruction type field that the host can carry out on the device using instruction variant 1. The table also outlines whether an instruction supports instruction gluing.

NOTE: If a data instruction is glued to the back of one of the instructions, the data instruction must have an instruction type field configured to 128.

Table 21-25: Typical Device Operations - Variant 1

Operation	Instruction Type	Assumed Instruction Gluing
Read	1	Yes – Glue with data instruction
Read In XIP	1	Yes – Glue with data instruction
Write	0	Yes, if data length is greater than two bytes
Set WEL	0	No
Clear WEL	0	No

Table 21-25: Typical Device Operations - Variant 1 (Continued)

Operation	Instruction Type	Assumed Instruction Gluing
Erase Sector	0	No
Chip Erase	0	No
Program Suspend	0	No
Program Resume	0	No
Read Device Register	1	No
Write Device Register	0	No
Setup Read Burst	0	No
Clear Flag Register	0	No
Enter Deep Powerdown	0	No
Exit Deep Powerdown	0	No
Reset Enable	0	No
Soft Reset	2	No
Read JEDEC ID	1	No
Read SFDP	1	Yes – Glue with data instruction
Enter Default Protocol Mode	0	No
Enter SDR Octal	0	No
Global Protection	0	No
Global Unprotection	0	No
Protect Sector	0	No
Unprotect Sector	0	No
Enter 4-Byte Address Mode	0	No
Exit 4-Byte Address Mode	0	No
SPI_NAND Page Read	0	Yes – Glue with data instruction
SPI_NAND Read From Cache	0	Yes – Glue with data instruction
SPI_NAND Page Read Page Cache Random	0	No
SPI_NAND Page Read Page Cache Last	0	No
SPI_NAND Program Load	0	Yes, if data length is greater than two bytes
SPI_NAND Program Load Random	0	Yes, if data length is greater than two bytes
SPI_NAND Program Execute	0	Yes, if data length is greater than two bytes
SPI_NAND Set Feature	0	Yes, if data length is greater than two bytes
SPI_NAND Get Feature	1	Yes – Glue with data instruction

Table 21-25: Typical Device Operations - Variant 1 (Continued)

Operation	Instruction Type	Assumed Instruction Gluing
SPI_NAND Read ID	32	Yes – Glue with data or variant instruction

The *Typical Device Operations* table shows typical operations and the associated value in the instruction type field that the host can carry out on the device using instruction variant 2. The table outlines some of the instruction fields that must be forced to a fixed value to carry out the operation, and whether an instruction supports instruction gluing.

The variant should be used when trying to formulate a generic or custom command sequence that is not a known protocol-specific instruction. For this instruction, select an instruction type field of 96. This instruction is permitted to be glued with another variant 3 instruction or with a data sequence instruction.

Table 21-26: Typical Device Operations - Variant 2

Operation	Instruction Type	Assumed Instruction Gluing
Write/Read for HyperRAM Devices	32	Yes – Glue with data or variant instruction
WREN 1	33	No
WREN 2	34	No
Profile 2 Read	35	Yes – Glue with data instruction
Profile 2 Program	36	Yes, if the data length is greater than two bytes. Can glue with data or another variant
Erase Sector	37	No
Chip Erase	38	No
Erase Suspend	39	No
Erase Resume	40	No
Program Suspend	41	No
Program Resume	42	No
Read Status	43	Yes – Glue with data instruction
Read Volatile Memory Register	44	Yes – Glue with data instruction
Read Non-Volatile Memory Register	45	Yes – Glue with data instruction
Read Customer Memory Register	47	Yes – Glue with data instruction
Status Register Clear	48	No
Write Volatile Memory Register	49	No
Write Non-Volatile Memory Register Enable	50	No
Write Custom Memory Register Enable	51	Yes – Complete with another variant 2 instruction with instruction type = 32
Enter Deep Powerdown	52	No

Table 21-26: Typical Device Operations - Variant 2 (Continued)

Operation	Instruction Type	Assumed Instruction Gluing
Reset/ASO Exit	53	No
Enter Default Protocol Mode	54	No
Read SFDP	55	Yes – Glue with data instruction

Switching Work Modes

STIG work mode provides an interface to execute low-level transactions on the xSPI flash interface. It is expected to be used in conjunction with ACMD or direct work mode. Therefore, switching into and out of STIG mode can become a common requirement. The host software must enter and exit STIG mode when the target xSPI flash device is in the idle state.

Before the host switches to STIG mode, it must wait for all operations that address the same xSPI flash device to be finished. Before the host exits STIG mode, it must check whether the device is in an idle state. Waiting for the last STIG command to complete execution may not be enough for cases where the command caused the xSPI device go into a busy state. In such cases, the host can poll the status register of the device. It can issue STIG instructions to execute read status commands and continue to do this until the memory device has reached an idle state. Once the device is in the idle state, the host can exit STIG mode.

Programming STIG Mode

Complete the following steps to program STIG mode.

1. Wait for xSPI flash controller to be in the idle state, then, poll the `XSPI_GSTAT.CTL_BUSY` bit.
2. Configure the `XSPI_WORKMODE_CTL` field to 2'b01 to enable STIG work mode.
3. Configure the command registers (`XSPI_CMD1` to `XSPI_CMD4`) .
4. Enable the STIG DONE interrupt when completion of the requested STIG transaction must raise external interrupt or sdma_trigg for the AXI bus completer interface transactions. Set the `XSPI_INT_EN.STIG_DONE_EN`, `XSPI_INT_EN.SDMA_TRIGG_EN`, and `XSPI_INT_EN.GINT_EN` bits to enable these interrupts.
5. Start the STIG operation by writing to the `XSPI_CMD0` register. Set `XSPI_GSTAT.GCMD_ENG_MC_BUSY` = 1.
6. Based on the instruction type field, the host must select one of the following:
   - Gluing with a data sequence, proceed to the next step.
   - For other instructions without the gluing condition, skip to step 9.
   - Otherwise, return to configuring the command register step.
7. When the instruction that contains the data sequence command indicates a read direction, and the number of data to be read is one or two bytes and the STATUS_SOURCE field in the instruction =1, then skip to step 9.

In all other cases, proceed to the next step.

8. The host transfers data through the AXI host interface of the controller to complete the STIG operation. In this step, XSPI_GSTAT.GCMD_ENG_MC_BUSY =1.
   a. Poll the XSPI_ISTAT.SDMA_TRIGG bit, or alternatively wait for the interrupt if the SDMA_TRIGG interrupt was unmasked. When XSPI_ISTAT.SDMA_TRIGG =1, proceed.
   b. Read the XSPI_SDMA_SIZ and XSPI_SDMA_TRD_STAT registers to determine the number of bytes to transfer, the thread ID and transfer direction.
   c. Clear the XSPI_ISTAT.SDMA_TRIGG flag.
   d. The external host can now transfer data. For program operations, it is expected that the external host will issue one or more AXI write requests with a total size matching the data transfer indicated in step b. For read operations, it is expected that the external host will issue one or more AXI read requests with a total size matching the data transfer indicated in step b. The external host should issue an address that matches the one reported in the XSPI_SDMA_ADDR0 and XSPI_SDMA_ADDR1 registers.
   e. If the host ignores any of the requirements described above and starts the data transfer when the host DMA is not ready, the XSPI_ISTAT.SDMA_ERR flag is set. If XSPI_INT_EN.SDMA_ERR_EN = 1, an interrupt is also triggered. When XSPI_DMA_CTL.SDMA_ERR_RSP =1, an error response is returned. When XSPI_DMA_CTL.SDMA_ERR_RSP =0, an OK response is returned.
   f. If the host sends an unsupported transaction to the bus completer interface, the bus completer DMA ignores the access and the SDMA_ERR flag in the XSPI_ISTAT register is set. If an AXI error response is detected on the system bus during host DMA transfer, the fail and bus error bits are set in the last operation status. The last operation status can be checked in the XSPI_CMD_STAT register.
   g. Performing transactions in wrong direction (that is, a write transaction to host DMA port in case of sending read operation command to xSPI device) is forbidden and causes the XSPI_ISTAT.SDMA_ERR flag to be set.When XSPI_INT_EN.SDMA_ERR_EN = 1, an interrupt is also triggered. When XSPI_DMA_CTL.SDMA_ERR_RSP =1, an error response is returned. When XSPI_DMA_CTL.SDMA_ERR_RSP =0, an OK response is returned.
9. Wait for the STIG transaction completion status. Poll the COMPLETE bit in the XSPI_CMD_STAT register or view the status through the external interrupt from the STIG_DONE (when correctly enabled by the host). In this step, XSPI_GSTAT.GCMD_ENG_BUSY =1.
10. Check the STIG status by reading the XSPI_CMD_STAT register. In this step, the generic command engine busy flags are programmed to a low value.
11. If another STIG operation must be executed, go to step 3. Otherwise, if the work mode must be changed, wait for the controller idle state by polling the XSPI_GSTAT.CTL_BUSY bit.

STIG Mode Status

The host can check the latest STIG operation status by reading the XSPI_CMD_STAT register.

After every new operation is triggered, the COMPLETE bit is cleared and other bits in the status register are not valid. When the operation completes, the COMPLETE bit is set and other bits in the status register become valid.

Table 21-27: STIG Mode XSPI_CMD_STAT Bit Descriptions

Name	Bits	Description
DATA_FROM_DEV	31:16	This field holds one or two bytes read from memory device. Fields of the DATA_SEQ instruction must be configured as follow: DATA_NO_OF_BYTES = 1 – one byte, or 2 – two bytes STATUS_SOURCE = 1. In other cases, data from the flash device is available on the bus completer data interface.
COMPLETE	15	When set, it denotes that the controller has updated status information and the operation is complete. This bit is set even when the operation ended as a failure.
FAIL	14	When set, it denotes that the operation failed to complete successfully.
Reserved	13:4	N/A
DQS_ERROR	3	The bit indicates that an incorrect DQS pulses number was detected during the data read operation. This is information from the PHY that either the DQS strobe did not appear during read (for example, the device is not connected to the controller) or the RD_DEL_SEL signal value is incorrect (field in the XSPI_PHY_GATE_LPBK_CTL) and data read from flash device are corrupted and read FIFO pointers in the PHY are misaligned. The DLL RESET or RESET signals clear the underrun/overflow flags in the PHY and clear all PHY read data pointers. One of these flags is required by the PHY before continuing to work after a DQS_ERROR assertion.
CRC_ERROR	2	This bit indicates that a CRC error has been received
BUS_ERROR	1	When set, it indicates that the controller received an error response on the system DMA interface.
CMD_ERROR	0	This bit indicates that an invalid command sequence has been detected.

STIG Limitations

STIG mode has the following limitations:

- The HOLD# functionality (alternative function of DQ3 pin of some legacy SPI devices) is not supported in STIG mode. Since STIG mode is fully software controlled, the HOLD# function is not particularly useful. Transfer management can instead be handled by higher-level layers (system layers) throttling when STIG transfers are triggered.
- The xSPI flash controller does not support wrap data flash transfer commands (all data to/from device are considered linear).
- The maximum number of bytes that can be read when using a generic STIG instruction type is limited to 64 bytes. Similarly, a maximum of 64 bytes can be read for all legacy HyperFlash and xSPI profile 2 instructions other than the READ_PROFILE_2 and READ_SFDP_P2.
- If XIP mode is used, the read operation that is used to either enter or exit XIP mode should not attempt to transfer more than 64B.

Sequencing STIG Operation

Internally, the STIG instructions are held in the main data FIFO. Based on the instruction type, glued instructions are sent and buffered in a second-level internal cache FIFO. This cache FIFO can accommodate up to 16 glued sequences (if more glued sequences are requested, an incorrect sequence error is generated). The instructions are processed together when the INSTR_LINK field = 0 of the last instruction in the instruction chain.

Internally, the controller translates the instructions and breaks them down into the correct sequence required by the memory device based on the type of the command. The general structure of xSPI frame is shown in the ***STIG Frame Structure*** diagram.

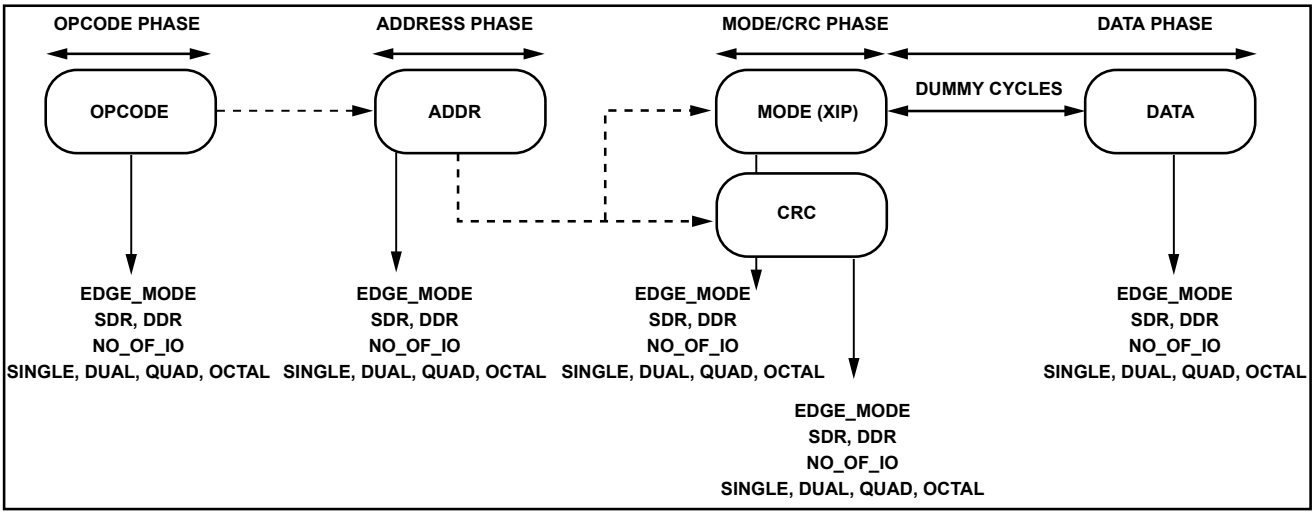


Figure 21-5: STIG Frame Structure

The blocks correspond to xSPI transfer phases. Depending on the individual command layout , some phases may be dropped; however, the order is always maintained.(Refer to JESD216 or a particular xSPI device specification). The EDGE_MODE and NO_Of_IOS can also differ in between phases. Per JESD216, only incremental increases over the number of IOs used or number of clock edges used is permitted. For example, the (opcode=1S, addr=8D, data=8D) transfer variant may be supported by current devices, but, a variant like (opcode=1S, addr=8D, data=4S)

is not. JESD216 does not allow for transfers with a number of active IO lines that differs from one another more than once. Also, note that if any of the phases are configured to be 8D and the associated NO_OF_BYTES corresponds to an odd value, the controller generates an error and does not accept the instruction. The 8D variant always requires an even number of bytes. The flexible command sequence structure of the controller allows any configuration of EDGE_MODE and NO_OF_IOS that provides a future proven solution for new variants of future revisions of JESD216.

Event Handling

The controller reports errors slightly differently depending on the selected work mode:

- For STIG or ACMD PIO work modes, the status is reported using the `XSPI_CMD_STAT` register at offset 0x44.
- For ACMD CDMA work mode, the status is reported using the status field of descriptor associated with a specified thread and the `XSPI_CMD_STAT` register.
- For direct work mode - the DIR_* bits in the `XSPI_ISTAT` register reflect the status.

The ***Error Descriptions*** table identifies the errors that can occur.

Table 21-28: Error Descriptions

Error	Description
DQS Error	This type of error is reported when the attached PHY detects a DQS underrun/overflow situation. When the DQS error is detected then data stream transferred from the xSPI flash memory can be corrupted. The host should assert DLL_RST_N or RST_N to clear DQS underrun/overflow flags in the PHY and clear all PHY read data pointers.
System Bus Error	For ACMD or STIG work modes, for the AXI bus completer interface, this error is reported when: 1. The host attempts to access the bus completer interface before it is permitted (refer to the required steps in STIG and ACMD sections of this document). 2. The host issues an unsupported burst type. The controller only supports incremental (non-wrapping) bursts. For the AXI bus requester interface (not applicable for STIG or direct mode) the error may occur when an AXI error response is returned on RRESP or BRESP.

Table 21-28: Error Descriptions (Continued)

Error	Description
CMD Error	STIG mode: This error occurs when the specific STIG sequence as defined in this document is not followed properly. To rectify, review the sequence carefully and repeat. Direct mode: In this mode, the DIR_CMD_ERR interrupt is triggered when the host has issued an AXI transaction that cannot be executed by the controller for one of the following reasons. • The command extension is not enabled when the opcode phase reflects octal DDR mode. • The number of address bytes is not even when the address phase reflects octal DDR mode. • The address value is not even when the address phase reflects octal DDR mode. • The number of data bytes is not even when the data phase reflects octal DDR mode, unless RWDS byte masking functionality is enabled through the direct access configuration register (only available for direct work modes). • The number of bytes for the status checking sequence is not even when data phase reflects octal DDR mode. • Device and controller are configured to work in XIP mode but, the write access was made from the host. • Device and controller are configured to work in XIP mode but, the sending of mode byte (enable/disable) is disabled. • The DATA_PER_ADDR parameter is set and transfer address or transfer size is not even, unless RWDS byte masking functionality is enabled through the direct access configuration register (only available for direct work modes). • Word size (sAWSIZE or sARSIZE) is greater than the selected page size in profile 1 or profile 2 - HF. ACMD mode: Conditions which cause the CMD_ERROR being triggered are described in the section, Rules and Limitations.
CRC Error	When CRC checking is enabled, this error is reported when the controller has detected a CRC error.
Device Error	A device error is reported when the attached memory device returned fail after checking the status of the last erase/program/read operation. It is applicable only for ACMD and direct work modes. The status of a read operation is only checked for SPI NAND devi-

Table 21-28: Error Descriptions (Continued)

Error	Description
	ces. Detecting an error in this case means that the device returned an uncorrectable ECC error.
ECC Error	An ECC correctable error is reported when the attached memory device returns a correctable error during a read operation and when the XSPI_STAT_SEQ_CFG10.ECC_FAIL_EN field is set in the register.

Device Discovery

Device discovery (DD) is used to identify the memory device and pre-configure controller registers to the required state automatically following power on reset. Automated device discovery is implemented in the controller using the SFDP as defined in JEDEC standard JESD216. The SFDP standard provides a consistent method of describing the functional and feature capabilities of serial flash devices in a standard set of parameter tables. The SFDP must be implemented in the attached memory device for discovery to be successful.

Device discovery can be triggered in two ways — automatically, following power on reset (by default, this always occurs, unless the controller input discovery_inhibit is set (=1), or manually using the discovery control XSPI_DISCOVERY_GCTL register.

Discovery Modes

Device discovery can automatically identify the memory device at power on reset and initialize many of the controller registers. However, the user can alternatively choose to trigger device discovery manually. There are three modes of operation for device discovery: full automatic, preconfigured SFDP, and preconfigured non-SFDP.

Full Automatic Mode Detection - In this default mode, the controller must first determine how to communicate with an unknown device. Per JESD216, if the memory device supports device discovery, it must also support the read SFDP command. The controller, therefore, attempts to communicate by sending read SFDP commands in different transfer formats. Each time it waits for an indication that the device has interpreted the command successfully. When a valid SFDP header is recognized, the controller knows which transfer format has been successful. It proceeds automatically to fetch and decode the SFDP database residing within the device. As the database is fetched from the device, the controller updates the controller registers accordingly.

To trigger device discovery using this mode as part of power on reset initialization, ensure inputs XSPI_DISCOVERY_GCTL.INHIBIT = 0 and XSPI_DISCOVERY_GCTL.NUM_LINES = 0.

To trigger device discovery using this mode from the registers, write the following values to the XSPI_DISCOVERY_GCTL register.

- XSPI_DISCOVERY_GCTL.REQ_TYP = 0
- XSPI_DISCOVERY_GCTL.NUM_LINES = 0
- XSPI_DISCOVERY_GCTL.REQ = 1

When operating in full automatic mode detection, the initial querying of the memory device to identify the correct transfer mode is performed as part of the controller's initialization process. Once completed, the XSPI_DISCOVERY_GCTL register (at 0x260) is automatically updated.

Preconfigured SFDP mode - If the user already knows what memory device the controller is connected to and that SFDP is supported, the controller does not need to query the device with different transfer formats of the read SFDP command. Device discovery is triggered as in full automatic detection mode, but it bypasses the initial sequence to identify the required transfer mode. This can speed up the device identification process.

To trigger device discovery using this mode as part of power on reset initialization, ensure inputs XSPI_DISCOVERY_GCTL.INHIBIT = 0 and XSPI_DISCOVERY_GCTL.NUM_LINES !=0. To trigger device discovery using this mode from registers, write the following values to the XSPI_DISCOVERY_GCTL register:

- XSPI_DISCOVERY_GCTL.REQ_TYP = 0
- XSPI_DISCOVERY_GCTL.NUM_LINES != 0 (choose the value suited to the attached device, refer to the XSPI_DISCOVERY_GCTL register definition for details)
- XSPI_DISCOVERY_GCTL.REQ = 1

Preconfigured non-SFDP mode - In this mode, the controller does not attempt to fetch the SFDP database from the memory device. This alternative configuration method allows the controller to operate with a pre-known memory device. It is optional, and the user can alternatively program each controller register manually.

To trigger device discovery using this mode as part of power on reset initialization, ensure inputs XSPI_DISCOVERY_GCTL.INHIBIT = 1 and XSPI_DISCOVERY_GCTL.NUM_LINES !=0. To trigger device discovery using this mode from the registers, write the following values to the XSPI_DISCOVERY_GCTL register:

- XSPI_DISCOVERY_GCTL.REQ_TYP = 1
- XSPI_DISCOVERY_GCTL.NUM_LINES != 0 (choose the value suited to the attached device, refer to the XSPI_DISCOVERY_GCTL register definition for details)
- XSPI_DISCOVERY_GCTL.REQ = 1

Discovery Flow

The ***Discovery Flow Diagram*** shows the operations and decisions that are part of the device discovery process.If device discovery is inhibited, or no valid SFDP header or parameter page header is recognized during the flow, the SFDP database/parameter page cannot be obtained from the device and all controller registers retain the reset values.

The host must not trigger any operation while device discovery is in progress. The discovery process completion is marked by the controller setting the XSPI_DISCOVERY_GCTL.PASS bit. When full automatic mode is in use, discovery is complete when either the INIT_COMP pin is high, or XSPI_GSTAT.INIT_PASS =1.

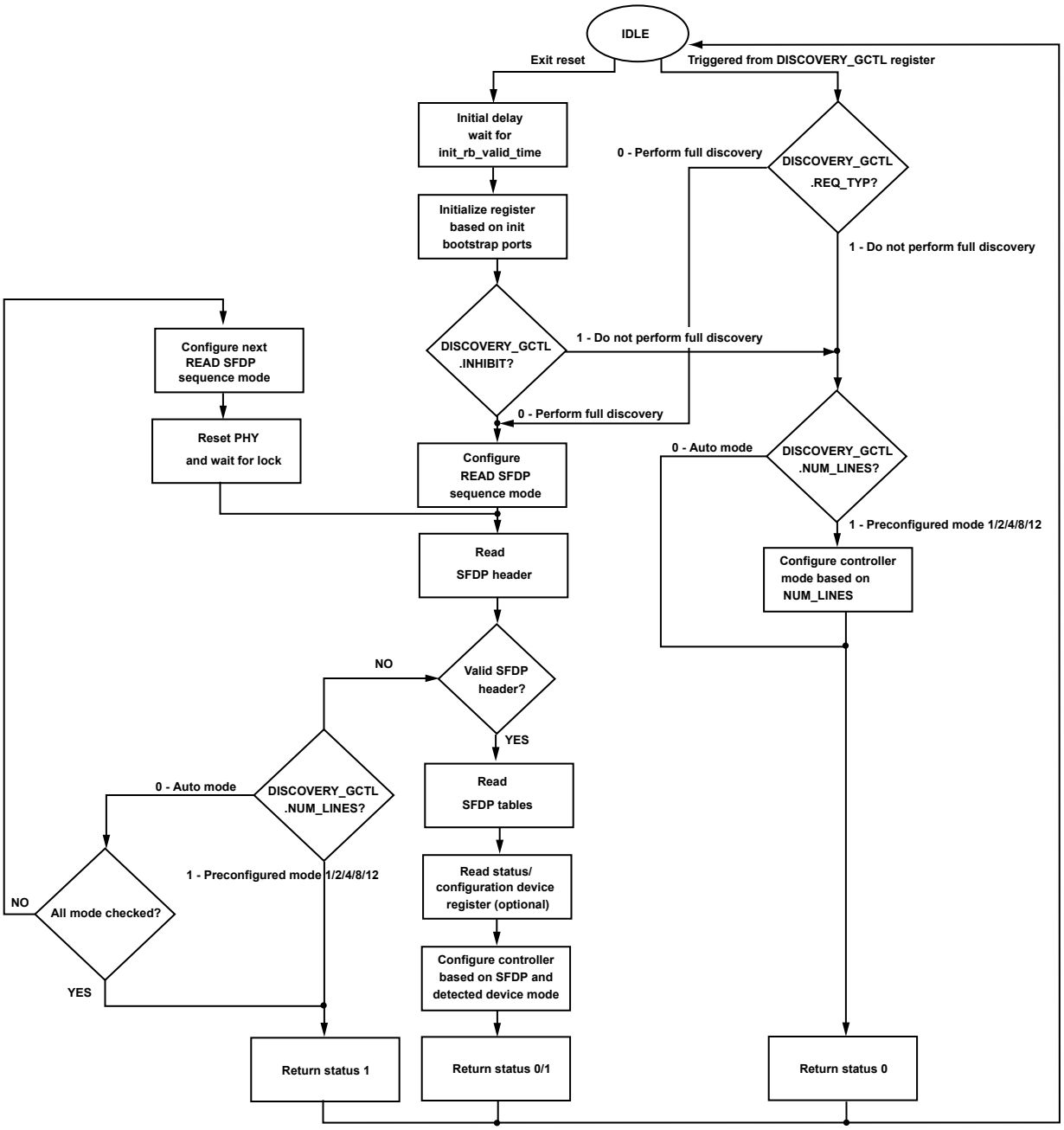


Figure 21-6: Device Discovery Flow

SFDP Command Structure

When full automatic mode detection is active, the controller must determine how to communicate with the device. It communicates by repeatedly sending read SFDP commands in different transfer formats. This mode is selected when `XSPI_DISCOVERY_GCTL.NUM_LINES` signals is configured to 4'd0. The algorithm that the controller executes differs from the suggested algorithm presented in the JESD216 specification document. The algorithm starts with single mode, goes through dual, quad, and octal modes, and finishes with the read SFDP command (compliant with legacy HyperFlash and xSPI profile 2.0).

The read SFDP commands are sent in the following order. If the device does not successfully interpret any of these steps, the device discovery is deemed to have failed.

1. 1S-1S-1S — opcode, address and data phases used for reading SFDP database are sent on one line in SDR mode. Command uses three address bytes and eight dummy clock cycles.
2. 1D-1D-1D — opcode, address and data phases used for reading SFDP database are sent on one line in DDR mode. Command uses three address bytes and eight dummy clock cycles.
3. 2S-2S-2S — opcode, address and data phases used for reading SFDP database are sent on two lines in SDR mode. Command uses three address bytes and eight dummy clock cycles.
4. 2D-2D-2D — opcode, address and data phases used for reading SFDP database are sent on two lines in DDR mode. Command uses three address bytes and eight dummy clock cycles.
5. 4S-4S-4S — opcode, address and data phases used for reading SFDP database are sent on four lines in SDR mode. Command uses three address bytes and eight dummy clock cycles.
6. 4D-4D-4D (Quad DDR* - presented as 4S-4D-4D in JESD216 specification) — opcode, address, and data phases used for reading SFDP database are sent on four lines. The opcode phase is sent in DDR mode with enabled command extension value. Address and data phases are sent in DDR mode. Command uses 3 address bytes and eight dummy clock cycles.
7. 4D-4D-4D (Quad DDR) — opcode, address and data phases used for reading SFDP database are sent on four lines in DDR mode. Command uses three address bytes and eight dummy clock cycles.
8. 8S-8S-8S — opcode, address and data phases used for reading SFDP database are sent on eight lines in SDR mode. Command uses three address bytes and eight dummy clock cycles.
9. 8S-8S-8S — opcode, address and data phases used for reading SFDP database are sent on eight lines in SDR mode. Command uses four address bytes and 20 dummy clock cycles. The opcode is sent twice and the second one is logical negation of the first one.
10. 8D-8D-8D — opcode, address and data phases used for reading SFDP database are sent on eight lines in DDR mode. Command uses four address bytes and 8 dummy clock cycles. The opcode extension is repetition of the opcode.
11. 8D-8D-8D — opcode, address and data phases used for reading SFDP database are sent on eight lines in DDR mode. Command sends the address over two clock cycles (meaning four bytes are send as this is octal DDR), but only three address bytes are valid. The opcode extension is repetition of the opcode.
12. 8D-8D-8D — opcode, address and data phases used for reading SFDP database are sent on eight lines in DDR mode. Command uses four address bytes and 20 dummy clock cycles. The opcode extension is logical negation of the opcode.
13. 8D-8D-8D — read SFDP command for legacy HyperFlash and xSPI profile 2.0 memory devices with 15 latency clock cycles. When preconfigured mode is selected, the above step is not required.

To use preconfigured mode in device discovery, configure the parameters of the read SFDP (READ_SFDP) instructions using `XSPI_DISCOVERY_GCTL` register fields shown in the following table.

Table 21-29: Read SFDP Command Variations - XSPI_DISCOVERY_GCTL Register

NUM_LINES	ABNUM	DMY_CNT	CMD_TYP[1]	CMD_TYP[0]	OE_EN	OE_VAL	Comment
1	N/A	N/A	N/A	0	N/A	N/A	Read SFDP command in 1S-1S-1S mode with three address bytes and eight dummy clock cycles.
1	N/A	N/A	N/A	1	N/A	N/A	Read SFDP command in 1D-1D-1D mode with three address bytes and eight dummy clock cycles.
2	N/A	N/A	N/A	0	N/A	N/A	Read SFDP command in 2S-2S-2S mode with three address bytes and eight dummy clock cycles.
2	N/A	N/A	N/A	1	N/A	N/A	Read SFDP command in 2D-2D-2D mode with three address bytes and eight dummy clock cycles.
4	N/A	N/A	0	0	N/A	N/A	Read SFDP command in 4S-4S-4S mode with three address bytes and eight dummy clock cycles.
4	N/A	N/A	1	0	N/A	N/A	Read SFDP command in 4D-4D-4D mode (quad DDR* - presented as 4S-4D-4D in

Table 21-29: Read SFDP Command Variations - XSPI_DISCOVERY_GCTL Register (Continued)

NUM_LINES	ABNUM	DMY_CNT	CMD_TYP[1]	CMD_TYP[0]	OE_EN	OE_VAL	Comment
							JESD216 specification) with three address bytes and with eight dummy clock cycles.
4	N/A	N/A	N/A	1	N/A	N/A	Read SFDP command in 4D-4D-4D mode (quad DDR) with three address bytes and with eight dummy clock cycles.
8	0/1	0/1	N/A	0	0	N/A	Read SFDP command in 8S-8S-8S mode with three or four address bytes and with eight or 20 dummy clock cycles. The opcode is sent one time.
8	0/1	0/1	N/A	0	1	1	Read SFDP command in 8S-8S-8S mode with three or four address bytes and with eight or 20 dummy clock cycles. The opcode is sent two times and the second (command extension) is a logical negation of the first one.
8	0/1	0/1	N/A	0	1	0	Read SFDP command in 8S-8S-8S mode

Table 21-29: Read SFDP Command Variations - XSPI_DISCOVERY_GCTL Register (Continued)

NUM_LINES	ABNUM	DMY_CNT	CMD_TYP[1]	CMD_TYP[0]	OE_EN	OE_VAL	Comment
							with three or four address bytes and with eight or 20 dummy clock cycles. The opcode is sent two times and the second (command extension) is a logical repetition of the first one.
8	0/1	0/1	N/A	1	N/A	0	Read SFDP command in 8D-8D-8D mode with four address bytes and with eight or 20 dummy clock cycles. The opcode extension is a repetition of the opcode. If three address bytes are selected, the controller sends the address over two clock cycles (meaning four bytes are sent due to octal DDR). But, only the first three address bytes are valid.
8	0/1	0/1	N/A	1	N/A	1	Read SFDP command in 8D-8D-8D mode with four address bytes and with eight or 20 dummy clock cycles. The opcode

Table 21-29: Read SFDP Command Variations - XSPI_DISCOVERY_GCTL Register (Continued)

NUM_LINES	ABNUM	DMY_CNT	CMD_TYP[1]	CMD_TYP[0]	OE_EN	OE_VAL	Comment
							extension is the logical negation of the opcode. If three address bytes are selected controller sends the address over two clock cycles (meaning four bytes are sent due to octal DDR), but only the first three address bytes are valid.

Table 21-29: Read SFDP Command Variations - XSPI_DISCOVERY_GCTL Register (Continued)

NUM_LINES	ABNUM	DMY_CNT	CMD_TYP[1]	CMD_TYP[0]	OE_EN	OE_VAL	Comment
12	hf_bound_en	latency_cnt[4]	latency_cnt[3]	latency_cnt[2]	latency_cnt[1]	latency_cnt[0]	Read SFDP command in 8D-8D-8D mode for xSPI profile 2.0 with variable number of latency clock cycles and enabled or disabled detection of the inter pages crossing latency clock cycles. This value should be configured to N-1, where N is the value expected by the memory device. If device expects 16 latency clock cycles then set 15 latency clock cycles as follow: DMY_CNT = 0 CMD_TYP = 3 OE_EN = 1 OE_VAL = 1
14	N/A	N/A	0/1	0/1	N/A	0	Read parameter page command in 1-1-1 mode for legacy SPI NAND device (at POR command is sent to device first). CMD_TYP field/port is used to set

Table 21-29: Read SFDP Command Variations - XSPI_DISCOVERY_GCTL Register (Continued)

NUM_LINES	ABNUM	DMY_CNT	CMD_TYP[1]	CMD_TYP[0]	OE_EN	OE_VAL	Comment
							XSPI_SEQ_GCTL1.PLANE_CNT parameter value.
14	N/A	N/A	0/1	0/1	N/A	1	Read parameter page command in 1-1-1 mode for legacy SPI NAND device (at PoR command is not send to device). CMD_TYP field/port is used to set XSPI_SEQ_GCTL1.PLANE_CNT parameter value.

SPI NAND Workflow

When connected to an SPI NAND memory device, the device discovery executes the following commands to read the parameter page database from the flash device:

1. Wait for MISCREG_XSPI_CTL1.RB_VLD_TM- this step is done only when device discovery is enabled at PoR. The RB_VALID_TIME is taken from the MISCREG_XSPI_CTL1 32-bit controller input.
2. Send 8'hFF reset command if MISCREG_XSPI_CTL0.EXTP_VL bootstrap port is 0, otherwise the reset command is not sent - this step is done only when device discovery is enabled at PoR.
3. Wait for the device ready state by reading the OIP bit in the flash device (get feature command 8'h0F with one address byte 8'hC0) - this step is done only when device discovery is enabled at PoR and if MISCREG_XSPI_CTL0.EXTP_VL - bootstrap port is 1'b0.
4. Read configuration register (get feature command 8'h0F with one address byte 8'hB0).
5. Configure bit 6th in the configuration register to enter parameter page read mode (configure feature command 8'h1F with one address byte 8'hB0).
6. Send read page 13h command with the block address of 24'h0001.
7. Wait for device ready state by reading the OIP bit in the flash device (get feature command 8'h0F with one address byte 8'hC0).
8. Send the read from cache 8'h0B command with the page address of 16'd0 and eight dummy clock cycles to fetch the parameter page signature.

9. Send the read from cache 8'h0B command with the page address of 16'd80 and eight dummy clock cycles to fetch the page size, page size extension and block size.
10. Restore the value in the configuration register read in step 3 (configure feature command 8'h1F with one address byte 8'hB0).
11. If a valid parameter page header was received, then configure the controller to SPI NAND mode.
12. Return status.

Discovery Status

When device discovery is performed as part of controller initialization, the status can be interpreted by looking at the INIT_FAIL controller output as follows.

NOTE: In the *Discovery Status Parameters* table, the controller's parameters related to CRC are updated to the values from the bootstrap ports (discovery_crc_*) except where invalid values of `XSPI_DISCOVERY_GCTL.NUM_LINES` are configured, or where the controller failed to detect the device accurately.

Table 21-30: Discovery Status Parameters

`XSPI_DISCOVERY_GCTL.INHIBIT`	**`XSPI_DISCOVERY_GCTL.NUM_LINES`**[3:0]	**Description** **(INIT_FAIL status is valid when INIT_PASS = 1)**
1	0	INIT_FAIL = 2'b00 – Device discovery is disabled Note that the PHY registers are still updated separately – see the section xSPI PHY Controller.
1	1, 2, 4, 8 or 12	INIT_FAIL = 2'b00 – The SFDP database was not read. The controller was, however, configured to the selected mode.
1	14	INIT_FAIL = 2'b00 – The SFDP database was not read. The controller was, however, configured to SPI NAND mode.
1	3, 5, 6, 7, 9, 10, 11, 13 or 15	INIT_FAIL = 2'b00 – The SFDP database was not read. Invalid configuration of `XSPI_DISCOVERY_GCTL.NUM_LINES`. The controller registers remain unchanged.
0	0	INIT_FAIL = 2'b00 – Device discovery enabled and detected an xSPI profile 1 device or profile 2 device during auto-detection mode. INIT_FAIL = 2'b10 – Device discovery enabled and detected a legacy SPI device during auto-detection mode INIT_FAIL = 2'b01 – Device discovery enabled but did not detect a device during auto-detection mode.

Table 21-30: Discovery Status Parameters (Continued)

XSPI_DISCOVERY_GCTL.INHIBIT	**XSPI_DISCOVERY_GCTL.NUM_LINES**[3:0]	**Description** **(INIT_FAIL status is valid when INIT_PASS = 1)**
0	1, 2, 4, 8 or 12	INIT_FAIL = 2'b00 – Device discovery enabled and detected an xSPI profile 1 device or profile 2 device during preconfigured detection mode. INIT_FAIL = 2'b10 – Device discovery enabled and detected a legacy SPI device during pre-configured detection mode. INIT_FAIL = 2'b01 – Device discovery enabled but did not detect a device during preconfigured-detection mode.
0	14	INIT_FAIL = 2'b00 – Device discovery enabled and detected a legacy SPI NAND device during pre-configured detection mode. INIT_FAIL = 2'b01 – Device discovery enabled but did not detect a legacy SPI NAND device during preconfigureddetection mode.
0	3, 5, 6, 7, 9, 10, 11, 13 or 15	INIT_FAIL = 2'b01 – Invalid configuration on XSPI_DISCOVERY_GCTL.NUM_LINES. The controller registers remain unchanged.

When device discovery is triggered from the registers, the same status can be interpreted using the XSPI_DISCOVERY_GCTL.FAIL field of the discovery control register.

XSPI_DISCOVERY_GCTL.INHIBIT	**XSPI_DISCOVERY_GCTL.NUM_LINES**[3:0]	**Description** **(XSPI_DISCOVERY_GCTL.FAIL** **status is valid when XSPI_DISCOVERY_GCTL.PASS** **= 1)**
0	0	XSPI_DISCOVERY_GCTL.FAIL = 2'b00 – Device discovery enabled and detected an xSPI profile 1 device or profile 2 device during auto-detection mode. XSPI_DISCOVERY_GCTL.FAIL = 2'b10 – Device discovery enabled and detected a legacy SPI device during auto-detection mode. XSPI_DISCOVERY_GCTL.FAIL = 2'b01 – Device discovery enabled but did not detect a device during auto-detection mode.

XSPI_DISCOVERY_GCTL.INHIBIT	XSPI_DISCOVERY_GCTL.NUM_LINES[3:0]	Description (XSPI_DISCOVERY_GCTL.FAIL status is valid when XSPI_DISCOVERY_GCTL.PASS = 1)
0	1, 2, 4, 8 or 12	XSPI_DISCOVERY_GCTL.FAIL = 2'b00 – Device discovery enabled and detected an xSPI profile 1 device or profile 2 device during preconfigured detection mode. XSPI_DISCOVERY_GCTL.FAIL = 2'b10 – Device discovery enabled and detected a legacy SPI device during pre-configured detection mode. XSPI_DISCOVERY_GCTL.FAIL = 2'b01 – Device discovery enabled but did not detect device during preconfigured detection mode. Presents how to configure device discovery module in preconfigured detection mode.
0	14	XSPI_DISCOVERY_GCTL.FAIL = 2'b00 – Device discovery enabled and detected a legacy SPI NAND device during pre-configured detection mode. XSPI_DISCOVERY_GCTL.FAIL = 2'b01 – Device discovery enabled but did not detect a legacy SPI NAND device during preconfigured detection mode.
0	3, 5, 6, 7, 9, 10, 11, 13 or 15	XSPI_DISCOVERY_GCTL.FAIL = 2'b01 – Invalid configuration of XSPI_DISCOVERY_GCTL.NUM_LINES. The controller registers remain unchanged.
1	1, 2, 4, 8 or 12	XSPI_DISCOVERY_GCTL.FAIL = 2'b00 – The SFDP database was not read. The controller is configured to the selected mode.
1	14	XSPI_DISCOVERY_GCTL.FAIL = 2'b00 – The SFDP database was not read. The controller is configured to SPI NAND mode.

Discovery Restrictions

As defined in the JESD216 standard, flash devices can implement various parameter tables. But, only the JEDEC basic flash parameter table is mandatory. The following restrictions apply to xSPI profile 1 and profile 2 memory devices with a SFDP revision number is greater than or equal to 16'h0107:

- For xSPI profile 1 memory devices that are configured to any mode other than single, the SFDP database must implement the SCCR (status, control and configuration register) map for the xSPI memory devices parameter table. This implementation allows the controller to update its registers accurately. If the device does not implement the map fully, some of the sequence-related registers of the controller can get commonly used values instead.

- For xSPI profile 2 memory device, the SFDP database must implement the SCCR map for xSPI profile 2 memory devices parameter table. This implementation allows the controller to update its registers accurately. If the device does not implement the map fully, some of the sequence-related registers of the controller can get commonly used values instead.

Discovery Limitations

Note the following device discovery limitations:

- The device discovery module must not be used to detect a device that is configured to operate in XIP mode.
- The device discovery module must not be used to detect the HyperRAM device. Only profile 1, profile 2 - HF and SPI NAND devices are supported for device discovery.

Legacy Devices

When the number of dummy clock cycles for a read command (READ_SEQ_PL_DUMMY_CNT) parameter cannot be read from the device status/configuration register, the controller configures a default value for single, dual, quad modes or 16/20 for octal modes. The device discovery module ends with a 2'b10 status, but, with the wrong number of dummy clock cycles. In a similar way, if the number of dummy clock cycles for a status checking command (READ_SEQ_PL_DEV_RDY_DUMMY_CNT) cannot be read from the from the SFDP database, the device discovery module configures a default value. See the section Controller Reconfiguration.

Device Specific Configuration

Macronix MX25 octal devices - To configure the controller to STR-OPI mode without trying to read SFDP database, device discovery must be run in not full discovery pre-configuration mode:

- `XSPI_DISCOVERY_GCTL.INHIBIT` = 1 or `XSPI_DISCOVERY_GCTL.REQ_TYP` = 1
- `XSPI_DISCOVERY_GCTL.NUM_LINES` = 8
- `XSPI_DISCOVERY_GCTL.ABNUM` = 1
- `XSPI_DISCOVERY_GCTL.DMY_CNT`= 1
- `XSPI_DISCOVERY_GCTL.CMD_TYP` = 0
- `XSPI_DISCOVERY_GCTL.OE_EN` = 1
- `XSPI_DISCOVERY_GCTL.OE_VAL` = 1.

To configure the controller to DTR-OPI mode without trying to read SFDP database, device discovery must be run in not full discovery pre-configuration mode:

- `XSPI_DISCOVERY_GCTL.INHIBIT` = 1 or `XSPI_DISCOVERY_GCTL.REQ_TYP`= 1
- `XSPI_DISCOVERY_GCTL.NUM_LINES` = 8
- `XSPI_DISCOVERY_GCTL.ABNUM` = 1
- `XSPI_DISCOVERY_GCTL.DMY_CNT` = 1

- XSPI_DISCOVERY_GCTL.CMD_TYP = 1
- XSPI_DISCOVERY_GCTL.OE_EN = 0 or 1
- XSPI_DISCOVERY_GCTL.OE_VAL = 1

Controller Reconfiguration

When device discovery completes successfully, the controller automatically updates various registers in the controller. The registers depend on the type of memory device attached. For profile 1 or SPI NAND devices, the registers used to define how the command (opcode), address, and data are transferred between the controller and device. The *Device Discovery Parameter* tables show the parameters in the device sequence registers that require updates.

Table 21-31: Device Discovery Parameters - Profile 1 or SPI NAND devices

Detected/Selected Mode	Command Phase (*cmd_ios, *cmd_edge, *cmd_ext_en, *cmd_ext_val)	Address Phase (*addr_ios, *addr_edge,*addr_cnt)	Data Phase (*data_ios, *data_edge)
Single SDR	0,0, 0, I	0*,0, 3(4)**	0*,0
Single DDR	0,1, 0, I	0*,1, 3(4)**	0*,1
Dual SDR	1,0, 0, I	1,0, 3(4)**	1,0
Dual	1,1, 0, I	1,1, 3(4)**	1,1
Quad SDR	2,0, 0, I	2,0, 3(4)**	2,0
Quad DDR	2,1, 0, I	2,1, 3(4)**	2,1
Quad DDR	2,1, 1, N	2,1, 3(4)**	2,1
Octal SDR without command extension	3,0, 0, I	3,0, 3(4)**	3,0
Octal SDR with command extension repeated	3,0, 1, R	3,0, 3(4)**	3,0
Octal SDR with command extension inverted	3, 0, 1, I	3, 0, 3(4)**	3, 0
Octal DDR with command extension repeated	3, 1, 1, R	3, 1, 4**	3, 1
Octal DDR with command extension inverted	3, 1, 1, I	3, 1, 4**	3, 1
SPI NAND	0, 0, 0, R	0, 0, 2	0, 0

NOTE: • * = single SDR/DDR mode, the number of IOs for address and data phases (for read and program commands) can, in many devices, be reprogrammed higher than what is shown in the table; but, the lower value is chosen as a result of device discovery.

- ** = the number of address bytes for read/program/erase sector commands is configured to four when:
  - the device operates with four address bytes, or
  - when SFDP indicates support for four address byte commands, or
  - when octal DDR mode is detected (in which case only an even number of address bytes can be transferred during the address phase).
  - In all other cases, the device discovery module configures the number of address bytes based on the number of address bytes that is used to detect the device.
- R = the command extension is repeated
- I = the command extension is inverted
- N = command value consists of the first half of the opcode (bits[7:4]) repeated twice, and command extension value consists of second half of the opcode (bits[3:0]) repeated twice. For example, if the attached memory requires 0x12 opcode for a program page operation, then PROG_SEQ_PL_CMD_VAL is configured to 8'h11 and PROG_SEQ_PL_CMD_EXT_VAL is configured to 8'h22. This mode is presented as 4S-4D-4D in the JESD216 specification.

Common Sequence Register Updates

For profile 1, profile 2 (HF) or SPI NAND devices, the ***Device Discovery Parameters*** table shows the common sequence parameters in the `XSPI_SEQ_GCTL0` and `XSPI_SEQ_GCTL1` registers. The registers are updated after discovery.

Table 21-32: Device Discovery Parameters - Profile 1, Profile 2(HF) or SPI NAND devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
TYP	• 0 when a legacy SPI or xSPI profile 1 device is detected • 1 when a legacy HyperFlash or xSPI profile 2 device is detected • 3 when a legacy SPI NAND device is detected.	• 0 for legacy SPI or xSPI profile 1 configuration • 1 for legacy HyperFlash or xSPI profile 2 configuration • 3 for SPI NAND configuration
DATA_PER_ADDR	• 0 when legacy SPI, xSPI profile 1 and SPI NAND devices are detected • 1 when a legacy HyperFlash or xSPI profile 2 device is detected	• 0 for legacy SPI, xSPI profile 1, or SPI NAND configuration • 1 for legacy HyperFlash or xSPI profile 2 configuration

Table 21-32: Device Discovery Parameters - Profile 1, Profile 2(HF) or SPI NAND devices (Continued)

Parameter Name	Full Discovery Process	Not Full Discovery Process
PSIZ_PGM	Value is read from the SFDP database or parameter page	4'd8 for legacy SPI or xSPI profile 1 configuration 4'd9 for legacy HyperFlash or xSPI profile 2 configuration 4'd11 for SPI NAND configuration if `XSPI_DISCOVERY_GCTL.ABNUM` bootstrap port/register field is 1'b0. Otherwise, 4'd12.
PSIZ_RD	For SPI NAND devices, the value is read from the parameter page. Otherwise, 4'd15 value is configured.	4'd15 for profile 1 and profile 2 - HF 4'd11 for SPI NAND configuration if `XSPI_DISCOVERY_GCTL.ABNUM` bootstrap port/register field is 1'b0. Otherwise, 4'd12.
DATA_SWAP	1 when the SFDP header is swapped in octal DDR mode. Otherwise, 0.	1 for legacy SPI or xSPI profile 1 when octal DDR mode is configured, number of address bytes is 4, and number of dummy cycles is 20 1 for legacy HyperFlash or xSPI profile 2 configuration. Otherwise, 0.
CRC_EN	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_EN after controller initialization process only	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_EN after controller initialization process only
CRC_VAR	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_VARIANT after controller initialization process only	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_VARIANT after controller initialization process only
CRC_OE	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_CHUNK after controller initialization process only	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_OE after controller initialization process only
CHUNK_SIZ	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_CHUNK_SIZE after controller initialization process only	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_CHUNK_SIZE after controller initialization process only
UAL_CHUNK_EN	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_UAL_CHUNK_EN after controller initialization process only	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_UAL_CHUNK_EN after controller initialization process only
UAL_CHUNK_CHK	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_UAL_CHUNK_CHK after controller initialization process only	Parameter is updated from bootstrap port signal DISCOVERY_SEQ_CRC_UAL_CHUNK_CHK after controller initialization process only

Table 21-32: Device Discovery Parameters - Profile 1, Profile 2(HF) or SPI NAND devices (Continued)

Parameter Name	Full Discovery Process	Not Full Discovery Process
PSIZ_EXT	For SPI NAND devices, the value is read from the parameter page	64 value is configured if XSPI_DISCOVERY_GCTL.DMY_CNT = 1'b0. Otherwise, 128.
PCA_SIZ	For SPI NAND devices when page size equals 4KB, 1'b1 value is set. Otherwise, 1'b0 value is configured.	Value is configured to the value of XSPI_DISCOVERY_GCTL.ABNUM field/ bootstrap port
PLANE_CNT	Value is configured to the value of XSPI_DISCOVERY_GCTL.CMD_TYP register field/bootstrap port	Value is configured to the value of XSPI_DISCOVERY_GCTL.CMD_TYP register field/bootstrap port
PPER_BLOCK	Value is read from the parameter page	6 (64 pages per block) value is configured
TCMS_EN	1'd0 value is configured	1'd0 value is configured

Reset Sequence Parameters for Profile 1 Devices

For profile 1 devices, the reset sequence registers are used when a reset command is issued to the memory device. The parameters are identified in the XSPI_RST_SEQ_CFG0 and XSPI_RST_SEQ_CFG1 registers. The registers are updated as shown in the ***Reset Sequence Parameters - Profile 1 Devices*** table.

Table 21-33: Reset Sequence Parameters - Profile 1 Devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_RST_SEQ_CFG0.CMD0_EN	0 when the device supports the soft reset command - 0xF0	Always 1
XSPI_RST_SEQ_CFG0.CMD0_EN	Always 8'h66 command is configured	Always 8'h66 command is configured
XSPI_RST_SEQ_CFG0.CMD1_VALUE	When the device supports soft reset command, 8'hF0. Otherwise, 8'h99 command is configured.	Always 8'h99 command is configured
XSPI_RST_SEQ_CFG1.P1_DAT_VALUE	Always 8'hD0 value is configured	Always 8'hD0 value is configured
XSPI_RST_SEQ_CFG0.DAT_EN	Always 0	Always 0
XSPI_ERS_SEQ_CFG0.CMD_VALUE	Read from the SFDP database	8'h20 command when XSPI_DISCOVERY_GCTL.ABNUM = 0, 8'h21 command when XSPI_DISCOVERY_GCTL.ABNUM = 1
XSPI_ERS_SEQ_CFG1.P1_SECT_SIZ_VALUE	Read from the SFDP database	Always 4'd12 (4kB)
XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_VALUE	Always 8'hC7 command is configured	Always 8'hC7 command is configured

Erase Sequence Register Updates

For profile 1 devices, the erase sequence registers are used when an erase_sector or erase_all command is issued to the memory device. The parameters are identified in the XSPI_ERS_SEQ_CFG0, XSPI_ERS_SEQ_CFG1, and XSPI_ERS_SEQ_CFG2 registers. The registers are updated as shown in the ***Erase_Sector / Erase_All Sequence Parameters*** table.

Table 21-34: Erase_Sector / Erase_All Sequence Parameters - Profile 1 Devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_ERS_SEQ_CFG0.CMD_VALUE	8'hD8 command is configured	8'hD8 command is configured
XSPI_ERS_SEQ_CFG1.P1_SECT_SIZ_VALUE	Calculated as follow: 2^N, where N=column address size + pages row address size	5'd18 value is configured
XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_VALUE	Always 8'hC7 command is configured	Always 8'hC7 command is configured

For SPI NAND devices, the erase sequence registers are used when an erase_sector command is issued to the memory device. The parameters are identified in the XSPI_ERS_SEQ_CFG0 and XSPI_ERS_SEQ_CFG1 registers. The registers are updated as shown in the ***Erase_Sector Sequence Parameters*** table.

Table 21-35: Erase_Sector Sequence Parameters - SPI NAND Devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_ERS_SEQ_CFG0.CMD_VALUE	8'hD8 command is configured	8'hD8 command is configured
XSPI_ERS_SEQ_CFG1.P1_SECT_SIZ_VALUE	Calculated as follow: 2^N, where N = column address size + pages row address size	5'd18 value is configured
XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_VALUE	Always 8'hC7 command is configured	Always 8'hC7 command is configured

Write Enable Sequence Register Updates

For profile 1 or SPI NAND devices, the write enable sequence parameters are identified in the XSPI_WE_SEQ_CFG0 register. The registers are used when the controller needs to prepare the device for a program operation. The ***Write Enable Sequence Parameters*** table shows the parameters for full and not full discovery.

Table 21-36: Write Enable Sequence Parameters - Profile 1 and SPI NAND devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_WE_SEQ_CFG0.P1_CMD_VALUE	Always 8'h06 command is configured	Always 8'h06 command is configured
XSPI_WE_SEQ_CFG0.P1_EN	Always1'b1 value is set	Always1'b1 value is set

Program Sequence Register Updates

For profile 1, profile 2 HyperFlash, or SPI NAND devices, the program sequence parameters are identified in the XSPI_PROG_SEQ_CFG0, XSPI_PROG_SEQ_CFG1, and XSPI_PROG_SEQ_CFG2 registers. The registers

used when the controller needs to perform a program (write) operation. The parameters are shown in the ***Program Sequence Parameters*** tables.

Table 21-37: Program Sequence Parameters - Profile 1

Detected/Selected Mode	Parameter Name	Full Discovery Process	Not Full Discovery Process
Single	XSPI_PROG_SEQ_CFG0.P1_CMD_VALUE	8'h8E command is configured (1-8-8 with 4 address bytes) when the SFDP database indicates support for this command by device 8'h84 command is configured (1-1-8 with 4 ad-dress bytes) when SFDP database indicates support for this command by device 8'h3E command is configured (1-4-4 with 4 address bytes) when SFDP database indicates indicates support for command by device 8'h34 command is configured (1-1-4 with 4 address bytes) when SFDP database indicates indicates support for this command by device 8'h12 command is configured (1-1-1 with 4 address bytes) when SFDP database indicates indicates support for this command by device Otherwise, 8'h02 command is configured (1-1-1 with 3 or 4 address bytes)	8'h02 command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 0 8'h12 command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 1
Single		6'd0 value is configured	6'd0 value is configured
Dual, Quad, Octal	XSPI_PROG_SEQ_CFG0.P1_CMD_VALUE	8'h12 command is configured when the SFDP database indicates support for this command by device Otherwise, 8'h02 command is configured	8'h02 command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 0 8'h12 command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 1
Dual, Quad, Octal	XSPI_PROG_SEQ_CFG0.P1_DMY_CNT	6'd0 value is configured	6'd0 value is configured

Table 21-38: Program Sequence Parameters - Profile 2 HF Devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_PROG_SEQ_CFG2.P2_MSK_CMD_MOD	Always 0 is configured	Always 0 is configured
XSPI_PROG_SEQ_CFG2.P2_BURST_TYP	Always 1 (linear burst) is set	Always 1 (linear burst) is set
XSPI_PROG_SEQ_CFG2.P2_TARGET	Always 0 (memory space) is configured	Always 0 (memory space) is configured

Table 21-38: Program Sequence Parameters - Profile 2 HF Devices (Continued)

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_PROG_SEQ_CFG2.P2_LATENCY_CNT	Parameter is not updated after discovery procedure since it is related with profile 2 - HR which is not supported by the device discovery module	Parameter is not updated after discovery procedure since it is related with profile 2 - HR which is not supported by the device discovery module

Table 21-39: Program Sequence Parameters - SPI NAND Devices

Parameter Name	Parameter Value
XSPI_PROG_SEQ_CFG0.P1_CMD_VALUE	8'h02 command is configured

Read Sequence Register Updates

For profile 1, profile HyperFlash, or SPI NAND devices, read sequence registers are identified in XSPI_READ_SEQ_CFG0, XSPI_READ_SEQ_CFG1 and XSPI_READ_SEQ_CFG2 registers. The registers used when the controller needs to perform a read operation. The parameters are shown in the ***Read Sequence Parameters*** table.

Table 21-40: Read Sequence Parameters - Profile 1 Devices

Detected / Selected Mode	Parameter Name	Full Discovery Process	Not Full Discovery Process
Single	P1_CMD_VALUE	When the SFDP database indicates supporting the command by device: 8'hCC command is configured (1-8-8 with 4 address bytes) 8'h7C command is configured (1-1-8 with 4 address bytes) 8'hEC command is configured (1-4-4 with 4 address bytes) 8'h6C command is configured (1-1-4 with 4 address bytes) 8'hBC command is configured (1-2-2 with 4 address bytes) 8'h3C command is configured (1-1-2 with 4 address bytes) 8'h0C command is configured (1-1-1 with 4 address bytes) Read command value for 1-8-8 mode with 3 or 4 address bytes is read from SFDP database and configured Read command value for 1-1-8 mode with 3 or 4 address bytes is read from SFDP database and configured Read command value for 1-4-4 mode with 3 or 4 address bytes is read from SFDP database and configured Read command value for 1-1-4 mode with 3 or 4 address bytes is read from SFDP database and configured Read command value for 1-2-2 mode with 3 or 4 address bytes is read from SFDP database and configured Read command value for 1-1-2 mode with 3 or 4 address bytes is read from SFDP database and configured Otherwise, 8'h0B command value (1-1-1 with 3 or 4 address bytes)	8'h03 command is configured when `XSPI_DISCOVERY_GCTL.ABNUM` = 0 and `XSPI_DISCOVERY_GCTL.DMY_CNT` = 0 8'h13 command is configured when `XSPI_DISCOVERY_GCTL.ABNUM` = 1 and `XSPI_DISCOVERY_GCTL.DMY_CNT` = 0 8'h0B command is configured when `XSPI_DISCOVERY_GCTL.ABNUM` = 0 and `XSPI_DISCOVERY_GCTL.DMY_CNT` = 1 8'h0C command is configured when `XSPI_DISCOVERY_GCTL.ABNUM` = 1 and `XSPI_DISCOVERY_GCTL.DMY_CNT` = 1
Single	P1_DMY_CNT	Number of dummy clock cycles is read from the SFDP database. For 8'0B and 8'h0C read commands 8 number of dummy cycles are set.	0 dummy clock cycles are configured when `XSPI_DISCOVERY_GCTL.DMY_CNT` = 0, 8 dummy clock cycles are configured when `XSPI_DISCOVERY_GCTL.DMY_CNT` = 1.

Table 21-40: Read Sequence Parameters - Profile 1 Devices (Continued)

Detected / Selected Mode	Parameter Name	Full Discovery Process	Not Full Discovery Process
Dual	P1_CMD_VALUE	Read command is read from the SFDP database.	8'h0B command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 0 8'h0C command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 1
Dual	P1_DMY_CNT	If possible, number of dummy clock cycles is read from the memory configuration/status register. Otherwise, the number of dummy clock cycles is read from the SFDP database.	8 dummy clock cycles are configured when XSPI_DISCOVERY_GCTL.DMY_CNT = 0 10 dummy clock cycles are configured when XSPI_DISCOVERY_GCTL.DMY_CNT = 1
Quad	P1_CMD_VALUE	Read command is read from the SFDP database.	8'h0B command is configured when XSPI_DISCOVERY_GCTL.ABNUM= 0, 8'h0C command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 1.
Quad	P1_DMY_CNT	If possible, the number of dummy clock cycles is read from the memory configuration/status register. Otherwise, the number of dummy clock cycles is read from the SFDP database.	8 dummy clock cycles are configured when XSPI_DISCOVERY_GCTL.DMY_CNT = 0
Octal	P1_CMD_VALUE	8'h0C command is sconfigured when the SFDP database indicates supporting this command by device. Otherwise, 8'h0B command is configured	8'hEC command is configured when XSPI_DISCOVERY_GCTL.ABNUM= 1 XSPI_DISCOVERY_GCTL.DMY_CNT = 1, XSPI_DISCOVERY_GCTL.CMD_TYP[0] = 0 and XSPI_DISCOVERY_GCTL.OE_EN = 1 8'h EE command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 1, XSPI_DISCOVERY_GCTL.DMY_CNT = 1 XSPI_DISCOVERY_GCTL.CMD_TYP[0] =1 8'h 0B command is configured when XSPI_DISCOVERY_GCTL.ABNUM = 0 8'h0C command is configured when XSPI_DISCOVERY_GCTL.ABNUM= 1

Table 21-40: Read Sequence Parameters - Profile 1 Devices (Continued)

Detected / Selected Mode	Parameter Name	Full Discovery Process	Not Full Discovery Process
Octal	P1_DMY_CNT	If possible, number of dummy clock cycles is read from the memory configuration/status register. Otherwise, 16 dummy clock cycles are configured when 8 dummy clock cycles have been used for device detection, or, 20 dummy clock cycles are configured when 20 dummy clock cycles have been used for device detection	16 dummy clock cycles are configured when `XSPI_DISCOVERY_GCTL.DMY_CNT` = 0 20 dummy clock cycles is configured when `XSPI_DISCOVERY_GCTL.DMY_CNT` = 1

Table 21-41: Read Sequence Parameters - Profile 1 Devices (mode byte)

Parameter Name	Full Discovery Process	Not Full Discovery Process
`XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT`	If possible, (number of clock cycles to send the mode byte (mb_factor) is not greater than `XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT`), the value equals the P1_MB_DMY_CNT - mb_factor. Otherwise, 0 is configured. mb_factor = 2*(3 — P1_ADDR_IOS— P1_ADDR_EDGE)	Value is computed in the same way as for full discovery process
`XSPI_READ_SEQ_CFG1.P1_MB_EN`	If the basic flash parameter table indicates support for mode byte for the selected read command, then 1. Otherwise, 0.	Always 1'b0
`XSPI_XIP_GCTL.XIP_DIS_MB_VALUE`	Always 8'hFF value	Always 8'hFF value

*When the address phase is in octal DDR mode, then mb_factor = 1.

Table 21-42: Read Sequence Parameters - Profile 2 HF Devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
`XSPI_READ_SEQ_CFG2.P2_LATENCY_CNT`	If possible, the number of latency clock cycles is read from the device status/configuration register. Otherwise, the number of latency clock cycles is configured to the number that is used to read the SFDP database.	Number of latency clock cycles is configured to number that is used to read the SFDP database

Table 21-42: Read Sequence Parameters - Profile 2 HF Devices (Continued)

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_READ_SEQ_CFG2.P2_HF_BOUND_EN	Value is equal to the XSPI_DISCOVERY_GCTL.ABNUM for preconfigured mode, or, 1 for auto-detection mode	Value is equal to the XSPI_DISCOVERY_GCTL.ABNUM
XSPI_READ_SEQ_CFG2.P2_MSK_CMD_MOD	Always 0 is configured	Always 0 is configured
XSPI_READ_SEQ_CFG2.P2_BURST_TYP	Always 1 (linear burst) is configured	Always 1 (linear burst) is configured
XSPI_READ_SEQ_CFG2.P2_TARGET	Always 0 (memory space) is configured	Always 0 (memory space) is configured

Table 21-43: Read Sequence Parameters - SPI NAND Devices

Parameter Name	Parameter Value
XSPI_READ_SEQ_CFG0.P1_CMD_VALUE	8'h0B command is configured
XSPI_READ_SEQ_CFG0.P1_DMY_CNT	8 dummy clock cycles are configured

Table 21-44: Read Sequence Parameters for SPI-NAND Devices (mode byte)

Parameter Name	Parameter Value
XSPI_READ_SEQ_CFG1.P1_MB_EN	0
XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT	0 dummy clock cycles are configured
XSPI_XIP_GCTL.XIP_DIS_MB_VALUE	Always 8'hFF value
XSPI_READ_SEQ_CFG1.P1_CACHE_RANDOM_READ_EN	0

Status Sequence Register Updates

For profile 1, profile HyperFlash, or NAND devices, the status sequence registers are identified in the XSPI_STAT_SEQ _CFG register. The registers are used when the controller needs to check the status of the memory device. The parameters are shown in the ***Status Sequence Parameters*** table.

Table 21-45: Status Sequence Parameters - Profile 1 Devices

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_ERS_SEQ_CFG0.ADDR_CNT	When possible, the number of address bytes is read from the SFDP database. Otherwise, 2 address bytes are configured for octal DDR mode and 1 address byte is configured.	Four address bytes are configured when XSPI_DISCOVERY_GCTL.ABNUM= 1 and XSPI_DISCOVERY_GCTL.DMY_CNT=1
XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_ADDR_EN	Address phase is enabled when the SFDP database indicates a requirement of an address phase	Address phase is always disabled

Table 21-45: Status Sequence Parameters - Profile 1 Devices (Continued)

Parameter Name	Full Discovery Process	Not Full Discovery Process
	for reading the erase error bit. Otherwise, the address phase is disabled.	
XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_DMY_CNT	Number of dummy clock cycles is read from the SFDP database.	8 dummy clock cycles are configured for octal modes. Otherwise, 1 dummy clock cycle is configured when the data phase is in DDR mode and 0 for other cases.
XSPI_STAT_SEQ_CFG2.P1_ERS_FAIL_CMD_VALUE	Command is read from the SFDP database when it indicates support for an erase error bit. Otherwise, the 8'h05 command is configured.	8'h05 command is configured
XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_ADDR_EN	Address phase is enabled when the SFDP database indicates the requirement of an address phase for reading the program error bit. Otherwise, the address phase is disabled.	Address phase is always disabled
XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_DMY_CNT	Number of dummy clock cycles is read from the SFDP database	8 dummy clock cycles are configured for octal modes. Otherwise, 1 dummy clock cycle is configured when the data phase is in DDR mode and 0 for other cases.
XSPI_STAT_SEQ_CFG2.P1_PROG_FAIL_CMD_VALUE	Command is read from the SFDP database when it indicates support of the program error bit. Otherwise, the 8'h05 command is configured.	8'h05 command is configured
XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_ADDR_EN	Address phase is enabled when the SFDP database indicates the requirement of an address phase for reading the RDY/ BUSY bit. Otherwise, the address phase is disabled.	Address phase is enabled when XSPI_DISCOVERY_GCTL.ABNUM = 1,XSPI_DISCOVERY_GCTL.DMY_CNT =1, XSPI_DISCOVERY_GCTL.CMD_TYP[0] = 0, and XSPI_DISCOVERY_GCTL.OE_EN = 1
XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_ADDR_EN	Address phase is enabled when the SFDP database indicates the requirement of an address phase for reading the erase error bit. Otherwise, the address phase is disabled.	Address phase is always disabled
XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_DMY_CNT	Number of dummy clock cycles is read from the SFDP database	8 dummy clock cycles are configured for octal modes. Otherwise, 1 dummy clock cycle is configured when the data phase is in DDR mode and 0 for other cases.
XSPI_STAT_SEQ_CFG2.P1_ERS_FAIL_CMD_VALUE	Command is read from the SFDP database when it indicates support for the erase error bit. Otherwise, the 8'h05 command is configured.	8'h05 command is configured
XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_ADDR_EN	Address phase is enabled when the SFDP database indicates the requirement of an address phase for reading the program error bit. Otherwise, the address phase is disabled.	Address phase is always disabled

Table 21-45: Status Sequence Parameters - Profile 1 Devices (Continued)

Parameter Name	Full Discovery Process	Not Full Discovery Process
XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_DMY_CNT	Number of dummy clock cycles is read from the SFDP database	8 dummy clock cycles are configured for octal modes. Otherwise, 1 dummy clock cycle is configured when the data phase is in DDR mode and 0 for other cases.
XSPI_STAT_SEQ_CFG2.P1_PROG_FAIL_CMD_VALUE	Command is read from the SFDP database when it indicates support for the program error bit. Otherwise, the 8'h05 command is configured.	8'h05 command is configured
XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_ADDR_EN	Address phase is enabled when the SFDP database indicates the requirement of an address phase for reading the RDY/ BUSY bit. Otherwise, the address phase is disabled.	Address phase is enabled when XSPI_DISCOVERY_GCTL.ABNUM= 1, XSPI_DISCOVERY_GCTL.DMY_CNT = 1, XSPI_DISCOVERY_GCTL.CMD_TYP = 0 and XSPI_DISCOVERY_GCTL.OE_EN = 1 or when XSPI_DISCOVERY_GCTL.ABNUM= 1, XSPI_DISCOVERY_GCTL.DMY_CNT=1 and XSPI_DISCOVERY_GCTL.CMD_TYP[0] = 1 Otherwise, the address phase is disabled.
XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_DMY_CNT	Number of dummy clock cycles is read from the SFDP database.	4 dummy clock cycles are set if XSPI_DISCOVERY_GCTL.ABNUM= 1, XSPI_DISCOVERY_GCTL.DMY_CNT= 1, XSPI_DISCOVERY_GCTL.CMD_TYP = 0 and XSPI_DISCOVERY_GCTL.OE_EN is 1 or if XSPI_DISCOVERY_GCTL.ABNUM= 1, XSPI_DISCOVERY_GCTL.DMY_CNT= 1 and XSPI_DISCOVERY_GCTL.CMD_TYP[0] = 1 else 8 dummy clock cycles are configured for other octal modes Otherwise, 1 dummy clock cycles is configured when data phase in DDR mode and 0 in other all other modes
XSPI_STAT_SEQ_CFG2.P1_DEV_RDY_CMD_VALUE	Command is read from the SFDP database when it supports the RDY/BUSY bit. Otherwise, the 8'h05 command is configured.	8'h05 command is configured

Table 21-46: Status Sequence Parameters - Profile 2 HF Devices

Parameter Name	Parameter Value	Not Full Discovery Process
XSPI_STAT_SEQ_CFG4.P2_LATENCY_CNT	When possible, the number of latency clock cycles is read from the device status/configuration register. Otherwise, the number of latency clock cycles is configured to a number that is used to read the SFDP database.	Number of latency clock cycles is configured to a number that is used to read the SFDP database
XSPI_STAT_SEQ_CFG4.P2_MSK_CMD_MOD	Always 0	Always 0

Table 21-47: Status Sequence Parameters

Parameter Name	Parameter Value	Not Full Discovery Process
XSPI_STAT_SEQ_CFG5.PROG_FAIL_EN	1 when the SFDP database indicates support for the program error bit. Otherwise, 0.	Always 0
XSPI_STAT_SEQ_CFG5.PROG_FAIL_SIZ	1 when the data phase reflects octal DDR mode. Otherwise, 0.	1 when the data phase reflects octal DDR mode. Otherwise, 0.
XSPI_STAT_SEQ_CFG5.PROG_FAIL_VALUE	Value is read from the SFDP database when it indicates support for the program error bit. Otherwise, 0.	Always 0
XSPI_STAT_SEQ_CFG5.PROG_FAIL_IDX	Value is read from the SFDP database when it indicates support for the program error bit. Otherwise, 0.	Always 0
XSPI_STAT_SEQ_CFG8.PROG_FAIL_ADDR_VALUE	Value is read from the SFDP database when it indicates support for the program error bit. Otherwise, 0x00000000 address is configured.	Always 0x00000000 address is configured.
XSPI_STAT_SEQ_CFG5.ERS_FAIL_EN	1 when the SFDP database indicates support for the program error bit. Otherwise, 0.	Always 0
XSPI_STAT_SEQ_CFG5.ERS_FAIL_SIZ	1 when the data phase reflects octal DDR mode. Otherwise, 0.	1 when the data phase reflects octal DDR mode. Otherwise, 0.
XSPI_STAT_SEQ_CFG5.ERS_FAIL_VALUE	Value is read from the SFDP database when it indicates support for the erase error bit. Otherwise, 0.	Always 0
XSPI_STAT_SEQ_CFG5.ERS_FAIL_IDX	Value is read from the SFDP database when it indicates support for the erase error bit. Otherwise, 0.	Always 0
XSPI_STAT_SEQ_CFG9.ERS_FAIL_ADDR_VALUE	Value is read from the SFDP database if it indicatesupport for the erase error bit, Otherwise, 0x00000000 address is configured.	Always 0x00000000 address is configured
XSPI_STAT_SEQ_CFG5.DEV_RDY_EN	Always 1	Always 1

Table 21-47: Status Sequence Parameters (Continued)

Parameter Name	Parameter Value	Not Full Discovery Process
XSPI_STAT_SEQ_CFG5.DEV_RDY_SIZ	1 when the data phase reflects octal DDR mode. Otherwise, 0.	1 when the data phase reflects octal DDR mode. Otherwise, 0.
XSPI_STAT_SEQ_CFG5.DEV_RDY_VALUE	Value is read from the SFDP database if it indicates support for the RDY/BUSY bit, Otherwise, 0 for legacy SPI and xSPI profile 1 or 1 for legacy HyperFlash and xSPI profile 2.	0 for legacy SPI and xSPI profile 1 1 for legacy HyperFlash and xSPI profile 2
XSPI_STAT_SEQ_CFG5.DEV_RDY_IDX	Value is read from the SFDP database if it indicates support for the RDY/BUSY bit, Otherwise, 0 for legacy SPI and xSPI profile 1 or 7 for legacy HyperFlash and xSPI profile 2.	0 for legacy SPI and xSPI profile 1 7 for legacy HyperFlash and xSPI profile 2
XSPI_STAT_SEQ_CFG7.DEV_RDY_ADDR_VALUE	Value is read from the SFDP database if it indicates support for the RDY/BUSY bit, Otherwise, 0x00000000 address is configured.	Always 0x00000000 address is configured.

The following status sequence parameters are updated in the XSPI_STAT_SEQ_CFGn registers for SPI NAND devices following device discovery.

Table 21-48: Status Sequence Parameters for SPI NAND Devices

Parameter Name	Parameter Value
XSPI_ERS_SEQ_CFG0.ADDR_CNT	0 (one address byte)
XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_ADDR_EN	1
XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_DMY_CNT	0 dummy clock cycles are configured
XSPI_STAT_SEQ_CFG2.P1_ERS_FAIL_CMD_VALUE	8'h0F command is configured
XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_ADDR_EN	1
XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_DMY_CNTT	0 dummy clock cycles are configured
XSPI_STAT_SEQ_CFG2.P1_PROG_FAIL_CMD_VALUE	8'h0F command is configured
XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_ADDR_EN	1
XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_DMY_CNT	0 dummy clock cycles are configured
XSPI_STAT_SEQ_CFG2.P1_DEV_RDY_CMD_VALUE	8'h0F command is configured
XSPI_STAT_SEQ_CFG5.PROG_FAIL_SIZ	0
XSPI_STAT_SEQ_CFG5.PROG_FAIL_VALUE	1
XSPI_STAT_SEQ_CFG5.ERS_FAIL_IDX	3
XSPI_STAT_SEQ_CFG8.PROG_FAIL_ADDR_VALUE	32'h00000C0 value is configured

Table 21-48: Status Sequence Parameters for SPI NAND Devices (Continued)

Parameter Name	Parameter Value
XSPI_STAT_SEQ_CFG5.ERS_FAIL_EN	1
XSPI_STAT_SEQ_CFG5.ERS_FAIL_SIZ	0
XSPI_STAT_SEQ_CFG5.ERS_FAIL_VALUE	1
XSPI_STAT_SEQ_CFG5.ERS_FAIL_IDX	2.
XSPI_STAT_SEQ_CFG9.ERS_FAIL_ADDR_VALUE	32'h00000C0 value is configured
XSPI_STAT_SEQ_CFG5.DEV_RDY_EN	1
XSPI_STAT_SEQ_CFG5.DEV_RDY_SIZ	0
XSPI_STAT_SEQ_CFG5.DEV_RDY_VALUE	0
XSPI_STAT_SEQ_CFG5.DEV_RDY_IDX	0
XSPI_STAT_SEQ_CFG7.DEV_RDY_ADDR_VALUE	32'h00000C0 value is configured
XSPI_STAT_SEQ_CFG10.CRDY_VALUE	0
XSPI_STAT_SEQ_CFG10.CRDY_IDX	7
XSPI_STAT_SEQ_CFG10.ECC_FAIL_EN	0
XSPI_STAT_SEQ_CFG10.ECC_FAIL_VALUE	0
XSPI_STAT_SEQ_CFG10.ECC_FAIL_MSK	0

Boot Engine

xSPI controller provides an option to boot directly from the attached memory device using direct mode following controller reset. Alternatively, the controller can use a dedicated boot engine to transfer the boot code from the memory device to an area of host memory. The boot engine automatically triggers a page read command and probes the status of the transfer.

The boot engine assumes that the boot image is stored in the xSPI device connected to bank 0. The boot image is stored in the xSPI memory. It is composed of a configuration record and main data. The configuration record contains the following information:

- Main data size — the overall data size that need to be transferred during boot process
- Host address — the address on the host side where the boot image is transferred
- Image offset — the address inside the xSPI device from where the boot image is transferred
- SPI NAND page size — For NAND devices, this provides the SPI NAND device page size that is used to determine number of read iterations

After the boot process completes, the boot engine module writes the boot operation status. Only a single boot image copy is stored in the xSPI memory. This kind of memory is less prone to errors. So, no additional copies are required.

The figure shows the boot engine module functional diagram.

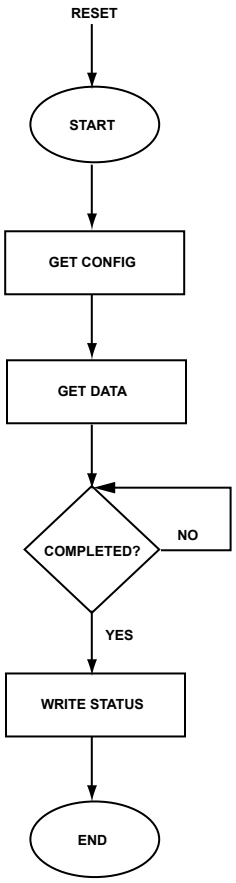


Figure 21-7: Boot Engine Flow

Configuring the Boot Engine

The boot engine is enabled by writing 1 to `MISCREG_XSPI_CTL0.BT_EN` field before doing a controller reset. Once reset is done, the controller fetches a 256-bit configuration record stored in the memory device. This record should have been previously stored in the memory device from address 0x0. The Boot Engine Configuration Record Layout figure provides the details of the required layout of the configuration record.

- Main data size — exists at memory device address 0x00. The lower 32 bits contain the size of the boot record information, the higher 32 bits are reserved.
- Image offset — exists at memory device address 0x08. The lower 48 bits contain the starting address of the boot image within the memory device. The higher 16 bits are reserved.
- Host address — exists at memory device address 0x10. The whole 64 bits contain the host address indicating where the boot image is copied.
- SPI NAND page size — exists at memory device address 0x18. It provides SPI NAND device page size which is used to determine the number of read iterations. Information is stored in the first 16 bits. The upper 48 bits are reserved.

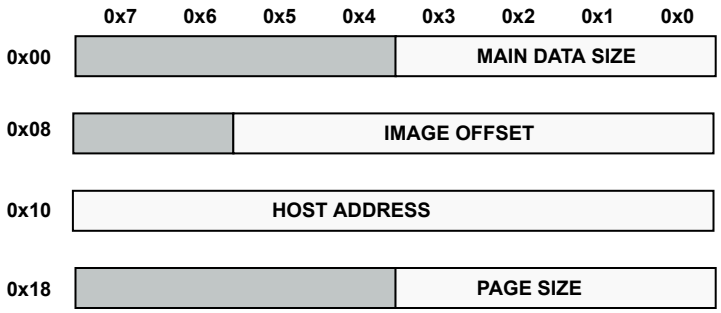


Figure 21-8: Boot Engine Configuration Record Layout

The register fields table shows the bits used by the boot engine to select valid commands and manage data transfer between the xSPI device and host memory. Register content is established by the device discovery process or register field reset vectors. The boot engine assumes that the information in these control registers are valid when boot process is triggered.

The following configuration register fields are used in the boot process:

Table 21-49: Registers for Boot

Register	Bit
XSPI_SEQ_GCTL0	- XSPI_SEQ_GCTL0.TYP - XSPI_SEQ_GCTL0.DAT_SWAP - XSPI_SEQ_GCTL0.UAL_CHUNK_CHK - XSPI_SEQ_GCTL0.UAL_CHUNK_EN - XSPI_SEQ_GCTL0.CHUNK_SIZ - XSPI_SEQ_GCTL0.CRC_OE - XSPI_SEQ_GCTL0.CRC_VAR - XSPI_SEQ_GCTL0.CRC_EN - XSPI_SEQ_GCTL0.DAT_PER_ADDR
XSPI_XIP_GCTL	- XSPI_XIP_GCTL.XIP_DIS_MB_VALUE
XSPI_READ_SEQ_CFG0	- XSPI_READ_SEQ_CFG0.P1_DMY_CNT - XSPI_READ_SEQ_CFG0.P1_DAT_EDGE - XSPI_READ_SEQ_CFG0.P1_DAT_IOS - XSPI_READ_SEQ_CFG0.P1_ADDR_EDGE - XSPI_READ_SEQ_CFG0.P1_ADDR_IOS - XSPI_READ_SEQ_CFG0.P1_ADDR_CNT - XSPI_READ_SEQ_CFG0.P1_CMD_EDGE - XSPI_READ_SEQ_CFG0.P1_CMD_IOS - XSPI_READ_SEQ_CFG0.P1_CMD_VALUE

Table 21-49: Registers for Boot (Continued)

Register	Bit
XSPI_READ_SEQ_CFG1	- XSPI_READ_SEQ_CFG1.P1_CMD_EXT_EN - XSPI_READ_SEQ_CFG1.P1_CMD_EXT_VALUE - XSPI_READ_SEQ_CFG1.P1_MB_EN - XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT
XSPI_READ_SEQ_CFG2	- XSPI_READ_SEQ_CFG2.P2_LATENCY_CNT - XSPI_READ_SEQ_CFG2.P2_HF_BOUND_EN - XSPI_READ_SEQ_CFG2.P2_MSK_CMD_MOD - XSPI_READ_SEQ_CFG2.P2_BURST_TYP - XSPI_READ_SEQ_CFG2.P2_TARGET
XSPI_STAT_SEQ_CFG0 (only for SPI NAND device)	- XSPI_STAT_SEQ_CFG0.P1_DAT_EDGE - XSPI_STAT_SEQ_CFG0.P1_DAT_IOS - XSPI_STAT_SEQ_CFG0.P1_CMD_EXT_EN - XSPI_STAT_SEQ_CFG0.P1_CMD_EDGE - XSPI_STAT_SEQ_CFG0.P1_CMD_IOS - XSPI_STAT_SEQ_CFG0.P1_ADDR_EDGE - XSPI_STAT_SEQ_CFG0.P1_ADDR_IOS - XSPI_STAT_SEQ_CFG0.P1_ADDR_CNT
XSPI_STAT_SEQ_CFG1(only for SPI NAND device)	- XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_ADDR_EN - XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_DMY_CNT
XSPI_STAT_SEQ_CFG2 (only for SPI NAND device)	- XSPI_STAT_SEQ_CFG2.P1_DEV_RDY_CMD_VALUE
XSPI_STAT_SEQ_CFG3 (only for SPI NAND device)	- XSPI_STAT_SEQ_CFG3.P1_DEV_RDY_CMD_EXT_VALUE
XSPI_STAT_SEQ_CFG5 (only for SPI NAND device)	- XSPI_STAT_SEQ_CFG5.DEV_RDY_SIZ - XSPI_STAT_SEQ_CFG5.DEV_RDY_VALUE - XSPI_STAT_SEQ_CFG5.DEV_RDY_IDX
XSPI_STAT_SEQ_CFG7 (only for SPI NAND device)	- XSPI_STAT_SEQ_CFG7.DEV_RDY_ADDR_VALUE

xSPI PHY Controller

The PHY handles the low-level timings of the data, address, and control signals between the device and controller. The PHY has its own programmable register group.

PHY Initialization

Before the host can communicate with the memory device, software must initialize the PHY and lock it. The soft PHY contains three synthesizable delay lines (DLLs) that are built from 256 delay elements. The first delay line is the controller delay line and it measures how many delay elements are needed to match the XSPI_CLK clock

period. The other two delay lines are targets, one for the transmission TX outbound path and the other for the reception RX inbound path.

The TX target delay line can use the controller delay line measurements to influence how much delay to add to transmitted data based on a percentage of the clock period. This improves the eye diagram (a common indicator of the quality of signals in high-speed digital transmissions) of the memory device. Similarly, the RX target delay line can shift the read sampling clock as a percentage of the clock period to find the ideal sampling point for the inbound read data. Before this can happen however, the controller delay line must be configured and this is done immediately after power on reset (PoR). This is PHY locking, a fully internal PHY function that attempts to lock the bus requestor delay line. Until the PHY is locked, there can be no access to or from the memory device.

When choosing to disable the controller delay line (that is, saturation mode), the target delay line cannot be programmed based on a percentage of the clock period and instead must be manually configured. This is suitable for when operating the PHY at slower data rates, when the controller delay line does not have enough delay elements to measure the complete clock period. Running at these slower rates does not require fine tuning of the target delay lines (when running at SDR) and manual configured is acceptable.

NOTE: PHY locking should not be confused with PHY training. PHY training is a software application (not part of the controller deliverables) that can be implemented to help tune the PHY to find the pbest sampling point and allow the highest possible `XSPI_CLK` frequency. The controller does not implement PHY training.

To lock the controller delay line and have the general PHY ready to transfer data, some of the PHY registers must be programmed with appropriate values that are dependent on the `XSPI_CLK` frequency. The PHY runs at the same frequency as the attached memory device clock and uses system parameters such as board skew, pad delays, and so forth. Failure to program suitable values can result in the PHY inability to lock and unable to properly wake up.

Initial PHY Registers to Configure

The PHY registers that must be initially configured before attempting to lock the PHY are described in the following sections:

- XSPI_PHY_DLL_CTL Register
- XSPI_PHY_DQ_TR Register
- XSPI_PHY_DQS_TR Register
- XSPI_PHY_GATE_LPBK_CTL Register
- XSPI_PHY_DLL_REQ_CTL Register
- XSPI_PHY_DLL_COMP_CTL Register

XSPI_PHY_DLL_CTL Register

This register generally controls how the PHY target delay lines are routinely updated to compensate for drift caused by dynamic factors such as temperature or open-circuit voltage (OCV). It also contains register settings that are dependent on the attached memory device and operating mode.

The XSPI_PHY_DLL_CTL.SDR_EDGE_ACT[21:21] bit controls on which clock edge the PHY controller transmits write data. It is only required for SDR operation and when the PHY controller defaults to propagating write data on the clock rising edge. Generally keep at the default state of zero, unless the memory device (in SDR operation) is unable to sample the data.

The XSPI_PHY_DLL_CTL.DQS_LST_DAT_DRP_EN[20:20] bit must only be set to one, when all of the following conditions are met:

- The attached memory device issues read data on the negative clock edge.
- The memory device supports DQS (transfer two data words per clock cycle) and the PHY has been set up to sample read data using that DQS signal. The DQS that the memory device sends contains one extra edge than normal and the PHY must compensate for the added edge.

XSPI_PHY_DQ_TR Register

This register controls the timing of the main data (DQ) bus as it switches from the PHY to the memory device. Adjustments can be made to control when the attached DQ pads switch direction (that is, to add delay when the output enable switches from input to output and from output to input). There is also an option to define additional latency on the data bus itself, in units of one-half clock cycle.

NOTE: The XSPI_PHY_DQ_TR.DAT_OE_END field (which defaults to 0x2 and controls when the output enable switches from output to input) must always be configured to a minimum value of 0x1. This register does not influence the locking procedure.

XSPI_PHY_DQS_TR Register

This register determines the read data sampling method used by the PHY. See PHY Read Sampling Method Selection for more details.

CAUTION: This register must be updated to reflect the read data sampling method to successfully read any data from the device.

XSPI_PHY_GATE_LPBK_CTL Register

This register controls the gate for read data sampling to occur and enables the loopback BIST test modes within the PHY. There are fields that must be set up prior to locking. When the register fields are not mentioned in this section, the field can be left in its default state.

- XSPI_PHY_GATE_LPBK_CTL.GATE_CFG – Use when the PHY is not using DQS from the device to sample read data. (Determined by the read data sampling method described in PHY Read Sampling Method Selection.) PHY uses information from the controller to determine which cycle the read data should be returning from the device, and when it should be sampling the DQ bus. When the round-trip delay of the entire read operation (based on WRITE CLK PAD delay + PCB trace to device + PCB trace from device + DQ PAD delay) exceeds one clock cycle, then the PHY must also delay the cycle in which it samples that data.

  When DQS from the device is not used to sample read data, the XSPI_PHY_GATE_LPBK_CTL.GATE_CFG field = floor (round-trip delay in ns / clock period in ns)

- XSPI_PHY_GATE_LPBK_CTL.RD_DEL_SEL = ceiling (round-trip delay in ns / clock period in ns) + 2

XSPI_PHY_DLL_REQ_CTL Register

CAUTION: With a small value, the initial lock time is longer. Values less than 0x04 may cause no lock by the delay line.

Use these register fields to configure and direct the controller delay line, prior to the PHY locking.

- XSPI_PHY_DLL_REQ_CTL.DLL_BYPASS_MODE – When set (=1) (default), it disables the controller delay line and the PHY always locks immediately. This forces saturation mode . When the user wants the PHY to operate in saturation mode, this field should be set and the PHY requires no further effort to lock.
- XSPI_PHY_DLL_REQ_CTL.PHASE_DT_SEL – This defaults to zero, but Analog Devices recommends it be configured to 0x2.
- XSPI_PHY_DLL_REQ_CTL.DLL_START_PT – The controller delay line can take some time to measure the clock period and then signal that the lock was achieved. It must start from a fixed number of delay elements and either add or remove delay elements until it finds the appropriate delay matching the clock period. Configuring this field can speed up the locking time. This field should be configured such that it is not greater than 7/8^{ths} of the clock period (use the worst-case element delay).

Example: When the clock frequency is 200 MHz (5 ns cycle time) and a worst-case element delay of 80 ps, this field equals [5 * (7/8) / .080 = 54] elements. When the locking time is not a concern, a small value such as 0x04 can be used instead to ensure that the delay line does not lock on a harmonic.

XSPI_PHY_DLL_COMP_CTL Register

This register controls the TX and RX target delay lines. When the PHY is operating in saturation mode, the number of delay elements can be manually programmed for both delay lines. Otherwise, each delay line can be precisely delayed by steps (1/256th of the clock period). The XSPI_PHY_DLL_COMP_CTL register does not need to be updated to initially lock the PHY, but it must be given appropriate values for the memory device to be able to sample the data sent by the PHY, and for the PHY to sample the data sent by the device.

On the transmit side, the DQ line can be delayed using the TX target delay line. On the receive side, the sampling clock (which depends on the chosen read data sampling method described in PHY Read Sampling Method Selection) is delayed by the RX target delay line. Choose initial values for these fields—setup times, skew between DQS and the data, board delays, and so forth—all affect where the eye diagram begins.

Initial values for the RX target delay line can be calculated using the steps found in the following section. The delivered script automatically calculates the required values, but these are for additional guidance.

IMPORTANT: Performing a full training sequence is the only way to calculate the most optimum delay value to find the center of the eye diagram.

Initial Value Selection for XSPI_PHY_DLL_COMP_CTL Register

1. Express the quarter_cycle_period in picoseconds. For example, when the clock period is 10 ns, configure this field equal to 2500 ps.

2. For the `delay_of_each_element` variable, calculate the expected delay through each implemented PHY delay element in ps. The delay elements within the PHY are made up of two NAND2 gates as selected in the hand instantiated cells. For example, when the delay through each NAND2 is 12 ps, the delay element = 24 ps.
3. Determine the `max_dqs_dq_skew` (in ns) maximum skew between the DQS strobe and the DQ data obtained from the memory device datasheet `tdqsq` value.
4. The `max_dqs_dq_skew_perclk` skew as a percentage of the clock equals `(max_dqs_dq_skew)` divided by the clock period.
5. The `rx_setup time` in ns is the time the device takes to drive the read data following the edge of the clock. This time is specified in the memory device datasheet. Note there are specific times for SDR and for each edge to valid data in DDR. Pick the largest value (worst case).
6. Calculate the `rx_setup_gt_half` value when `rx_setup` is greater than one-half the clock cycle. Determine the difference between these two values in nanoseconds. Formula: `rx_setup_gt_half` = rx_setup – (clock_period / 2).
7. Calculate the `rx_setup_gt_full` value when `rx_setup+4` is greater than a clock cycle. Determine the difference between these two values in nanoseconds. Formula: `rx_setup_gt_full` = rx_setup + 4 – clock_period.
8. Calculate the `tx_setup_de` value. Let `tx_setup` be the setup time in ps, that the target memory device requires before it samples write data. This is often referred to as `tDVCH` in the device data sheet. Divide the `tx_setup` time by the delay of each element to get the number of delay elements needed to cover the set up.
9. Calculate the `tx_skew_margin_de` value for the TX skew margin. Let `quarter_cycle_de` be the quarter_cycle_period variable divided by the delay of each element to get the number of delay elements needed to cover the ¼ of the clock period. Divide again by five to get the number of delay elements needed to cover 1/20th of the clock period. This value is chosen for the TX skew margin.

Value Selection for RX Target Delay Lines

Use the ***RX Target Delay Line Parameters*** table to configure initial values for the RX target delay line.

Table 21-50: RX Target Delay Line Parameters

Parameters	Initial Value to Store in RX Delay Line Field
Saturation Mode = 1 Using DQS from device	`ceiling`((max_dqs_dq_skew*1000) / delay_of_each_element) + 1
Saturation Mode = 1 Using Internal or External Loopback DQS `rx_setup` ≤ half clock period	1

Table 21-50: RX Target Delay Line Parameters (Continued)

Parameters	Initial Value to Store in RX Delay Line Field
Saturation Mode = 1 Using Internal or External Loopback DQS rx_setup > half clock period rx_setup + 4 ≤ clock period	Setup time is quite large relative to the clock period. Need to delay initial sample point accordingly. `ceiling`((rx_setup_gt_half*1000) / delay_of_each_element) + 1
Saturation Mode = 1 Using Internal or External Loopback DQS rx_setup + 4 > clock period	Setup time is very large relative to the clock period. For SDR modes, Analog Devices recommends setting `SDR_EDGE_ACT` to force the capture edge to be negative. `ceiling`((rx_setup_gt_full*1000) / delay_of_each_element) + 1
Saturation Mode = 0 Using DQS from device `max_dqs_dq_skew_perclk` > 1	255
Saturation Mode = 0 Using DQS from device `max_dqs_dq_skew_perclk` ≤ 1	`max_dqs_dq_skew_perclk` * 256 + 0.01*256
Saturation Mode = 0 Using Internal or External Loopback DQS rx_setup ≤ half clock period	0
Saturation Mode = 0 Using Internal or External Loopback DQS rx_setup > half clock period rx_setup + 4 ≤ clock period	`ceiling`(256*(rx_setup_gt_half/clk_period)) + 25; Adding 25 extra elements to guarantee minimal skew.
Saturation Mode = 0 Using Internal or External Loopback `rx_setup` + 4 > clock period	`ceiling`(256*(rx_setup_gt_full/clk_period)) + 25; Adding 25 extra elements to guarantee minimal skew.

CAUTION: The maximum value that can be programmed for the controller delay line is 255. When any calculation exceeds 255, configure the value to 255.

Value Selection for TX Target Delay Lines

Use the ***TX Target Delay Line Parameters*** table to configure initial values for the TX target delay line.

Table 21-51: TX Target Delay Line Parameters

Parameters	Initial Value to Store in TX Delay Line Field
Saturation Mode = 1 flash setup is greater than quarter of cycle - skew margin (tx_setup_de > (quarter_cycle_de – tx_skew_margin_de))	0x5
Saturation Mode = 1 flash setup is less than or equal to quarter of cycle - skew margin (tx_setup_de ≤ (quarter_cycle_de – tx_skew_margin_de))	Set to 25% That is, 25% ceiling(quarter_cycle_de)
Saturation Mode = 0 flash setup is greater than quarter of a cycle (tx_setup_de > quarter_cycle_de)	0x15
Saturation Mode = 0 flash setup is less than or equal to quarter of cycle (tx_setup_de = quarter_cycle_de)	0x33

CAUTION: The maximum value that can be programmed for the controller delay line is 255. When any calculation exceeds 255, configure the value to 255.

PHY Re-initialization

Once the host is operational, it can update the PHY registers and trigger a re-initialization or locking procedure using bits 24 and 25 of the `XSPI_PHY_DLL_CTL` register. A re-initialization must be performed whenever the PHY registers are modified or when the user has changed the `XSPI_CLK` frequency.

Resynchronization of Target Delay Lines

While the PHY is running, the target delay lines can still occasionally require resynchronization as a result of temperature changes, OCV, and how much that change alters the delay through the delay elements. The frequency at which this resync needs to happen depends on how much change those dynamic factors can make on the delay through the delay line and whether that change is sufficient that the sampling point is no longer within the valid data window.

There is no need to resync when the `DLL_LOCK_VALUE` has not changed or has only changed by a small amount. Analog Devices recommends that software monitors the `XSPI_PHY_DLLOB0.DLL_LOCK_VALUE` field and triggers a resync when the `XSPI_PHY_DLLOB0.DLL_LOCK_VALUE` has changed by a significant amount since the last resync (for example, 10%). Alternatively, a user can choose to resync periodically, often enough to mitigate these changes in delay, but which does not impact overall performance.

During a resynchronization, the memory device cannot be accessed. For automatic recurring delay line resynchronizations, configure the `XSPI_PHY_DLL_UPDT_CNT.RESYNC_CNT` field to a value other than 0x0. The value defines the time interval between each resync in `xspi_clk cycles`.

CAUTION: The memory interface must be idle during resync operation.

When a resync happens while the controller is communicating with the memory device, the resync is automatically withheld until the memory operation completes. Similarly, when a resync is already in progress, the controller withholds any normal operation triggered by the host until that resync completes. The `XSPI_PHY_DLL_CTL.RESYNC_IDLE_CNT` field defines the wait time after a resync operation completes when normal traffic can resume.

PHY Read Sampling Method Selection

To use the PHY, decide which read sampling mechanism to use. Essentially, there are two options the user can select from when using the PHY:

- DQS Mode for PHY Read Sampling
- Non-DQS Mode (Loopback) for PHY Read Sampling

DQS Mode for PHY Read Sampling

The highest device clock frequencies and, therefore, performance is achieved when the memory device drives DQS. DQS is an independent signal that is driven parallel to the read data and acts as a source synchronous clock from the device to the PHY. The PHY takes the DQS from the device, passes it through the receive delay line (for programmable incremental delay steps) and uses the resulting signal as a strobe to sample incoming read data. To support this option, the memory device being targeted must support DQS. Additionally, a physical hardware DQS pin connection must be provided. Ensure that bit 20 of the `XSPI_PHY_DQS_TR` register is cleared (=0).

Non-DQS Mode (Loopback) for PHY Read Sampling

The memory device sometimes does not supply DQS or lacks DQS hardware support on board. The alternative option is to mimic the data sampling strobe, referred to as the DQS loopback options of the PHY. Generally, this mode is referred as on-DQS (loopback) mode.

Non-DQS Loopback Mode With a Pad

Internal loopback through the `pad_mem_rebar_t pad`. This still requires the integrator to implement the `pad_mem_rebar_t pad`, although it is not bonded out to a pin. This option can be chosen when dynamic factors like OCV are not important (perhaps because the operating frequency is slower or only SDR operation is needed). This loopback function is performed within the pad and includes pad delays. Depending on the chosen pads, pad delays can be significant, so including these in the loopback path can be important. To support this option, the user must implement the `pad_mem_rebar_t pad`, but it is not connected to anything on the board. Bits 20 and 21 of the`XSPI_PHY_DQS_TR` register must be set (=1). Bit 22 of the `XSPI_PHY_DQS_TR` register must be cleared (=0).

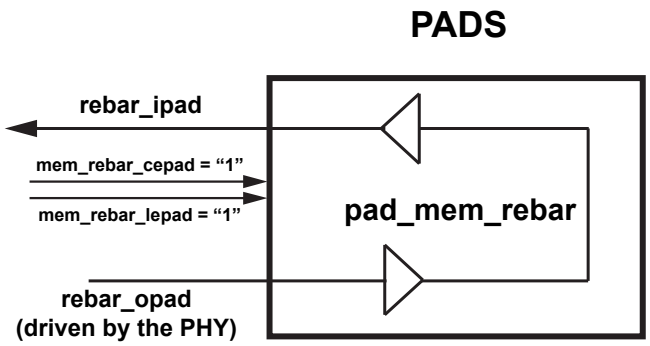


Figure 21-9: Logic Diagram for Non-DQS (Loopback) Mode With Pad

Memory Device Data Integrity

Some memory devices support advanced data integrity checking functions (for example, Macronix MX25). The controller is compatible with the following device capabilities.

ECC Data Integrity

Some memory devices have built-in Error Checking and Correcting (ECC). During a page program operation, the device automatically creates the ECC. During a read operation, the device checks and when necessary, automatically corrects errors. These devices can output an ECC status signal (ECS#) to signal the ECC correction status. The status is optionally passed to the controller using the `XSPI_DFI_GP_OPEN_DRAIN` input, which can be visible with the interrupt status register.

CRC Parity Check

Some memory devices have built-in cyclic redundancy check (CRC) capabilities. For write operations, the controller must send the CRC code to the device for checking. For read operations, the device must send the CRC code and the code must be validated using the controller.

The controller can generate the CRC on the address and calculate it with an exclusive-OR of the address bytes. This is sent to the device immediately after the address bytes. The controller can also generate a CRC on a chunk (programmable number of bytes ranging from 8 bytes through 512 bytes) of the write data. A CRC code is inserted into the outbound data stream immediately following that data chunk. Configuring the controller to use CRC depends on the active work mode. The CRC on the data bytes is calculated by exclusively-ORing the data bytes within a chunk.

CRC Using Direct or ACMD Mode

CRC is enabled using the `XSPI_SEQ_GCTL0`[8:8] bit. It can be configured to always assume that a CRC code is immediately present after the address bytes, or when it is to be inserted into the data stream after a chunk of bytes (independent of whether the stream is sending address bytes or data bytes). This operation is controlled using bit nine of the same register. Some devices require that the CRC be sent first with the CRC value and then repeated with an inverted value. Enable this option using the `XSPI_SEQ_GCTL0`[10:10] bit. Control the chunk size (number of bytes protected by the CRC) with the `XSPI_SEQ_GCTL0`[14:12] field. When the first CRC code

protects the address and is part of the data bytes, bit 16 should be set (=1). When checking that the first CRC code for read data is required, then bit 17 must also be set.

CRC Using STIG Mode

The parameters defined in the XSPI_SEQ_GCTL0 register apply also for the direct and ACMD modes available directly in the glued data STIG instruction described in the Glued Data Instruction section. Enable the CRC using the XSPI_SEQ_GCTL0.CRC_EN field. Control the chunk size (number of bytes protected by the CRC) using the XSPI_SEQ_GCTL0.CHUNK_SIZ field. When the first CRC code protects the address and part of the data bytes, the XSPI_SEQ_GCTL0.UAL_CHUNK_EN field must be cleared (=0). When needing to check the first CRC code for read data, theXSPI_SEQ_GCTL0.UAL_CHUNK_CHK bit must be set (=1).

Miscellaneous Low-level Configurations

The following configurations are supported using the ACMD, direct, and STIG modes.

Write Protection

Some devices that operate in single or dual mode support the use of a write protect signal using the DQ2 pin. The controller can support this feature by using the XSPI_MINICTL_WP_CTL register. See the XSPI_MINICTL_WP_CTL register description for configuration details.

Device Reset

When operating in single or dual mode, some devices have a dedicated #RESET pin or support the hardware reset using the DQ3 pin. The controller supports the driving of reset on any bank (chip select) with the XSPI_MINICTL_WP_CTL register.

SPI Clock CPOL and CPHA Configuration

When operating in Single Data Rate (SDR), the transfers to and from the SPI memory device can operate in SPI mode 0 (polarity CPOL=0, phase CPHA=0, neither checked) or SPI mode 3 (CPOL=1, CPHA=1, both checked). It is unlikely this default configuration needs changing, but the option is available using the XSPI_MINICTL_CLKMODE register.

JEDEC Reset Delay Settings

When issuing a JEDEC reset instruction to the memory device, the Joint Electron Device Engineering Council Solid State Technology Association (JEDEC) standard requires the controller to adhere to the t_{CSL} and t_{CSH} timing parameters. When non-default settings are required, a user programs the XSPI_MINICTL_JEDEC_RST_TR register with specific values.

Device Delay Settings

The XSPI_MINICTL_DEV_DLY register can provide minimum delays between transactions at the memory device interface.

Reset Recovery Delay Settings

Following a hardware reset, the user can force a minimum time delay using the XSPI_MINICTL_RST_RECOV register, before the chip select moves to an active state.

t_{CMS} Timing Pause Refresh Settings

Some volatile memory devices can require a periodic refresh (for example, HyperRAM). This refresh action is referred to as t_{CMS} timing. Use the XSPI_MINICTL_DEV_ACTIVE_MAX register and the XSPI_SEQ_GCTL0.TCMS_EN bit for the controller to configure a maximum time that the chip select remains in an active state to perform a periodic refresh operation. The controller automatically breaks up larger transactions. This action is transparant to the host interface.

Legacy HyperFlash or Profile 2 Address Settings

By default, the host address must be rearranged to form the Command/Address (CA) structure for many legacy HyperFlash and profile 2 devices. Typically, the CA field has a reserved section; some bits of the host address must be shifted. The XSPI_MINICTL_HF_OFFSET register provides the user a method to configure the mapping of these reserved bits.

Multiple Device Connections to Controller

The xSPI controller can connect to a maximum of eight memory devices using the chip select (CS). Throughout this hardware reference, each device connected to an individual chip select is referred to as a bank. Mixed device types are permitted, but, cannot be used simultaneously. For example, when bank zero is connected to a quad SPI NOR flash device and bank one is connected to a HyperBus style device, then some reconfiguration of the xSPI register space is required when the host switches between accessing these devices.

SPI NAND Device Operation Settings

For SPI NAND devices, the memory device address contains both row and column addresses. The ***SPI NAND Device Memory Address*** figure shows the field layout. The boundary between the row and column addresses can be configured using the XSPI_SEQ_GCTL1.PCA_SIZ field. When the selected sequences require a number of bits aligned to 8 (for example, when a sequence requires 16 bits and contains 4 dummy bits before the 12 bits of the column address), the added dummy bits are internally cleared (=0).

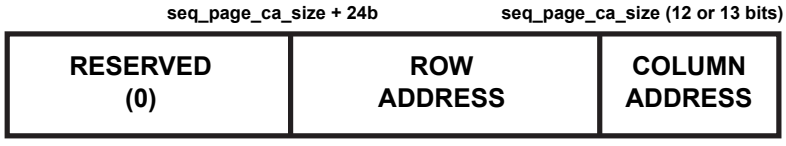


Figure 21-10: SPI NAND Device Memory Address

The SPI NAND devices require a specific page layout. The host software must program the controller to adhere to this field layout. It especially important not to overwrite the reserved areas as bad block data.

ADSP-2184x XSPI Register Descriptions

Expanded Serial Peripheral Interface (XSPI) contains the following registers.

Table 21-52: ADSP-2184x XSPI Register List

Name	Description
XSPI_BOOT_STAT	Boot Status Register
XSPI_CMD0	Command Register 0
XSPI_CMD1	Command Register 1
XSPI_CMD2	Command Register 2
XSPI_CMD3	Command Register 3
XSPI_CMD4	Command Register 4
XSPI_CMD5	Command Register 5
XSPI_CMD_STAT	Command Status Register
XSPI_CMD_STAT_PTR	Command Status Pointer Register
XSPI_DAC_CFG	DAC Configuration Register
XSPI_DAC_REMAPADDR0	Address Remapping Register 0
XSPI_DAC_REMAPADDR1	Address Remapping Register 1
XSPI_DISCOVERY_GCTL	Device Discovery Control Register
XSPI_DMA_CTL	DMA Interface Control Register
XSPI_DMA_ERRADDR_HI	DMA Error Address High Register
XSPI_DMA_ERRADDR_LO	DMA Error Address Low Register
XSPI_ERS_SEQ_CFG0	Erase Sequence Configuration Register 0
XSPI_ERS_SEQ_CFG1	Erase Sequence Configuration Register 1
XSPI_ERS_SEQ_CFG2	Erase Sequence Configuration Register 2
XSPI_FEATURES	Controller Features Register
XSPI_GSTAT	General Controller Status Register
XSPI_INT_EN	Interrupt Enable Register
XSPI_ISTAT	Interrupt Status Register
XSPI_LONGPOL_GCTL	Long Polling Count Register
XSPI_MINICTL_DEV_ACTIVE_MAX	Device Active Maximum Clock Cycle Register
XSPI_MINICTL_CLKMODE	Clock Mode Control Register
XSPI_MINICTL_DEV_DLY	Device Delay Register
XSPI_MINICTL_HF_OFFSET	HyperFlash Offset Register
XSPI_MINICTL_JEDEC_RST_TR	JEDEC Reset Delay Register

Table 21-52: ADSP-2184x XSPI Register List (Continued)

Name	Description
XSPI_MINICTL_RST_PIN_CTL	Hardware Reset Control Register
XSPI_MINICTL_RST_RECOV	Reset Recovery Delay Register
XSPI_MINICTL_WP_CTL	Write Protect Register
XSPI_PHY_DLLOB0	PHY DLL Observable Points 0 Register
XSPI_PHY_DLLOB1	PHY DLL Observable Points 1 Register
XSPI_PHY_DLL_CTL	PHY DLL Control Register
XSPI_PHY_DLL_REQ_CTL	Bus Requester DLL Control Register
XSPI_PHY_DLL_COMP_CTL	Bus Completer DLL Control Register
XSPI_PHY_DLL_UPDT_CNT	PHY DLL Resynchronization Register
XSPI_PHY_DQS_TR	PHY DQS Timing Register
XSPI_PHY_DQ_TR	PHY DQ Timing Register
XSPI_PHY_FEATURES	PHY Features Register
XSPI_PHY_GATE_LPBK_CTL	PHY Gate Loopback Control Register
XSPI_PHY_GCTL	PHY Global Control Register
XSPI_PHY_GPIO_CTL0	PHY GPIO Control Register 0
XSPI_PHY_GPIO_CTL1	PHY GPIO Control Register 1
XSPI_PHY_GPIO_STAT0	PHY GPIO Status Register 0
XSPI_PHY_GPIO_STAT1	PHY GPIO Status Register 1
XSPI_PHY_GTSEL	PHY Global Termination Control Register
XSPI_PHY_IE_TR	PHY DQS Input Enable Timing Register
XSPI_PHY_OB0	PHY Observable Points Register
XSPI_PHY_REVID	PHY Revision ID Register
XSPI_PHY_STATIC_TGL	PHY Static Aging Register
XSPI_PHY_WR_DESKEW_PAD_CTL0	PHY Deskew Write Register
XSPI_PROG_SEQ_CFG0	Program Sequence Configuration Register 0
XSPI_PROG_SEQ_CFG1	Program Sequence Configuration Register 1
XSPI_PROG_SEQ_CFG2	Program Sequence Configuration Register 2
XSPI_READ_SEQ_CFG0	Read Sequence Configuration Register 0
XSPI_READ_SEQ_CFG1	Read Sequence Configuration Register 1
XSPI_READ_SEQ_CFG2	Read Sequence Configuration Register 2
XSPI_REVID	Revision ID Register

Table 21-52: ADSP-2184x XSPI Register List (Continued)

Name	Description
XSPI_RST_SEQ_CFG0	Reset Sequence Configuration Register 0
XSPI_RST_SEQ_CFG1	Reset Sequence Configuration Register 1
XSPI_SDMA_ADDR0	Host DMA Buffer Address Register 0
XSPI_SDMA_ADDR1	Host DMA Buffer Address Register 1
XSPI_SDMA_SIZ	Host DMA Block Size Register
XSPI_SDMA_TRD_STAT	Host DMA Thread Status Register
XSPI_SEQ_GCTL0	Sequence Configuration Register 0
XSPI_SEQ_GCTL1	Sequence Configuration Register 1
XSPI_SHORTPOL_GCTL	Short Polling Count Register
XSPI_STAT_SEQ_CFG0	Status Checking Sequence Configuration Register 0
XSPI_STAT_SEQ_CFG1	Status Checking Sequence Configuration Register 1
XSPI_STAT_SEQ_CFG10	Status Checking Sequence Configuration Register 10
XSPI_STAT_SEQ_CFG2	Status Checking Sequence Configuration Register 2
XSPI_STAT_SEQ_CFG3	Status Checking Sequence Configuration Register 3
XSPI_STAT_SEQ_CFG4	Status Checking Sequence Configuration Register 4
XSPI_STAT_SEQ_CFG5	Status Checking Sequence Configuration Register 5
XSPI_STAT_SEQ_CFG7	Status Checking Sequence Configuration Register 7
XSPI_STAT_SEQ_CFG8	Status Checking Sequence Configuration Register 8
XSPI_STAT_SEQ_CFG9	Status Checking Sequence Configuration Register 9
XSPI_TRD_COMP_ISTAT	Auto Command Engine Interrupt Status Thread Register
XSPI_TRD_ERR_INT_EN	Thread Error Interrupt Enable Register
XSPI_TRD_ERR_ISTAT	Thread Error Interrupt Status Register
XSPI_TRD_STAT	Auto Command Engine Thread Status Register
XSPI_WE_SEQ_CFG0	WEL Sequence Configuration Register
XSPI_WORKMODE_CTL	Device Control Register
XSPI_XIP_GCTL	XIP Configuration Register

Boot Status Register

The XSPI_BOOT_STAT register provides the status of the most recent boot operation, triggered by the automated boot controller.

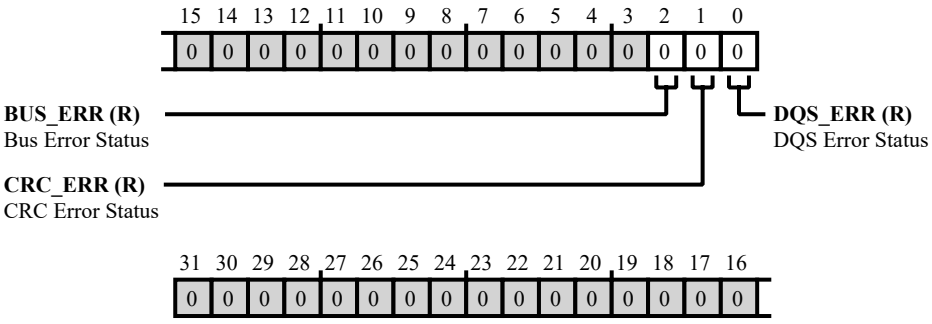


Figure 21-11: XSPI_BOOT_STAT Register Diagram

Table 21-53: XSPI_BOOT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	BUS_ERR	Bus Error Status. The XSPI_BOOT_STAT.BUS_ERR field describes bus status during the boot process. When set, the boot process failed due to the bus interface receiving an error response from the target.	
		0	No Error Detected
		1	Error Detected
1 (R/NW)	CRC_ERR	CRC Error Status. The XSPI_BOOT_STAT.CRC_ERR field describes CRC status during the boot process. When set, the boot process failed due to the CRC error on the xSPI interface.	
		0	No Error Detected
		1	Error Detected
0 (R/NW)	DQS_ERR	DQS Error Status. The XSPI_BOOT_STAT.DQS_ERR bit field describes the DQS status during the boot process. When set, the boot process failed due to the DQS error on the xSPI interface.	
		0	No Error Detected
		1	Error Detected

Command Register 0

Writing data to the XSPI_CMD0 register initiates a new transaction of the xSPI flash controller in CDMA/PIO and STIG work mode.

Encoding of the register depends on the selected work mode.

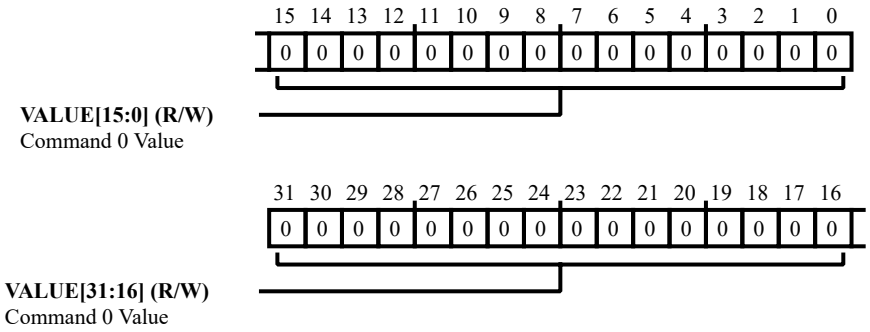


Figure 21-12: XSPI_CMD0 Register Diagram

Table 21-54: XSPI_CMD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command 0 Value. Command 0 register field.

Command Register 1

The XSPI_CMD1 register is only used in CDMA/PIO and STIG work mode. Field encoding of the register depends on the selected work mode.

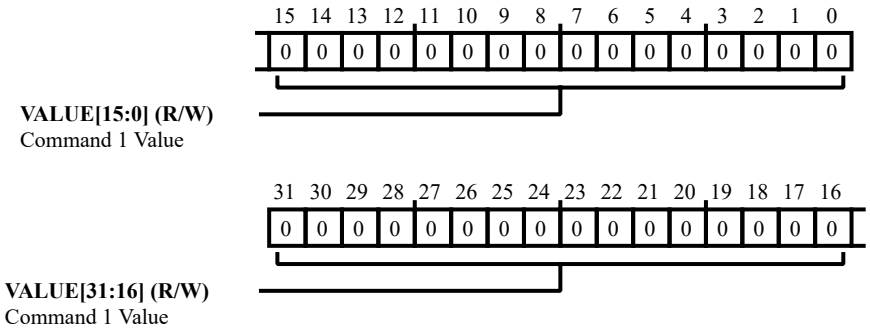


Figure 21-13: XSPI_CMD1 Register Diagram

Table 21-55: XSPI_CMD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command 1 Value. Command 1 register field.

Command Register 2

The XSPI_CMD2 register is only used in CDMA/PIO and STIG work mode. Field encoding of the register depends on the selected work mode.

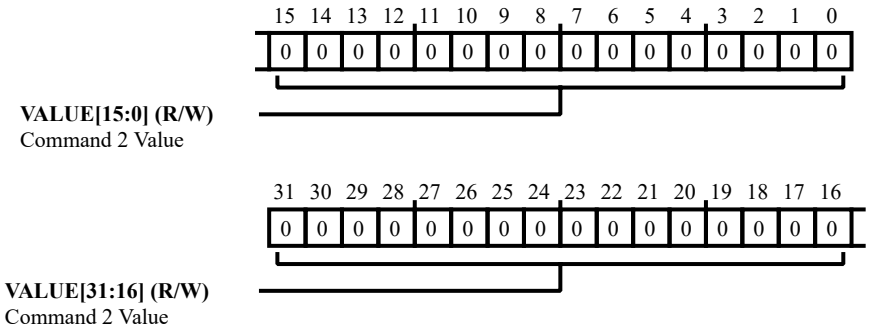


Figure 21-14: XSPI_CMD2 Register Diagram

Table 21-56: XSPI_CMD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command 2 Value. Command 2 register field.

Command Register 3

The XSPI_CMD3 register is only used in CDMA/PIO and STIG work mode; the definition changes depending on the work mode.

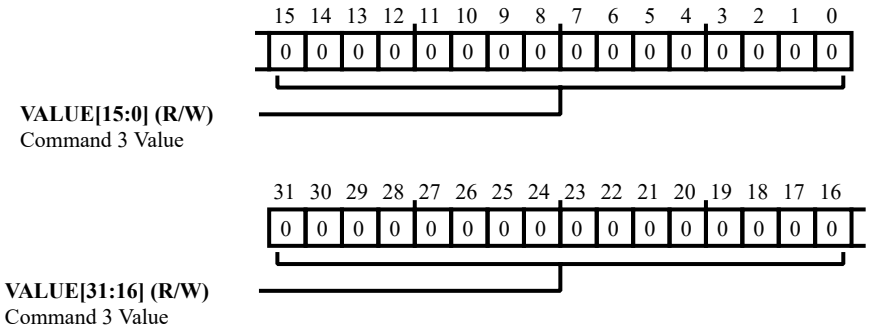


Figure 21-15: XSPI_CMD3 Register Diagram

Table 21-57: XSPI_CMD3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command 3 Value. Command 3 register field.

Command Register 4

The XSPI_CMD4 register is only used in CDMA/PIO and STIG work mode; the definition changes depending on the work mode.

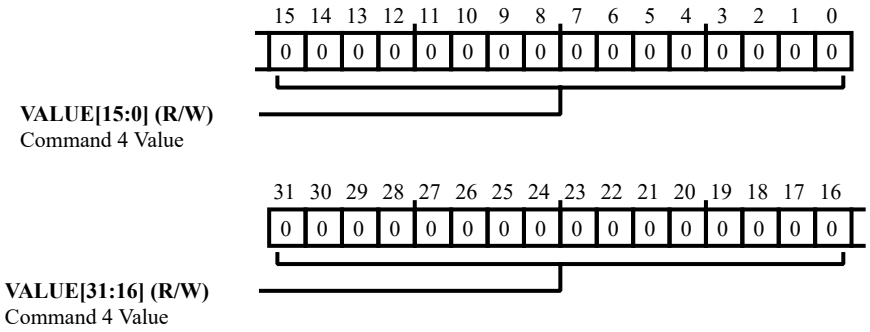


Figure 21-16: XSPI_CMD4 Register Diagram

Table 21-58: XSPI_CMD4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command 4 Value. Command 4 register field.

Command Register 5

The XSPI_CMD5 register is only used in CDMA/PIO and STIG work mode; the definition changes depending on the work mode.

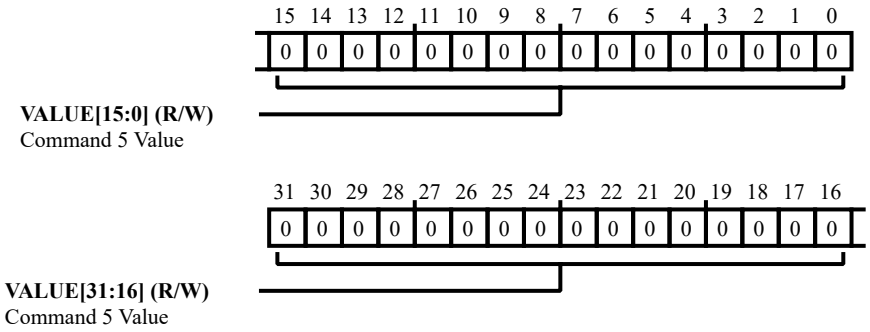


Figure 21-17: XSPI_CMD5 Register Diagram

Table 21-59: XSPI_CMD5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command 5 Value. Command 5 register field.

Command Status Register

The XSPI_CMD_STAT register reports the command status for the selected thread in ACMD and STIG work mode, when an xSPI flash transaction has completed.

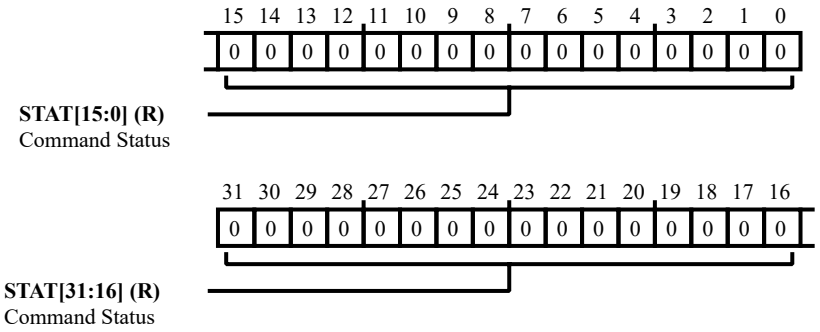


Figure 21-18: XSPI_CMD_STAT Register Diagram

Table 21-60: XSPI_CMD_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	STAT	Command Status. When operating in ACMD mode, the XSPI_CMD_STAT.STAT bit field reports the descriptor status associated with the selected thread (the thread to which this status belongs is selected via the XSPI_CMD_STAT_PTR register). When operating in STIG mode, this register reports the STIG completion status. Refer to section "STIG Mode Status" for a full bitwise definition.

Command Status Pointer Register

The XSPI_CMD_STAT_PTR register selects the thread that is used for outputting the command status in the XSPI_CMD_STAT register.

This register is relevant for ACMD mode only.

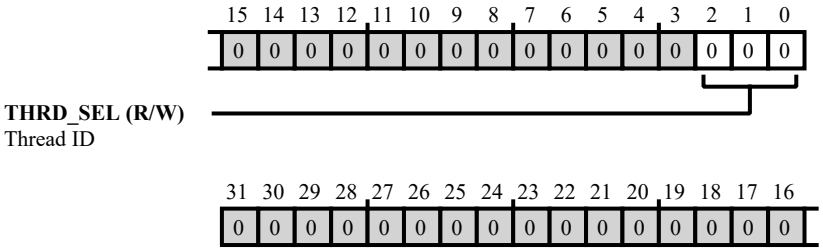


Figure 21-19: XSPI_CMD_STAT_PTR Register Diagram

Table 21-61: XSPI_CMD_STAT_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	THRD_SEL	Thread ID. The XSPI_CMD_STAT_PTR.THRD_SEL bit field indicates the thread ID status; the status is available in the XSPI_CMD_STAT register.

DAC Configuration Register

The XSPI_DAC_CFG register holds specific configuration information required while operating in direct work mode.

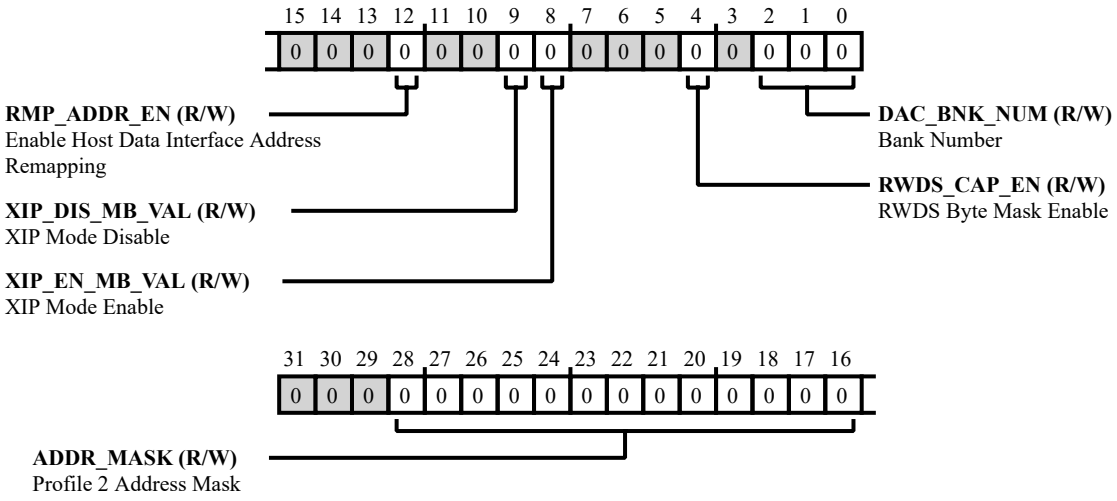


Figure 21-20: XSPI_DAC_CFG Register Diagram

Table 21-62: XSPI_DAC_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:16 (R/W)	ADDR_MASK	Profile 2 Address Mask. The XSPI_DAC_CFG.ADDR_MASK bit field enables masking bits [44:32] of the system address for read/write transfers for profile 2.
12 (R/W)	RMP_ADDR_EN	Enable Host Data Interface Address Remapping. When the XSPI_DAC_CFG.RMP_ADDR_EN bit is set (=1), the incoming AXI host address is adapted and sent to the flash device as (address - N), where N is the value stored in the remap address register (XSPI_DAC_REMAPADDRn).
9 (R/W)	XIP_DIS_MB_VAL	XIP Mode Disable. The XSPI_DAC_CFG.XIP_DIS_MB_VAL bit is used to trigger exit from XIP mode within the device. When XSPI_DAC_CFG.XIP_DIS_MB_VAL is set (=1), the controller sends mode bits specified in the XSPI_XIP_GCTL. XIP_DIS_MB_VALUE on the next read transaction. This disables XIP work mode for both the device and controller.
8 (R/W)	XIP_EN_MB_VAL	XIP Mode Enable. The XSPI_DAC_CFG.XIP_EN_MB_VAL bit is used to trigger entry to XIP mode within the device. When XSPI_DAC_CFG.XIP_EN_MB_VAL is set (=1), the controller sends mode bits as specified in XSPI_XIP_GCTL. XIP_EN_MB_VAL on the next read transaction. This switches both device and controller into XIP work mode.

Table 21-62: XSPI_DAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	RWDS_CAP_EN	RWDS Byte Mask Enable. When enabled, the controller supports byte masking in the connected device by automatically translating the incoming AXI host interface write strobes as needed. If the connected device does not support RWDS byte masking, the `XSPI_DAC_CFG.RWDS_CAP_EN` bit must be cleared. Note that when the `XSPI_DAC_CFG.RWDS_CAP_EN` bit is cleared, the controller assumes all bytes on the AXI write channel have an equivalent AXI write strobe set to 1. Also note that when this bit is cleared, the user cannot send byte writes to an octal DDR device or to a device that is 16-bit addressed.
2:0 (R/W)	DAC_BNK_NUM	Bank Number. The `XSPI_DAC_CFG.DAC_BNK_NUM` bit field indicates the bank (the attached memory device) targeted by the controller while operating in direct mode.

Address Remapping Register 0

When XSPI_DAC_CFG.RMP_ADDR_EN =1, the incoming AXI host interface address is adapted and sent to the flash device as (address - N), where N[31:0] is the value stored in this register.

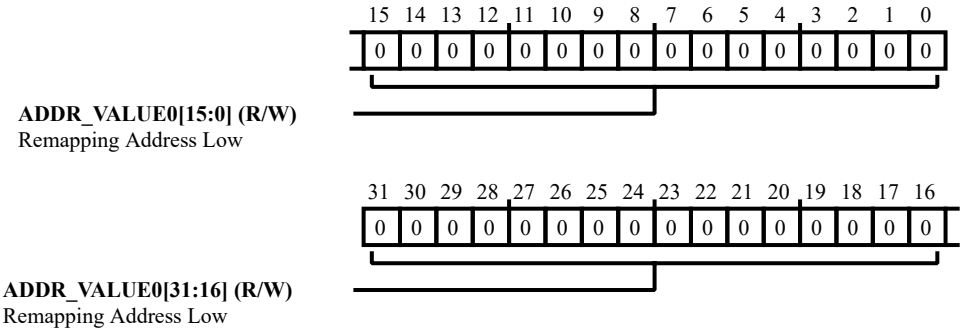


Figure 21-21: XSPI_DAC_REMAPADDR0 Register Diagram

Table 21-63: XSPI_DAC_REMAPADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR_VALUE0	Remapping Address Low. The `XSPI_DAC_REMAPADDR0.ADDR_VALUE0` bit field indicates the remapping of the incoming address on the AXI host interface to a different address used by the flash device. The value of this register must be aligned to 8 bytes.

Address Remapping Register 1

When XSPI_DAC_CFG.RMP_ADDR_EN =1, the incoming host interface address is adapted and sent to the flash device as (address - N), where N[63:32] is the value stored in this register.

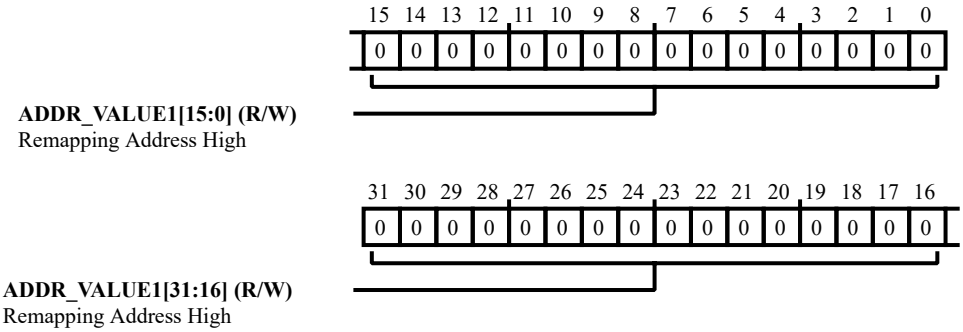


Figure 21-22: XSPI_DAC_REMAPADDR1 Register Diagram

Table 21-64: XSPI_DAC_REMAPADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR_VALUE1	Remapping Address High. The `XSPI_DAC_REMAPADDR1.ADDR_VALUE1` bit field indicates the remapping of the incoming address on the host interface to a different address used by the flash device.

Device Discovery Control Register

The XSPI_DISCOVERY_GCTL register controls device discovery.

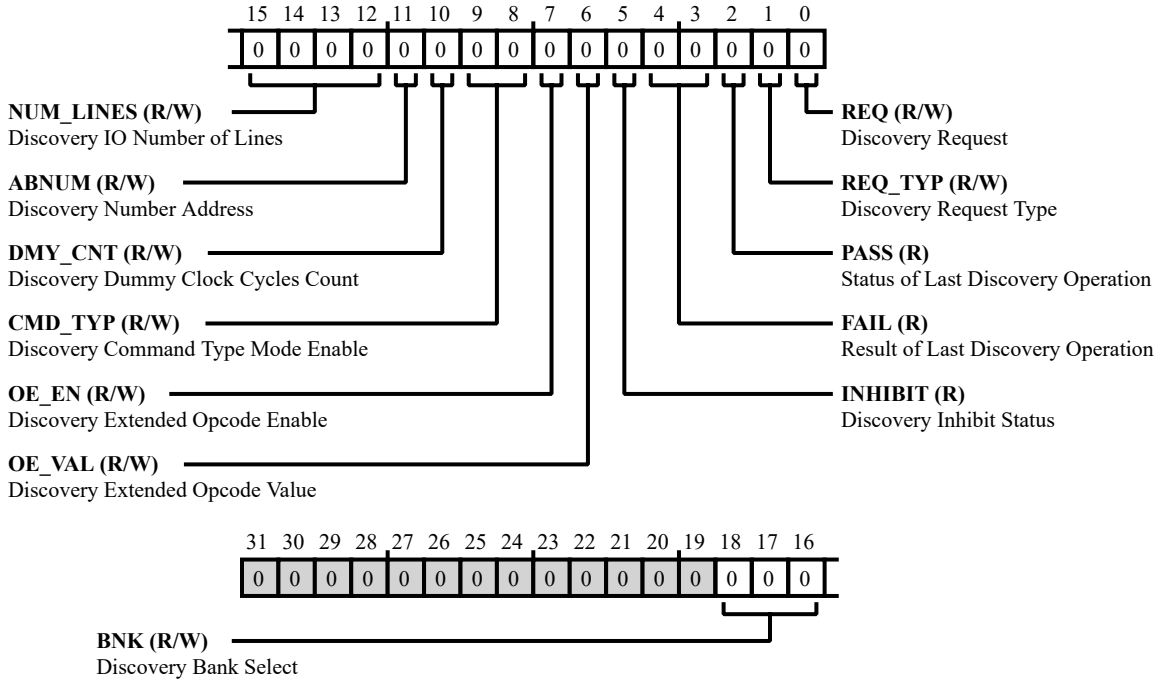


Figure 21-23: XSPI_DISCOVERY_GCTL Register Diagram

Table 21-65: XSPI_DISCOVERY_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	BNK	Discovery Bank Select. The XSPI_DISCOVERY_GCTL.BNK bit field indicates the discovery bank. Writing a value of 0x0-0x7 selects a bank. This field is updated after the initialization process.	
15:12 (R/W)	NUM_LINES	Discovery IO Number of Lines. This is a 4-bit value used in discovery mode. Writing a value to the XSPI_DISCOVERY_GCTL.NUM_LINES bit field selects number of xSPI I/Os used by device discovery. This field is updated after the initialization process.	
		0	Auto
		1	1 line
		2	2 lines
		3	Reserved
		4	4 lines

Table 21-65: XSPI_DISCOVERY_GCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	Reserved
		6	Reserved
		7	Reserved
		8	8 lines
		9	Reserved
		10	Reserved
		11	Reserved
		12	8 lines for legacy HyperFlash and xSPI profile
		13	Reserved
		14	1 line for legacy SPI NAND
		15	Reserved
11 (R/W)	ABNUM	Discovery Number Address. The `XSPI_DISCOVERY_GCTL.ABNUM` bit enables discovery 4-bit addressing. This field is updated after the initialization process.	
		0	3-bit addressing
		1	4-bit addressing
10 (R/W)	DMY_CNT	Discovery Dummy Clock Cycles Count. The `XSPI_DISCOVERY_GCTL.DMY_CNT` bit indicates the discovery number of dummy clock cycles. This bit is updated after the initialization process.	
		0	8 dummy clock cycles
		1	20 dummy clock cycles
9:8 (R/W)	CMD_TYP	Discovery Command Type Mode Enable. The `XSPI_DISCOVERY_GCTL.CMD_TYP` bit selects the discovery command type mode. This field is updated after the initialization process.	
		0	SDR mode enabled
		1	DDR mode enabled
		2	DTR mode enabled (quad mode only)
		3	Reserved

Table 21-65: XSPI_DISCOVERY_GCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	OE_EN	Discovery Extended Opcode Enable. The `XSPI_DISCOVERY_GCTL.OE_EN` bit enables the discovery extended opcode. This field is updated after the initialization process.	
		0	Disable
		1	Enable
6 (R/W)	OE_VAL	Discovery Extended Opcode Value. The `XSPI_DISCOVERY_GCTL.OE_VAL` bit field indicates the discovery extended opcode value. This field is updated after the initialization process.	
		0	Extended opcode is 8'hA5
		1	Extended opcode is 8'h5A
5 (R/NW)	INHIBIT	Discovery Inhibit Status. The `XSPI_DISCOVERY_GCTL.INHIBIT` bit indicates whether device discovery is inhibited at power on.	
		0	Discovery allow
		1	Discovery inhibit
4:3 (R/NW)	FAIL	Result of Last Discovery Operation. The `XSPI_DISCOVERY_GCTL.FAIL` bit field indicates the result of the last discovery operation. Valid when PASS = 1.	
		0	xSPI or SPI NAND device detected
		1	Fail
		2	Legacy SPI device detected
		3	N/A
2 (R/NW)	PASS	Status of Last Discovery Operation. The `XSPI_DISCOVERY_GCTL.PASS` bit indicates the status of the last discovery operation. `XSPI_DISCOVERY_GCTL.PASS` =1 when the device discovery operation has finished. The result can be read from the FAIL field.	
1 (R/W)	REQ_TYP	Discovery Request Type. The `XSPI_DISCOVERY_GCTL.REQ_TYP` bit indicates the discovery request type.	
		0	Full discovery. Perform full discovery process (try to detect device)
		1	Selected mode. Configure registers only to selected mode (not full discovery process)

Table 21-65: XSPI_DISCOVERY_GCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	REQ	Discovery Request. Writing 1 to the `XSPI_DISCOVERY_GCTL.REQ` bit triggers the device discovery operation. The `XSPI_DISCOVERY_GCTL.REQ` bit is cleared by hardware when the device discovery operation completes.

DMA Interface Control Register

The XSPI_DMA_CTL register is a common register for both the host and internal DMA interface.

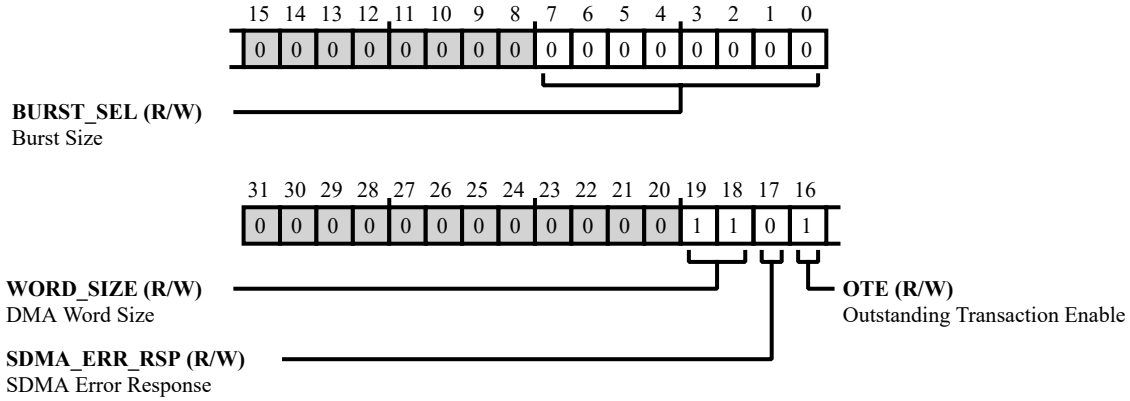


Figure 21-24: XSPI_DMA_CTL Register Diagram

Table 21-66: XSPI_DMA_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19:18 (R/W)	WORD_SIZE	DMA Word Size. The XSPI_DMA_CTL.WORD_SIZE bit field indicates the AXI transaction size used by the internal DMA interface to transfer data.	
		0	Byte
		1	16-bit word
		2	32-bit word
		3	64-bit word
17 (R/W)	SDMA_ERR_RSP	SDMA Error Response. AXI error responses can only occur in ACMD or STIG work modes. When XSPI_DMA_CTL.SDMA_ERR_RSP is set (=1), an AXI error response is returned if one of the following is true. 1. The host attempts to access the target interface before it is permitted (refer to the required steps in STIG and ACMD sections of this document). 2. The host issues an unsupported burst type. The controller only supports incremental bursts (non-wrapping) bursts currently. When XSPI_DMA_CTL.SDMA_ERR_RSP is cleared (=0), an OK response is returned.	
16 (R/W)	OTE	Outstanding Transaction Enable. The XSPI_DMA_CTL.OTE bit enables an outstanding transaction. This only applies to the internal DMA interface. The AXI host interface ignores this bit and accepts all incoming transactions.	

Table 21-66: XSPI_DMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	BURST_SEL	Burst Size. The `XSPI_DMA_CTL.BURST_SEL` bit field indicates the maximum AXI burst size used by the internal DMA interface. The maximum burst size can be calculated as `XSPI_DMA_CTL.BURST_SEL`+1. Note that this field must be changed only when controller is in an idle state.

DMA Error Address High Register

Internal DMA interface error address [63:32]. The XSPI_DMA_ERRADDR_HI register holds the upper 32 bits of the address of the AXI request on the system bus requester data interface that caused the CDMA_TERR or DDMA_TERR bits in the XSPI_ISTAT register to be set. This register is overwritten when further error responses are detected.

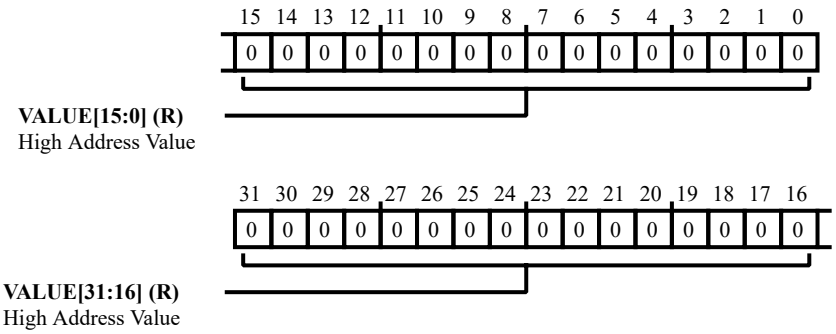


Figure 21-25: XSPI_DMA_ERRADDR_HI Register Diagram

Table 21-67: XSPI_DMA_ERRADDR_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	High Address Value. The XSPI_DMA_ERRADDR_HI.VALUE bit field indicates the address of the first request on the internal DMA interface that returned an error response.

DMA Error Address Low Register

Internal DMA interface error address [31:0]. The XSPI_DMA_ERRADDR_LO register holds the lower 32 bits of the address of the AXI request on the system bus requester data interface that caused the CDMA_TERR or DDMA_TERR bits in the XSPI_ISTAT register to be set. The XSPI_DMA_ERRADDR_LO register is overwritten when further error responses are detected.

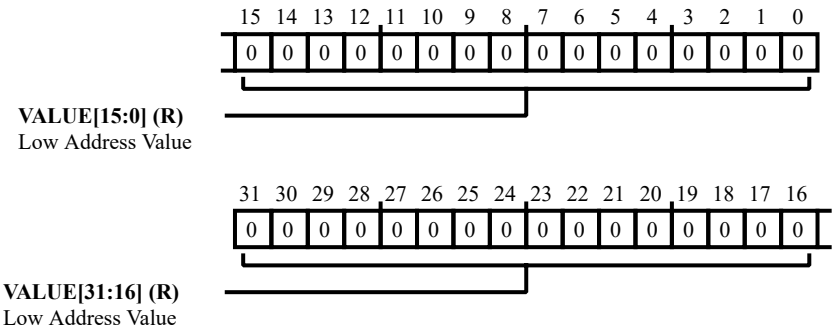


Figure 21-26: XSPI_DMA_ERRADDR_LO Register Diagram

Table 21-68: XSPI_DMA_ERRADDR_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Low Address Value. The XSPI_DMA_ERRADDR_LO.VALUE bit field indicates the address of the first request on the internal DMA interface that returned an error response.

Erase Sequence Configuration Register 0

The XSPI_ERS_SEQ_CFG0 register configures the erase_sector sequence for profile 1 and SPI NAND in ACMD work mode.

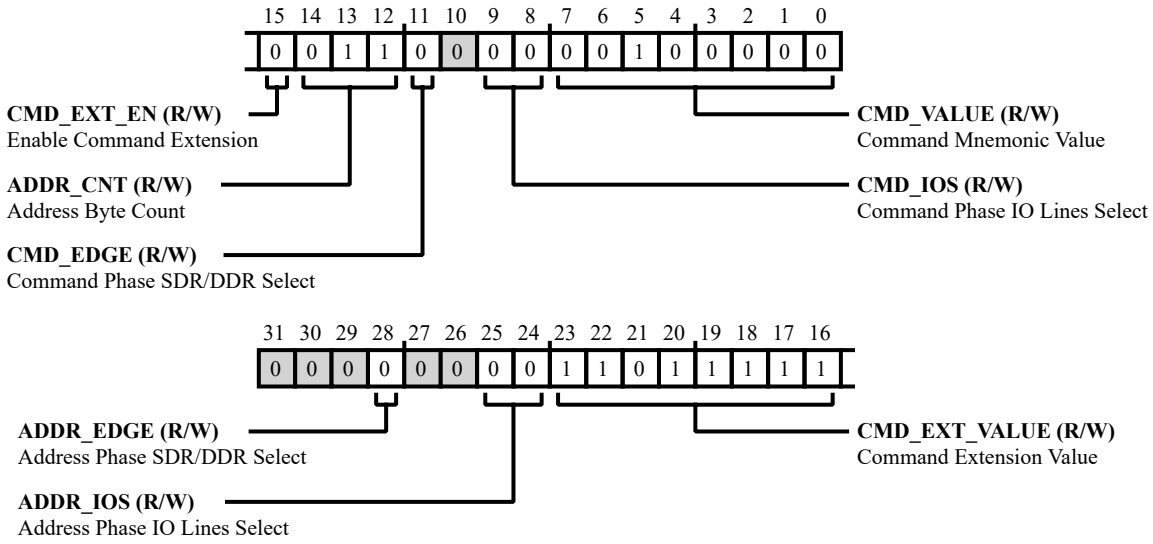


Figure 21-27: XSPI_ERS_SEQ_CFG0 Register Diagram

Table 21-69: XSPI_ERS_SEQ_CFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	ADDR_EDGE	Address Phase SDR/DDR Select. The XSPI_ERS_SEQ_CFG0.ADDR_EDGE bit selects between SDR/DDR mode for the address phase.	
25:24 (R/W)	ADDR_IOS	Address Phase IO Lines Select. The XSPI_ERS_SEQ_CFG0.ADDR_IOS bit field indicates the number of data lines used to send the address phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
23:16 (R/W)	CMD_EXT_VALUE	Command Extension Value. The XSPI_ERS_SEQ_CFG0.CMD_EXT_VALUE bit field indicates the command extension value, when enabled.	
15 (R/W)	CMD_EXT_EN	Enable Command Extension. The XSPI_ERS_SEQ_CFG0.CMD_EXT_EN bit enables command extension.	

Table 21-69: XSPI_ERS_SEQ_CFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/W)	ADDR_CNT	Address Byte Count. The `XSPI_ERS_SEQ_CFG0.ADDR_CNT` bit field indicates the number of address bytes.	
11 (R/W)	CMD_EDGE	Command Phase SDR/DDR Select. The `XSPI_ERS_SEQ_CFG0.CMD_EDGE` bit field selects between SDR/DDR mode for the command phase.	
9:8 (R/W)	CMD_IOS	Command Phase IO Lines Select. The `XSPI_ERS_SEQ_CFG0.CMD_IOS` bit field indicates the number of data lines used to send the command phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
7:0 (R/W)	CMD_VALUE	Command Mnemonic Value. The `XSPI_ERS_SEQ_CFG0.CMD_VALUE` bit field indicates the command mnemonic value.	

Erase Sequence Configuration Register 1

The XSPI_ERS_SEQ_CFG1 register configures the erase_sector sequence for profile 1 and SPI NAND in ACMD work mode.

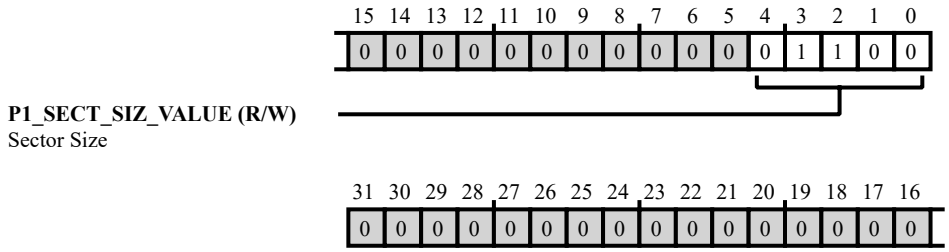


Figure 21-28: XSPI_ERS_SEQ_CFG1 Register Diagram

Table 21-70: XSPI_ERS_SEQ_CFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	P1_SECT_SIZ_VALUE	Sector Size. The XSPI_ERS_SEQ_CFG1.P1_SECT_SIZ_VALUE bit field indicates the sector size. Value encoded as $2^{erss_seq_p1_sect_size}$: 8'h00 - 1B 8'h01 - 2B 8'h02 - 4B ... 8'h0f - 32kB 8'h10 - 64kB ... 8'h1f - (2^{31})B

Erase Sequence Configuration Register 2

The XSPI_ERS_SEQ_CFG2 register configures the erase_all sequence for profile 1 in ACMD work mode.

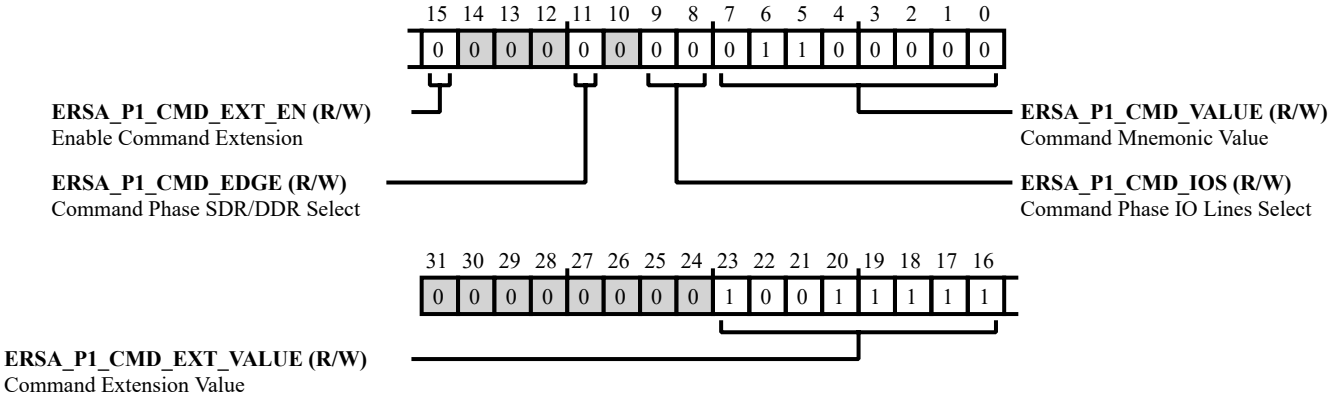


Figure 21-29: XSPI_ERS_SEQ_CFG2 Register Diagram

Table 21-71: XSPI_ERS_SEQ_CFG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	ERSA_P1_CMD_EXT_VALUE	Command Extension Value. The XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_EXT_VALUE bit field indicates the command extension value.	
15 (R/W)	ERSA_P1_CMD_EXT_EN	Enable Command Extension. The XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_EXT_EN bit field enables the command extension.	
11 (R/W)	ERSA_P1_CMD_EDGE	Command Phase SDR/DDR Select. The XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_EDGE bit selects between SDR/DDR mode for the command phase.	
9:8 (R/W)	ERSA_P1_CMD_IOS	Command Phase IO Lines Select. The XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_IOS bit field indicates the number of data lines used to send the command phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
7:0 (R/W)	ERSA_P1_CMD_VALUE	Command Mnemonic Value. The XSPI_ERS_SEQ_CFG2.ERSA_P1_CMD_VALUE bit field indicates the command mnemonic value.	

Controller Features Register

The XSPI_FEATURES register shows the available hardware features of the controller.

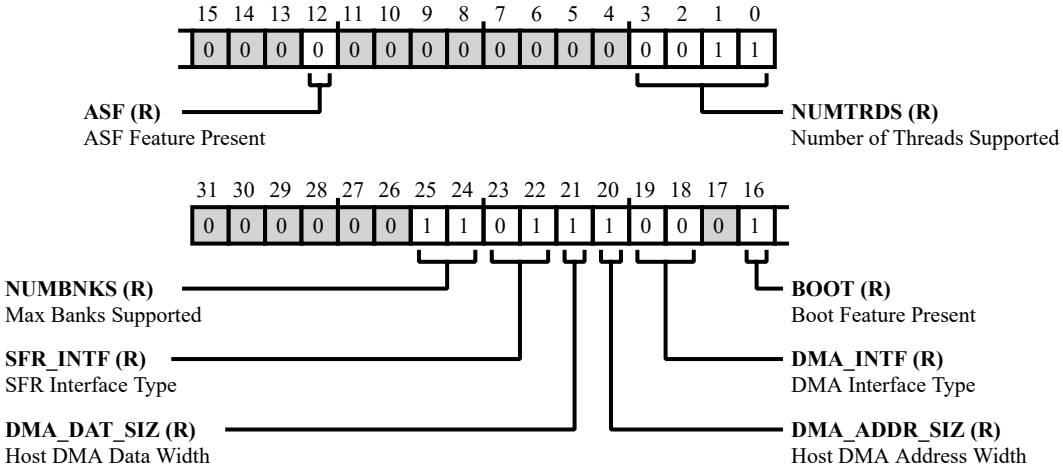


Figure 21-30: XSPI_FEATURES Register Diagram

Table 21-72: XSPI_FEATURES Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	NUMBNKS	Max Banks Supported. The XSPI_FEATURES.NUMBNKS bit field indicates the maximum number of banks supported by hardware.	
		0	One bank
		1	Two banks
		2	Four banks
		3	Eight banks
23:22 (R/NW)	SFR_INTF	SFR Interface Type. The XSPI_FEATURES.SFR_INTF bit field indicates the SFR interface type 1-APB, other values reserved.	
21 (R/NW)	DMA_DAT_SIZ	Host DMA Data Width. The XSPI_FEATURES.DMA_DAT_SIZ bit indicates the host DMA data width.	
		0	32 bit
		1	64 bit

Table 21-72: XSPI_FEATURES Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	DMA_ADDR_SIZ	Host DMA Address Width. The `XSPI_FEATURES.DMA_ADDR_SIZ` bit indicates the host DMA address width.	
		0	32 bit
		1	64 bit
19:18 (R/NW)	DMA_INTF	DMA Interface Type. The `XSPI_FEATURES.DMA_INTF` bit field indicates the DMA interface type (0-AXI4 other values reserved).	
16 (R/NW)	BOOT	Boot Feature Present. The `XSPI_FEATURES.BOOT` bit indicates whether the boot feature is present.	
12 (R/NW)	ASF	ASF Feature Present. The `XSPI_FEATURES.ASF` bit indicates whether the ASF feature is present.	
3:0 (R/NW)	NUMTRDS	Number of Threads Supported. The `XSPI_FEATURES.NUMTRDS` bit field indicates the number of threads available in the controller.	
		0	One thread
		1	Two threads
		2	Four threads
		3	Eight threads
		4-15	Reserved

General Controller Status Register

The XSPI_GSTAT register indicates the status of the controller.

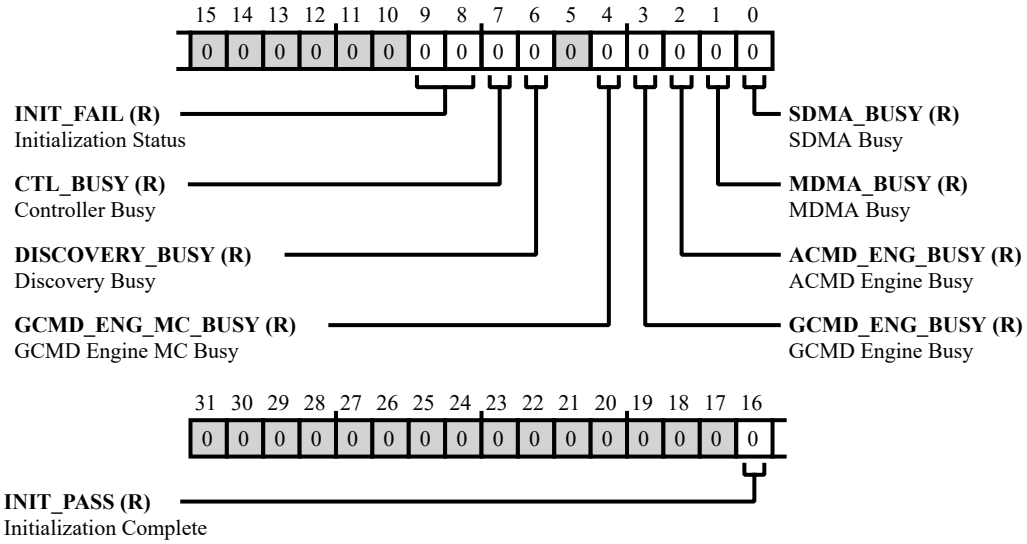


Figure 21-31: XSPI_GSTAT Register Diagram

Table 21-73: XSPI_GSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	INIT_PASS	Initialization Complete. When XSPI_GSTAT.INIT_PASS is set (=1), the xSPI controller has completed its reset and initialization process.	
9:8 (R/NW)	INIT_FAIL	Initialization Status. The XSPI_GSTAT.INIT_FAIL bit field indicates the initialization process status.	
		0	xSPI device detected
		1	Initialization has failed
		2	Legacy SPI device detected
		3	N/A
7 (R/NW)	CTL_BUSY	Controller Busy. The XSPI_GSTAT.CTL_BUSY bit indicates whether controller is in the busy state. Note that this bit is also routed to the controller interface via the CTRL_BUSY pin.	
		0	Idle
		1	Busy

Table 21-73: XSPI_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/NW)	DISCOVERY_BUSY	Discovery Busy. When `XSPI_GSTAT.DISCOVERY_BUSY` is set (=1), device discovery is in progress. When device discovery is inhibited, this bit is also set during the PHY initialization procedure.
4 (R/NW)	GCMD_ENG_MC_BUSY	GCMD Engine MC Busy. The bit `XSPI_GSTAT.GCMD_ENG_MC_BUSY` is relevant for STIG mode only. This bit indicates when the controller is waiting for next/last instruction in a glued instruction chain or it is executing a requested sequence on the xSPI interface.
3 (R/NW)	GCMD_ENG_BUSY	GCMD Engine Busy. When operating in direct work mode, the `XSPI_GSTAT.GCMD_ENG_BUSY` is set (=1), when the direct CMD generator is busy. Note that no new requests on the host interface are accepted while `XSPI_GSTAT.GCMD_ENG_BUSY` is set. When operating in STIG work mode, the `XSPI_GSTAT.GCMD_ENG_BUSY` is set while the STIG engine is busy.
2 (R/NW)	ACMD_ENG_BUSY	ACMD Engine Busy. When `XSPI_GSTAT.ACMD_ENG_BUSY` is set (=1), the auto command engine is busy (ACMD mode only).
1 (R/NW)	MDMA_BUSY	MDMA Busy. When `XSPI_GSTAT.MDMA_BUSY` is set (=1), the internal DMA interface is busy.
0 (R/NW)	SDMA_BUSY	SDMA Busy. When `XSPI_GSTAT.SDMA_BUSY` is set (=1), the host interface is busy.

Interrupt Enable Register

If the selected bit of the XSPI_INT_EN register is set (=1), the rising edge of the corresponding bit in the XSPI_ISTAT register triggers an interrupt.

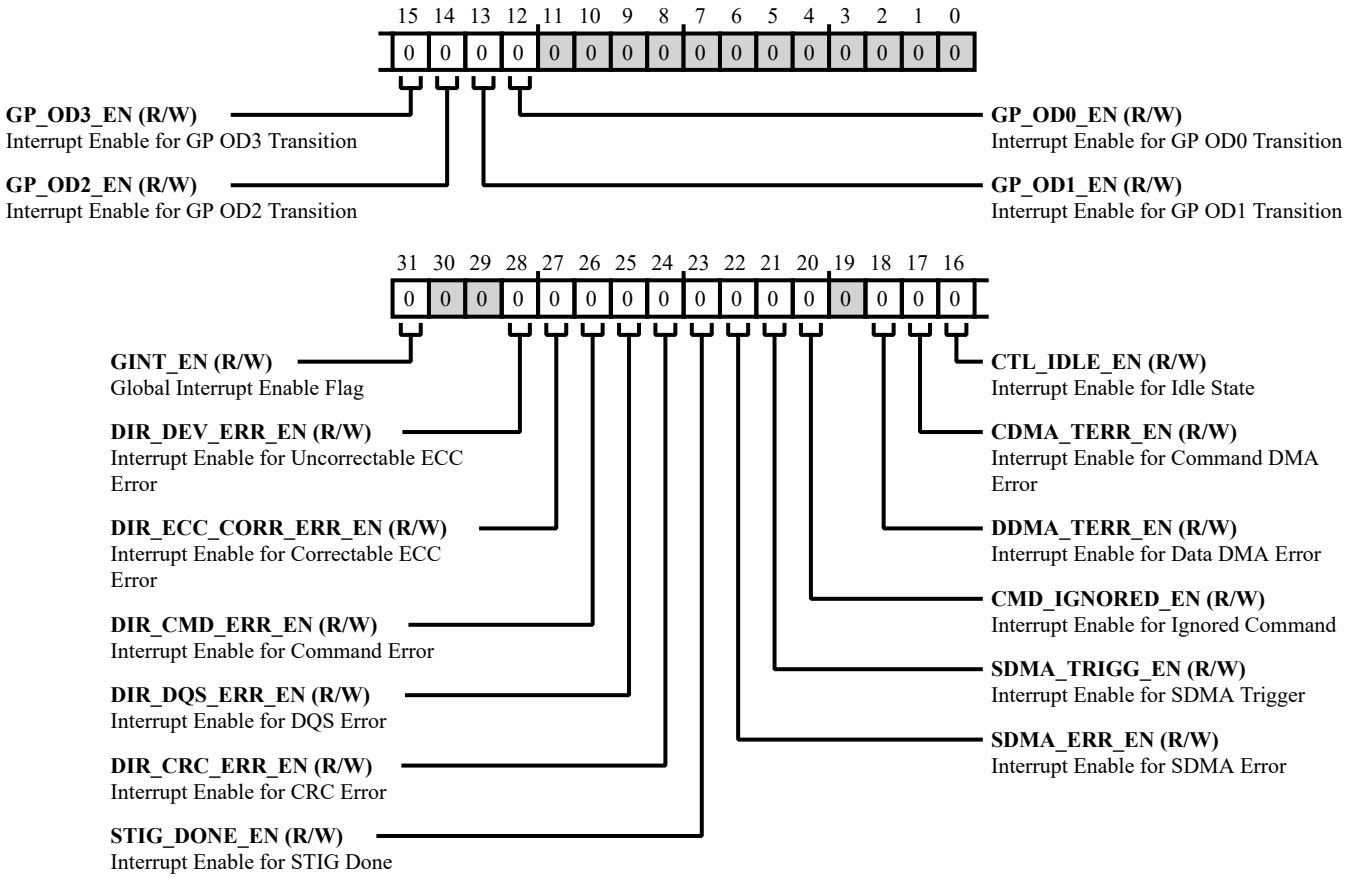


Figure 21-32: XSPI_INT_EN Register Diagram

Table 21-74: XSPI_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	GINT_EN	Global Interrupt Enable Flag. The XSPI_INT_EN.GINT_EN bit enables a global interrupt flag.
28 (R/W)	DIR_DEV_ERR_EN	Interrupt Enable for Uncorrectable ECC Error. The XSPI_INT_EN.DIR_DEV_ERR_EN bit enables an interrupt when an uncorrectable ECC or program fail error occurred in direct work mode.
27 (R/W)	DIR_ECC_CORR_ERR_EN	Interrupt Enable for Correctable ECC Error. The XSPI_INT_EN.DIR_ECC_CORR_ERR_EN bit enables an interrupt when a correctable ECC error occurred in direct work mode.

Table 21-74: XSPI_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	DIR_CMD_ERR_EN	Interrupt Enable for Command Error. The `XSPI_INT_EN.DIR_CMD_ERR_EN` bit enables an interrupt when an invalid command sequence has been detected in direct work mode.
25 (R/W)	DIR_DQS_ERR_EN	Interrupt Enable for DQS Error. The `XSPI_INT_EN.DIR_DQS_ERR_EN` bit enables an interrupt when the controller returns a DQS error after a read or status checking command in direct work mode.
24 (R/W)	DIR_CRC_ERR_EN	Interrupt Enable for CRC Error. The `XSPI_INT_EN.DIR_CRC_ERR_EN` bit enables an interrupt when the controller returns a CRC error after a read or status checking command in direct work mode.
23 (R/W)	STIG_DONE_EN	Interrupt Enable for STIG Done. The `XSPI_INT_EN.STIG_DONE_EN` bit enables an interrupt when an instruction in a glued chain is complete.
22 (R/W)	SDMA_ERR_EN	Interrupt Enable for SDMA Error. The `XSPI_INT_EN.SDMA_ERR_EN` bit enables an interrupt when an illegal access to the host DMA interface is detected.
21 (R/W)	SDMA_TRIGG_EN	Interrupt Enable for SDMA Trigger. The `XSPI_INT_EN.SDMA_TRIGG_EN` bit enables an interrupt when the trigger condition for the host DMA is met.
20 (R/W)	CMD_IGNORED_EN	Interrupt Enable for Ignored Command. The `XSPI_INT_EN.CMD_IGNORED_EN` bit enables an interrupt for detection of an ignored command.
18 (R/W)	DDMA_TERR_EN	Interrupt Enable for Data DMA Error. The `XSPI_INT_EN.DDMA_TERR_EN` bit enables an interrupt for detecting a data internal DMA error.
17 (R/W)	CDMA_TERR_EN	Interrupt Enable for Command DMA Error. The `XSPI_INT_EN.CDMA_TERR_EN` bit enables an interrupt for detecting an auto CMD engine target error.
16 (R/W)	CTL_IDLE_EN	Interrupt Enable for Idle State. The `XSPI_INT_EN.CTL_IDLE_EN` bit enables an interrupt for detecting that the controller has returned to the idle state.
15 (R/W)	GP_OD3_EN	Interrupt Enable for GP OD3 Transition. The `XSPI_INT_EN.GP_OD3_EN` bit enables an interrupt for detecting the high-to-low or low-to-high transition on the XSPI_DFI_GP_OPEN_DRAIN[3] input pin.

Table 21-74: XSPI_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	GP_OD2_EN	Interrupt Enable for GP OD2 Transition. The `XSPI_INT_EN.GP_OD2_EN` bit enables an interrupt for detecting the high-to-low or low-to-high transition on the XSPI_DFI_GP_OPEN_DRAIN[2] input pin.
13 (R/W)	GP_OD1_EN	Interrupt Enable for GP OD1 Transition. The `XSPI_INT_EN.GP_OD1_EN` bit enables an interrupt for detecting the high-to-low or low-to-high transition on the XSPI_DFI_GP_OPEN_DRAIN[1] input pin.
12 (R/W)	GP_OD0_EN	Interrupt Enable for GP OD0 Transition. The `XSPI_INT_EN.GP_OD0_EN` bit enables an interrupt for detecting the high-to-low or low-to-high transition on the XSPI_DFI_GP_OPEN_DRAIN[0] input pin.

Interrupt Status Register

The XSPI_ISTAT register contains information about functional areas requiring servicing. Many of the bits serve as an indicator to further read and service various status registers. After servicing the interrupt source associated with a bit, the user must clear that interrupt source bit by writing a 1 to it.

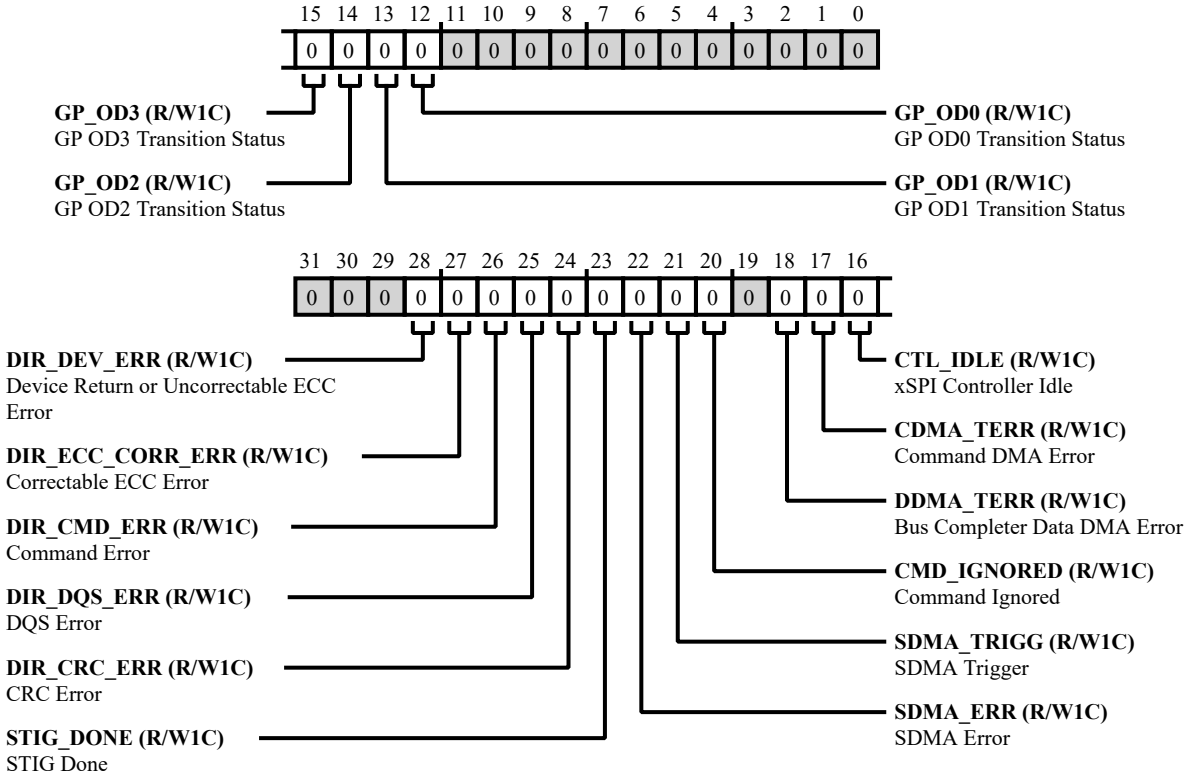


Figure 21-33: XSPI_ISTAT Register Diagram

Table 21-75: XSPI_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28 (R/W1C)	DIR_DEV_ERR	Device Return or Uncorrectable ECC Error. The XSPI_ISTAT.DIR_DEV_ERR bit is set (=1) when a program operation in direct mode failed. That is, the device returned the program fail bit when the status was checked. This bit is also set when an uncorrectable ECC error occurs in direct mode.
27 (R/W1C)	DIR_ECC_CORR_ERR	Correctable ECC Error. The XSPI_ISTAT.DIR_ECC_CORR_ERR bit is set (=1) when a correctable ECC error occurs in direct mode.

Table 21-75: XSPI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W1C)	DIR_CMD_ERR	Command Error. The XSPI_ISTAT.DIR_CMD_ERR bit is set (=1) when the host has attempted to do something that is unsupported or incorrect. For example, the host has triggered a write when the device is operating in XIP mode. The full list of error conditions is described in the Event Handling section.
25 (R/W1C)	DIR_DQS_ERR	DQS Error. The XSPI_ISTAT.DIR_DQS_ERR bit is set (=1) when an incorrect number of DQS pulses were detected during a direct mode read or status check. The status is passed from the soft PHY and could mean that the PHY is configured badly such that it expects DQS strobes but never received them or that the XSPI_PHY_GATE_LPBK_CTL .RD_DEL_SEL value is incorrect and has caused data corruption and pointer misalignment in the PHY. To resolve this issue, the PHY must be reset to clear the XSPI_PHY_OB0.DQS_UR and XSPI_PHY_OB0.DQS_OV flags and reset the read data pointers.
24 (R/W1C)	DIR_CRC_ERR	CRC Error. The XSPI_ISTAT.DIR_CRC_ERR bit is set (=1) when CRC checking is enabled and a CRC error after a read or status checking command in direct work mode has been detected.
23 (R/W1C)	STIG_DONE	STIG Done. The XSPI_ISTAT.STIG_DONE bit is set (=1) when the last instruction in a glued chain has completed.
22 (R/W1C)	SDMA_ERR	SDMA Error. The XSPI_ISTAT.SDMA_ERR bit is set (=1) when an illegal access to the target DMA interface is detected.
21 (R/W1C)	SDMA_TRIGG	SDMA Trigger. The XSPI_ISTAT.SDMA_TRIGG bit indicates when the trigger condition for the target DMA is met. It is relevant to STIG and ACMD mode only and used to instruct the host when an access on the host interface is to be performed. For details, see the Operating in ACMD or STIG mode section.
20 (R/W1C)	CMD_IGNORED	Command Ignored. In ACMD work mode, the controller detected that a command that was sent by the host was to an already busy thread and ignored it. In STIG work mode, the controller detected that a command was sent when the STIG engine was already busy and ignored it.
18 (R/W1C)	DDMA_TERR	Bus Completer Data DMA Error. ACMD mode only. The XSPI_ISTAT.DDMA_TERR bit is set (=1) when a system bus error was detected on the internal DMA interface during data reads or writes.

Table 21-75: XSPI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W1C)	CDMA_TERR	Command DMA Error. Command DMA target error. ACMD mode only. The `XSPI_ISTAT.CDMA_TERR` bit is set (=1) when a system bus error was detected on the internal DMA interface during descriptor reads or writes.
16 (R/W1C)	CTL_IDLE	xSPI Controller Idle. This is general status. The `XSPI_ISTAT.CTL_IDLE` bit indicates that the xSPI controller has returned to an idle state.
15 (R/W1C)	GP_OD3	GP OD3 Transition Status. The `XSPI_ISTAT.GP_OD3` bit indicates a high-to-low or low-to-high transition was detected on the XSPI_DFI_GP_OPEN_DRAIN[3] input pin.
14 (R/W1C)	GP_OD2	GP OD2 Transition Status. The `XSPI_ISTAT.GP_OD2` bit indicates a high-to-low or low-to-high transition detected on the XSPI_DFI_GP_OPEN_DRAIN[2] input pin.
13 (R/W1C)	GP_OD1	GP OD1 Transition Status. The `XSPI_ISTAT.GP_OD1` bit indicates a high-to-low or low-to-high transition detected on the XSPI_DFI_GP_OPEN_DRAIN[1] input pin.
12 (R/W1C)	GP_OD0	GP OD0 Transition Status. The `XSPI_ISTAT.GP_OD0` bit indicates a high-to-low or low-to-high transition detected on the XSPI_DFI_GP_OPEN_DRAIN[0] input pin.

Long Polling Count Register

The XSPI_LONGPOL_GCTL register contains the wait count value for long polling.

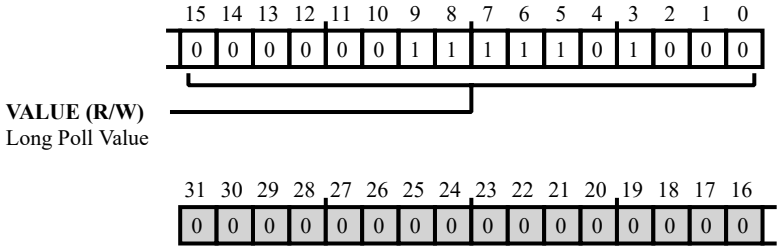


Figure 21-34: XSPI_LONGPOL_GCTL Register Diagram

Table 21-76: XSPI_LONGPOL_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Long Poll Value. The XSPI_LONGPOL_GCTL.VALUE bit field indicates the number of system clock cycles after an erase/write operation has been issued before the controller starts to check device status (ready/busy and fail/pass). First status checking polling happens after at least this many number of system clock cycles. The next status checking will happen every XSPI_SHORTPOLL_GCTL.VALUE cycles. The long polling value should be significantly larger the short polling value.

Device Active Maximum Clock Cycle Register

The XSPI_MINICTL_DEV_ACTIVE_MAX register is used to introduce a maximum number of xSPI clock cycles through which the CS number is kept active (low) on the memory interface. It is sometimes referred to as tCMS or tCEM timing. It is generally only relevant for RAM devices that must be periodically refreshed. The refresh is typically handled internal to the device. But, to fit this refresh operation along with normal controller requests, the controller my need to limit its access time. The user can program this register to force the controller to obey that requirement.

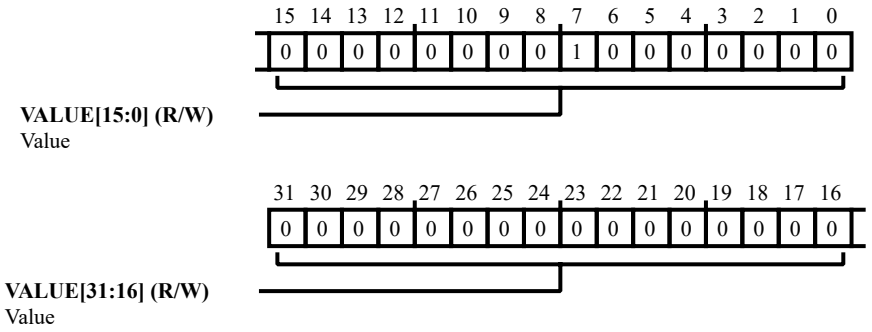


Figure 21-35: XSPI_MINICTL_DEV_ACTIVE_MAX Register Diagram

Table 21-77: XSPI_MINICTL_DEV_ACTIVE_MAX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Value. The XSPI_MINICTL_DEV_ACTIVE_MAX.VALUE bit field is only valid when XSPI_SEQ_GTL0.TCMS_EN is set (=1). When using the STIG work mode, this is also only valid when the TCMS_EN bit of the STIG instruction is set. The XSPI_MINICTL_DEV_ACTIVE_MAX.VALUE bit field must only be enabled while working with RAM devices that require timing constraint for chip select low pulse width (the most common name is tCMS or tCEM in device specification).

Clock Mode Control Register

The XSPI_MINICTL_CLKMODE register defines the CPOL/CPHA settings for legacy SPI mode (defaulting to mode 0).

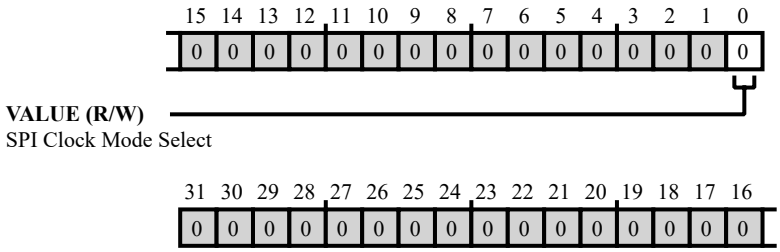


Figure 21-36: XSPI_MINICTL_CLKMODE Register Diagram

Table 21-78: XSPI_MINICTL_CLKMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	VALUE	SPI Clock Mode Select. The XSPI_MINICTL_CLKMODE.VALUE bit indicates the SPI clock mode. For DDR transfers, this bit must be cleared (=0) to meet DDR flash timings. For SDR transfers, allowable values are as follows: 0 - SPI mode 0 (clock is low when SPI bus is in idle) 1 - SPI mode 3 (clock is high when SPI bus is in idle)

Device Delay Register

The XSPI_MINICTL_DEV_DLY register is used to introduce relative device selection delays with respect to generated xSPI flash interface.

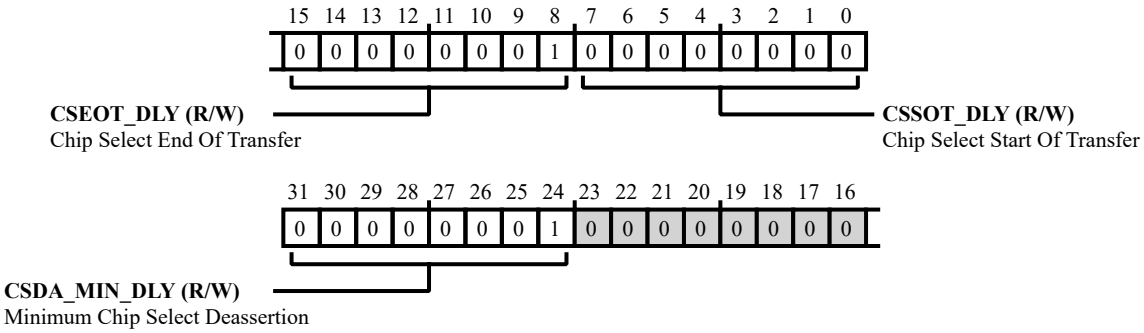


Figure 21-37: XSPI_MINICTL_DEV_DLY Register Diagram

Table 21-79: XSPI_MINICTL_DEV_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	CSDA_MIN_DLY	Minimum Chip Select Deassertion. The XSPI_MINICTL_DEV_DLY.CSDA_MIN_DLY bit field indicates the Minimum Chip Select (CSDA_MIN) deassertion timing.
15:8 (R/W)	CSEOT_DLY	Chip Select End Of Transfer. The XSPI_MINICTL_DEV_DLY.CSEOT_DLY bit field indicates the Chip Select End Of Transfer (CSEOT). It improves the CS deassertion device timing to the last active clock edge.
7:0 (R/W)	CSSOT_DLY	Chip Select Start Of Transfer. The XSPI_MINICTL_DEV_DLY.CSSOT_DLY bit field indicates the Chip Select Start Of Transfer (CSSOT). It improves the CS deassertion device timing to the first active clock edge.

HyperFlash Offset Register

The XSPI_MINICTL_HF_OFFSET register is only applicable for legacy HyperFlash or xSPI profile 2 devices and can be used to configure the reserved area of the Command/Address (CA) field.

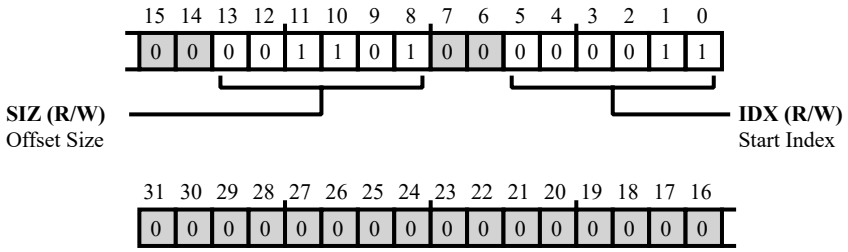


Figure 21-38: XSPI_MINICTL_HF_OFFSET Register Diagram

Table 21-80: XSPI_MINICTL_HF_OFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:8 (R/W)	SIZ	Offset Size. The XSPI_MINICTL_HF_OFFSET.SIZ bit field indicates the offset size of the reserved area in the command format.
5:0 (R/W)	IDX	Start Index. The XSPI_MINICTL_HF_OFFSET.IDX bit field indicates the starting index of the reserved area in the command format.

JEDEC Reset Delay Register

The XSPI_MINICTL_JEDEC_RST_TR register is used to introduce relative device selection delays applicable for JEDEC reset instruction.

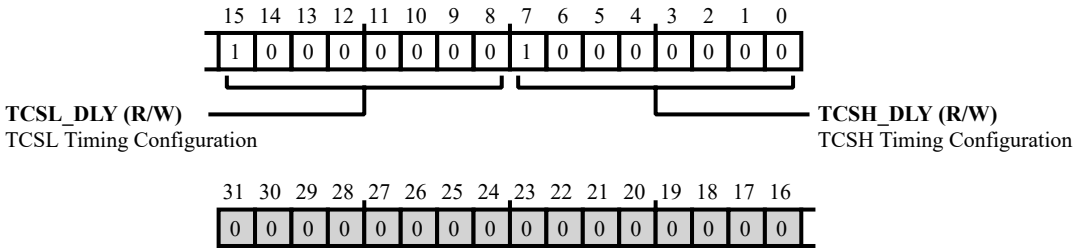


Figure 21-39: XSPI_MINICTL_JEDEC_RST_TR Register Diagram

Table 21-81: XSPI_MINICTL_JEDEC_RST_TR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	TCSL_DLY	TCSL Timing Configuration. The XSPI_MINICTL_JEDEC_RST_TR.TCSL_DLY bit field indicates how many xSPI clock cycles constitute tCSL timing of JEDEC reset instruction.
7:0 (R/W)	TCSH_DLY	TCSH Timing Configuration. The XSPI_MINICTL_JEDEC_RST_TR.TCSH_DLY bit field indicates how many xSPI clock cycles constitute tCSH timing of JEDEC reset instruction.

Hardware Reset Control Register

The XSPI_MINICTL_RST_PIN_CTL register configures the software controlled hardware reset.

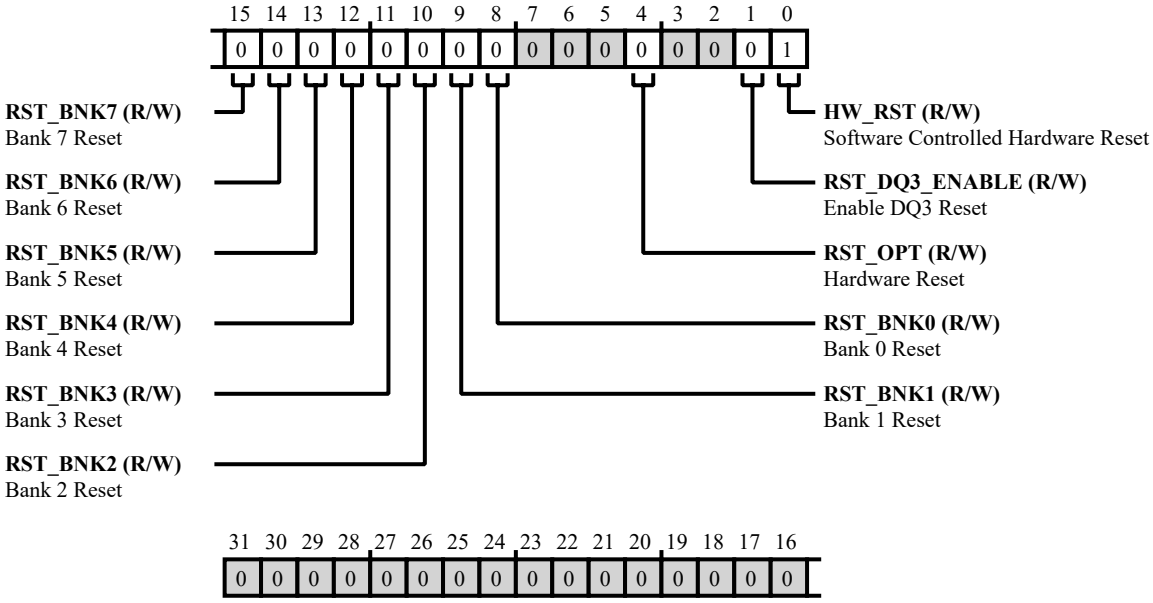


Figure 21-40: XSPI_MINICTL_RST_PIN_CTL Register Diagram

Table 21-82: XSPI_MINICTL_RST_PIN_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	RST_BNK7	Bank 7 Reset. The XSPI_MINICTL_RST_PIN_CTL.RST_BNK7 bit activates software controlled hardware reset signal on bank 7 (such as a device connected to chip select bit 7). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable
14 (R/W)	RST_BNK6	Bank 6 Reset. The XSPI_MINICTL_RST_PIN_CTL.RST_BNK6 bit activates software controlled hardware reset signal on bank 6 (such as a device connected to chip select bit 6). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable

Table 21-82: XSPI_MINICTL_RST_PIN_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	RST_BNK5	Bank 5 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_BNK5` bit activates software controlled hardware reset signal on bank 5 (such as a device connected to chip select bit 5). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable
12 (R/W)	RST_BNK4	Bank 4 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_BNK4` bit activates software controlled hardware reset signal on bank 0 (such as a device connected to chip select bit 4). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable
11 (R/W)	RST_BNK3	Bank 3 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_BNK3` bit activates software controlled hardware reset signal on bank 3 (such as a device connected to chip select bit 3). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable
10 (R/W)	RST_BNK2	Bank 2 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_BNK2` bit activates software controlled hardware reset signal on bank 2 (such as a device connected to chip select bit 2). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable
9 (R/W)	RST_BNK1	Bank 1 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_BNK1` bit activates software controlled hardware reset signal on bank 1 (such as a device connected to chip select bit 1). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable

Table 21-82: XSPI_MINICTL_RST_PIN_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	RST_BNK0	Bank 0 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_BNK0` bit activates the software controlled hardware reset signal on bank 0 (such as a device connected to chip select bit 0). It is applicable only for devices that support the RESET pin.	
		0	Disable
		1	Enable
4 (R/W)	RST_OPT	Hardware Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_OPT` bit indicates the hardware reset option.	
		0	RESET pin. Device RESET# pin is used for toggling device hardware reset functionality.
		1	DQ3 pin. Device DQ3 pin is used for toggling device hardware reset functionality.
1 (R/W)	RST_DQ3_ENABLE	Enable DQ3 Reset. The `XSPI_MINICTL_RST_PIN_CTL.RST_DQ3_ENABLE` bit enables passing RESET to the DQ3 port of the device (by switching direction of DQ3 pad).	
0 (R/W)	HW_RST	Software Controlled Hardware Reset. The value of this field is directly routed to the DQ3 or RESET# output signal (depending on XSPI_MINICTRL_RST_PIN_CTL. RST_OPT). The value can be changed only when xSPI flash interface is in an idle state. The value of the software controlled hardware reset signal is overwritten when DQ3 is a valid transaction pin. The controller does not check the device hardware reset setup/hold timings - this must be ensured by the host. The host is also responsible for triggering a suitable reset method by selecting the corresponding bank number and reset method (as defined in this register). The controller does not drive RESET value during read data phase of any active transfer (as transfer direction switches in this transfer part).	

Reset Recovery Delay Register

The XSPI_MINICTL_RST_RECOV register is used to introduce relative reset recovery delay with respect to generated xSPI flash interface.

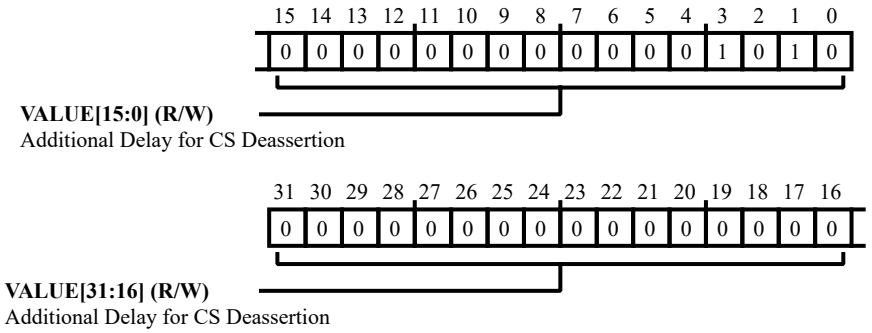


Figure 21-41: XSPI_MINICTL_RST_RECOV Register Diagram

Table 21-83: XSPI_MINICTL_RST_RECOV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Additional Delay for CS Deassertion. The XSPI_MINICTL_RST_RECOV.VALUE bit field defines the additional delay for CS deassertion to accommodate device reset recovery timing.

Write Protect Register

The XSPI_MINICTL_WP_CTL register controls write protection.

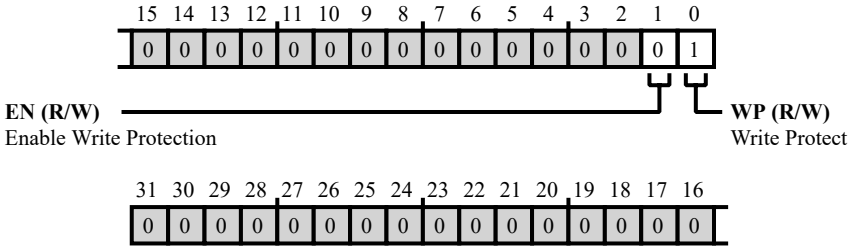


Figure 21-42: XSPI_MINICTL_WP_CTL Register Diagram

Table 21-84: XSPI_MINICTL_WP_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	EN	Enable Write Protection. The XSPI_MINICTL_WP_CTL.EN bit enables passing the write protect signal to the device (by switching direction of DQ2 pad).
0 (R/W)	WP	Write Protect. The XSPI_MINICTL_WP_CTL.WP bit indicates the write protect signal for all devices. Value of this register is directly routed to the DQ2 output signal. The value can be changed only when xSPI flash interface is in an idle state. Value of the write protection signal is overwritten in case DQ2 is a valid transaction pin. The controller does not check the write protect setup/hold timings - this must be ensured by the host. The controller does not drive the write protect value during the read data phase of any active transfer. Write protection on DQ2 functionality is only supported by flash devices and when the controller is configured in single and dual SPI modes.

PHY DLL Observable Points 0 Register

The XSPI_PHY_DLLOB0 register holds observable points in the PHY.

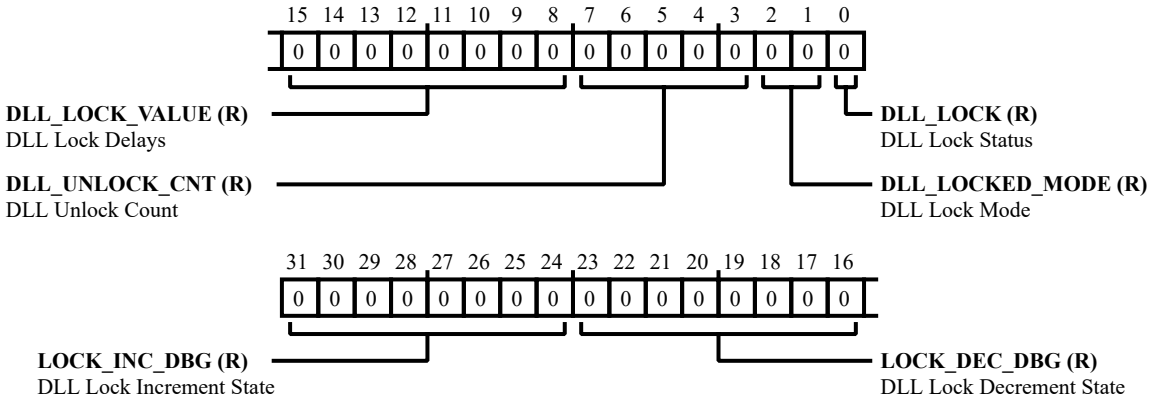


Figure 21-43: XSPI_PHY_DLLOB0 Register Diagram

Table 21-85: XSPI_PHY_DLLOB0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	LOCK_INC_DBG	DLL Lock Increment State. The XSPI_PHY_DLLOB0.LOCK_INC_DBG bit field holds the state of the cumulative DLL_LOCK_INC register when the DLL_UNLOCK_CNT field (bits [7:3]) of this parameter was triggered to increment or was last saturated at a value of 0x1f.
23:16 (R/NW)	LOCK_DEC_DBG	DLL Lock Decrement State. The XSPI_PHY_DLLOB0.LOCK_DEC_DBG bit field holds the state of the cumulative DLL_LOCK_DEC register when the DLL_UNLOCK_CNT field (bits [7:3]) of this parameter was triggered to decrement or was last saturated at a value of 0x1f.
15:8 (R/NW)	DLL_LOCK_VALUE	DLL Lock Delays. The XSPI_PHY_DLLOB0.DLL_LOCK_VALUE bit field indicates the number of delay elements that the DLL has determined for lock in either full clock or half clock mode. In full clock mode, this value equals the number of delay elements in one cycle. In half clock mode, this value equals the number of delay elements in one half clock cycle. In saturation mode, this value equals the maximum number of delay elements. The slaves use this value to configure their delays for the clock write and read DQS signals. This value is valid only when locking mechanism is done.
7:3 (R/NW)	DLL_UNLOCK_CNT	DLL Unlock Count. The XSPI_PHY_DLLOB0.DLL_UNLOCK_CNT bit field indicates the number of times that the bus requester DLL consecutive increment or decrement value programmed in the DLL_LOCK_NUM field (bits [18:16]) of the XSPI_PHY_DLL_REQ_CTL register has been triggered. The DLL_UNLOCK_CNT saturates at a value of 0x1f. Asserting the XSPI_PHY_DLL_CTL.DLL_RST resets this counter to 0.

Table 21-85: XSPI_PHY_DLLOB0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:1 (R/NW)	DLL_LOCKED_MODE	DLL Lock Mode. The `XSPI_PHY_DLLOB0.DLL_LOCKED_MODE` bit field indicates the mode in which the DLL has achieved the lock.	
		0	Full Clock Mode. The bus requester delay line was long enough to lock on one full clock cycle of delay. In this mode, the DLL_LOCK_VALUE field (bits [15:8]) of this parameter indicates the number of delays in full clock cycles.
		1	Reserved
		2	Half Clock Mode. The bus requester delay line was not long enough to lock one full cycle of delay but could lock on a half-cycle of delay. In this mode, the DLL_LOCK_VALUE field (bits [15:8]) of this parameter indicates the number of delays in one half clock cycles.
		3	Saturation Mode. The bus requester delay line was not long enough to lock on a full or a half-clock cycle. In this mode, the encoder value is fixed at the maximum delay line setting and the bus requester DLL will be disabled. The bus completer delay lines continue to use the fractional delays based upon the fixed saturation value of the delay line.
0 (R/NW)	DLL_LOCK	DLL Lock Status. The `XSPI_PHY_DLLOB0.DLL_LOCK` bit indicates the status of the DLL locking when the DLL lock logic is found (not inc AND not dec) OR (an inc then dec) OR (a dec then inc). When `XSPI_PHY_DLL_REQ_CTL.DLL_START_PT` is configured to be smaller than half a clock period, the first found (a dec then inc) is not the DLL locking point , but DLL_LOCK is asserted. 0 - DLL has not locked 1 - DLL is locked	
		0	Not Locked
		1	Locked

PHY DLL Observable Points 1 Register

The XSPI_PHY_DLLOB1 register holds observable points in the PHY.

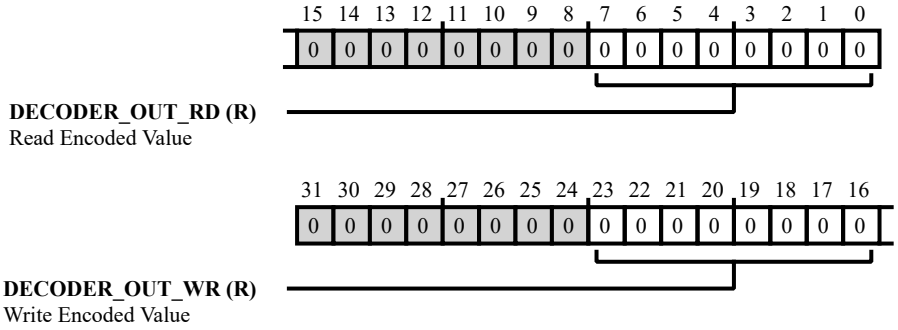


Figure 21-44: XSPI_PHY_DLLOB1 Register Diagram

Table 21-86: XSPI_PHY_DLLOB1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/NW)	DECODER_OUT_WR	Write Encoded Value. The XSPI_PHY_DLLOB1.DECODER_OUT_WR bit field holds the encoded value for the clock write delay line for this slice.
7:0 (R/NW)	DECODER_OUT_RD	Read Encoded Value. The XSPI_PHY_DLLOB1.DECODER_OUT_RD bit field holds the encoded value for the read delay line for this slice.

PHY DLL Control Register

The XSPI_PHY_DLL_CTL register configures the resynchronization of the bus completer DLL of the PHY. When the PHY is used with the xSPI controller, this register is automatically updated by device discovery during initialization.

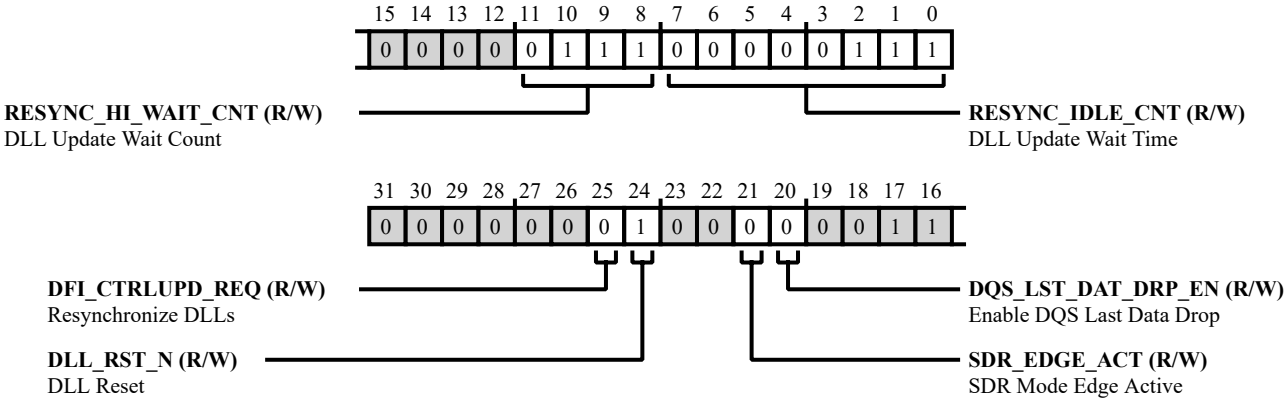


Figure 21-45: XSPI_PHY_DLL_CTL Register Diagram

Table 21-87: XSPI_PHY_DLL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25 (R/W)	DFI_CTRLUPD_REQ	Resynchronize DLLs. The XSPI_PHY_DLL_CTL.DFI_CTRLUPD_REQ bit signals to resynchronize the DLLs and read and write FIFO pointers. To send the update request to the PHY, the host must first set this bit (=1), then, wait until the bit is cleared (=0). Do not use this signal when automatic resynchronization is enabled (XSPI_PHY_DLL_UPDT_CNT is not zero).
24 (R/W)	DLL_RST_N	DLL Reset. The XSPI_PHY_DLL_CTL.DLL_RST_N bit resets the DLLs of the PHY and starts searching for lock again.
21 (R/W)	SDR_EDGE_ACT	SDR Mode Edge Active. The XSPI_PHY_DLL_CTL.SDR_EDGE_ACT bit indicates how the PHY samples data on the edges of the sampling clock. In SDR mode, only one sample is needed. When XSPI_PHY_DLL_CTL.SDR_EDGE_ACT is cleared (=0), the controller propagates data from the positive edge of the PHY sampling clock. When XSPI_PHY_DLL_CTL.SDR_EDGE_ACT is set (=1), the controller propagates data from the negative edge of PHY sampling clock. In DDR mode, this bit must be cleared.

Table 21-87: XSPI_PHY_DLL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20 (R/W)	DQS_LST_DAT_DRP_EN	Enable DQS Last Data Drop. XSPI_PHY_DLL_CTL.DQS_LST_DAT_DRP_EN is set (=1) when the flash device issues data on the negative edge of the flash clock and returns them with DQS, and the PHY is configured to sample data in DQS mode. In this case, the number of DQS edges equals to number of requested data + 1. When XSPI_PHY_DLL_CTL.DQS_LST_DAT_DRP_EN is set, the controller internally requests this redundant data at the end of the transfer cleaning up the PHY FIFO.
17 (R/W)	EXTENDED_WR_MODE	. This PHY register field is not applicable for xSPI flash Controller.
16 (R/W)	EXTENDED_RD_MODE	. This PHY register field is not applicable for xSPI flash controller.
11:8 (R/W)	RESYNC_HI_WAIT_CNT	DLL Update Wait Count. The XSPI_PHY_DLL_CTL.RESYNC_HI_WAIT_CNT bit field defines the number of xSPI clock cycles for which the DLL update request (XSPI_PHY_DLL_CTL.DFI_CTRLUPD_REQ) must be asserted to resynchronize the DLLs and read and write FIFO pointers.
7:0 (R/W)	RESYNC_IDLE_CNT	DLL Update Wait Time. The XSPI_PHY_DLL_CTL.RESYNC_IDLE_CNT bit field defines the wait time (in terms of xSPI clock cycles) between the deassertion of the DLL update request (XSPI_PHY_DLL_CTL.DFI_CTRLUPD_REQ) and resuming traffic to the PHY.

Bus Requester DLL Control Register

The XSPI_PHY_DLL_REQ_CTL register controls the bus requester DLL logic.

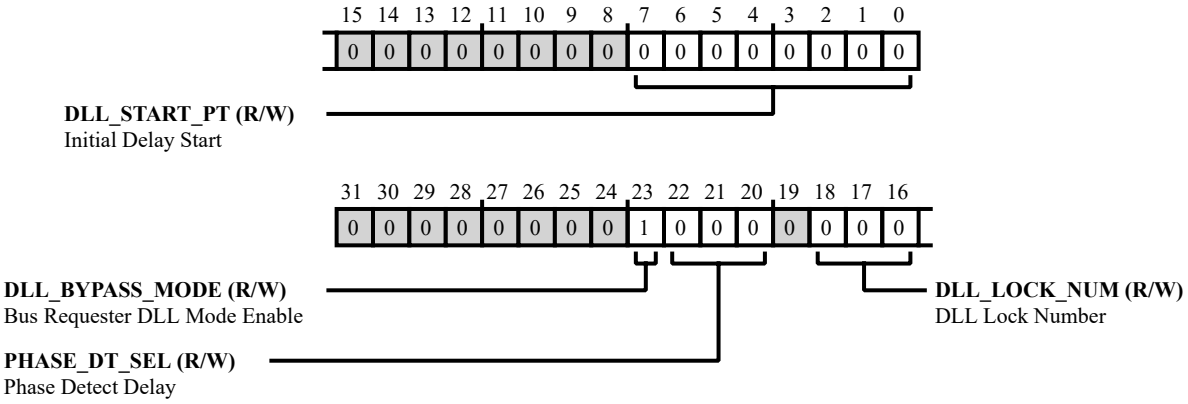


Figure 21-46: XSPI_PHY_DLL_REQ_CTL Register Diagram

Table 21-88: XSPI_PHY_DLL_REQ_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	DLL_BYPASS_MODE	Bus Requester DLL Mode Enable. The XSPI_PHY_DLL_REQ_CTL.DLL_BYPASS_MODE bit enables DLL bypass mode control. It controls the bypass mode of the bus requester and bus completer DLLs. The XSPI_PHY_DLL_REQ_CTL.DLL_BYPASS_MODE bit is intended to be used only for debug.	
		0	Normal Operational Mode. DLL functioning in normal mode where the bus completer delay line settings are used as fractional delay of the bus requester delay line encoder reading of the number of delays in one cycle.
		1	Bypass Mode On. Delays are defined in the XSPI_PHY_DLL_COMP_CTL register. The bus requester DLL is disabled with only 1 delay element in its delay line. The bus completer delay lines decode delays in absolute delay elements rather than as fractional delays. The XSPI_PHY_DLLOB0. DLL_LOCK is forced high.
22:20 (R/W)	PHASE_DT_SEL	Phase Detect Delay. The XSPI_PHY_DLL_REQ_CTL.PHASE_DT_SEL bit field indicates the number of delay elements to be inserted between the phase detect flip-flops. Defaults to 0x0, although the recommended value is 2 elements. But, if a lock condition is not detected, the user should increase the number of delay elements.	
		0	One Delay Element
		1	Two Delay Elements

Table 21-88: XSPI_PHY_DLL_REQ_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	Three Delay Elements
		3	Four Delay Elements
		4	Five Delay Elements
		5	Six Delay Elements
		6	Seven Delay Elements
		7	Eight Delay Elements
18:16 (R/W)	DLL_LOCK_NUM	DLL Lock Number. The `XSPI_PHY_DLL_REQ_CTL.DLL_LOCK_NUM` bit field holds the number of consecutive increment or decrement indications that trigger an unlock condition and increment the XSPI_PHY_DLLOB0.DLL_UNLOCK_CNT field and either the XSPI_PHY_DLLOB0.LOCK_DEC_DBG or XSPI_PHY_DLLOB0.LOCK_DEC_DBG fields.	
7:0 (R/W)	DLL_START_PT	Initial Delay Start. The `XSPI_PHY_DLL_REQ_CTL.DLL_START_PT` bit field indicates the initial delay value for the DLL. This value is also used as the increment value if the initial value is less than a half-clock cycle. This field must be set such that it is not greater than 7/8ths of a clock period given the worst case element delay. For example, if the frequency is 200MHz (5ns cycle time) with a worst case element 80ps delay, this field must be configured to = 5 * (7/8) / .080 = 54 elements. This calculation helps determine the start point which achieves the fastest lock. However, a small value such as 0x04 may be used instead to ensure that the DLL does not lock on a harmonic. Note that with a small value like this, the initial lock time is longer. Value smaller than 0x04 may cause no lock by DLL.	

Bus Completer DLL Control Register

The XSPI_PHY_DLL_COMP_CTL register controls the bus completer DLL.

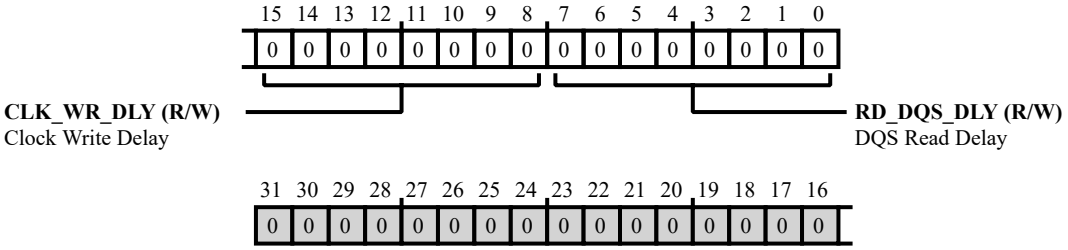


Figure 21-47: XSPI_PHY_DLL_COMP_CTL Register Diagram

Table 21-89: XSPI_PHY_DLL_COMP_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	CLK_WR_DLY	Clock Write Delay. The XSPI_PHY_DLL_COMP_CTL.CLK_WR_DLY bit field controls the clock write delay line which adjusts the write DQ bit timing in 1/256th steps of the clock period in normal DLL locked mode. In bypass mode, this field directly programs the number of delay elements.
7:0 (R/W)	RD_DQS_DLY	DQS Read Delay. The XSPI_PHY_DLL_COMP_CTL.RD_DQS_DLY bit field controls the read DQS delay which adjusts the timing in 1/256th of the clock period when in normal DLL locked mode. In bypass mode, this field directly programs the number of delay elements.

PHY DLL Resynchronization Register

The XSPI_PHY_DLL_UPDT_CNT register configures the resynchronization of the bus completer DLL of the PHY.

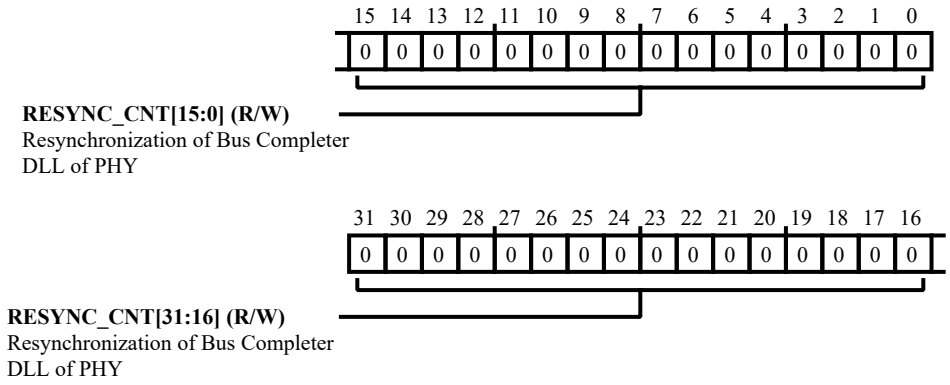


Figure 21-48: XSPI_PHY_DLL_UPDT_CNT Register Diagram

Table 21-90: XSPI_PHY_DLL_UPDT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RESYNC_CNT	Resynchronization of Bus Completer DLL of PHY. The XSPI_PHY_DLL_UPDT_CNT.RESYNC_CNT bit field defines the time interval (in terms of xSPI clock) to send an update (XSPI_PHY_DLL_CTL.DFI_CTRLUPD_REQ=1) to the PHY to re-synchronize the bus completer DLL values with that of the bus requester DLL and to also re-synchronize the read and write FIFO pointers in the read path. If the value in this field is zero, the controller will not further DLL update requests to the PHY. DFI_CTRLUPD_REQ signal can be controlled directly by the host using XSPI_PHY_DLL_CTL.DFI_CTRLUPD_REQ. Note: While this feature is enabled, access to the PHY registers shall not be performed.

PHY DQS Timing Register

The XSPI_PHY_DQS_TR register controls the DQS related timing.

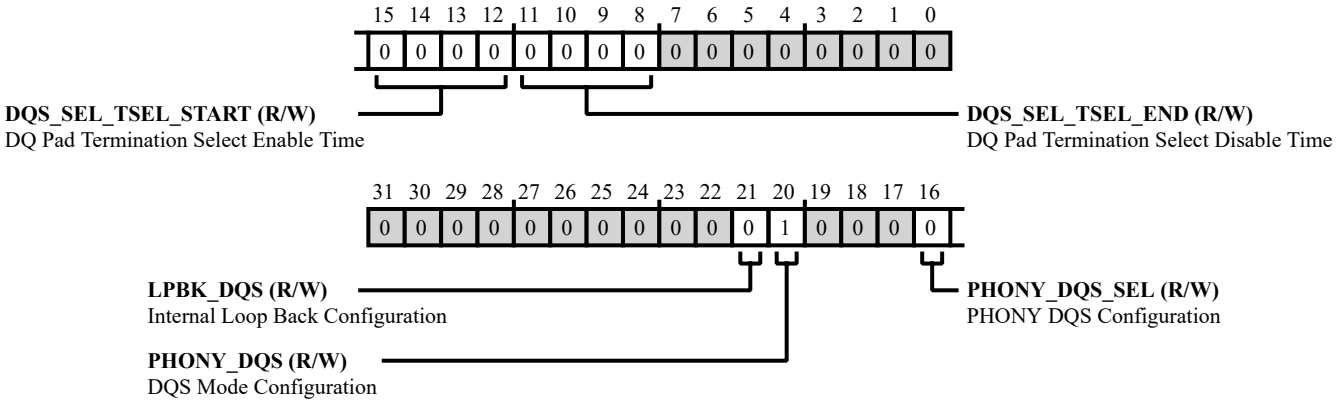


Figure 21-49: XSPI_PHY_DQS_TR Register Diagram

Table 21-91: XSPI_PHY_DQS_TR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W)	LPBK_DQS	Internal Loop Back Configuration. The XSPI_PHY_DQS_TR.LPBK_DQS bit is used in conjunction with bits 22 and 20 to control how read data is sampled by the PHY. The XSPI_PHY_DQS_TR.LPBK_DQS bit is only valid when bit 20 is set (=1), meaning it is only relevant when read data is not being sampled by DQS from the memory device. If bit 20 is cleared (=0), the XSPI_PHY_DQS_TR.LPBK_DQS bit should also be cleared. When using the PHY with the xSPI controller, this bit must be set to the same value as bit 20.This bit selects which internal source is used by the PHY to sample the read data. This is internally generated and is passed out of the PHY via rebar_opad If bit 22 of this register is '0', then it will be internally looped back within the rebar pad. If bit 22 of this register is '1', the rebar_opad will be passed through the rebar pad and it will be the responsibility of the integrator to loop that back to the PHY into the lpbk_dqs pin. 0 - Use phony DQS for data capture 1 - Use lpbk_dqs for data capture. [/list].

Table 21-91: XSPI_PHY_DQS_TR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	PHONY_DQS	DQS Mode Configuration. The `XSPI_PHY_DQS_TR.PHONY_DQS` bit is used in conjunction with bits 22 and 21 to control how read data is sampled by the PHY. The `XSPI_PHY_DQS_TR.PHONY_DQS` bit selects whether the read data sent by the memory device is sampled by DQS supplied by the memory device, or by a signal locally generated within the PHY.	
		0	Use DQS from device
		1	Use internally generated DQS
16 (R/W)	PHONY_DQS_SEL	PHONY DQS Configuration. When `XSPI_PHY_DQS_TR.PHONY_DQS_SEL` is cleared (=0), the PHONY_DQS is synchronous with rising edge of the CLK_PHY before sending to the entry flops. When `XSPI_PHY_DQS_TR.PHONY_DQS_SEL` is set (=1), the PHONY_DQS signal is synchronous with the falling edge of CLK_PHY before sending to the entry flops.	
15:12 (R/W)	DQS_SEL_TSEL_START	DQ Pad Termination Select Enable Time. The `XSPI_PHY_DQS_TR.DQS_SEL_TSEL_START` bit field defines the DQ pad dynamic termination select enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution.	
11:8 (R/W)	DQS_SEL_TSEL_END	DQ Pad Termination Select Disable Time. The `XSPI_PHY_DQS_TR.DQS_SEL_TSEL_END` bit field defines the DQ pad dynamic termination select disable time. Larger values increase the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution.	

PHY DQ Timing Register

The XSPI_PHY_DQ_TR register controls the DQ related timing.

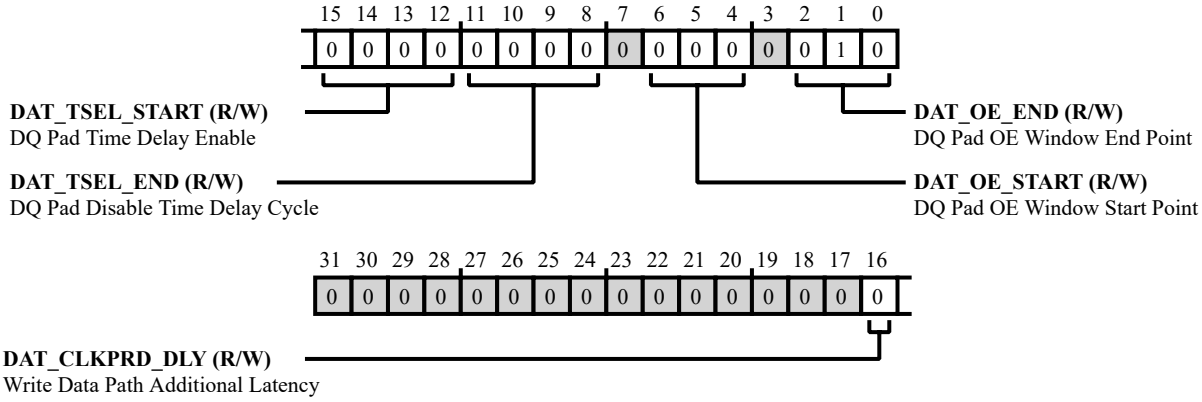


Figure 21-50: XSPI_PHY_DQ_TR Register Diagram

Table 21-92: XSPI_PHY_DQ_TR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	DAT_CLKPRD_DLY	Write Data Path Additional Latency. The XSPI_PHY_DQ_TR.DAT_CLKPRD_DLY bit field defines additional latency on the write data path. It also adds a clock cycle delay for the data OE path which is equivalent of adding 2 to DAT_OE_END and DAT_OE_START.
15:12 (R/W)	DAT_TSEL_START	DQ Pad Time Delay Enable. The XSPI_PHY_DQ_TR.DAT_TSEL_START bit field defines the DQ pad dynamic termination select enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution.
11:8 (R/W)	DAT_TSEL_END	DQ Pad Disable Time Delay Cycle. The XSPI_PHY_DQ_TR.DAT_TSEL_END bit field defines the DQ pad dynamic termination select disable time. Larger values increase the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution.
6:4 (R/W)	DAT_OE_START	DQ Pad OE Window Start Point. The XSPI_PHY_DQ_TR.DAT_OE_START bit field adjusts the starting point of the DQ pad output enable window. Lower numbers pull the rising edge earlier in time and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/2 cycle resolution.
2:0 (R/W)	DAT_OE_END	DQ Pad OE Window End Point. The XSPI_PHY_DQ_TR.DAT_OE_END bit field adjusts the ending point of the DQ pad output enable window. Lower numbers pull the falling edge earlier in time and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/2 cycle resolution.

PHY Features Register

The XSPI_PHY_FEATURES register shows the available hardware features.

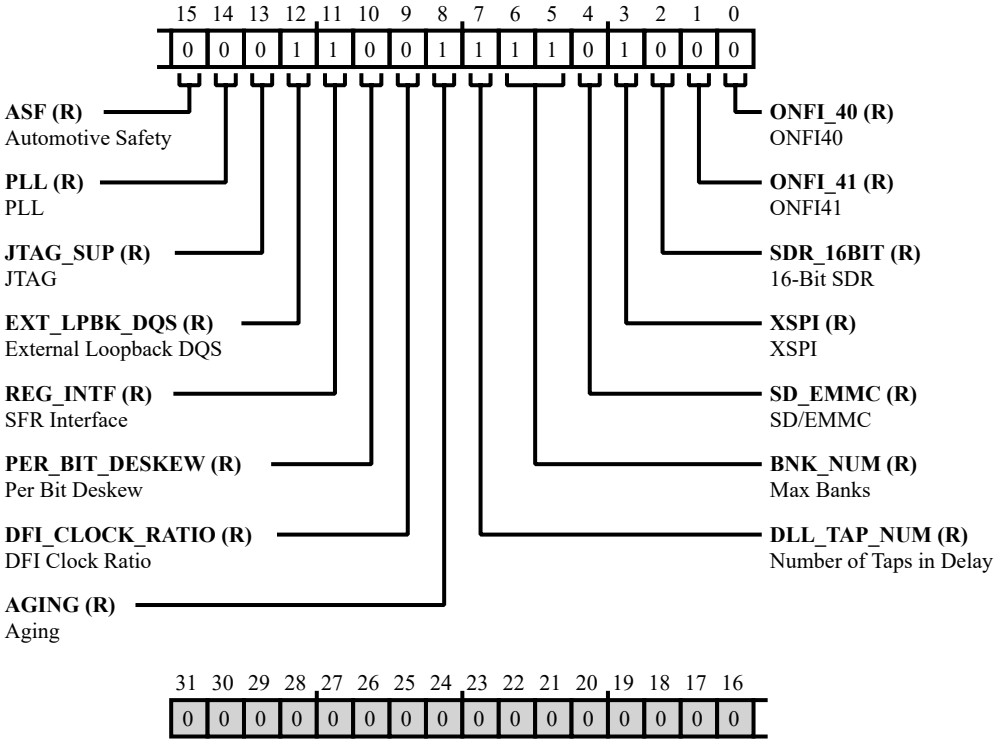


Figure 21-51: XSPI_PHY_FEATURES Register Diagram

Table 21-93: XSPI_PHY_FEATURES Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/NW)	ASF	Automotive Safety. The XSPI_PHY_FEATURES.ASF bit indicates support for an automotive safety feature.
14 (R/NW)	PLL	PLL. The XSPI_PHY_FEATURES.PLL bit indicates support for PLL.
13 (R/NW)	JTAG_SUP	JTAG. The XSPI_PHY_FEATURES.JTAG_SUP bit indicates support for JTAG multiplexing.
12 (R/NW)	EXT_LPBK_DQS	External Loopback DQS. The XSPI_PHY_FEATURES.EXT_LPBK_DQS bit indicates support for an external LPBK_DQS IO pad.

Table 21-93: XSPI_PHY_FEATURES Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	REG_INTF	SFR Interface. The `XSPI_PHY_FEATURES.REG_INTF` bit indicates the SFR interface type. This is an encoded value.	
		0	DFI
		1	APB
10 (R/NW)	PER_BIT_DESKEW	Per Bit Deskew. The `XSPI_PHY_FEATURES.PER_BIT_DESKEW` bit indicates support for per-bit deskew.	
9 (R/NW)	DFI_CLOCK_RATIO	DFI Clock Ratio. The `XSPI_PHY_FEATURES.DFI_CLOCK_RATIO` bit indicates support for a clock ratio on the DFI interface. This is an encoded value.	
		0	1:1
		1	1:2
8 (R/NW)	AGING	Aging. The `XSPI_PHY_FEATURES.AGING` bit indicates support for aging in delay lines.	
7 (R/NW)	DLL_TAP_NUM	Number of Taps in Delay. The `XSPI_PHY_FEATURES.DLL_TAP_NUM` bit indicates the number of taps in the delay line.	
		0	128
		1	256
6:5 (R/NW)	BNK_NUM	Max Banks. The `XSPI_PHY_FEATURES.BNK_NUM` bit field indicates the maximum number of banks supported by hardware. This is an encoded value.	
		0	One bank
		1	Two banks
		2	Four banks
		3	Eight banks
4 (R/NW)	SD_EMMC	SD/EMMC. The `XSPI_PHY_FEATURES.SD_EMMC` bit indicates support for SD/eMMC.	
3 (R/NW)	XSPI	XSPI. The `XSPI_PHY_FEATURES.XSPI` bit indicates support for xSPI.	

Table 21-93: XSPI_PHY_FEATURES Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/NW)	SDR_16BIT	16-Bit SDR. The `XSPI_PHY_FEATURES.SDR_16BIT` bit indicates support for 16-bit in ONFI SDR work mode.
1 (R/NW)	ONFI_41	ONFI41. The `XSPI_PHY_FEATURES.ONFI_41` bit indicates support for ONFI4.1 - NAND flash.
0 (R/NW)	ONFI_40	ONFI40. The `XSPI_PHY_FEATURES.ONFI_40` bit indicates support for ONFI4.0 - NAND flash.

PHY Gate Loopback Control Register

The XSPI_PHY_GATE_LPBK_CTL register controls the gate and loopback control related timing.

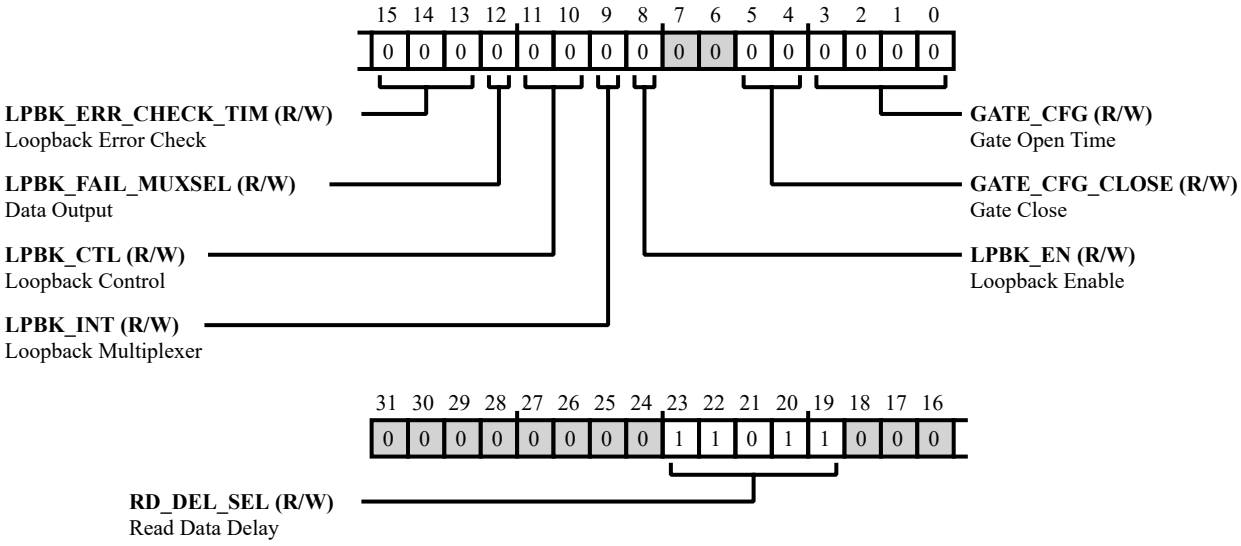


Figure 21-52: XSPI_PHY_GATE_LPBK_CTL Register Diagram

Table 21-94: XSPI_PHY_GATE_LPBK_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:19 (R/W)	RD_DEL_SEL	Read Data Delay. The XSPI_PHY_GATE_LPBK_CTL.RD_DEL_SEL bit defines the read data delay. It indicates the number of cycles to delay the dfi_rddata_en signal prior to enabling the read FIFO. After this delay, the read pointers begin incrementing the read FIFO. If sync_method is set (=1) the value of this field must take into account the synchronization time of the pointers in the entry FIFO (adding three clock cycles should be sufficient).	
15:13 (R/W)	LPBK_ERR_CHECK_TIM	Loopback Error Check. The XSPI_PHY_GATE_LPBK_CTL.LPBK_ERR_CHECK_TIM bit field indicates the cycle delay between the LFSR and loopback error check logic to ensure that the LFSR sourced data and data being looped back arrive at the same clock cycle for comparison. This value is related to the XSPI_PHY_GATE_LPBK_CTL .RD_DEL_SEL field, and is equal to 7 - RD_DEL_SEL.	
12 (R/W)	LPBK_FAIL_MUXSEL	Data Output. The XSPI_PHY_GATE_LPBK_CTL.LPBK_FAIL_MUXSEL bit selects the data output type for XSPI_PHY_OB0.LPBK_DQ_DAT.	
		0	Return the expected data
		1	Return the actual data

Table 21-94: XSPI_PHY_GATE_LPBK_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	LPBK_CTL	Loopback Control. The `XSPI_PHY_GATE_LPBK_CTL.LPBK_CTL` bit field indicates the loopback control.	
		0	Normal operation mode
		1	Loopback start. Enable loopback write mode.
		2	Loopback stop. Stop loopback to check error register.
		3	Clear. Clear loopback registers.
9 (R/W)	LPBK_INT	Loopback Multiplexer. The `XSPI_PHY_GATE_LPBK_CTL.LPBK_INT` bit controls the loopback read multiplexer.	
		0	External loopback
		1	Internal loopback
8 (R/W)	LPBK_EN	Loopback Enable. The `XSPI_PHY_GATE_LPBK_CTL.LPBK_EN` bit controls the internal write multiplexer.	
		0	Normal operation
		1	Enable loopback
5:4 (R/W)	GATE_CFG_CLOSE	Gate Close. The `XSPI_PHY_GATE_LPBK_CTL.GATE_CFG_CLOSE` bit field allows extending the closing of the DQS gate. Recommended value is zero. rebar_dfi - Read enable output signal. It is an internal signal between the xSPI PHY and xSPI controller. dfi_cebar - Peripheral select output (CS) signal. It is an internal signal between the xSPI PHY and xSPI controller.	
3:0 (R/W)	GATE_CFG	Gate Open Time. Coarse adjust of gate open time. The `XSPI_PHY_GATE_LPBK_CTL.GATE_CFG` bit field is the number of cycles to delay the dfi_rddata_en signal prior to opening the gate in full cycle increments. Decreasing this value pulls the gate earlier in time. This field must be programmed such that the gate signal lands in the valid DQS gate window.	

PHY Global Control Register

The XSPI_PHY_GCTL register handles the global control settings for the PHY.

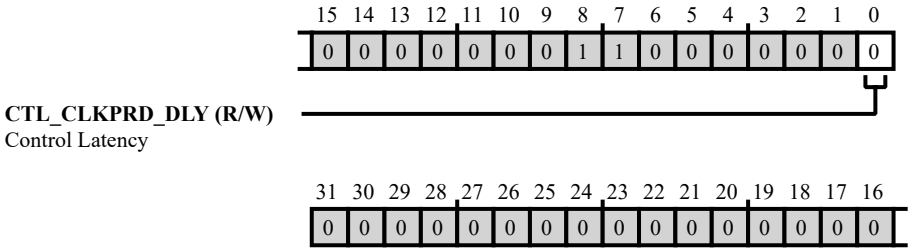


Figure 21-53: XSPI_PHY_GCTL Register Diagram

Table 21-95: XSPI_PHY_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	CTL_CLKPRD_DLY	Control Latency. The XSPI_PHY_GCTL.CTL_CLKPRD_DLY bit indicates an additional latency on the control signals WE/RE/CE/WP.

PHY GPIO Control Register 0

The XSPI_PHY_GPIO_CTL0 register is a general purpose register. The [31:0] vector is brought to the PHY I/Os. These pins can be used to control any static settings that may be required for the connected I/O pads.

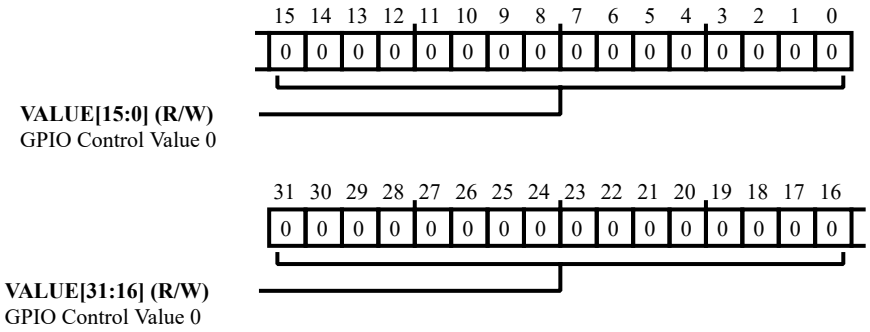


Figure 21-54: XSPI_PHY_GPIO_CTL0 Register Diagram

Table 21-96: XSPI_PHY_GPIO_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	GPIO Control Value 0. The [31:0] vector is brought to the PHY I/Os. The XSPI_PHY_GPIO_CTL0.VALUE bit field can used to control any static settings that may be required for the connected IO pads.

PHY GPIO Control Register 1

The XSPI_PHY_GPIO_CTL1 register is a general purpose register. The [31:0] vector is brought to the PHY I/Os. These pins can be used to control any static settings that may be required for the connected I/O pads.

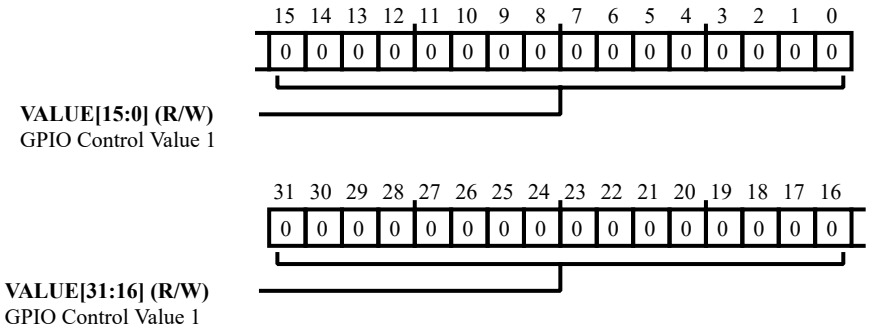


Figure 21-55: XSPI_PHY_GPIO_CTL1 Register Diagram

Table 21-97: XSPI_PHY_GPIO_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	GPIO Control Value 1. The [31:0] vector is brought to the PHY I/Os. The XSPI_PHY_GPIO_CTL1.VALUE bit field can used to control any static settings that may be required for the connected IO pads.

PHY GPIO Status Register 0

The XSPI_PHY_GPIO_STAT0 register is a general purpose register. A [31:0] vector is brought from the PHY I/Os to this register. This can be used as a status register.

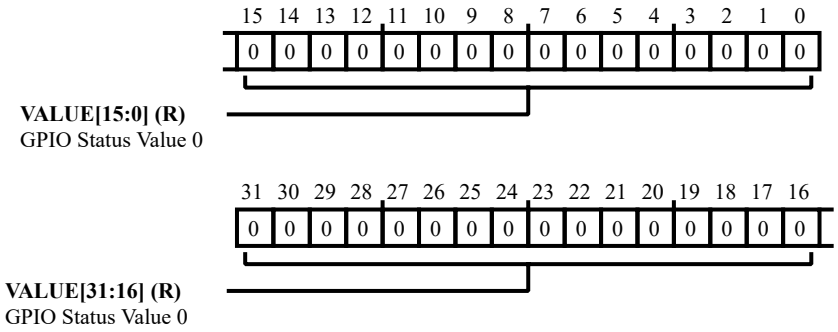


Figure 21-56: XSPI_PHY_GPIO_STAT0 Register Diagram

Table 21-98: XSPI_PHY_GPIO_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	GPIO Status Value 0. A [31:0] vector is brought from the PHY IOs to this register. The XSPI_PHY_GPIO_STAT0.VALUE bit field can used as a status register.

PHY GPIO Status Register 1

The XSPI_PHY_GPIO_STAT1 register is a general purpose register. A [31:0] vector is brought from the PHY I/Os to this register. This can be used as a status register.

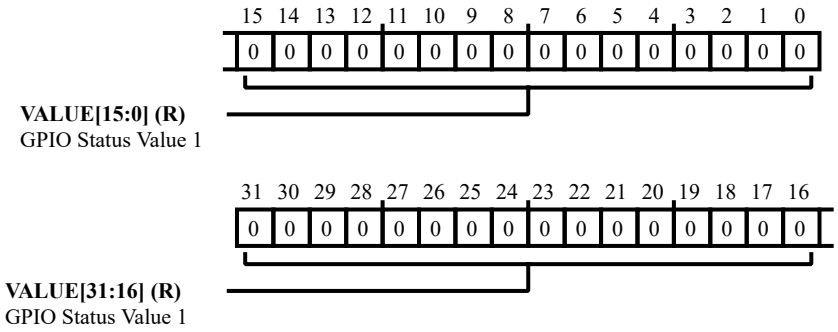


Figure 21-57: XSPI_PHY_GPIO_STAT1 Register Diagram

Table 21-99: XSPI_PHY_GPIO_STAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	GPIO Status Value 1. A [31:0] vector is brought from the PHY I/Os to this register. The XSPI_PHY_GPIO_STAT1.VALUE bit field can be used as a status register.

PHY Global Termination Control Register

The XSPI_PHY_GTSEL register handles the global control settings for the termination selects for reads.

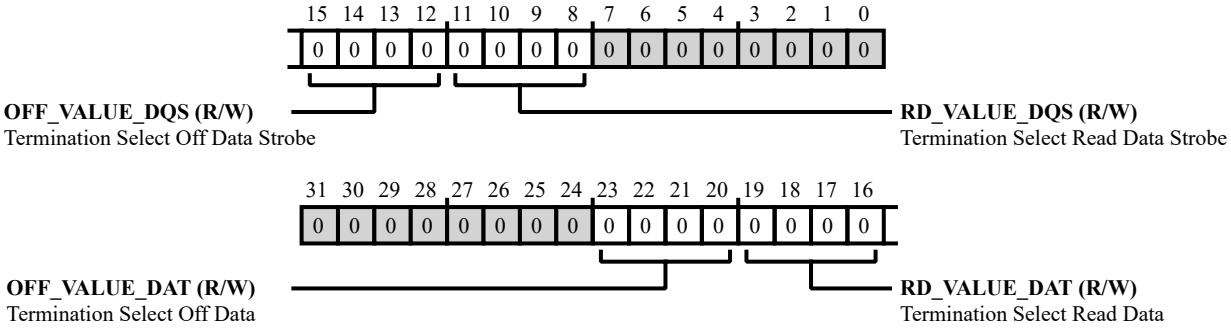


Figure 21-58: XSPI_PHY_GTSEL Register Diagram

Table 21-100: XSPI_PHY_GTSEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:20 (R/W)	OFF_VALUE_DAT	Termination Select Off Data. The XSPI_PHY_GTSEL.OFF_VALUE_DAT bit field indicates the termination select off value for the data.
19:16 (R/W)	RD_VALUE_DAT	Termination Select Read Data. The XSPI_PHY_GTSEL.RD_VALUE_DAT bit field indicates the termination select read value for the data.
15:12 (R/W)	OFF_VALUE_DQS	Termination Select Off Data Strobe. The XSPI_PHY_GTSEL.OFF_VALUE_DQS bit field indicates the termination select off value for the data strobe.
11:8 (R/W)	RD_VALUE_DQS	Termination Select Read Data Strobe. The XSPI_PHY_GTSEL.RD_VALUE_DQS bit field indicates the termination select read value for the data strobe.

PHY DQS Input Enable Timing Register

The XSPI_PHY_IE_TR register controls the DQS related timing.

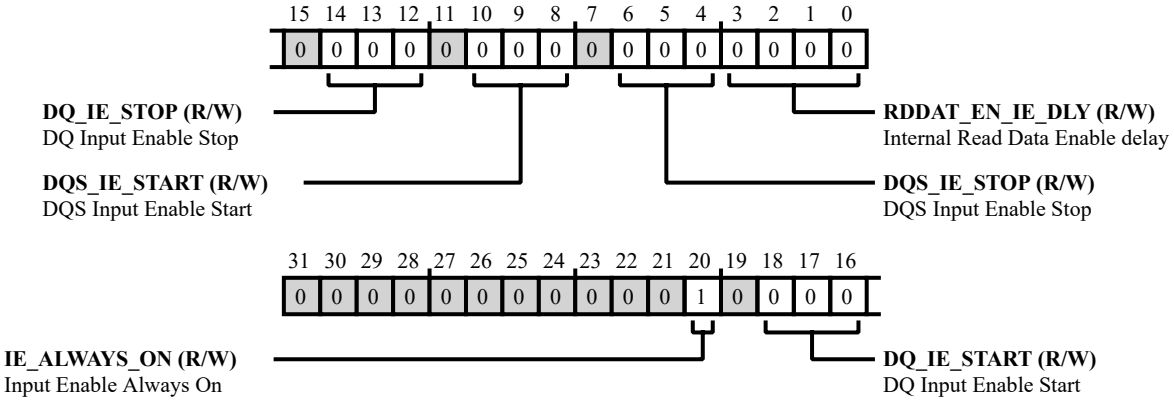


Figure 21-59: XSPI_PHY_IE_TR Register Diagram

Table 21-101: XSPI_PHY_IE_TR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20 (R/W)	IE_ALWAYS_ON	Input Enable Always On. The XSPI_PHY_IE_TR.IE_ALWAYS_ON bit forces the input enable(s) to be on always.
18:16 (R/W)	DQ_IE_START	DQ Input Enable Start. The XSPI_PHY_IE_TR.DQ_IE_START bit field indicates the start position for the DQ input enable.
14:12 (R/W)	DQ_IE_STOP	DQ Input Enable Stop. The XSPI_PHY_IE_TR.DQ_IE_STOP bit field indicates the stop position for the DQ input enable.
10:8 (R/W)	DQS_IE_START	DQS Input Enable Start. The XSPI_PHY_IE_TR.DQS_IE_START bit field indicates the start position for the DQS input enable.
6:4 (R/W)	DQS_IE_STOP	DQS Input Enable Stop. The XSPI_PHY_IE_TR.DQS_IE_STOP bit field indicates the stop position for the DQS input enable.
3:0 (R/W)	RDDAT_EN_IE_DLY	Internal Read Data Enable delay. The XSPI_PHY_IE_TR.RDDAT_EN_IE_DLY bit field indicates the number of clocks of delay for the DFI_RDDATA_EN signal to line it up with the true (normal) DFI read data position. The controller must deliver an early version of the read data enable to allow time for the input pads to turn on and this field allows the PHY to create the original timing.

PHY Observable Points Register

The XSPI_PHY_OB0 register holds observable points in the PHY.

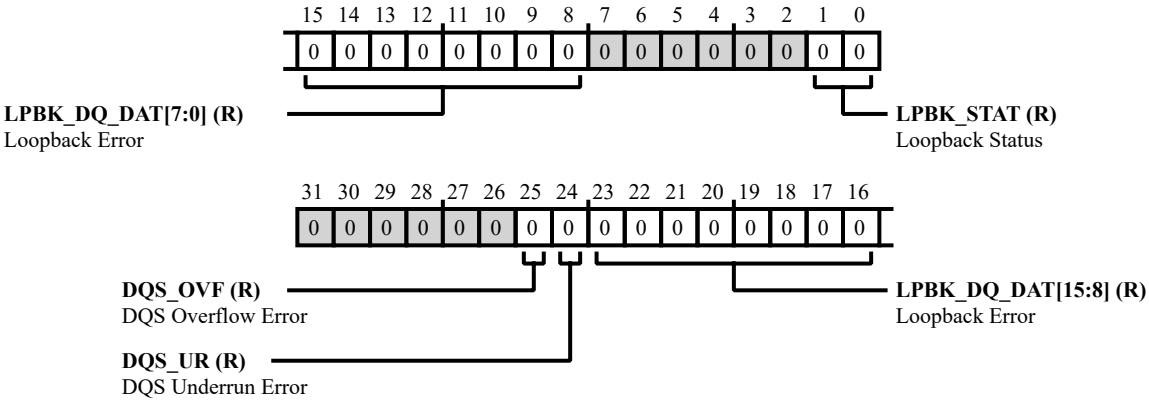


Figure 21-60: XSPI_PHY_OB0 Register Diagram

Table 21-102: XSPI_PHY_OB0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25 (R/NW)	DQS_OVF	DQS Overflow Error. The XSPI_PHY_OB0.DQS_OVF bit indicates that the logic gate was closed too late. For example, the number of DQS strobes exceed the capacity of the entry FIFO. It indicates that XSPI_PHY_GATE_LPBK_CTL.RD_DEL_SE signal value is too high and DFI_RDDATA are corrupted. It is possible that overflow status is asserted with underrun status - in such case the overflow takes the precedence. The XSPI_PHY_DLL_CTL.DLL_RST_N bit or RST_N signal clears this flag.
24 (R/NW)	DQS_UR	DQS Underrun Error. The XSPI_PHY_OB0.DQS_UR bit indicates that the logic gate had to be forced closed. It indicates that either the DQS strobe did not appear during read or XSPI_PHY_GATE_LPBK_CTL.RD_DEL_SEL signal value is too low and DFI_RDDATA are corrupted. The XSPI_PHY_DLL_CTL.DLL_RST_N bit or RST_N signal clears this flag.
23:8 (R/NW)	LPBK_DQ_DAT	Loopback Error. If errors are encountered in loopback test, the XSPI_PHY_OB0.LPBK_DQ_DAT bit field reports the actual data or the expected data, depending on the setting of the XSPI_PHY_GATE_LPBK_CTL.LPBK_FAIL_MUXSEL bit. This field is not clear by the clear state of the loopback. If there are no errors in loopback test, the value is zero (or value from previous state).

Table 21-102: XSPI_PHY_OB0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/NW)	LPBK_STAT	Loopback Status. The `XSPI_PHY_OB0.LPBK_STAT` bit field indicates the loopback status. Bit0 - loopback start; Defines the status of the loopback mode. 0 = Not in loopback mode; 1 = In loopback mode. Bit1 - loopback status; Defines the status of the loopback mode. 0 = Last Loopback test had no errors; 1 = Last loopback test contained data errors.

PHY Revision ID Register

The XSPI_PHY_REVID register contains the release identification number.

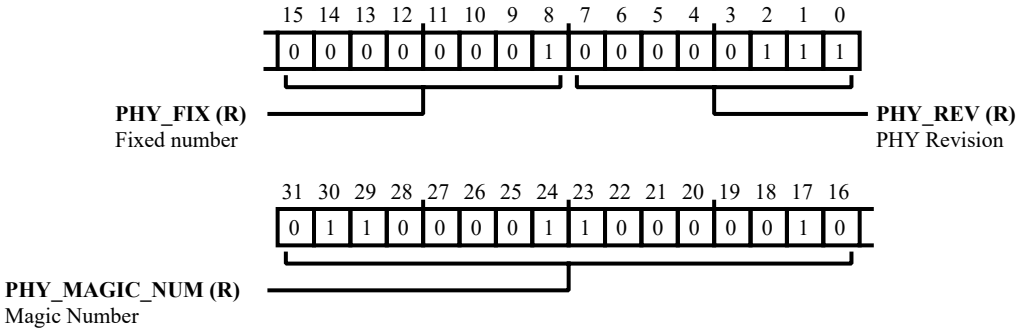


Figure 21-61: XSPI_PHY_REVID Register Diagram

Table 21-103: XSPI_PHY_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	PHY_MAGIC_NUM	Magic Number. The XSPI_PHY_REVID.PHY_MAGIC_NUM bit field indicates the magic number.
15:8 (R/NW)	PHY_FIX	Fixed number. The XSPI_PHY_REVID.PHY_FIX bit field indicates the fixed number (minor revision number).
7:0 (R/NW)	PHY_REV	PHY Revision. The XSPI_PHY_REVID.PHY_REV bit field indicates the PHY revision number.

PHY Static Aging Register

The XSPI_PHY_STATIC_TGL register controls the static aging feature of the PHY.

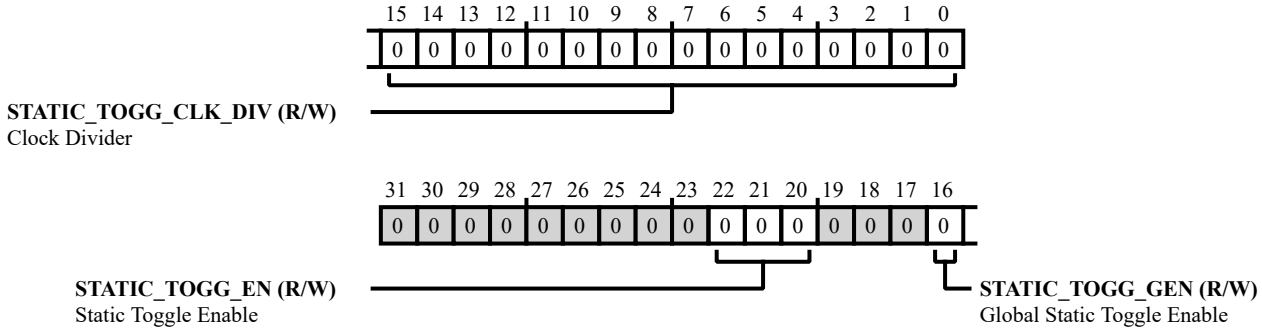


Figure 21-62: XSPI_PHY_STATIC_TGL Register Diagram

Table 21-104: XSPI_PHY_STATIC_TGL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22:20 (R/W)	STATIC_TOGG_EN	Static Toggle Enable. The XSPI_PHY_STATIC_TGL.STATIC_TOGG_EN bit field enables the toggle signal during static activity. When cleared (=0), the feature is disabled. Bit 0 - Bus requester delay line enable Bit 1 - Read path delay line enable Bit 2 - Write path delay line enable
16 (R/W)	STATIC_TOGG_GEN	Global Static Toggle Enable. The XSPI_PHY_STATIC_TGL.STATIC_TOGG_GEN bit enables the toggle signal during static activity.
15:0 (R/W)	STATIC_TOGG_CLK_DIV	Clock Divider. The XSPI_PHY_STATIC_TGL.STATIC_TOGG_CLK_DIV bit field indicates the clock divider used to create the toggle signal.

PHY Deskew Write Register

The XSPI_PHY_WR_DESKEW_PAD_CTL0 register holds the values of the phase detect block for each DQ bit on the write path.

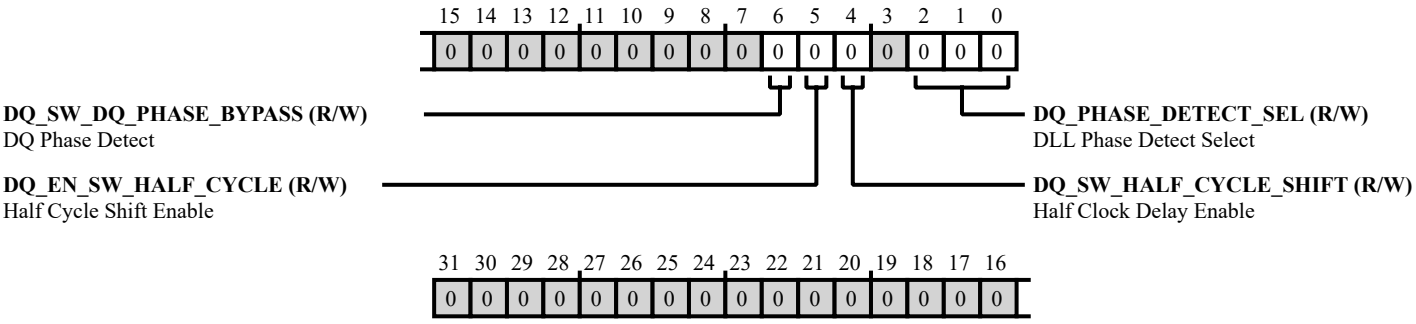


Figure 21-63: XSPI_PHY_WR_DESKEW_PAD_CTL0 Register Diagram

Table 21-105: XSPI_PHY_WR_DESKEW_PAD_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DQ_SW_DQ_PHASE_BYPASS	DQ Phase Detect. When XSPI_PHY_WR_DESKEW_PAD_CTL0.DQ_SW_DQ_PHASE_BYPASS is set (=1), the clock write delay line configuration is used to determine the HALF_CYCLE_SHIFT. When XSPI_PHY_WR_DESKEW_PAD_CTL0.DQ_SW_DQ_PHASE_BYPASS is cleared (=0), the phase detect circuit is used to determine the HALF_CYCLE_SHIFT.	
		0	Use phase detect circuit
		1	Use the clock write delay line configuration. A delay line configuration of 0x00-0x7f means HALF_CYCLE_SHIFT= 0 and a delay line configuration of 0x80-0xff means HALF_CYCLE_SHIFT = 1.
5 (R/W)	DQ_EN_SW_HALF_CYCLE	Half Cycle Shift Enable. When XSPI_PHY_WR_DESKEW_PAD_CTL0.DQ_EN_SW_HALF_CYCLE is set (=1), it enables the software half cycle shift. This determines whether the write data is transferred to the clock write domain on the positive or negative edge of the PHY clock. This field is valid when DQ_SW_DQ_PHASE_BYPASS =0. Note: If the user chooses to control the half cycle shift manually, it is important that DQ_SW_HALF_CYCLE_SHIFT = 0 if the delay is less than a 1/2 cycle and DQ_SW_HALF_CYCLE_SHIFT = 1 if the delay is greater than a 1/2 cycle. It is recommended to allow the hardware to control this automatically.	
		0	Hardware automatically controls shifting needed for the write level delay line.
		1	DQ_SW_HALF_CYCLE_SHIFT field defines the shift

Table 21-105: XSPI_PHY_WR_DESKEW_PAD_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	DQ_SW_HALF_CYCLE_SHIFT	Half Clock Delay Enable. When `XSPI_PHY_WR_DESKEW_PAD_CTL0.DQ_SW_HALF_CYCLE_SHIFT` is set (=1), a half clock delay is added to the write data path.	
		0	No effect
		1	Adds a half clock delay to the write data path
2:0 (R/W)	DQ_PHASE_DETECT_SEL	DLL Phase Detect Select. The `XSPI_PHY_WR_DESKEW_PAD_CTL0.DQ_PHASE_DETECT_SEL` bit field indicates the DLL phase detect selector for DQ generation to handle the clock domain crossing between the clock and clock write signal. It indicates the number of delay elements to be inserted between the phase detect flip-flops. Defaults to 0x0.	
		0	One delay element
		1	Two delay elements
		2	Three delay elements
		3	Four delay elements
		4	Five delay elements
		5	Six delay elements
		6	Seven delay elements
		7	Eight delay elements

Program Sequence Configuration Register 0

The XSPI_PROG_SEQ_CFG0 register configures the program sequence for profile 1 and SPI NAND in ACMD and direct work modes.

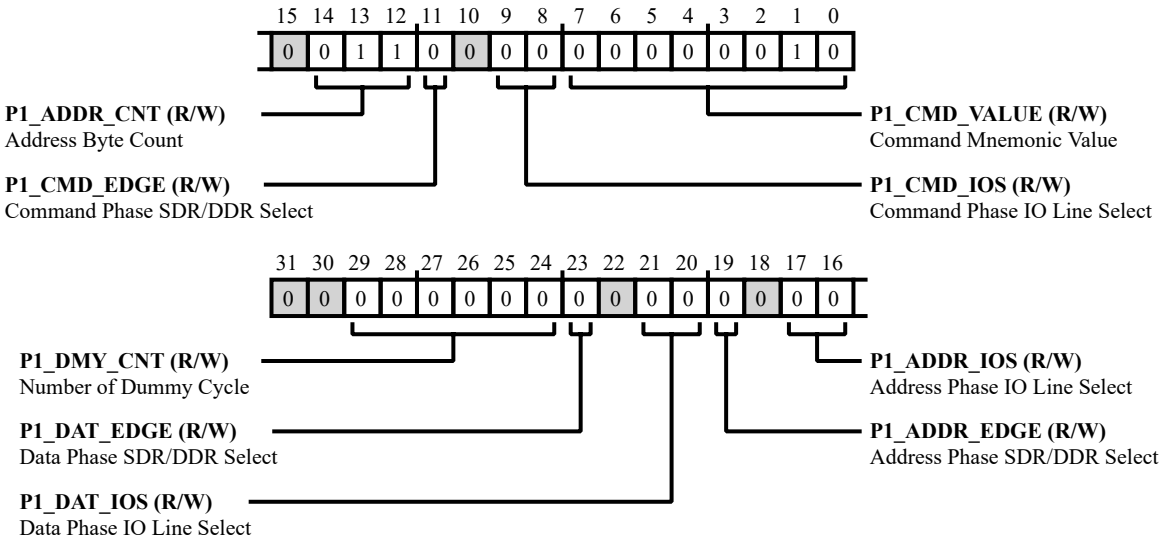


Figure 21-64: XSPI_PROG_SEQ_CFG0 Register Diagram

Table 21-106: XSPI_PROG_SEQ_CFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:24 (R/W)	P1_DMY_CNT	Number of Dummy Cycle. The XSPI_PROG_SEQ_CFG0.P1_DMY_CNT bit field indicates the number of dummy cycles in profile 1. If 0, dummy cycles are disabled.	
23 (R/W)	P1_DAT_EDGE	Data Phase SDR/DDR Select. The XSPI_PROG_SEQ_CFG0.P1_DAT_EDGE bit selects between SDR/DDR mode for the data phase.	
21:20 (R/W)	P1_DAT_IOS	Data Phase IO Line Select. The XSPI_PROG_SEQ_CFG0.P1_DAT_IOS bit field indicates the number of data lines used to send the data phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
19 (R/W)	P1_ADDR_EDGE	Address Phase SDR/DDR Select. The XSPI_PROG_SEQ_CFG0.P1_ADDR_EDGE bit selects between SDR/DDR mode for the address phase.	

Table 21-106: XSPI_PROG_SEQ_CFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	P1_ADDR_IOS	Address Phase IO Line Select. The `XSPI_PROG_SEQ_CFG0.P1_ADDR_IOS` bit field indicates the number of data lines used to send the address phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
14:12 (R/W)	P1_ADDR_CNT	Address Byte Count. The `XSPI_PROG_SEQ_CFG0.P1_ADDR_CNT` bit field indicates the number of address bytes.	
11 (R/W)	P1_CMD_EDGE	Command Phase SDR/DDR Select. The `XSPI_PROG_SEQ_CFG0.P1_CMD_EDGE` bit selects between SDR/DDR mode for the command phase.	
9:8 (R/W)	P1_CMD_IOS	Command Phase IO Line Select. The `XSPI_PROG_SEQ_CFG0.P1_CMD_IOS` bit field indicates the number of data lines used to send the command phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
7:0 (R/W)	P1_CMD_VALUE	Command Mnemonic Value. The `XSPI_PROG_SEQ_CFG0.P1_CMD_VALUE` bit field indicates the command mnemonic value.	

Program Sequence Configuration Register 1

The XSPI_PROG_SEQ_CFG1 register configures the program sequence for profile 1 and SPI NAND in ACMD and direct work modes.

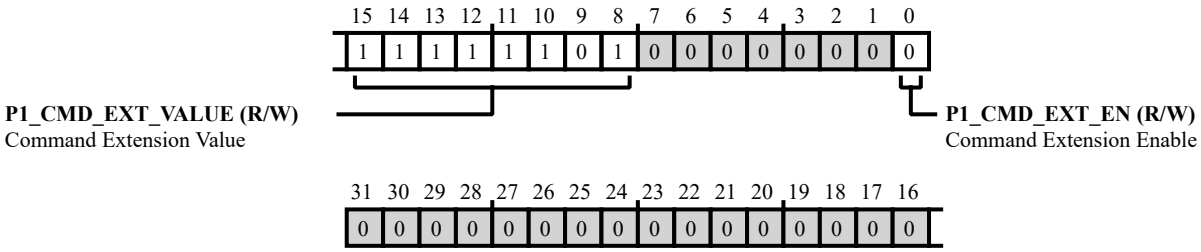


Figure 21-65: XSPI_PROG_SEQ_CFG1 Register Diagram

Table 21-107: XSPI_PROG_SEQ_CFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	P1_CMD_EXT_VALUE	Command Extension Value. The XSPI_PROG_SEQ_CFG1.P1_CMD_EXT_VALUE bit field indicates the command extension value.
0 (R/W)	P1_CMD_EXT_EN	Command Extension Enable. The XSPI_PROG_SEQ_CFG1.P1_CMD_EXT_EN bit enables the command extension.

Program Sequence Configuration Register 2

The XSPI_PROG_SEQ_CFG2 register configures the program sequence for profile 2 in ACMD and direct work modes.

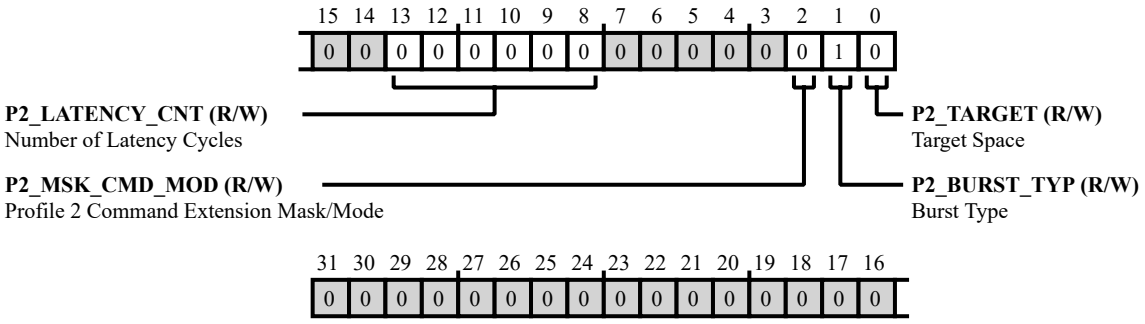


Figure 21-66: XSPI_PROG_SEQ_CFG2 Register Diagram

Table 21-108: XSPI_PROG_SEQ_CFG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13:8 (R/W)	P2_LATENCY_CNT	Number of Latency Cycles. The XSPI_PROG_SEQ_CFG2.P2_LATENCY_CNT bit field indicates the number of latency cycles for profile 2 - HR only. When XSPI_PROG_SEQ_CFG2.P2_LATENCY_CNT is cleared (=0), latency cycles are disabled. Configure the XSPI_PROG_SEQ_CFG2.P2_LATENCY_CNT bit field to N-1, where N is the number of latency clock cycles expected by the memory device.	
2 (R/W)	P2_MSK_CMD_MOD	Profile 2 Command Extension Mask/Mode. The XSPI_PROG_SEQ_CFG2.P2_MSK_CMD_MOD bit indicates the profile 2 command extension variant. This influences bits [44:40] of the CA. When XSPI_PROG_SEQ_CFG2.P2_MSK_CMD_MOD is set (=1), CA[44:40] is configured to all ones. In direct work mode, when XSPI_PROG_SEQ_CFG2.P2_MSK_CMD_MOD is cleared (=0), bits [44:40] of the CA are configured to (sAWADDR[45:41] logically AND'ed with DAC_ADDR_MASk[12:8]). In ACMD work mode, when XSPI_PROG_SEQ_CFG2.P2_MSK_CMD_MOD is cleared (=0), bits [44:40] of the CA are cleared (= 0).	
1 (R/W)	P2_BURST_TYP	Burst Type. The XSPI_PROG_SEQ_CFG2.P2_BURST_TYP bit indicates the burst type; it corresponds to the 45th Command/Address (CA) bit assignment. Although the user can set wrapped burst, the controller itself does not currently support wrapping bursts and therefore must always have this bit set (=1).	
		0	Wrapped burst
		1	Linear burst

Table 21-108: XSPI_PROG_SEQ_CFG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	P2_TARGET	Target Space. The `XSPI_PROG_SEQ_CFG2.P2_TARGET` bit indicates the target space, it corresponds to the 46th Command/Address (CA) bit assignment.	
		0	Memory space
		1	Register space

Read Sequence Configuration Register 0

The XSPI_READ_SEQ_CFG0 register configures the read sequence for profile 1 and SPI NAND in ACMD and direct work modes.

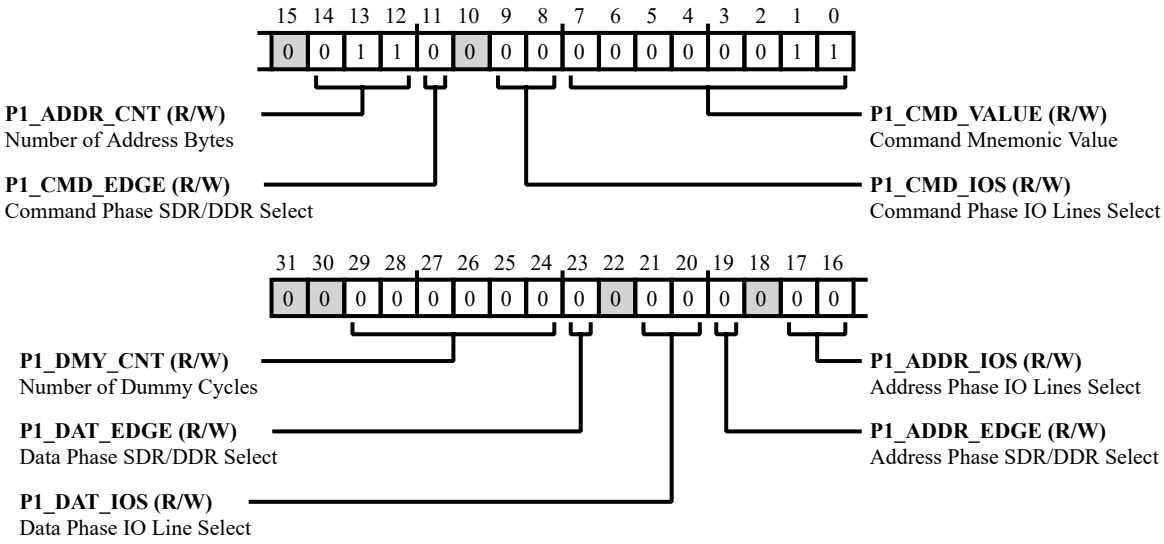


Figure 21-67: XSPI_READ_SEQ_CFG0 Register Diagram

Table 21-109: XSPI_READ_SEQ_CFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:24 (R/W)	P1_DMY_CNT	Number of Dummy Cycles. The XSPI_READ_SEQ_CFG0.P1_DMY_CNT bit field indicates the number of dummy cycles. If 0 - dummy cycles disabled. This field is used when sending mode-bits is disabled. Otherwise, the read_seq_p1_mb_dummy_cnt must be used.	
23 (R/W)	P1_DAT_EDGE	Data Phase SDR/DDR Select. The XSPI_READ_SEQ_CFG0.P1_DAT_EDGE bit selects between SDR/DDR mode for data phase.	
21:20 (R/W)	P1_DAT_IOS	Data Phase IO Line Select. The XSPI_READ_SEQ_CFG0.P1_DAT_IOS bit field indicates the number of data lines used to send the data phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used

Table 21-109: XSPI_READ_SEQ_CFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P1_ADDR_EDGE	Address Phase SDR/DDR Select. The `XSPI_READ_SEQ_CFG0.P1_ADDR_EDGE` bit selects between SDR/DDR mode for address phase.	
17:16 (R/W)	P1_ADDR_IOS	Address Phase IO Lines Select. The `XSPI_READ_SEQ_CFG0.P1_ADDR_IOS` bit field indicates the number of data lines used to send the address phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
14:12 (R/W)	P1_ADDR_CNT	Number of Address Bytes. The `XSPI_READ_SEQ_CFG0.P1_ADDR_CNT` bit field indicates the number of address bytes.	
11 (R/W)	P1_CMD_EDGE	Command Phase SDR/DDR Select. The `XSPI_READ_SEQ_CFG0.P1_CMD_EDGE` bit selects between SDR/DDR mode for command phase.	
9:8 (R/W)	P1_CMD_IOS	Command Phase IO Lines Select. The `XSPI_READ_SEQ_CFG0.P1_CMD_IOS` bit field indicates the number of data lines used to send the command phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
7:0 (R/W)	P1_CMD_VALUE	Command Mnemonic Value. The `XSPI_READ_SEQ_CFG0.P1_CMD_VALUE` bit field indicates the command mnemonic value.	

Read Sequence Configuration Register 1

The XSPI_READ_SEQ_CFG1 register configures the read sequence for profile 1 and SPI NAND in ACMD and direct work modes.

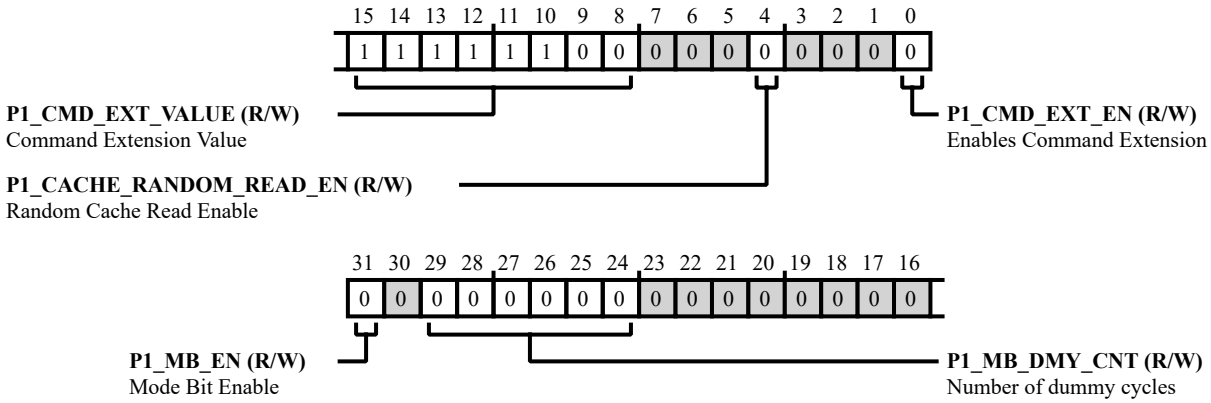


Figure 21-68: XSPI_READ_SEQ_CFG1 Register Diagram

Table 21-110: XSPI_READ_SEQ_CFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	P1_MB_EN	Mode Bit Enable. When XSPI_READ_SEQ_CFG1.P1_MB_EN is set (=1), the mode bits as defined in the XSPI_XIP_GCTL.XIP_DIS_MB_VAL field, are sent following the address bytes.
29:24 (R/W)	P1_MB_DMY_CNT	Number of dummy cycles. The XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT bit field indicates the number of dummy cycles. When XSPI_READ_SEQ_CFG1.P1_MB_DMY_CNT is cleared (=0), dummy cycles are disabled. This field is used when the sending mode bits function is enabled.
15:8 (R/W)	P1_CMD_EXT_VALUE	Command Extension Value. The XSPI_READ_SEQ_CFG1.P1_CMD_EXT_VALUE bit enables the command extension value.
4 (R/W)	P1_CACHE_RAN-DOM_READ_EN	Random Cache Read Enable. The XSPI_READ_SEQ_CFG1.P1_CACHE_RANDOM_READ_EN bit changes behavior of the read sequence to utilize the read page cache random/read page cache random last commands. This field is not used in direct mode. This bit is only applicable to the SPI NAND profile.

Table 21-110: XSPI_READ_SEQ_CFG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	P1_CMD_EXT_EN	Enables Command Extension. The `XSPI_READ_SEQ_CFG1.P1_CMD_EXT_EN` bit enables command extension.

Read Sequence Configuration Register 2

The XSPI_READ_SEQ_CFG2 register configures the read sequence for profile 2 in ACMD and direct work modes.

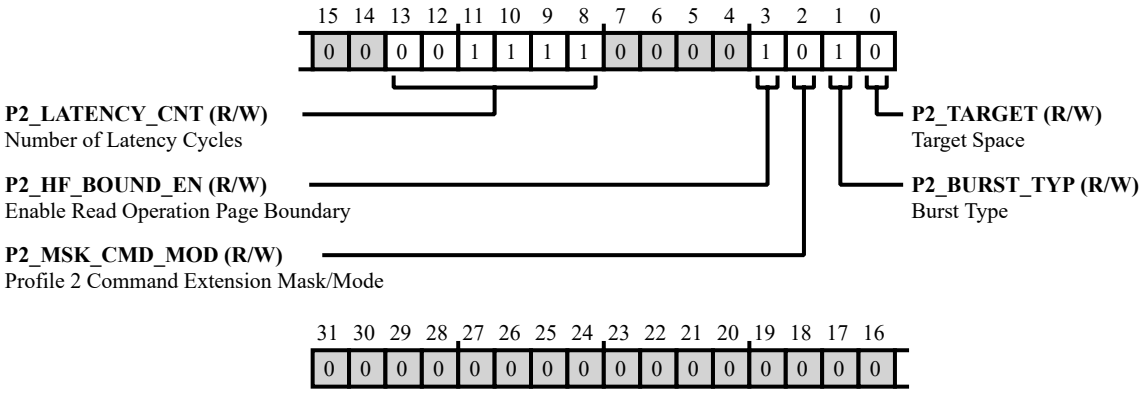


Figure 21-69: XSPI_READ_SEQ_CFG2 Register Diagram

Table 21-111: XSPI_READ_SEQ_CFG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13:8 (R/W)	P2_LATENCY_CNT	Number of Latency Cycles. The XSPI_READ_SEQ_CFG2.P2_LATENCY_CNT bit field indicates the number of latency cycles. When XSPI_READ_SEQ_CFG2.P2_LATENCY_CNT is cleared (=0), latency clock cycles are disabled. Configure XSPI_READ_SEQ_CFG2.P2_LATENCY_CNT to N-1, where N is the number of latency clock cycles expected by the memory device.	
3 (R/W)	P2_HF_BOUND_EN	Enable Read Operation Page Boundary. The XSPI_READ_SEQ_CFG2.P2_HF_BOUND_EN bit field is used by the controller to calculate read transaction crossing page boundary. It is valid only when profile 2 - HF is selected.	
		0	Disable
		1	Enable

Table 21-111: XSPI_READ_SEQ_CFG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	P2_MSK_CMD_MOD	Profile 2 Command Extension Mask/Mode. The `XSPI_READ_SEQ_CFG2.P2_MSK_CMD_MOD` bit field indicates the profile 2 command extension variant.This bit influences bits [44:40] of Command/Address. When `XSPI_READ_SEQ_CFG2.P2_MSK_CMD_MOD` is set (=1), bits [44:40] of Command/Address is set to 1. In direct work mode, when `XSPI_READ_SEQ_CFG2.P2_MSK_CMD_MOD` is cleared (=0), bits [44:40] of Command/Address is set to (sARADDR[45:41] logically AND'ed with dac_addr_mask[12:8]). In ACMD work mode, when `XSPI_READ_SEQ_CFG2.P2_MSK_CMD_MOD` is cleared (=0), bits [44:40] of Command/Address is cleared.	
1 (R/W)	P2_BURST_TYP	Burst Type. The `XSPI_READ_SEQ_CFG2.P2_BURST_TYP` bit indicates the burst type; it corresponds to 45th Command/Address (CA) bit assignment. Although the user can set wrapped burst, the controller itself does not currently support wrapping bursts; therefore, configure `XSPI_READ_SEQ_CFG2.P2_BURST_TYP` = 1.	
		0	Wrapped burst
		1	Linear burst
0 (R/W)	P2_TARGET	Target Space. The `XSPI_READ_SEQ_CFG2.P2_TARGET` bit indicates the target space; it corresponds to 46th Command/Address (CA) bit assignment.	
		0	Memory space
		1	Register space

Revision ID Register

The XSPI_REVID register contains the release identification number.

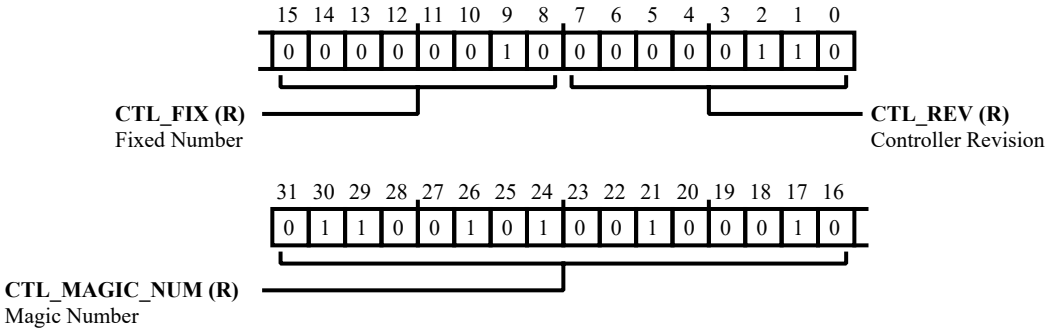


Figure 21-70: XSPI_REVID Register Diagram

Table 21-112: XSPI_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	CTL_MAGIC_NUM	Magic Number. The XSPI_REVID.CTL_MAGIC_NUM bit field indicates the controller's magic number. It is a unique number characteristic for the xSPI controller.
15:8 (R/NW)	CTL_FIX	Fixed Number. The XSPI_REVID.CTL_FIX bit field indicates the fixed number (minor revision number).
7:0 (R/NW)	CTL_REV	Controller Revision. The XSPI_REVID.CTL_REV bit field indicates the controller revision number.

Reset Sequence Configuration Register 0

The XSPI_RST_SEQ_CFG0 register configures the reset sequence for profile 1 and SPI NAND in ACMD work mode.

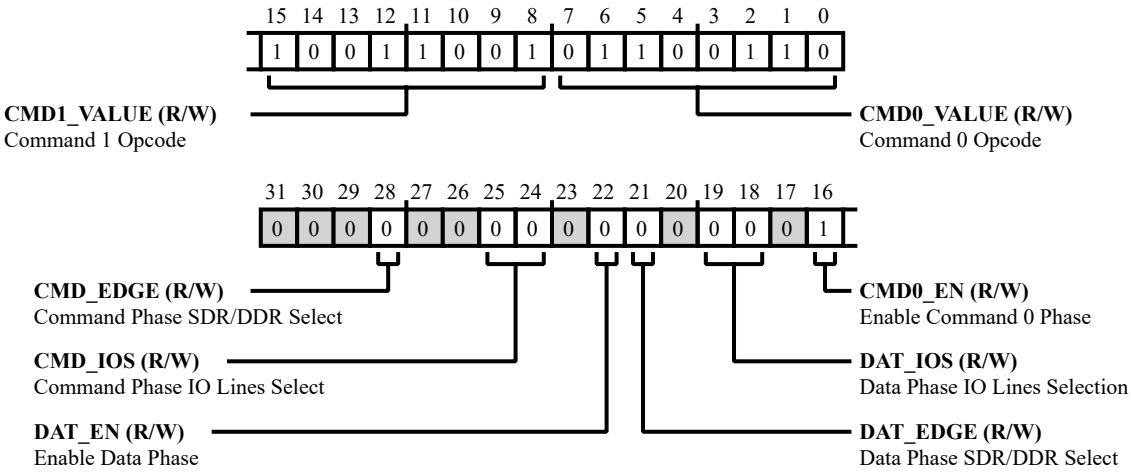


Figure 21-71: XSPI_RST_SEQ_CFG0 Register Diagram

Table 21-113: XSPI_RST_SEQ_CFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	CMD_EDGE	Command Phase SDR/DDR Select. The XSPI_RST_SEQ_CFG0.CMD_EDGE bit selects between SDR/DDR mode for both command phases.	
25:24 (R/W)	CMD_IOS	Command Phase IO Lines Select. The XSPI_RST_SEQ_CFG0.CMD_IOS bit field indicates the number of data lines used to send commands for both command phases.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
22 (R/W)	DAT_EN	Enable Data Phase. The XSPI_RST_SEQ_CFG0.DAT_EN bit enables sending the data phase (confirmation byte in) following the CMD1 phase.	
21 (R/W)	DAT_EDGE	Data Phase SDR/DDR Select. The XSPI_RST_SEQ_CFG0.DAT_EDGE bit selects between SDR/DDR mode for the data phase (confirmation byte in) following the CMD1 phase.	

Table 21-113: XSPI_RST_SEQ_CFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:18 (R/W)	DAT_IOS	Data Phase IO Lines Selection. The `XSPI_RST_SEQ_CFG0.DAT_IOS` bit field indicates the number of lines used to send the data phase (confirmation byte in) following the CMD1 phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
16 (R/W)	CMD0_EN	Enable Command 0 Phase. The `XSPI_RST_SEQ_CFG0.CMD0_EN` bit enables the CMD0 phase.	
15:8 (R/W)	CMD1_VALUE	Command 1 Opcode. The `XSPI_RST_SEQ_CFG0.CMD1_VALUE` bit field indicates the command mnemonic value for the CMD1 phase.	
7:0 (R/W)	CMD0_VALUE	Command 0 Opcode. The `XSPI_RST_SEQ_CFG0.CMD0_VALUE` bit field indicates the command mnemonic value for the CMD0 phase.	

Reset Sequence Configuration Register 1

The XSPI_RST_SEQ_CFG1 register configures the reset sequence for profile 1 and SPI NAND in ACMD work mode.

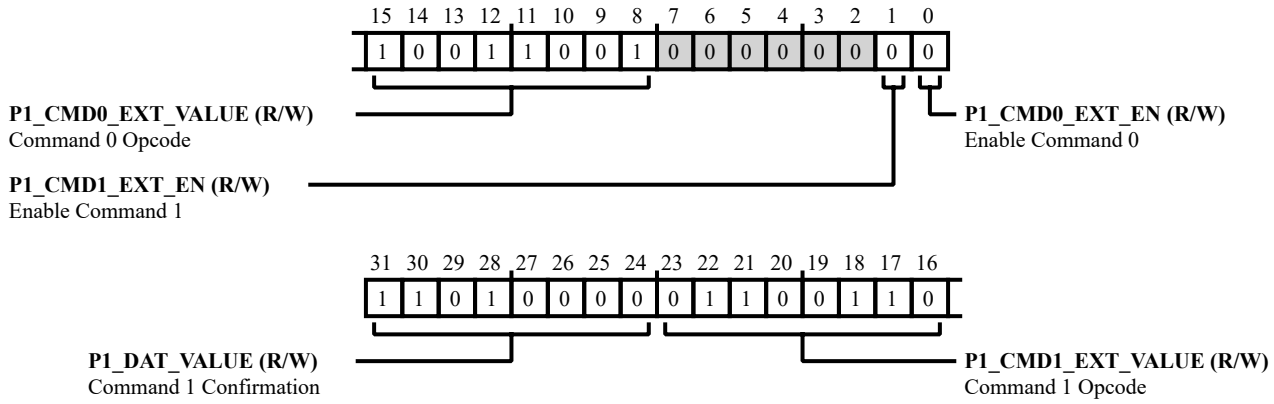


Figure 21-72: XSPI_RST_SEQ_CFG1 Register Diagram

Table 21-114: XSPI_RST_SEQ_CFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	P1_DAT_VALUE	Command 1 Confirmation. Value of confirmation byte in (if enabled) the following CMD1 phase.
23:16 (R/W)	P1_CMD1_EXT_VALUE	Command 1 Opcode. Command extension value of the CMD1 phase (if enabled).
15:8 (R/W)	P1_CMD0_EXT_VALUE	Command 0 Opcode. Command extension value of the CMD0 phase (if enabled).
1 (R/W)	P1_CMD1_EXT_EN	Enable Command 1. Command extension enable for the CMD1 phase.
0 (R/W)	P1_CMD0_EXT_EN	Enable Command 0. Command extension enable for the CMD0 phase.

Host DMA Buffer Address Register 0

The XSPI_SDMA_ADDR0 register stores the buffer address in the host memory that is used as a sink/source for the host DMA transfer. The host DMA address is based on the memory pointer field that was programed by the host as part of the CDMA/PIO command.

A single CDMA/PIO command can trigger multiple transfers on the host interface, so the host DMA address value is automatically incremented and updated before each DMA transfer.

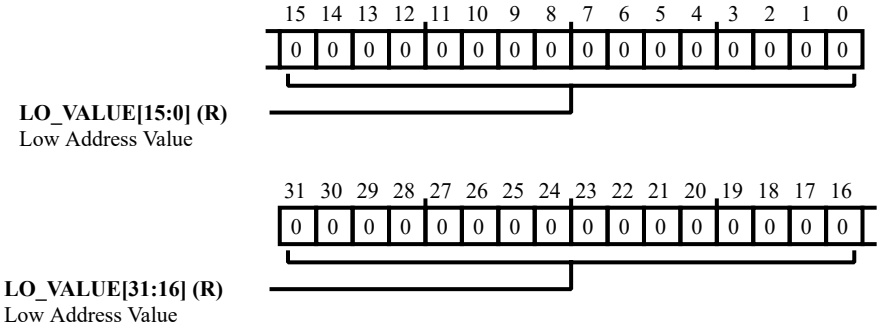


Figure 21-73: XSPI_SDMA_ADDR0 Register Diagram

Table 21-115: XSPI_SDMA_ADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	LO_VALUE	Low Address Value. The XSPI_SDMA_ADDR0.LO_VALUE bit field indicates the lower part of the SDMA destination/source address.

Host DMA Buffer Address Register 1

The XSPI_SDMA_ADDR1 register stores the buffer address in the host memory that is used as a sink/source for the host DMA transfer. The host DMA address is based on the memory pointer field that was programed by the host as part of the CDMA/PIO command.

A single CDMA/PIO command can trigger multiple transfers on the host interface, so the host DMA address value is automatically incremented and updated before each DMA transfer.

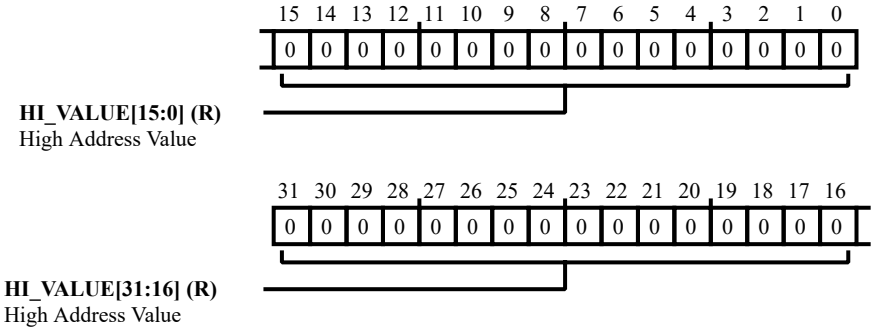


Figure 21-74: XSPI_SDMA_ADDR1 Register Diagram

Table 21-116: XSPI_SDMA_ADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	HI_VALUE	High Address Value. The XSPI_SDMA_ADDR1.HI_VALUE bit field indicates the higher part of the SDMA destination/source address.

Host DMA Block Size Register

The XSPI_SDMA_SIZ register contains the transferred data block size for the host DMA module.

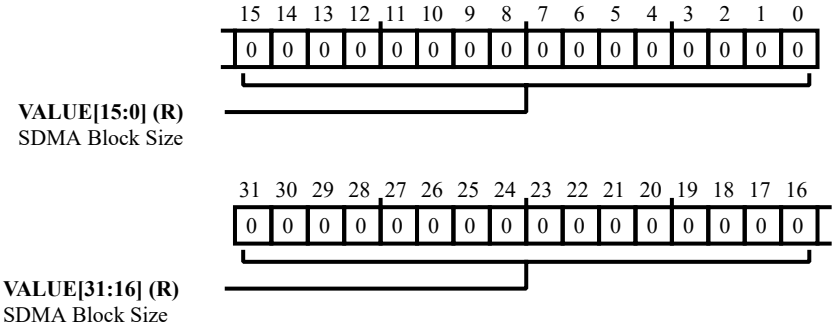


Figure 21-75: XSPI_SDMA_SIZ Register Diagram

Table 21-117: XSPI_SDMA_SIZ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	SDMA Block Size. The XSPI_SDMA_SIZ.VALUE bit field indicates the transferred data block size in bytes for the host DMA module. Data size is rounded up to the data bus word size.

Host DMA Thread Status Register

The XSPI_SDMA_TRD_STAT register contains information for the current host DMA transaction as related to the execution thread.

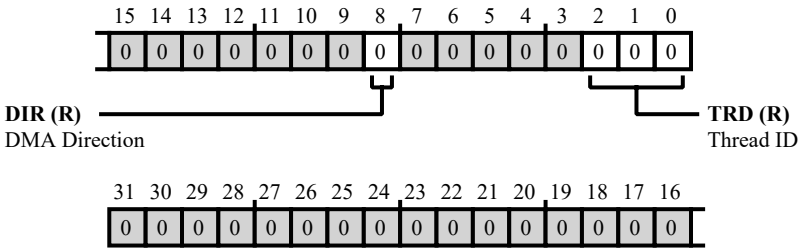


Figure 21-76: XSPI_SDMA_TRD_STAT Register Diagram

Table 21-118: XSPI_SDMA_TRD_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8 (R/NW)	DIR	DMA Direction. The XSPI_SDMA_TRD_STAT.DIR bit field indicates the transfer direction related to current host DMA transfer (0-read; 1-write).
2:0 (R/NW)	TRD	Thread ID. The XSPI_SDMA_TRD_STAT.TRD bit field indicates the thread number associated with transferred data block for the host DMA module.

Sequence Configuration Register 0

The XSPI_SEQ_GCTL0 register configures common values for sequences in CDMA, PIO and direct work modes.

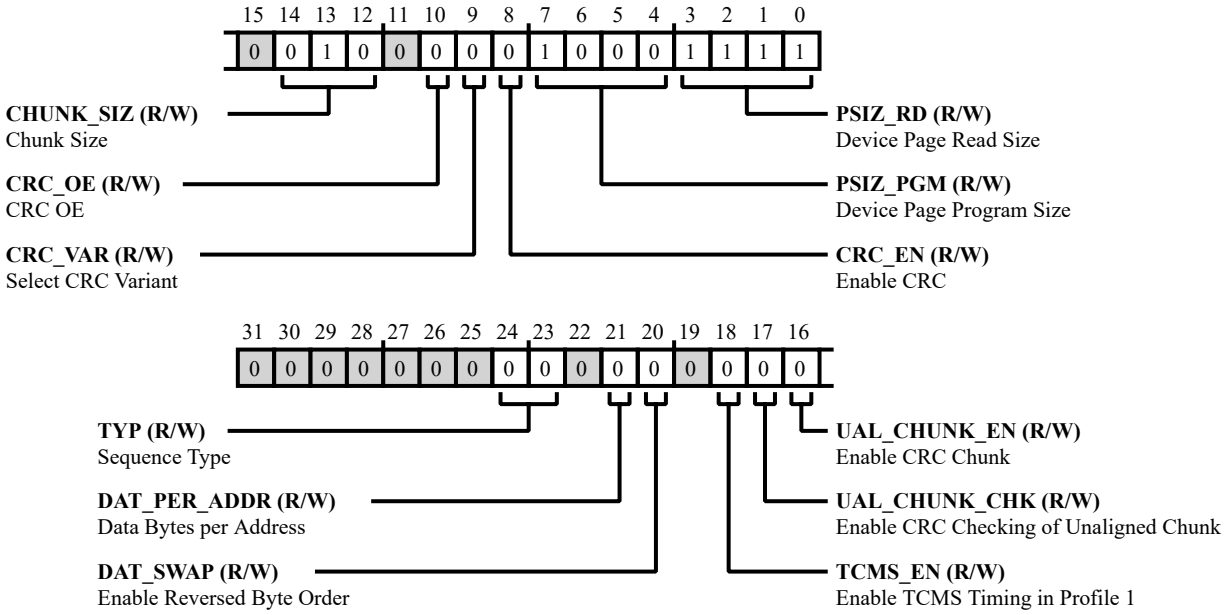


Figure 21-77: XSPI_SEQ_GCTL0 Register Diagram

Table 21-119: XSPI_SEQ_GCTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24:23 (R/W)	TYP	Sequence Type. The XSPI_SEQ_GCTL0.TYP bit field indicates the sequence type (common for all sequences).	
		0	Profile 1
		1	Profile 2 - HF (HyperFlash)
		2	Profile 2 - HR (HyperRAM)
		3	SPI NAND

Table 21-119: XSPI_SEQ_GCTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	DAT_PER_ADDR	Data Bytes per Address. The `XSPI_SEQ_GCTL0.DAT_PER_ADDR` bit indicates the data organization of the xSPI memory. Note that the xSPI address/pointer specified during sending CDMA/PIO and direct command is always byte-aligned (for example, when this field is set, the xSPI address must be even).	
		0	1B per single memory address. No translation of xSPI address.
		1	2B per single memory address. Translation of xSPI address from byte address to word address is performed automatically.
20 (R/W)	DAT_SWAP	Enable Reversed Byte Order. When set (=1), the `XSPI_SEQ_GCTL0.DAT_SWAP` bit enables reverse byte order. The `XSPI_SEQ_GCTL0.DAT_SWAP` bit can be set only when data phase reflects octal DDR mode. In other modes the `XSPI_SEQ_GCTL0.DAT_SWAP` bit must be cleared (=0).	
		0	Disable
		1	Enable
18 (R/W)	TCMS_EN	Enable TCMS Timing in Profile 1. When set (=1), the `XSPI_SEQ_GCTL0.TCMS_EN` bit enables tCMS timing in profile 1. Refer to the XSPI_MINICTL_DEV_ACTIVE_MAX register at offset 0x1018 for further details.	
		0	Disable
		1	Enable
17 (R/W)	UAL_CHUNK_CHK	Enable CRC Checking of Unaligned Chunk. Setting the `XSPI_SEQ_GCTL0.UAL_CHUNK_CHK` bit enables the checking of a CRC unaligned chunk from the flash device. It can be set (=1) only if crc_ual_chunk_en = 1. Otherwise, the `XSPI_SEQ_GCTL0.UAL_CHUNK_CHK` bit must be cleared (=0) .	
		0	Disable
		1	Enable
16 (R/W)	UAL_CHUNK_EN	Enable CRC Chunk. Setting the `XSPI_SEQ_GCTL0.UAL_CHUNK_EN` bit enables taking into consideration the command address to determine after how many bytes CRC data slice is expected to be returned by the flash device.	
		0	Disable
		1	Enable

Table 21-119: XSPI_SEQ_GCTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/W)	CHUNK_SIZ	Chunk Size. The `XSPI_SEQ_GCTL0.CHUNK_SIZ` field indicates the number of bytes after which CRC occurs.	
		0	N/A
		1	8B
		2	16B
		3	32B
		4	64B
		5	128B
		6	256B
		7	512B
10 (R/W)	CRC_OE	CRC OE. The `XSPI_SEQ_GCTL0.CRC_OE` field determines whether the controller expects the xSPI device to toggle CRC data on both SPI clock edges in CRC->CRC# sequence.	
		0	Disable
		1	Enable
9 (R/W)	CRC_VAR	Select CRC Variant. The `XSPI_SEQ_GCTL0.CRC_VAR` bit indicates the CRC variant.	
		0	Address Transfer Phase Only. CRC is calculated for all bytes of address transfer phase only and put on the bus after the address transfer phase.
		1	Sequence. CRC is calculated for all bytes in sequence and put on the bus after all bytes in the sequence.
8 (R/W)	CRC_EN	Enable CRC. When set (=1), the `XSPI_SEQ_GCTL0.CRC_EN` bit enables dynamic CRC calculation that is based on all previous bytes in the current sequence. Software puts this value on the xSPI flash interface. Not required by Legacy Hyper flash and xSPI Profile 2.0 Devices but can be useful for external flash Monitor to control data integrity.	
		0	Disable
		1	Enable

Table 21-119: XSPI_SEQ_GCTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/W)	PSIZ_PGM	Device Page Program Size. The `XSPI_SEQ_GCTL0.PSIZ_PGM` bit field indicates the page size of device being used for program operations. Number of bytes in page = 2^{page_size}. Allowed values are: 4'b0000 - 1B 4'b0001 - 2B ... 4'b1000 - 256B 4'b1001 - 512B 4'b1010 - 1024B 4'b1011 - 2048B 4'b1100 - 4096B 4'b1111 - N/A This field is not used in direct mode for profile 2 - HR or when SPI NAND device is selected.
3:0 (R/W)	PSIZ_RD	Device Page Read Size. The `XSPI_SEQ_GCTL0.PSIZ_RD` bit field indicates the page size of the device used for read operations. Number of bytes in page = 2^{page_size}. Allowed values are: 4'b0000 - 1B 4'b0001 - 2B ... 4'b1000 - 256B 4'b1001 - 512B 4'b1010 - 1024B 4'b1011 - 2048B 4'b1100 - 4096B 4'b1101 - N/A ... 4'b1111 - unlimited (controller sends all data with single xSPI command)

Sequence Configuration Register 1

The XSPI_SEQ_GCTL1 register configures common values for sequences in CDMA, PIO and direct work modes.

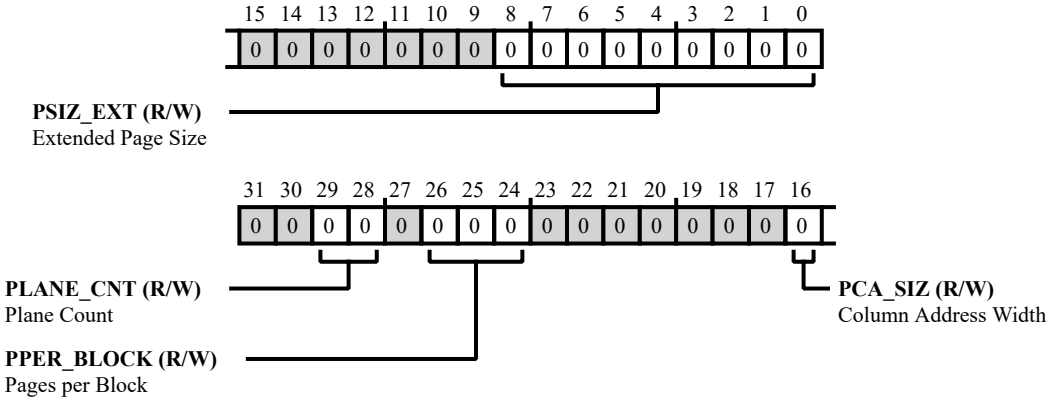


Figure 21-78: XSPI_SEQ_GCTL1 Register Diagram

Table 21-120: XSPI_SEQ_GCTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/W)	PLANE_CNT	Plane Count. The XSPI_SEQ_GCTL1.PLANE_CNT bit field indicates the number of planes in the SPI NAND device (encoded as 2^N).	
		0	Single plane
		1	Two planes
		2	Four planes
		3	Reserved
26:24 (R/W)	PPER_BLOCK	Pages per Block. The XSPI_SEQ_GCTL1.PPER_BLOCK bit field indicates the number of pages per blocks for SPI NAND device (encoded as 2^N):	
		0	1 page per block
		1	2 pages per block
		2	4 pages per block
		3	8 pages per block
		4	16 pages per block
		5	32 pages per block
		6	64 pages per block
		7	128 pages per block

Table 21-120: XSPI_SEQ_GCTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	PCA_SIZ	Column Address Width. The `XSPI_SEQ_GCTL1.PCA_SIZ` bit indicates the width of the column address for SPI NAND devices. The value of this field is used to calculate the next page address when the data size specified in the sequence exceeds the current page capacity.	
		0	12 bit address width
		1	13 bit address width
8:0 (R/W)	PSIZ_EXT	Extended Page Size. The `XSPI_SEQ_GCTL1.PSIZ_EXT` bit field indicates the extended page size area (spare area size) for SPI NAND devices. The number of data bytes transmitted to/from each page is extended by a value of this field. This field is not used in direct mode.	

Short Polling Count Register

The XSPI_SHORTPOL_GCTL register contains the status monitor cycle count value.

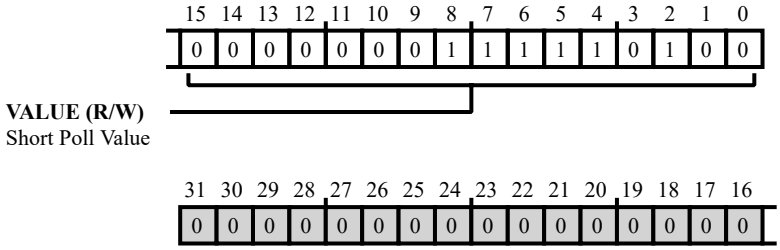


Figure 21-79: XSPI_SHORTPOL_GCTL Register Diagram

Table 21-121: XSPI_SHORTPOL_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Short Poll Value. The XSPI_SHORTPOL_GCTL.VALUE bit field indicates the minimum number of system clocks after a long polling delay before the controller starts to poll for status if the controller was busy during the first status poll attempt. The long polling value should be significantly larger the short polling value.

Status Checking Sequence Configuration Register 0

The XSPI_STAT_SEQ_CFG0 register configures the status checking sequence for profile 1 and SPI NAND in ACMD and direct work modes.

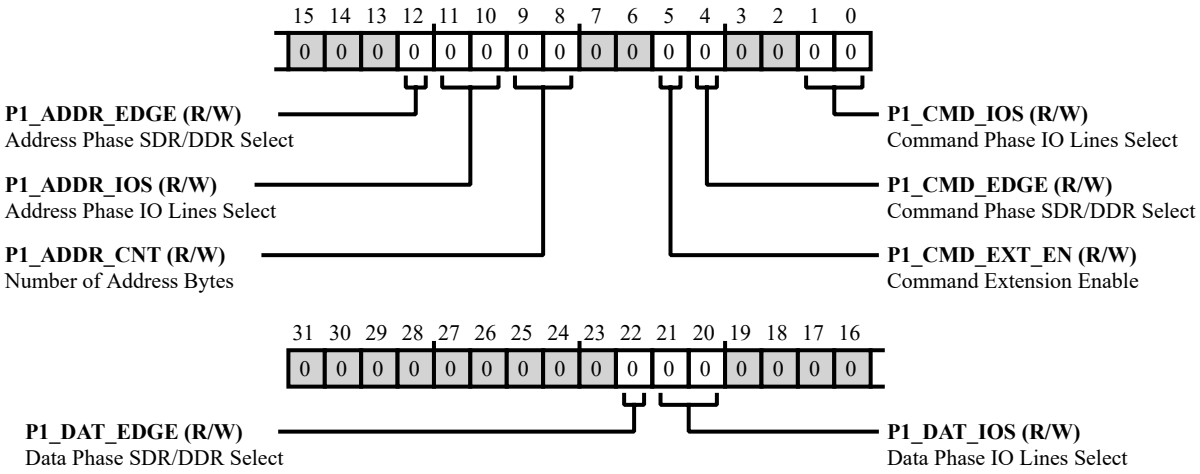


Figure 21-80: XSPI_STAT_SEQ_CFG0 Register Diagram

Table 21-122: XSPI_STAT_SEQ_CFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P1_DAT_EDGE	Data Phase SDR/DDR Select. The XSPI_STAT_SEQ_CFG0.P1_DAT_EDGE bit selects between SDR/DDR mode for data phase.	
21:20 (R/W)	P1_DAT_IOS	Data Phase IO Lines Select. The XSPI_STAT_SEQ_CFG0.P1_DAT_IOS bit field indicates the number of data lines used to send the data phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
12 (R/W)	P1_ADDR_EDGE	Address Phase SDR/DDR Select. The XSPI_STAT_SEQ_CFG0.P1_ADDR_EDGE bit selects between SDR/DDR mode for address phase.	

Table 21-122: XSPI_STAT_SEQ_CFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	P1_ADDR_IOS	Address Phase IO Lines Select. The `XSPI_STAT_SEQ_CFG0.P1_ADDR_IOS` bit field indicates the number of data lines used to send the address phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used
9:8 (R/W)	P1_ADDR_CNT	Number of Address Bytes. The `XSPI_STAT_SEQ_CFG0.P1_ADDR_CNT` bit field indicates the number of address bytes for all status sequences.	
		0	One address byte
		1	Two address bytes
		2	Three address bytes
		3	Four address bytes
5 (R/W)	P1_CMD_EXT_EN	Command Extension Enable. The `XSPI_STAT_SEQ_CFG0.P1_CMD_EXT_EN` bit enables the command extension.	
4 (R/W)	P1_CMD_EDGE	Command Phase SDR/DDR Select. The `XSPI_STAT_SEQ_CFG0.P1_CMD_EDGE` bit selects between SDR/DDR mode for command phase.	
1:0 (R/W)	P1_CMD_IOS	Command Phase IO Lines Select. The `XSPI_STAT_SEQ_CFG0.P1_CMD_IOS` bit field indicates the number of data lines used to send the command.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used

Status Checking Sequence Configuration Register 1

The XSPI_STAT_SEQ_CFG1 register configures the status checking sequence for profile 1 and SPI NAND ACMD and direct work modes.

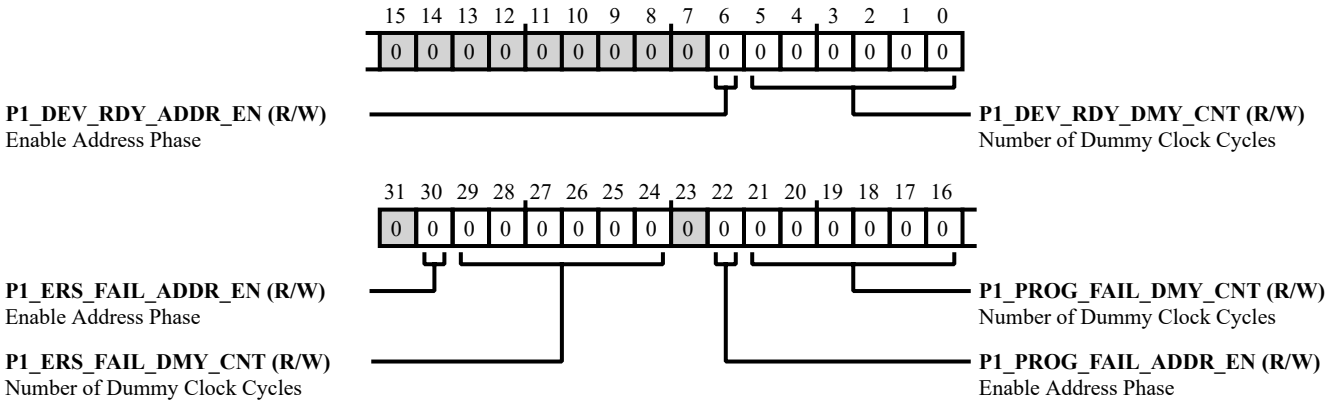


Figure 21-81: XSPI_STAT_SEQ_CFG1 Register Diagram

Table 21-123: XSPI_STAT_SEQ_CFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	P1_ERS_FAIL_ADDR_EN	Enable Address Phase. The XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_ADDR_EN bit enables the address phase for checking the fail status after the erase operation. This field is not used in direct work mode.
29:24 (R/W)	P1_ERS_FAIL_DMY_CNT	Number of Dummy Clock Cycles. The XSPI_STAT_SEQ_CFG1.P1_ERS_FAIL_DMY_CNT bit field indicates the number of dummy clock cycles used to check the fail status after the erase operation. This field is not used in direct work mode.
22 (R/W)	P1_PROG_FAIL_ADDR_EN	Enable Address Phase. The XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_ADDR_EN bit enables the address phase for checking the fail status after the program operation.
21:16 (R/W)	P1_PROG_FAIL_DMY_CNT	Number of Dummy Clock Cycles. The XSPI_STAT_SEQ_CFG1.P1_PROG_FAIL_DMY_CNT bit field indicates the number of dummy clock cycles used to check the fail status after the program operation.
6 (R/W)	P1_DEV_RDY_ADDR_EN	Enable Address Phase. The XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_ADDR_EN bit enables the address phase for checking the ready/busy status after the program/erase and soft reset/read (only for SPI NAND) operation.

Table 21-123: XSPI_STAT_SEQ_CFG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	P1_DEV_RDY_DMY_CNT	Number of Dummy Clock Cycles. The `XSPI_STAT_SEQ_CFG1.P1_DEV_RDY_DMY_CNT` bit field indicates the number of dummy clock cycles needed to check the ready/busy status after program/erase or soft reset/read (only for SPI NAND) operation.

Status Checking Sequence Configuration Register 10

The XSPI_STAT_SEQ_CFG10 register configures the status checking sequence for SPI NAND devices in ACMD and direct work modes.

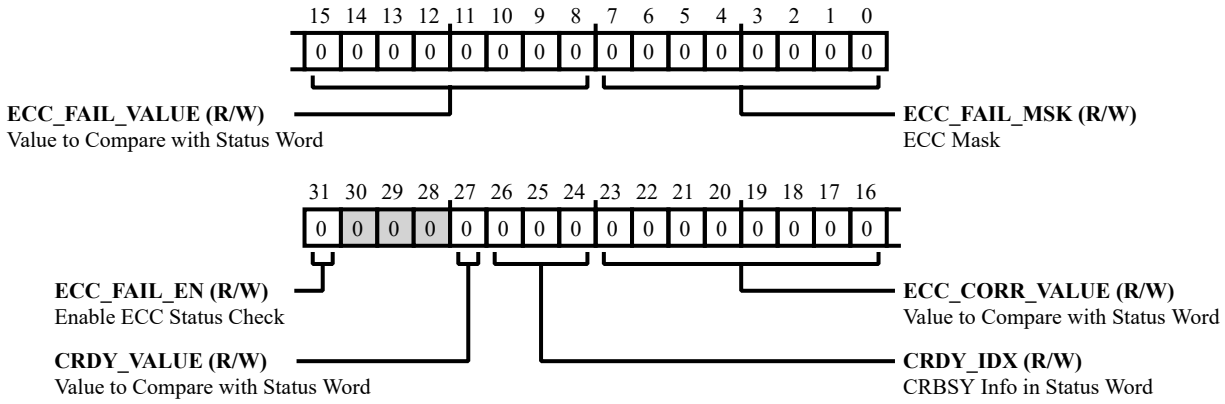


Figure 21-82: XSPI_STAT_SEQ_CFG10 Register Diagram

Table 21-124: XSPI_STAT_SEQ_CFG10 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	ECC_FAIL_EN	Enable ECC Status Check. The XSPI_STAT_SEQ_CFG10.ECC_FAIL_EN bit enables checking ECC status after a read page operation.
27 (R/W)	CRDY_VALUE	Value to Compare with Status Word. The XSPI_STAT_SEQ_CFG10.CRDY_VALUE bit indicates the value to be compared with the selected status bit in order to detect if the device is in a ready state after the read page cache random operation (CRBSY bit). This field is not used in direct work mode.
26:24 (R/W)	CRDY_IDX	CRBSY Info in Status Word. The XSPI_STAT_SEQ_CFG10.CRDY_IDX bit field indicates which bit of the status word contains the Cache Read Busy (CRBSY) bit information for the SPI NAND read page cache random operation. This field is not used in direct work mode.
23:16 (R/W)	ECC_CORR_VALUE	Value to Compare with Status Word. The XSPI_STAT_SEQ_CFG10.ECC_CORR_VALUE bit field indicates the value to be compared with the status word masked by the XSPI_STAT_SEQ_CFG10.ECC_FAIL_MSK field in order to detect if the device returned a correctable ECC error during SPI NAND page read operation. This can be used to detect a single range of correctable errors returned by the xSPI device.

Table 21-124: XSPI_STAT_SEQ_CFG10 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	ECC_FAIL_VALUE	Value to Compare with Status Word. The `XSPI_STAT_SEQ_CFG10.ECC_FAIL_VALUE` bit field indicates the value to be compared with the status word masked by the XSPI_STAT_SEQ_CFG10.ECC_FAIL_MSK field in order to detect if the device returned an uncorrectable ECC error during SPI NAND page read operation.
7:0 (R/W)	ECC_FAIL_MSK	ECC Mask. The `XSPI_STAT_SEQ_CFG10.ECC_FAIL_MSK` bit field indicates the mask used to select which bits of status word carries the ECC status.

Status Checking Sequence Configuration Register 2

The XSPI_STAT_SEQ_CFG2 register configures the status checking sequence for profile 1 and SPI NAND in ACMD and direct work modes.

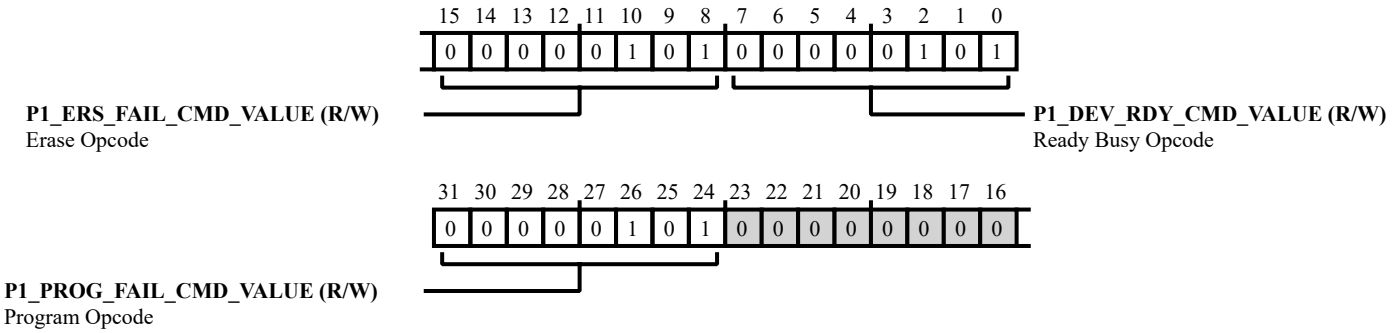


Figure 21-83: XSPI_STAT_SEQ_CFG2 Register Diagram

Table 21-125: XSPI_STAT_SEQ_CFG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	P1_PROG_FAIL_CMD_VALUE	Program Opcode. The XSPI_STAT_SEQ_CFG2.P1_PROG_FAIL_CMD_VALUE bit field contains the command mnemonic value that is used to check the fail status after program operation.
15:8 (R/W)	P1_ERS_FAIL_CMD_VALUE	Erase Opcode. The XSPI_STAT_SEQ_CFG2.P1_ERS_FAIL_CMD_VALUE bit field contains the command mnemonic value that is used to check the fail status after the erase operation. This field is not used in direct work mode.
7:0 (R/W)	P1_DEV_RDY_CMD_VALUE	Ready Busy Opcode. The XSPI_STAT_SEQ_CFG2.P1_DEV_RDY_CMD_VALUE bit field contains the command mnemonic value that is used to check the ready/busy status after program/erase and soft erase/read (only for SPI NAND) operation.

Status Checking Sequence Configuration Register 3

The XSPI_STAT_SEQ_CFG3 register configures the status checking sequence for profile 1 and SPI NAND in ACMD and direct work modes.

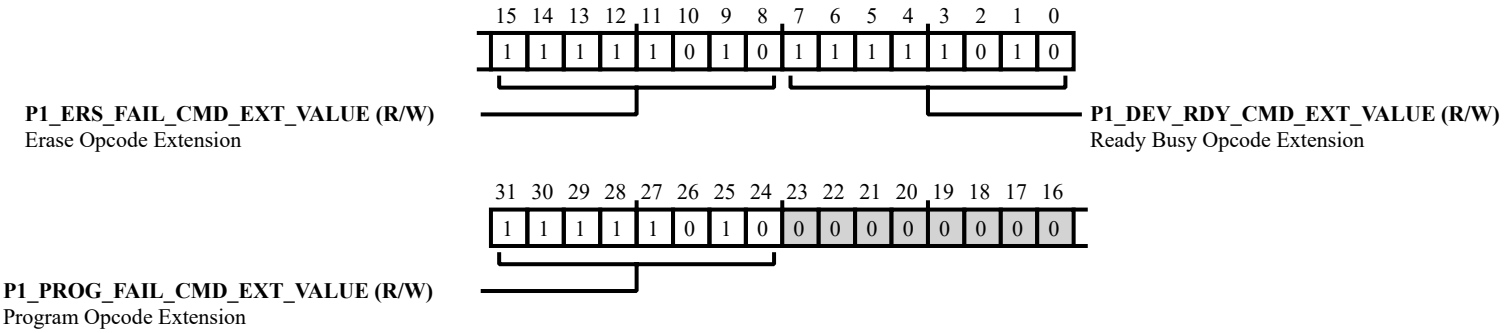


Figure 21-84: XSPI_STAT_SEQ_CFG3 Register Diagram

Table 21-126: XSPI_STAT_SEQ_CFG3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	P1_PROG_FAIL_CMD_EXT_VALUE	Program Opcode Extension. The XSPI_STAT_SEQ_CFG3.P1_PROG_FAIL_CMD_EXT_VALUE bit field contains the command mnemonic value that is used to check the fail status after the program operation.
15:8 (R/W)	P1_ERS_FAIL_CMD_EXT_VALUE	Erase Opcode Extension. The XSPI_STAT_SEQ_CFG3.P1_ERS_FAIL_CMD_EXT_VALUE bit field contains the command mnemonic value that is used to check the fail status after the erase operation. This field is not used in direct work mode.
7:0 (R/W)	P1_DEV_RDY_CMD_EXT_VALUE	Ready Busy Opcode Extension. The XSPI_STAT_SEQ_CFG3.P1_DEV_RDY_CMD_EXT_VALUE bit field contains the command mnemonic value that is used to check the ready/busy status after the program/erase and soft reset/read (only for SPI NAND) operation.

Status Checking Sequence Configuration Register 4

The XSPI_STAT_SEQ_CFG4 register configures the status checking sequence for profile 2 - HF in ACMD and direct work modes.

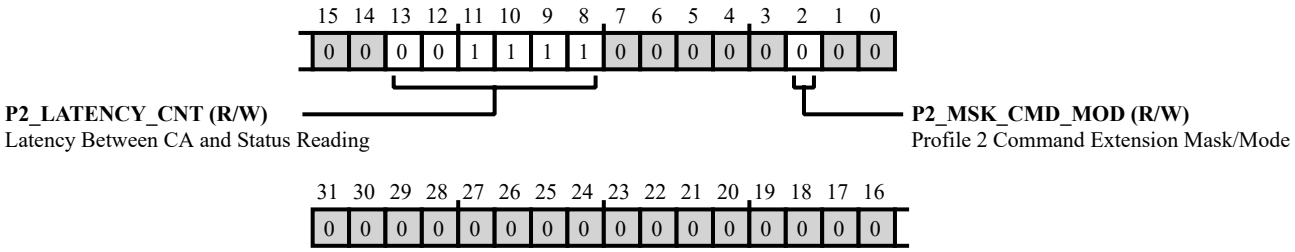


Figure 21-85: XSPI_STAT_SEQ_CFG4 Register Diagram

Table 21-127: XSPI_STAT_SEQ_CFG4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:8 (R/W)	P2_LATENCY_CNT	Latency Between CA and Status Reading. The XSPI_STAT_SEQ_CFG4.P2_LATENCY_CNT bit field indicates the number of latency cycles between the CA and status reading. Configure XSPI_STAT_SEQ_CFG4.P2_LATENCY_CNT to N-1, where N is the number of latency clock cycles expected by the memory device.
2 (R/W)	P2_MSK_CMD_MOD	Profile 2 Command Extension Mask/Mode. The XSPI_STAT_SEQ_CFG4.P2_MSK_CMD_MOD bit field indicates the profile 2 command extension variant. The value of this field influences bits[44:40] of the Command/Address (CA). If this bit is set (=1), bits [44:40] of the CA are set (=1). Otherwise, bits [44:40] of the CA are cleared (=0).

Status Checking Sequence Configuration Register 5

The XSPI_STAT_SEQ_CFG5 register configures the status checking sequence for profile 1, SPI NAND and profile 2 - HF in ACMD and direct work modes.

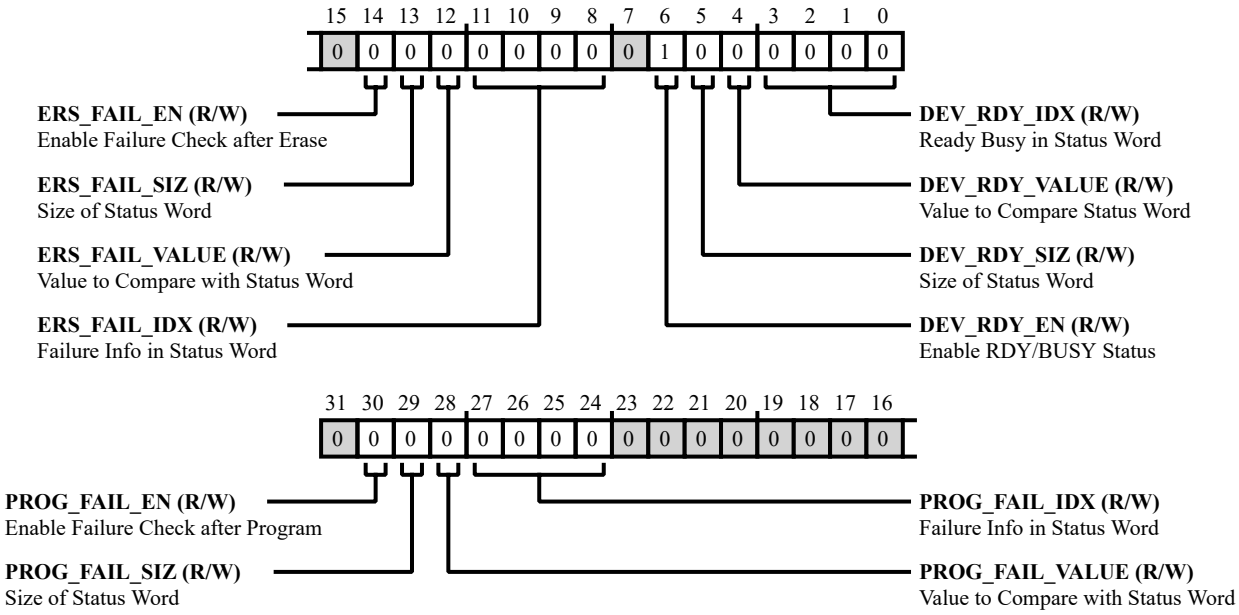


Figure 21-86: XSPI_STAT_SEQ_CFG5 Register Diagram

Table 21-128: XSPI_STAT_SEQ_CFG5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	PROG_FAIL_EN	Enable Failure Check after Program. The XSPI_STAT_SEQ_CFG5.PROG_FAIL_EN bit enables checking the fail status after program operation.
29 (R/W)	PROG_FAIL_SIZ	Size of Status Word. The XSPI_STAT_SEQ_CFG5.PROG_FAIL_SIZ bit indicates the size of the status word (0-1B, 1-2B).
28 (R/W)	PROG_FAIL_VALUE	Value to Compare with Status Word. The XSPI_STAT_SEQ_CFG5.PROG_FAIL_VALUE bit indicates the value to be compared with selected status bit in order to detect that the device is in fail state after program operation.
27:24 (R/W)	PROG_FAIL_IDX	Failure Info in Status Word. The XSPI_STAT_SEQ_CFG5.PROG_FAIL_IDX bit field indicates which bit of the status word contains the failure information after the program command.

Table 21-128: XSPI_STAT_SEQ_CFG5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	ERS_FAIL_EN	Enable Failure Check after Erase. The `XSPI_STAT_SEQ_CFG5.ERS_FAIL_EN` bit enables checking the fail status after an erase operation. This field is not used in direct work mode.
13 (R/W)	ERS_FAIL_SIZ	Size of Status Word. The `XSPI_STAT_SEQ_CFG5.ERS_FAIL_SIZ` bit indicates the size of status word (0-1B, 1-2B). This field is not used in direct work mode.
12 (R/W)	ERS_FAIL_VALUE	Value to Compare with Status Word. The `XSPI_STAT_SEQ_CFG5.ERS_FAIL_VALUE` bit indicates the value to be compared with the selected status bit in order to detect that the device is in the ready state after erase operation. This field is not used in direct work mode.
11:8 (R/W)	ERS_FAIL_IDX	Failure Info in Status Word. The `XSPI_STAT_SEQ_CFG5.ERS_FAIL_IDX` bit field indicates which bit of the status word contains the failure information after the erase command. This field is not used in direct work mode.
6 (R/W)	DEV_RDY_EN	Enable RDY/BUSY Status. The `XSPI_STAT_SEQ_CFG5.DEV_RDY_EN` bit enables checking the ready/busy status after program/erase operations.
5 (R/W)	DEV_RDY_SIZ	Size of Status Word. The `XSPI_STAT_SEQ_CFG5.DEV_RDY_SIZ` bit indicates the size of status word (0-1B, 1-2B).
4 (R/W)	DEV_RDY_VALUE	Value to Compare Status Word. The `XSPI_STAT_SEQ_CFG5.DEV_RDY_VALUE` bit indicates the value to be compared with selected status bit in order to detect that the device is ready after a program/erase or soft reset/read (only for SPI NAND) operation.
3:0 (R/W)	DEV_RDY_IDX	Ready Busy in Status Word. The `XSPI_STAT_SEQ_CFG5.DEV_RDY_IDX` bit field indicates which bit of the status word contains the ready/busy information to be polled after a program/erase or soft erase/read (only for SPI NAND) operation.

Status Checking Sequence Configuration Register 7

The XSPI_STAT_SEQ_CFG7 register configures the status checking sequence for profile 1, SPI NAND and profile 2 - HF in ACMD and direct work modes.

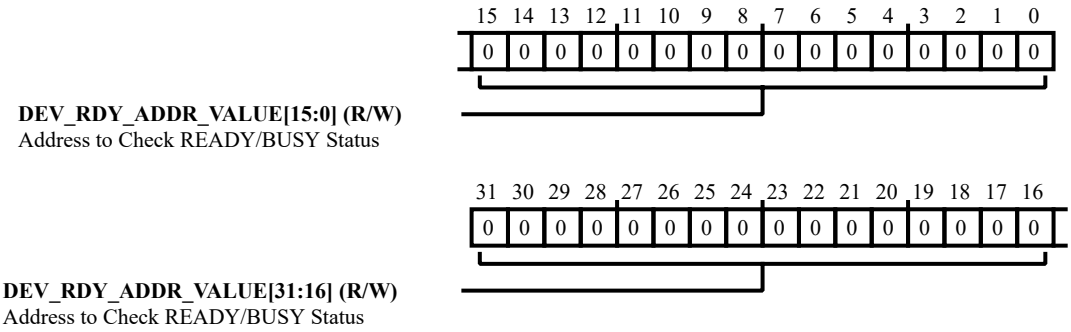


Figure 21-87: XSPI_STAT_SEQ_CFG7 Register Diagram

Table 21-129: XSPI_STAT_SEQ_CFG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DEV_RDY_ADDR_VALUE	Address to Check READY/BUSY Status. The XSPI_STAT_SEQ_CFG7.DEV_RDY_ADDR_VALUE bit field indicates the value of the address used to check the ready/busy status after the program/erase and soft reset/read (only for SPI NAND) operation.

Status Checking Sequence Configuration Register 8

The XSPI_STAT_SEQ_CFG8 register configures the status checking sequence for profile 1, SPI NAND and profile 2 - HF in ACMD and direct work modes.

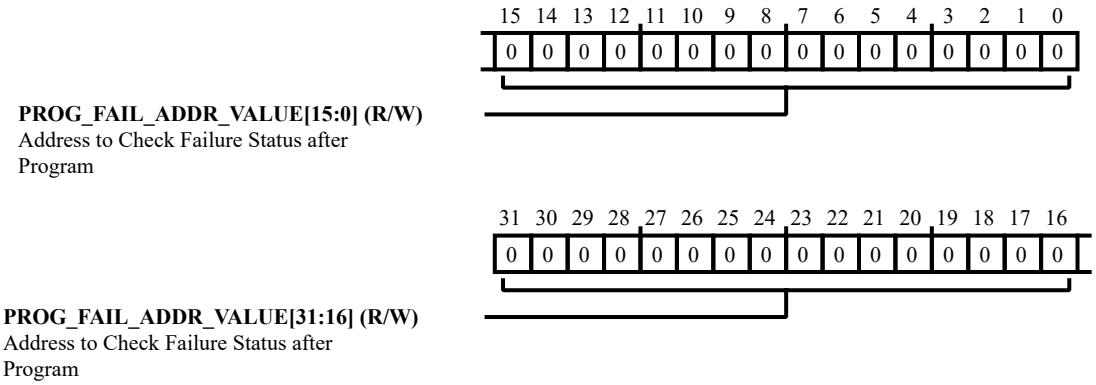


Figure 21-88: XSPI_STAT_SEQ_CFG8 Register Diagram

Table 21-130: XSPI_STAT_SEQ_CFG8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PROG_FAIL_ADDR_VALUE	Address to Check Failure Status after Program. The XSPI_STAT_SEQ_CFG8.PROG_FAIL_ADDR_VALUE bit field indicates the value of the address used to check the fail status after the program operation.

Status Checking Sequence Configuration Register 9

The XSPI_STAT_SEQ_CFG9 register configures the status checking sequence for profile 1, SPI NAND and profile 2 - HF in ACMD work mode.

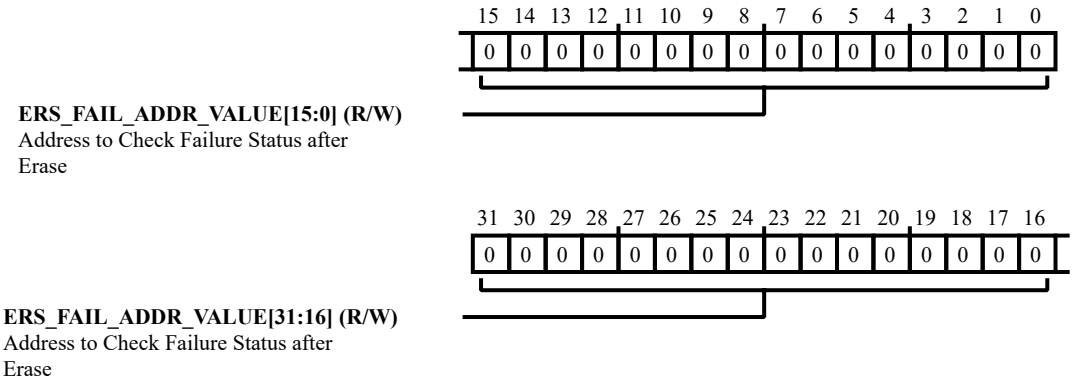


Figure 21-89: XSPI_STAT_SEQ_CFG9 Register Diagram

Table 21-131: XSPI_STAT_SEQ_CFG9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ERS_FAIL_ADDR_VALUE	Address to Check Failure Status after Erase. The XSPI_STAT_SEQ_CFG9.ERS_FAIL_ADDR_VALUE bit field indicates the value of the address used to check fail status after erase operation.

Auto Command Engine Interrupt Status Thread Register

Each bit of the field corresponds to an auto command engine thread and holds the descriptor status for that selected thread. It is set only when INT bit of descriptor is set.

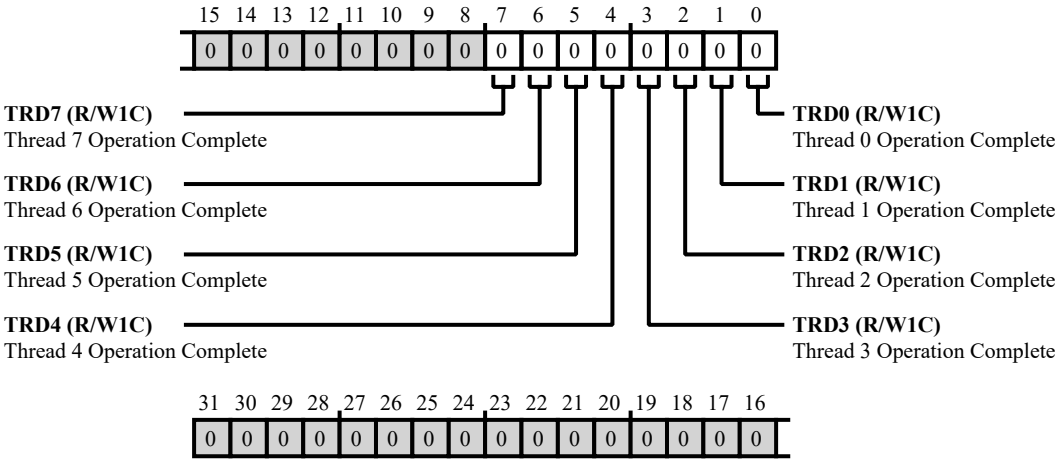


Figure 21-90: XSPI_TRD_COMP_ISTAT Register Diagram

Table 21-132: XSPI_TRD_COMP_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W1C)	TRD7	Thread 7 Operation Complete. Thread 7 operation complete flag.
6 (R/W1C)	TRD6	Thread 6 Operation Complete. Thread 6 operation complete flag.
5 (R/W1C)	TRD5	Thread 5 Operation Complete. Thread 5 operation complete flag.
4 (R/W1C)	TRD4	Thread 4 Operation Complete. Thread 4 operation complete flag.
3 (R/W1C)	TRD3	Thread 3 Operation Complete. Thread 3 operation complete flag.
2 (R/W1C)	TRD2	Thread 2 Operation Complete. Thread 2 operation complete flag.
1 (R/W1C)	TRD1	Thread 1 Operation Complete. Thread 1 operation complete flag.
0 (R/W1C)	TRD0	Thread 0 Operation Complete. Thread 0 operation complete flag.

Thread Error Interrupt Enable Register

If the selected bit of the XSPI_TRD_ERR_INT_EN register is set, the rising edge of the corresponding bit in the XSPI_TRD_COMP_ISTAT register causes the setting of the external interrupt line.

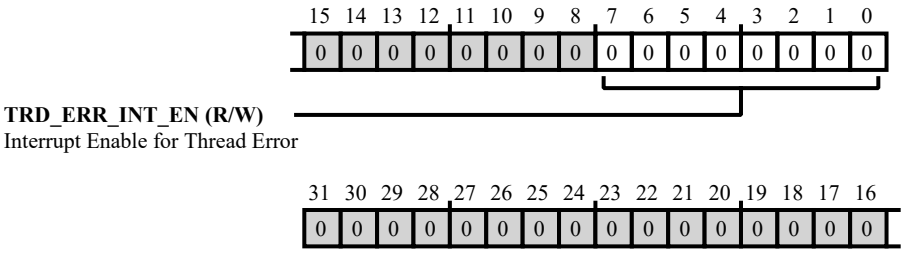


Figure 21-91: XSPI_TRD_ERR_INT_EN Register Diagram

Table 21-133: XSPI_TRD_ERR_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	TRD_ERR_INT_EN	Interrupt Enable for Thread Error. Interrupt enable for detecting a thread error.

Thread Error Interrupt Status Register

The XSPI_TRD_ERR_ISTAT register can be read to determine whether a particular thread (in the auto command engine) has encountered an error condition.

To get more information on the error, software must read the status field of the descriptor (CDMA mode) or the appropriate status register (PIO mode).

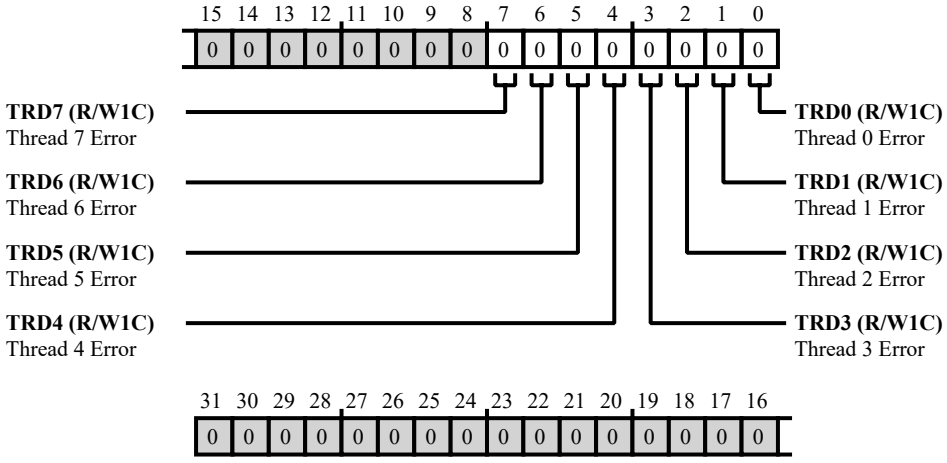


Figure 21-92: XSPI_TRD_ERR_ISTAT Register Diagram

Table 21-134: XSPI_TRD_ERR_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W1C)	TRD7	Thread 7 Error. Thread 7 error.
6 (R/W1C)	TRD6	Thread 6 Error. Thread 6 error.
5 (R/W1C)	TRD5	Thread 5 Error. Thread 5 error.
4 (R/W1C)	TRD4	Thread 4 Error. Thread 4 error.
3 (R/W1C)	TRD3	Thread 3 Error. Thread 3 error.
2 (R/W1C)	TRD2	Thread 2 Error. Thread 2 error.
1 (R/W1C)	TRD1	Thread 1 Error. Thread 1 error.

Table 21-134: XSPI_TRD_ERR_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W1C)	TRD0	Thread 0 Error. Thread 0 error.

Auto Command Engine Thread Status Register

The XSPI_TRD_STAT register indicates the auto command engine thread status (ACMD mode only).

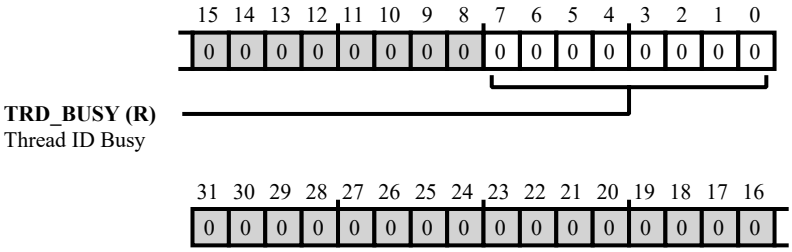


Figure 21-93: XSPI_TRD_STAT Register Diagram

Table 21-135: XSPI_TRD_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	TRD_BUSY	Thread ID Busy. The XSPI_TRD_STAT.TRD_BUSY bit indicates the auto command engine thread busy status. When set, the corresponding thread is currently busy.

WEL Sequence Configuration Register

The XSPI_WE_SEQ_CFG0 register configures the Write Enable Latch (WEL) sequence for profile 1 and SPI NAND in ACMD and direct work modes.

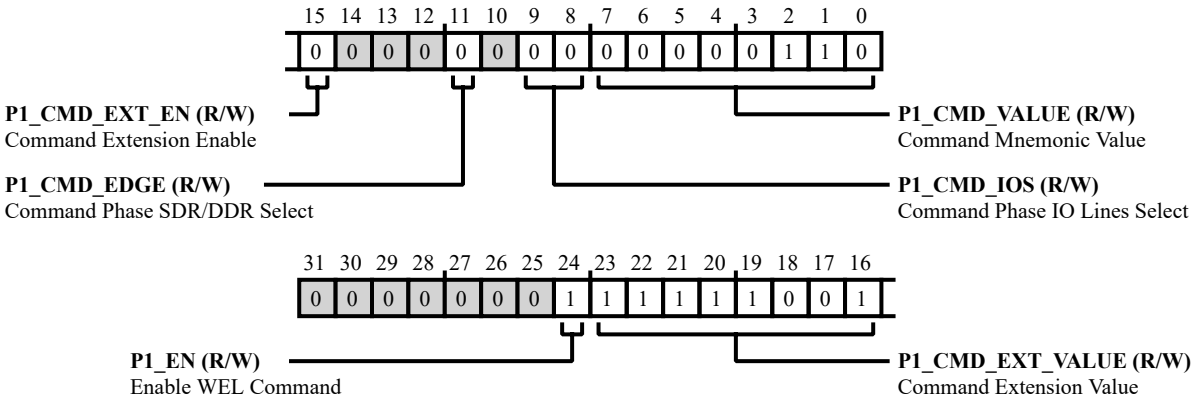


Figure 21-94: XSPI_WE_SEQ_CFG0 Register Diagram

Table 21-136: XSPI_WE_SEQ_CFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	P1_EN	Enable WEL Command. The XSPI_WE_SEQ_CFG0.P1_EN bit enables the WEL command.	
23:16 (R/W)	P1_CMD_EXT_VALUE	Command Extension Value. The XSPI_WE_SEQ_CFG0.P1_CMD_EXT_VALUE bit field indicates the command extension value.	
15 (R/W)	P1_CMD_EXT_EN	Command Extension Enable. The XSPI_WE_SEQ_CFG0.P1_CMD_EXT_EN bit enables the command extension.	
11 (R/W)	P1_CMD_EDGE	Command Phase SDR/DDR Select. The XSPI_WE_SEQ_CFG0.P1_CMD_EDGE bit selects between SDR/DDR mode for command phase.	
9:8 (R/W)	P1_CMD_IOS	Command Phase IO Lines Select. The XSPI_WE_SEQ_CFG0.P1_CMD_IOS bit field indicates the number of data lines used to send the command phase.	
		0	One data line used (for example, serial)
		1	Two data lines used
		2	Four data lines used
		3	Eight data lines used

Table 21-136: XSPI_WE_SEQ_CFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	P1_CMD_VALUE	Command Mnemonic Value. The `XSPI_WE_SEQ_CFG0.P1_CMD_VALUE` bit field indicates the command mnemonic value.

Device Control Register

The XSPI_WORKMODE_CTL register controls the device working modes.

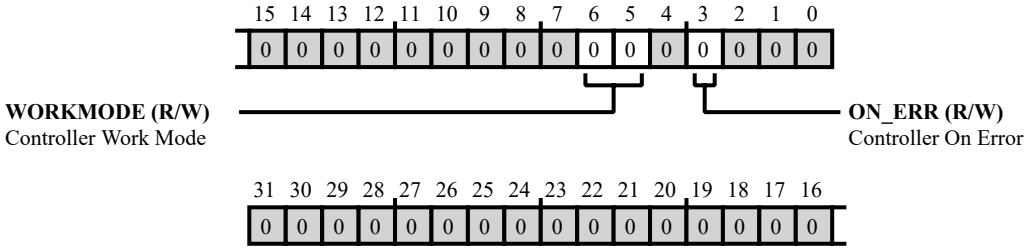


Figure 21-95: XSPI_WORKMODE_CTL Register Diagram

Table 21-137: XSPI_WORKMODE_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:5 (R/W)	WORKMODE	Controller Work Mode. The XSPI_WORKMODE_CTL.WORKMODE bit field indicates the work mode of the controller. Allowed values are 00 - Direct mode 01 - STIG mode 11 - ACMD mode
3 (R/W)	ON_ERR	Controller On Error. This control bit is relevant only for ACMD work mode. If the XSPI_WORKMODE_CTL.ON_ERR bit is cleared (=0) and an error occurs, the controller drops execution of all further operations programmed in a single thread at the page (for read or program commands) or sector (for the sector erase command) boundary. The XSPI_WORKMODE_CTL.ON_ERR bit applies to both command DMA and PIO work modes. When XSPI_WORKMODE_CTL.ON_ERR is set (=1), execution continues.

XIP Configuration Register

The XSPI_XIP_GCTL register configures the controller in XIP work mode in CDMA, PIO and direct work modes.

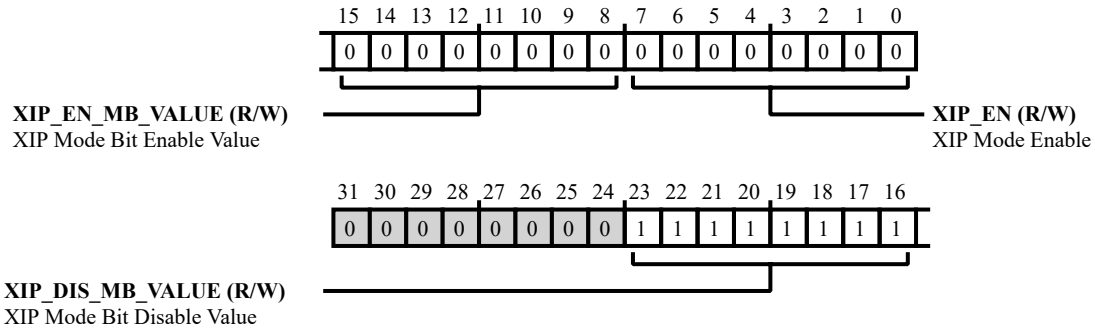


Figure 21-96: XSPI_XIP_GCTL Register Diagram

Table 21-138: XSPI_XIP_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	XIP_DIS_MB_VALUE	XIP Mode Bit Disable Value. The XSPI_XIP_GCTL.XIP_DIS_MB_VALUE bit field indicates the value of the mode bits required to disable XIP mode.
15:8 (R/W)	XIP_EN_MB_VALUE	XIP Mode Bit Enable Value. The XSPI_XIP_GCTL.XIP_EN_MB_VALUE bit field indicates the value of the mode bits required to enable XIP mode.
7:0 (R/W)	XIP_EN	XIP Mode Enable. The XSPI_XIP_GCTL.XIP_EN bit field indicates the XIP mode enabled for a selected memory bank. When XIP mode is enabled, only read sequences are valid. Invoking any other command sequence is ignored and CMD_ERROR/DSC_ERROR/XSPI_ISTAT.DIR_CMD_ERR =1.

22 Universal Asynchronous Receiver/Transmitter (UART)

The UART module is a full-duplex peripheral compatible with PC-style industry-standard UARTs. The UART converts data between serial and parallel formats. The serial communication follows an asynchronous protocol that supports various word lengths, stop bits, bit rates, and parity-generation options. Multiple events can generate interrupts.

The UART is logically compliant to EIA-232E, EIA-422, EIA-485 and LIN standards, but usually requires external transceiver devices to meet electrical requirements. In IrDA (Infrared Data Association) mode, the UART meets the half-duplex IrDA SIR (9.6/115.2 Kbps rate) protocol. In multi-drop bus mode, the UART meets the full-duplex MDB/ICP v2.0 protocol.

The UART module supports partial modem status and control functionality to allow for hardware flow control.

The UART is a DMA-capable peripheral with separate transmit and receive DMA controller channels. The use of DMA requires minimal software intervention as the DMA engine moves the data. The UART can also use a programmed core mode of operation. The core mode requires software management of the data flow using either interrupts or polling.

The UART can use one of the peripheral timers for a hardware-assisted auto-baud detection mechanism. The timers are external to the UART.

NOTE: The UARTs (UART0, UART1, and UART2) operate in the SCLK0 domain.

UART Features

Each UART includes the following features.

- 5–8 data bits
- Programmable extra stop bit and programmable extra half-stop bit
- Even, odd, and sticky parity bit options
- Extra 8-stage receive FIFO with programmable threshold interrupt request
- Flexible transmit and receive interrupt request timing

- Three interrupt outputs for receive, transmit, and status
- Independent DMA operation for receive and transmit
- Programmable automatic request to send (RTS)/clear to send (CTS) hardware flow control
- False start bit detection
- SIR IrDA operation mode
- MDB/ICP v2.0 operation mode
- Internal loopback
- Improved bit rate granularity
- LIN break command/Inter-frame gap transmission support

Table 22-1: UART Specifications

Feature	Availability
Protocol	
Controller-Capable	Yes
Target-Capable	Yes
Transmission Simplex	Yes
Transmission Half-Duplex	Yes
Transmission Full-Duplex	Yes
Access Type	
Data Buffer	Yes
Core Data Access	Yes
DMA Data Access	Yes
DMA Channels	2 (per UART Port)
DMA Descriptor	Yes
Boot Capable	Yes (Target Mode)
Local Memory	No
Clock Operation	SCLK/16

UART Functional Description

The following sections provide details on the UARTs functionality.

ADSP-2184x UART Register List

The Universal Asynchronous Receiver/Transmitter module (UART) is a full-duplex peripheral compatible with PC-style industry-standard UARTs. The UART converts data between serial and parallel formats. The serial communication follows an asynchronous protocol that supports various word length, stop bit, parity, and interrupt generation options. A set of registers governs UART operations. For more information on UART functionality, see the UART register descriptions.

Table 22-2: ADSP-2184x UART Register List

Name	Description
UART_CLK	Clock Rate Register
UART_CTL	Control Register
UART_IMSK	Interrupt Mask Register
UART_IMSK_CLR	Interrupt Mask Clear Register
UART_IMSK_SET	Interrupt Mask Set Register
UART_RBR	Receive Buffer Register
UART_RSR	Receive Shift Register
UART_RXCNT	Receive Counter Register
UART_SCR	Scratch Register
UART_STAT	Status Register
UART_TAIP	Transmit Address/Insert Pulse Register
UART_THR	Transmit Hold Register
UART_TSR	Transmit Shift Register
UART_TXCNT	Transmit Counter Register

ADSP-2184x UART Interrupt List

Table 22-3: ADSP-2184x UART Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
367	UART0_TXDMA	UART0 Transmit DMA	Level	20
368	UART0_RXDMA	UART0 Receive DMA	Level	21
369	UART0_STAT	UART0 Status	Level	
370	UART1_TXDMA	UART1 Transmit DMA	Level	34
371	UART1_RXDMA	UART1 Receive DMA	Level	35
372	UART1_STAT	UART1 Status	Level	
373	UART2_TXDMA	UART2 Transmit DMA	Level	37

Table 22-3: ADSP-2184x UART Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
374	UART2_RXDMA	UART2 Receive DMA	Level	38
375	UART2_STAT	UART2 Status	Level	
376	UART0_TXDMA_ERR	UART0 Transmit DMA Error	Level	
377	UART0_RXDMA_ERR	UART0 Receive DMA Error	Level	
378	UART1_TXDMA_ERR	UART1 Transmit DMA Error	Level	
379	UART1_RXDMA_ERR	UART1 Receive DMA Error	Level	
380	UART2_TXDMA_ERR	UART2 Transmit DMA Error	Level	
381	UART2_RXDMA_ERR	UART2 Receive DMA Error	Level	

ADSP-2184x UART Trigger List

Table 22-4: ADSP-2184x UART Trigger List Generators

Trigger ID	Name	Description	Sensitivity
191	UART0_TXDMA	UART0 Transmit DMA	Edge
192	UART0_RXDMA	UART0 Receive DMA	Edge
193	UART1_TXDMA	UART1 Transmit DMA	Edge
194	UART1_RXDMA	UART1 Receive DMA	Edge
195	UART2_TXDMA	UART2 Transmit DMA	Edge
196	UART2_RXDMA	UART2 Receive DMA	Edge

Table 22-5: ADSP-2184x UART Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
232	UART0_TXDMA	UART0 Transmit DMA	Pulse
233	UART0_RXDMA	UART0 Receive DMA	Pulse
234	UART1_TXDMA	UART1 Transmit DMA	Pulse
235	UART1_RXDMA	UART1 Receive DMA	Pulse
236	UART2_TXDMA	UART2 Transmit DMA	Pulse
237	UART2_RXDMA	UART2 Receive DMA	Pulse

ADSP-2184x UART DMA Channel List

Table 22-6: ADSP-2184x UART DMA Channel List

DMA ID	DMA Channel Name	Description
DMA20	UART0_TXDMA	UART0 Transmit DMA
DMA21	UART0_RXDMA	UART0 Receive DMA
DMA34	UART1_TXDMA	UART1 Transmit DMA
DMA35	UART1_RXDMA	UART1 Receive DMA
DMA37	UART2_TXDMA	UART2 Transmit DMA
DMA38	UART2_RXDMA	UART2 Receive DMA

UART Block Diagram

The ***UART Block Diagram*** figure shows a simplified block diagram of one UART module and how it interconnects to the processor system.

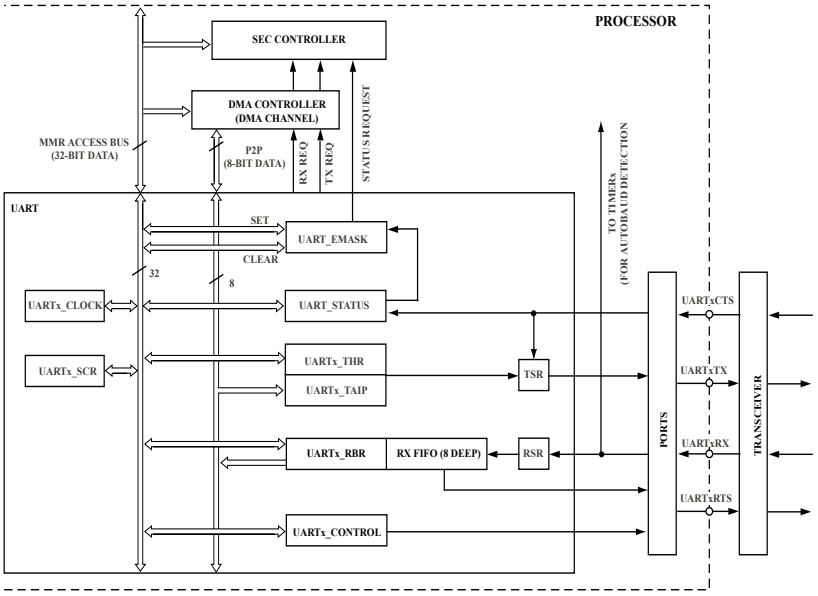


Figure 22-1: UART Block Diagram

UART Architectural Concepts

The following sections provide information about the UART architecture.

Internal Interface

The UART is a DMA-capable peripheral with support for separate transmit and receive DMA controller channels. It operates in either DMA or programmed core modes. The core mode requires software management of the data flow

using either interrupts or polling. The DMA method requires minimal software intervention, as the DMA engine itself moves the data. The UART_RBR and UART_THR registers also connect to one of the peripheral DMA buses.

All UART registers are 32 bits, and the registers connect to the peripheral MMR bus. Not all MMRs can be used, and unused bits are zero-filled. The UART has three interrupt outputs described as follows.

- The transmit and receive request outputs can function as DMA requests and connect to the DMA controller. Therefore, if the DMA is not enabled, the DMA controller simply forwards the request to the system event controller (SEC).
- The status interrupt output connects directly to the SEC . On many processors, the alternative capture input () of one of the GP timers also senses the UART_RX pin. When configured in capture mode, the processor can then use the GP timer to detect the bit rate of the received signal.

External Interface

Each UART features a UART_RX (receive) pin and a UART_TX (transmit) pin available through the general-purpose ports. These two pins usually connect to an external transceiver device that meets the electrical requirements of full-duplex or half-duplex standards. For example, EIA-232, EIA-422, 4-wire EIA-485 for full-duplex or 2-wire EIA-485, LIN for half-duplex. Additionally, the UART features a pair of clear-to-send, input pins (UART_CTS), and request-to-send, output pins (UART_RTS) for hardware flow control. UART signals are multiplexed with other functions at the pin level.

Hardware Flow Control

To prevent the UART transmitter from sending data while the receiving counterpart is not ready, the UART features a UART_RTS/UART_CTS hardware flow control mechanism. The UART_RTS signal is an output that connects to the UART_CTS input of the communication partner. If data transfer is bidirectional, the figure shows the ***UART Hardware Flow Control*** handshake.

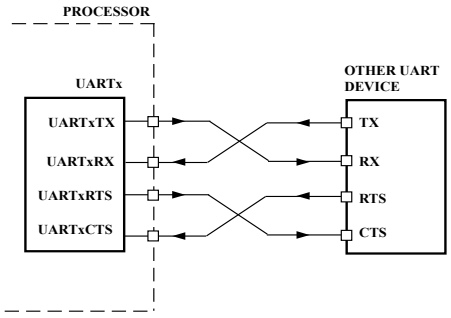


Figure 22-2: UART Hardware Flow Control

In both DMA and core mode, the receiver can deassert the UART_RTS signal to indicate that its receive buffer is almost full. Continued data transfers can cause an overrun error. The transmitter pauses when the UART_CTS input is in a deasserted state. In this state, the transmitter completes transmission of the data currently held in the transmit shift register (UART_TSR) but it does not continue with the data in the transmit hold register (UART_THR). If the UART_CTS pin is asserted again, the transmitter resumes and loads the content of UART_THR register into the UART_TSR register.

Bit Rate Generation

The peripheral clock (SCLK) and the 16-bit divisor in the UART_CLK register characterize the sample clock. The UART uses the UART_CTL.EN bit to enable the clock. By default, every serial bit is oversampled 16 times. The bit clock is 1/16th of the sample clock. If not in IrDA mode, the bit clock can equal the sample clock if the UART_CLK.EDBO bit is set, so that the following equation applies:

$$\text{Bit Rate} = \text{SCLK}/(16^{(1-\text{EDBO})} \times \text{Divisor})$$

The ***UART Bit Rate Examples with 100 MHz SCLK*** table provides example divide factors required to support standard baud rates at an SCLK of 100 MHz.

Table 22-7: UART Bit Rate Examples with 100 MHz SCLK

Bit Rate (bits/sec)	EDBO = 0			EDBO = 1		
	DL	Actual	% Error	DL	Actual	% Error
2400	2604	2400.15	0.006	41667	2399.98	0.001
4800	1302	4800.31	0.006	20833	4800.08	0.002
9600	651	9600.61	0.006	10417	9599.69	0.003
19200	326	19171.78	0.147	5208	19201.23	0.006
38400	163	38343.56	0.147	2604	38402.46	0.006
57600	109	57339.45	0.452	1736	57603.69	0.006
115200	54	115740.74	0.469	868	115207.37	0.006
921600	7	892857.14	3.119	109	917431.19	0.452
1500000	4	1562500	4.167	67	1492537.31	0.498
3000000	2	3125000	4.167	33	3030303.03	1.01
6250000	1	6250000	0	16	6250000	0

NOTE: Carefully select the SCLK frequencies. Even multiples of the bit rate can decrease the error percentage.

Setting the bit clock equal to the sample clock (UART_CLK.EDBO =1) improves the bit rate granularity and the bit clock matches with the bit rate of the communication partner. Use UART_CLK.EDBO mode only when bit rate accuracy is not acceptable in the UART_CLK.EDBO=0 mode.

The UART_CLK.EDBO=1 mode is not intended to increase the speed of operation beyond the electrical limitations of the UART transfer protocol.

Autobaud Detection

At the chip level, the $\overline{\text{UART_RX}}$ pin is typically routed to an alternate capture input (TIMER_ACI[nn]) of a general-purpose timer. When working in width capture mode, the processor uses this general-purpose timer to detect the bit rate applied to the $\overline{\text{UART_RX}}$ pin automatically by an external device. It often uses the capture capabilities of the timer to supervise the bit rate at run time. If the UART communicates with any device supplied

by a weak clock oscillator that drifts over time, the processor can then readjust its UART bit rate dynamically, as required.

Often, the processor uses autobaud detection for initial bit rate negotiations where it is most likely a completer device waiting for the host to send a predefined autobaud character. This situation is common for UART booting. Do not enable the UART_CTL.EN bit while autobaud detection is in-process, to prevent the UART from starting a receive operation with incorrect bit rate matching. Alternatively, set the UART_CTL.LOOP_EN bit to disconnect the UART from its UART_RX pin.

A software routine can detect the pulse widths of serial stream bit cells. The sample base of the timer is synchronous with the UART operation (all derived from the same SCLK). The UART uses pulse widths to calculate the bit rate divider as follows:

Divisor = TIMER_TMR[n]_WID/($16^{(1–EDBO)}$ × Number of captured UART bits)

To increase the number of timer counts and the resolution of the captured signal, do not measure just the pulse width of a single bit. Instead, enlarge the pulse of interest over more bits. Traditionally, a NULL character (ASCII 0x00) is used in autobaud detection, as shown in the ***Autobaud Detection*** figure.

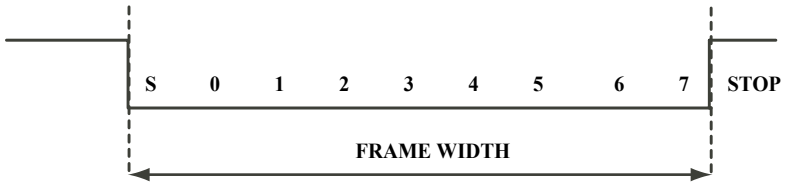


Figure 22-3: Autobaud Detection

Because the example frame encloses 8 data bits and 1 start bit, apply the following formula:

Divisor = TIMER_TMR[n]_WID/($16^{(1–EDBO)}$ × 9)

NOTE: For processor-specific mapping of timer alternate capture inputs to the UARTs of the processor, see "Width Capture (WIDCAP) Mode" in the "*General-Purpose Timer (TIMER)*" chapter.

Real receive signals often have asymmetrical falling and rising edges, and the sampling logic level is not exactly in the middle of the signal voltage range. At higher bit rates, such pulse-width-based autobaud detection does not always return adequate results without extra conditioning of the analog signal. Measure signal periods to work around this issue.

For example, predefine the ASCII character “@” (0x40) as the autobaud detection character and measure the period between two subsequent falling edges. As shown in the ***Autobaud Detection Character 0x40*** figure, measure the period between the falling edge of the start bit and the falling edge after bit 6. Since this period encloses 8 bits, apply the following formula:

Divisor = TIMER_TMR[n]_PER/($16^{(1–EDBO)}$ × 8)

or:

- Divisor = TIMER_TMR[n]_PER>> 7, if UART_CLK.EDBO=0

- Divisor = TIMER_TMR[n]_PER>> 3, if UART_CLK.EDBO=1

The ***Autobaud Detection Character 0x40*** figure shows the ASCII "@" (0x40) detection character.

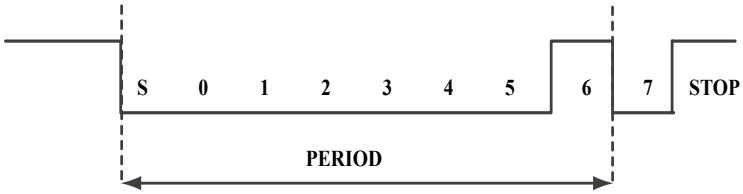


Figure 22-4: Autobaud Detection Character 0x40

UART Debug Features

The UART can automatically calculate and transmit a parity bit. The ***UART Parity*** table summarizes parity behavior assuming 8-bit data words (UART_CTL.WLS=b#11).

Table 22-8: UART Parity

PEN	STP	EPS	Data (hex)	Data (binary, LSB first)	Parity
0	x	x	x	x	None
1	0	0	0x60	0000 0110	1
1	0	0	0x57	1110 1010	0
1	0	1	0x60	0000 0110	0
1	0	1	0x57	1110 1010	1
1	1	0	x	x	1
1	1	1	x	x	0

The two force error bits, UART_CTL.FPE and UART_CTL.FFE, are intended for test purposes. They are useful for debugging software, especially in loopback mode.

The UART can be set to internal loopback mode (UART_CTL.LOOP_EN=1). Loopback mode disconnects the input of the receiver from the receive pin and internally redirects the transmit output to the receiver. The transmit pin remains active and continues to transmit data externally as well. Loopback mode also forces the UART_RTS pin to deassert, disconnects the UART_STAT.CTS bit from the UART_CTS input pin, and connects the internal version of UART_RTS to the UART_STAT.CTS bit.

Additionally, the UART_TX pin can be forced to zero asynchronously using the UART_CTL.SB bit.

UART Operating Modes

The following sections describe the main operating modes of the UART.

- UART Mode
- IrDA SIR Mode

- Multi-Drop Bus Mode

UART Mode

The UART mode follows an asynchronous serial communication protocol with these options:

- 1 start bit
- 5–8 data bits
- Address bit (available in MDB mode only)
- None, even, odd or sticky parity
- One, 1½, or two stop bits (1½ stop bits valid only in 5-bit word length)

The `UART_CTL` register controls the format of received and transmitted character frames. Data is always transmitted and received with the least significant bit (LSB) first.

The ***Bit Stream on a UART TX Pin Transmitting an "S" Character (0x53)*** figure shows a typical physical bit stream measured on a `UART_TX` pin.

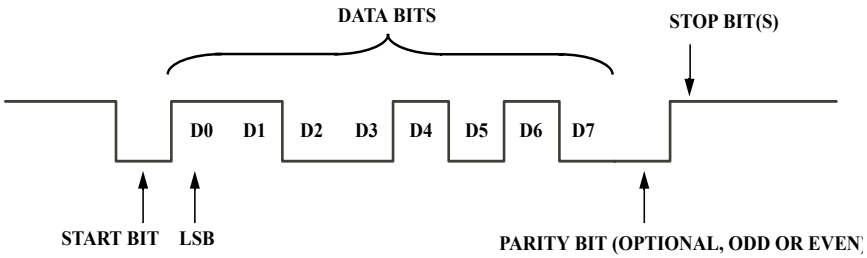


Figure 22-5: Bit Stream on a UART TX Pin Transmitting an "S" Character (0x53)

IrDA SIR Mode

The UART also supports serial data communication by way of infrared signals, according to the recommendations of the Infrared Data Association (IrDA). The physical layer known as IrDA SIR (9.6/115.2 Kbps rate) is based on return-to-zero-inverted (RZI) modulation. The UART does not support pulse position modulation.

Using the 16x data rate clock, RZI modulation is achieved by inverting and modulating the non-return-to-zero (NRZ) code normally transmitted by the UART. On the receive side, the UART uses a 16x clock to determine an IrDA pulse sample window, from which it recovers the RZI modulated NRZ code.

NOTE: The `UART_CLK.EDBO` bit is not valid in IrDA mode. Clear (=0) this bit in this mode.

Multi-Drop Bus Mode

The UART uses a protocol for point-to-point connections as well as in networks where the EIA-485 standard is representative of UART-based bus systems. The EIA-232 standard defines point-to-point connections. In such networks, node addressing is important.

In a multidrop bus (MDB) network, for example, an address bit enhances the UART frame. The address bit is inserted between the data bits and the optional parity bit. To configure the UART for MDB mode, set the mode of operation bits (UART_CTL.MOD [5:4]) to 01.

By convention, the address bit is transmitted low for regular data bytes. When set, it marks special address bytes that require the attention of all nodes on the network.

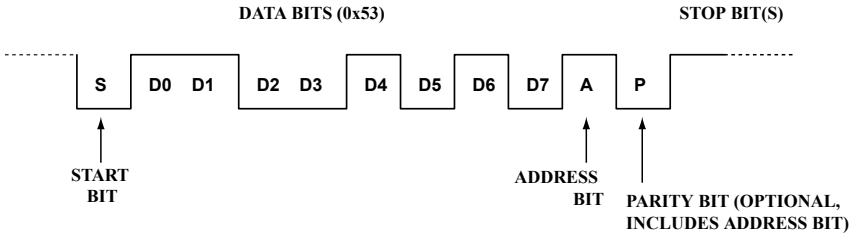


Figure 22-6: UART Frame with Address Bit

All transmit operations are processed through the transmit buffer register (UART_THR), so all DMA data transmissions clear the address bit. If data is written to the transmit address or insert pulse register (UART_TAIP) instead, the same transmit operation is initiated with the only exception that the address bit is sent high.

The UART uses the UART_STAT.ADDR bit of the receiver to signal whether the previously received frame had the address bit set or not. Hardware updates it every time a new frame is received. When the enable address word interrupt bit (UART_IMSK.EAWI) is set, the reception of an address byte triggers a special status interrupt request.

The address sticky bit (UART_STAT.ASTKY) is the sticky version of the UART_STAT.ADDR bit. Hardware sets it whenever the UART_STAT.ADDR bit is set. Software can clear the UART_STAT.ASTKY bit with a W1C operation.

In MDB mode, only address bytes progress to the receive FIFO by default. Data bytes are gated unless the UART_STAT.ASTKY bit is set. The receiver ignores all traffic on the UART bus. This way, the processor can go into low-power mode and interrupt activity does not load the processor every time a frame is transmitted on the UART bus. If, however, an address frame is transmitted, the receiver immediately samples all further traffic. A software routine can analyze the received data, decide whether it was of relevance for the local network node, and W1C the UART_STAT.ASTKY bit if it was not.

Software can overrule of the hardware address frame detection by setting the UART_STAT.ADDR bit and (indirectly) the UART_STAT.ASTKY bit with a W1S operation.

The MDB mode follows an asynchronous serial communication protocol with the following options.

- 1 start bit
- 5–8 data bits
- Address bit
- None, even, odd or sticky parity
- One, 1½, or two stop bits (1½ stop bits are valid only in 5-bit word length)

NOTE: If the address bit and parity bit are both enabled, the parity check and generation includes the address bit in its parity calculation.

UART Data Transfer Modes

The UART can transfer data using both the core and DMA. Receive and transmit paths operate independently except that the bit rate and the frame format are identical for both transfer directions. Transmit and receive channels are both buffered. The UART_THR register buffers the transmit shift register (UART_TSR) and the UART_RBR register buffers the receive shift register (UART_RSR).

UART Mode Transmit Operation (Core)

In core mode, the processor core moves data to and from the UART. A write to the UART_THR register initiates the transmit operation. If no former operation is pending, the UART_THR register passes the data immediately to the UART_TSR register. There, it is shifted out at the bit rate characterized by the UART_CLK register, with start, stop, and parity bits appended as defined by the UART_CTL register.

The UART_THR register and the UART_TSR register can be modeled as a two-stage transmit buffer. The least significant bit (LSB) always transmits first. This bit is bit 0 of the value written to the UART_THR register.

UART Mode LIN Break Command

Some UART-based protocols demand synchronization methods that are not native to standard UART implementations. For example, the Local Interconnect Network (LIN) protocol requires a low-pulse of well-defined transmit length as a prologue to every multi-byte message. Its length must be at least 13 bit-times.

With previous UARTs, there were two options to implement this protocol:

- A null byte is transmitted with a temporarily lowered bit rate, or
- A software counter generates the period, and the asynchronous set break (SB) mechanisms pull the transmit pin low

Since both methods have their disadvantages, the newer UART introduces a new inter-frame gap technique.

The feature is not available in MDB or IrDA operating modes. However, in standard UART mode (bits UART_CTL.MOD [5:4] =00), a write to the UART_TAIP register initiates the transmission of an inter-frame pulse. If the transmit buffer is not empty, the UART first transmits all bytes in the queue. It only initiates with pulse generation after the last stop bit of the last byte has been shifted out.

The value written into the UART_TAIP register defines the nature and the duration of the transmitted pulse. Bits [6:0] control the duration in bit-times and bit [7] controls the value (duration = UART_TAIP[6:0] / UART_CLK[15:0]). If UART_TAIP[7] is set, and an active high pulse is issued, the number of stop bits is extended. If UART_TAIP[7] is cleared, a low pulse is generated. Invert the polarity using the UART_CTL.FCPOL bit. Writing a value of 13 into the UART_TAIP register generates the break command as required by the LIN protocol.

NOTE: If the UART_CTL.TPOLC bit is enabled, an inverted most-significant bit can be transmitted.

NOTE: If another transmission is pending (in the UART_TSR register), the UART_TAIP initiated pulse is queued until after all pending operations have finished and all stop bits are transmitted.

The transmission of break command/inter-frame gap precedes transmission of the number of stop bits as set in the UART_CTL.STB and UART_CTL.STBH bit fields.

The UART receiver can detect break commands through the break indicator (UART_STAT.BI) flag. This flag reports that an entire UART frame has been received in low state. It does not report whether the duration of the received low pulse was exact or at least 13 bit-times as LIN controllers transmit. Typically, the break indicator meets LIN requirements. The processor can use GP timers to determine the pulse width more precisely, if necessary.

Each $\overline{\text{UART_RX}}$ pin is also routed to a GP timer through its alternate capture input (TACI). This functionality is not only useful for bit rate detection (*autobaud*) but also helps to measure the pulse widths precisely on the $\overline{\text{UART_RX}}$ input. Additionally, the GP timers can issue an interrupt request or a fault condition when the received pulse width is shorter than a bit time or longer than the worst-case break condition. The windowed watchdog width mode of the GP timers controls this functionality.

UART Mode Receive Operation (Core)

The receive operation uses the same data format as the transmit configuration except that one valid stop bit is always sufficient. The UART_CTL.STB and UART_CTL.STBH bits have no impact on the receiver.

The UART receiver senses the falling edges of the receive input. When it detects an edge, the receiver starts sampling the input according to settings in the UART_CLK register. The receiver samples the start bit (majority sampling) close to its midpoint. If sampled low, it assumes a valid start condition. Otherwise, it discards the detected falling edge.

After detection of the start bit, the received word is shifted into the UART_RSR register.

After the corresponding stop bit is received, the content of the UART_RSR register is transferred to the 8-deep receive FIFO and is accessible by reading the UART_RBR register.

The receive FIFOs and the UART_RBR register act as a 9-stage receive buffer. If the stop bit of the ninth word is received before software reads the UART_RBR register, an overrun error is reported. Overruns protect data in the UART_RBR register and the receive FIFO from being overwritten by further data until the software clears the UART_STAT.OE bit. However, the data in the UART_RSR register is immediately destroyed as soon as the overrun occurs.

The sampling clock is 16 times faster than the bit clock. The receiver oversamples every bit 16 times and makes a majority-decision based on the middle three samples. This functionality improves immunity against noise and hazards on the line. The receiver disregards spurious pulses of less than two times the sampling clock period.

Normally, the receiver samples every incoming bit at exactly the 7th, 8th, and 9th sample clock. If, however, the UART_CLK.EDBO bit is set to 1, the receiver samples bits approximately at 7/16th, 8/16th, and 9/16th of their period. This configuration achieves better bit rate granularity and accuracy as required at high operation speeds. Hardware design must ensure that the incoming signal is stable between 6/16th and 10/16th of the nominal bit period.

Reception starts when the UART receiver detects a falling edge on the UART_RX input pin. The receiver attempts to see a start bit. The data is shifted into the UART_RSR register. After the ninth sample of the first, the receiver processes the stop bit and copies the received data to the 8-stage receive FIFO. The UART_RSR recovers for further data reception.

The receiver samples data bits close to their midpoint. Because the receiver clock is typically asynchronous to the data rate of the transmitter, the sampling point can drift relative to the center of the data bits. The sampling point is synchronized again with each start bit, so the error accumulates only over the length of a single word. The polarity of received data is selectable, using the UART_CTL.RPOLC bit.

NOTE: The receiver checks for only a single stop bit. After the third sample of the first stop bit has been received (at time 9/16th of the stop bit duration), the receiver immediately acts (status update). It then prepares for new falling edge detection (start detection).

IrDA Transmit Operation

To generate the IrDA pulse transmitted by the UART, the normal NRZ output of the transmitter is first inverted if the UART_CTL.TPOLC bit is configured for active-low operation. In this configuration, a zero is transmitted as a high pulse of 16 UART clock periods and a one is transmitted as a low pulse for 16 UART clock periods. Then, six UART clock periods delay the leading edge of the pulse. Similarly, eight UART clock periods truncate the trailing edge of the pulse. For a 16-cycle UART clock period, this operation results in the final representation of the original zero as a high pulse of only 3/16 clock periods. The ***IrDA Transmit Pulse*** figure shows how the pulse is centered around the middle of the bit time. The final IrDA pulse is fed to the off-chip infrared driver.

This modulation approach ensures a pulse width output from the UART of three cycles high out of every 16 UART clock cycles. As shown in the ***IrDA Transmit Pulse*** figure, the error terms associated with the bit rate generator are small and well within the tolerance of most infrared transceiver specifications.

NOTE: In IrDA mode, writes to the UART_TAIP register are equivalent to writes to the UART_THR register.

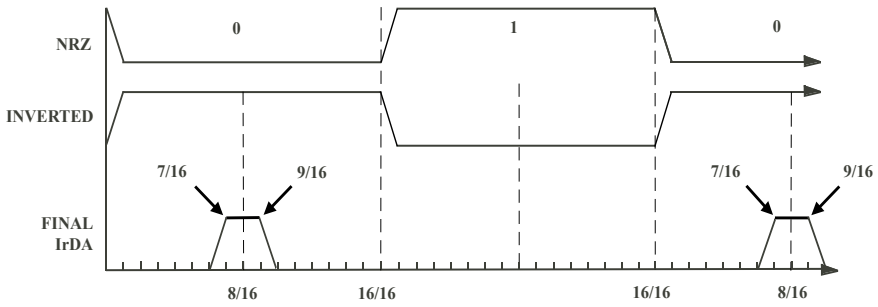


Figure 22-7: IrDA Transmit Pulse

IrDA Receive Operation

The IrDA receiver function is more complex than the transmit function. The receiver must discriminate the IrDA pulse and reject noise. The receiver looks for the IrDA pulse in a narrow window centered around the middle of the expected pulse.

Glitch filtering is accomplished by counting 16 system clocks from the time the receiver detects an initial pulse. If the pulse is absent when the counter expires, the receiver interprets it as a glitch. Otherwise, the receiver interprets it as a zero. This assessment is acceptable because glitches originating from on-chip capacitive cross-coupling typically do not last for more than a fraction of the system clock (SCLK) period. Appropriate shielding avoids sources outside of the chip and not part of the transmitter. The only other source of a glitch is the transmitter itself. The processor relies on the transmitter to perform within specification. If the transmitter violates the specification, unpredictable results can occur. The 4-bit counter adds an extra level of protection at a minimal cost.

NOTE: Because SCLK can change across systems, the longest glitch tolerated is inversely proportional to the SCLK frequency.

A counter that is clocked at the 16x bit-time sample clock determines the receive sampling window. The sampling window is resynchronized with each start bit by centering the sampling window around the start bit.

The polarity of receive data is selectable, using the `UART_CTL.RPOLC` bit. The ***IrDA Receiver Pulse Detection*** figure provides examples of each polarity type.

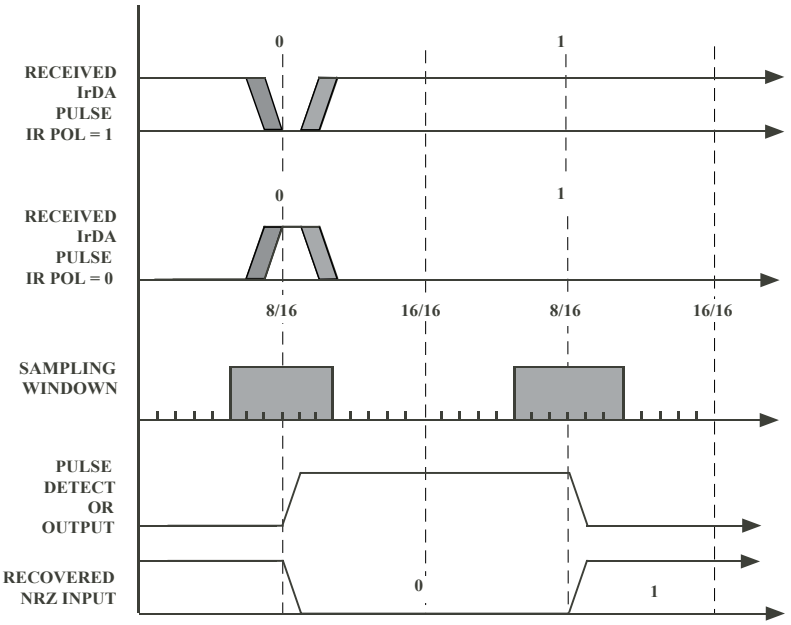


Figure 22-8: IrDA Receiver Pulse Detection

MDB Transmit Operation

In MDB mode, receive and transmit paths operate independently from each other, except for sharing bit rate and frame formats for both transfer directions.

Transmit operation is initiated by writing the `UART_THR` or `UART_TAIP` registers. A write to the `UART_THR` register transmits the written word with the appending address bit set low. A write to the `UART_TAIP` register transmits the written word with the appended address bit set high. The data is moved into the `UART_TSR` register, where it is shifted out at the bit rate programmed by the `UART_CLK` register, with start, stop, address, and parity bits appended, as required.

If DMA is enabled, the DMA engine always writes the data into the UART_THR register, and the written word is transmitted with the appending address bit set low.

The polarity of transmit data is selectable, using the UART_CTL.TPOLC bit.

MDB Receive Operation

Receive operations use the same data format as the transmit configuration, except that the number of stop bits is always assumed to be 1. After detection of the start bit, the received word is shifted into the UART_RSR register at the programmed bit.

Normally, the receiver samples every incoming bit at exactly the 7th, 8th, and 9th sample clock. If, however, the UART_CLK.EDBO bit is set, the receiver samples the bits approximately at 7/16th, 8/16th, and 9/16th of their period. This configuration achieves better bit rate granularity and accuracy needed at high operation speeds. Hardware design must ensure that the incoming signal is stable between 6/16th and 10/16th of the nominal bit period.

After the appropriate number of bits (including address, parity, and stop bits) is received, the UART_RSR register is transferred to the receive FIFO and accessible through the UART_RBR register.

The polarity of receive data is selectable, using the UART_CTL.RPOLC bit.

DMA Mode

In DMA mode, separate receive and transmit DMA channels move data between the UART and memory. The software does not have to move data; it just must set up the appropriate transfers either through the descriptor mechanism or through autobuffer mode.

DMA channels provide a 4-deep FIFO, resulting in total buffer capabilities of 6 words at the transmit side and 9 words at the receive side. In DMA mode, the bus activity and arbitration mechanism determine the latency. The processor loading and interrupt priorities do not determine the latency.

To enable UART DMA, first set up the system DMA control registers. Then, enable the UART_IMSK.ERBFI or UART_IMSK.ETBEI interrupts. This sequence is necessary because these interrupt request lines double as DMA request lines. With DMA enabled, once these requests are received, the DMA control unit generates a direct memory access. If DMA is not enabled, the UART interrupt is passed on to the system interrupt handling unit. The status interrupt for the UART goes directly to the system event controller (SEC), bypassing the DMA unit completely.

For transmit DMA, programs must set the DMA_CFG.SYNC bit. With this bit set, interrupt generation is delayed until the entire DMA FIFO is drained to the UART module. The UART transmit DMA interrupt service routine can disable the DMA or to clear the UART_IMSK.ETBEI control bit only when the DMA_CFG.SYNC bit is set. Otherwise, up to four data bytes can be lost.

When the UART_IMSK.ETBEI bit is set, an initial transmit DMA request is issued immediately. The program then clears the UART_IMSK.ETBEI bit through the DMA service routine.

In DMA transmit mode, the UART_IMSK.ETBEI bit enables the peripheral request to the DMA FIFO. The DMA_CFG.EN bit enables the strobe on the memory side. If the DMA count is less than the DMA FIFO depth,

which is 4, then the DMA interrupt can be requested before the UART_IMSK.ETBEI bit is set. If this behavior is unwanted, set the DMA_CFG.SYNC bit.

Regardless of the DMA_CFG.SYNC setting, the DMA stream has not left the UART transmitter completely at the time the interrupt request is generated. Transmission can abort in the middle of the stream, causing data loss, when the UART clock was disabled without extra synchronization with the UART_STAT.TEMT bit.

The UART provides functionality to avoid resource-consuming polling of the UART_STAT.TEMT bit. The UART_IMSK_SET.EDTPTI bit enables the UART_STAT.TEMT bit to trigger a DMA interrupt. To delay the DMA completion interrupt until the last data word of a STOP DMA has left the UART, keep the DMA_CFG.INT bit cleared and set the UART_IMSK_SET.EDTPTI bit instead. Then, the normal DMA completion interrupt is suppressed. Later, the UART_STAT.TEMT event triggers a DMA interrupt after the last word of the DMA has left the UART transmit buffers. If DMA_CFG.INT and UART_IMSK.EDTPTI are set, when finishing STOP mode, the DMA requests two interrupts.

The DMA of the UART module supports 8-bit and 16-bit operation, but not 32-bit operation. It does not support sign-extension.

Mixing DMA and Core Modes

Switching from DMA mode to core operation dynamically requires some consideration, especially for transmit operations. By default, the interrupt timing of the DMA is synchronized with the memory side of the DMA FIFOs. Normally, the transmit DMA completion interrupt is generated after the last byte is copied from the memory into the DMA FIFO. The transmit DMA interrupt service routine is not yet permitted to disable the DMA_CFG.EN bit. The interrupt is requested when the DMA_STAT.IRQDONE bit is set. The DMA_STAT.RUN bit, however, remains set until the data has completely left the transmit DMA FIFO.

When planning to switch from a DMA to core mode, set the DMA_CFG.SYNC bit in the word of the last descriptor or work unit before handing over control. Then, after the interrupt request occurs, software can write new data into the UART_THR register as soon as the UART_STAT.THRE bit permits. If the DMA_CFG.SYNC bit cannot be set, software can poll the DMA_STAT.RUN bit instead. Alternatively, using the UART_IMSK.EDTPTI bit can avoid expensive status bit polling.

When switching from core to DMA operation, ensure that the first DMA request is issued properly. If the DMA is enabled while the UART is still transmitting, no precaution is required. If, however, the DMA is enabled after the UART_STAT.TEMT bit is high, pulse the UART_IMSK.ETBEI bit to initiate DMA transmission.

Setting Up Hardware Flow Control

The following steps show how to set up UART hardware flow control:

1. Configure automatic or manual hardware flow control for the receiver through the UART_CTL.ARTS bit, or the transmitter through the UART_CTL.ACTS bit.
2. Configure $\overline{\text{UART_CTS}}$ and $\overline{\text{UART_RTS}}$ polarity through the UART_CTL.FCPOL bit.

On reset, when the UART is not yet enabled and the port multiplexing has not been programmed, the $\overline{\text{UART_RTS}}$ pin is not driven. Some applications require a resistor to pull the $\overline{\text{UART_RTS}}$ signal to either state during reset.

UART Event Control

Status flags in the UART_STAT register are available to signal data reception, parity, and error conditions, if necessary.

DMA and Interrupt Multiplexing

See the *Direct Memory Access (DMA)* chapter for information on DMA multiplexing. Several interrupts and DMA channels in the UART can be multiplexed.

NOTE: To operate in interrupt mode without using DMA channels, set the UART_IMSK.ELSI bit. This configuration redirects receive and transmit requests to the status interrupt output. The status interrupt goes directly to the SEC without going through the DMA controller.

Interrupt Masks

Each UART features a set of interrupt mask registers: UART_IMSK, UART_IMSK_SET, and UART_IMSK_CLR. The UART_IMSK register supports read/write operations. Writing ones to the UART_IMSK_SET register enables interrupts, writing ones to the UART_IMSK_CLR register disables them. Reads from either register return the enabled bits. This way, different interrupt service routines can control transmit, receive, and status interrupt requests independently and easily.

The UART module uses the UART_IMSK registers to enable requests for system handling of empty or full states of data registers. Unless polling is used as a means of action, the UART_IMSK.ERBFI and UART_IMSK.ETBEI bits in this register are normally set.

Each UART module has three interrupt outputs. It uses one for transmission, one for reception, and one for reporting status events. The UART module routes transmit and receive requests through the DMA controller. The status request goes directly to the system event controller (SEC).

If the associated DMA channel is enabled, the request functions as a DMA request. If the DMA channel is disabled, it simply forwards the request to the SEC. A DMA channel must be associated with the UART module to enable transmit and receive interrupts. Otherwise, transmit and receive requests cannot be forwarded.

NOTE: To operate in interrupt mode without using DMA channels, set the UART_IMSK.ELSI bit. This configuration redirects receive and transmit requests to the status interrupt request output. The status interrupt goes directly to the SEC without going through the DMA controller.

Interrupt Servicing

Interrupt service routines (ISRs) perform UART writes and reads. Separate interrupt lines are provided for transmit, receive, and status. The UART_IMSK register group enables the independent interrupts individually. To enable UART transmit interrupts, set the UART_CTL.EN bit.

The ISRs evaluate the status bits in the UART_STAT register to determine the signaling interrupt source. The system event controller for the processor assigns and unmasks interrupts. The ISRs must clear the interrupt latches explicitly. To reduce interrupt frequency on the receive side in core mode, use the UART_IMSK.ERFCI status

interrupt as an alternative to the regular UART_IMSK.ERBFI receive interrupt. Hardware must ensure that at least two (if UART_CTL.RFIT=0) or four (if UART_CTL.RFIT=1) words are available in the receive buffer by the time the interrupt is requested.

Transmit Interrupts

The UART module uses the UART_IMSK_SET.ETBEI bit to enable transmit interrupt requests.

The UART_THR and UART_TAIP registers are the same physical register, and both affect the signaling of the UART_STAT.TEMT, UART_STAT.TFI, and UART_STAT.THRE bits similarly.

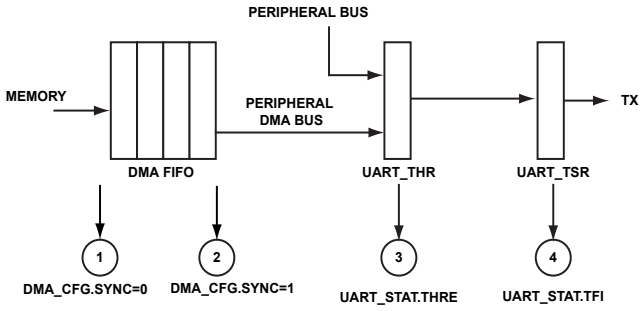


Figure 22-9: Transmit Interrupts

The UART module asserts the transmit request along with the UART_STAT.THRE bit, indicating that the transmit buffer is ready for new data. The UART_STAT.THRE bit resets to 1. When the UART_IMSK_SET.ETBEI bit is set, the UART module immediately issues an interrupt or DMA request. This way, no special handling of the first character is required when transmission of a string is initiated. Set the UART_IMSK_SET.ETBEI bit and let the interrupt service routine load the first character from memory and write it to the UART_THR register in the normal manner. ISRs can clear the UART_IMSK.ETBEI bit through the UART_IMSK_CLR register when the string transmission has completed.

Hardware clears the UART_STAT.THRE bit when new data is written to the UART_THR register. These write operations also clear the transmit interrupt request. However, they also initiate further transmission. If continued transmission is undesirable, the UART module can alternatively clear the transmit request through the UART_IMSK_CLR.ETBEI bit register. Transfers of data from the UART_THR register to the UART_TSR register reset this status flag in the UART_STAT register.

ISRs can interrogate the UART_STAT.TEMT bit to discover any ongoing transmission. The sticky counterpart of the UART_STAT.TEMT bit, UART_STAT.TFI, indicates when the transmit buffer has drained and can trigger a status interrupt. When data is pending in either one of these registers, the UART_STAT.TEMT flag is low. As soon as all data has left the UART_TSR register, the UART_STAT.TEMT bit goes high again and indicates that all pending transmit operations (including stop bits) have finished. Then, it is safe to disable the UART_CTL.EN bit or to three-state off-chip line drivers. Then, the UART module can generate an interrupt either through the status interrupt channel when the UART_IMSK.ETFI bit is set, or through the DMA controller when enabled by the UART_IMSK.EDTPTI bit.

When enabled by the UART_IMSK.ETBEI bit, the UART_STAT.THRE flag requests data along the peripheral command lines to the DMA controller (referred to as TXREQ). This signal is routed through the DMA controller. If the associated DMA channel is enabled, the TXREQ signal functions as a DMA request, otherwise the DMA controller simply forwards it to the SEC. Alternatively the UART_IMSK.ETXS bit can redirect the transmit interrupts to the UART status interrupt.

With interrupts disabled, the UART module can poll the status flags to determine when data is ready to move. Because polling is processor intensive; it is not typically used in real-time signal processing environments. Since read operations from UART_STAT registers have no side effects, different software threads can interrogate these registers without mutual impacts.

Polling the SEC_SSTAT[n] register without enabling the interrupts by the SEC_CCTL[n] register is an alternate method of operation to consider. Software can write up to two words into the UART_THR register before enabling the UART clock. As soon as the UART_CTL.EN bit is set, the UART module sends those two words.

Receive Interrupts

The UART module uses the UART_IMSK_SET.ERBFI bit to enable receive interrupt requests. If set, the UART_STAT.DR flag requests an interrupt on the dedicated RXREQ output, indicating that new data is available in the UART_RBR register. This signal is routed through the DMA controller. If the associated DMA channel is enabled, the RXREQ signal functions as a DMA request; otherwise the DMA controller simply forwards it to the SEC. Alternatively, if no DMA channel is assigned to the UART, the UART_IMSK.ERXS bit can redirect the receive interrupts to the UART status interrupt. When software reads the UART_RBR register, hardware clears the UART_STAT.DR bit again, which, in turn, clears the receive interrupt request.

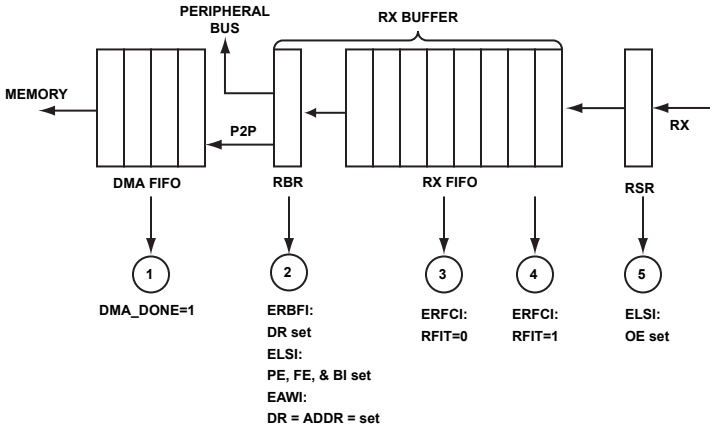


Figure 22-10: Receive Interrupts

Hardware updates the following:

- UART_STAT.DR bits
- UART_STAT.ADDR bits
- UART_STAT.ASTKY bits
- UART_STAT.PE bits

- UART_STAT.FE bits
- UART_STAT.BI bits
- UART_RBR register

The UART_STAT.OE bit is updated as soon as an overflow condition occurs (for example when a stop bit for a frame is received and the receive FIFO is full). When software does not read the UART_RBR register in time, the received data is protected from being overwritten by new data until software clears the UART_STAT.OE bit. Only the content of the UART_RSR register can be overwritten in the overrun case.

The UART module uses the UART_STAT.RFCS bit to monitor the state of the 8-deep receive FIFO. It uses the UART_CTL.RFIT bit to control the behavior of the buffer. If UART_CTL.RFIT is zero, the UART_STAT.RFCS bit is set when the receive buffer holds four or more words. If UART_CTL.RFIT is set, the UART_STAT.RFCS bit is set when the receive buffer holds seven or more words. Hardware clears the UART_STAT.RFCS bit when a core or DMA reads the UART_RBR register and when the buffer is flushed below the level of four (UART_CTL.RFIT=0) or seven (UART_CTL.RFIT=1). If the associated interrupt bit UART_IMSK.ERFCI is enabled, a status interrupt request is reported when the UART_STAT.RFCS bit is set.

If errors are detected during reception, an interrupt can be requested from the status interrupt output. This status interrupt request goes directly to the SEC. The bit enables status interrupt requests.

The controller detects the following error conditions, shown with their associated bits in the UART_STAT register.

- Overrun error (UART_STAT.OE bit)
- Parity error (UART_STAT.PE bit)
- Framing error or invalid stop bit (UART_STAT.FE bit)
- Break indicator (UART_STAT.BI bit)

Status Interrupts

The UART module uses status interrupt channels for the following purposes:

- Line status interrupt requests
- Flow control interrupt requests
- Receive FIFO threshold interrupt requests
- Transmission finished interrupt request

The UART module uses the UART_IMSK.ELSI bit to enable the line status interrupts. If set, the status interrupt request is asserted with one of the UART_STAT.BI, UART_STAT.FE, UART_STAT.PE, or UART_STAT.OE receive errors bits. A W1C operation to the UART_STAT register clears the error bits. Once all error conditions are cleared, the interrupt request deasserts.

The UART module uses the UART_IMSK_SET.ERFCI bit to enable the receive FIFO count interrupt. If set, a status interrupt request is generated when the UART_STAT.RFCS is active. The UART_STAT.RFCS bit

indicates a receive buffer threshold level. If the UART_CTL.RFIT bit is cleared, software can safely read two words out of the UART_RBR register by the time the UART_STAT.RFCS interrupt occurs.

If the UART_CTL.RFIT bit is set, software can safely read four words. The interrupt request and the UART_STAT.RFCS bit are cleared when the UART_RBR is read enough of times, so that the receive buffer drains below the threshold of two (UART_CTL.RFIT=0) or four (UART_CTL.RFIT=1). Because in DMA mode a status service routine may not be permitted to read UART_RBR, this interrupt is only recommended in core mode. In DMA mode, use this functionality for error recovery only.

The UART module uses the UART_IMSK_SET.EDSSI bit to enable the flow control interrupts. If active, a status interrupt is generated when the sticky UART_STAT.SCTS bit register is set, indicating that the UART_CTS input of the transmitter been reasserted. A W1C operation to the UART_STAT.SCTS bit clears the interrupt request.

The UART module uses the UART_IMSK_SET.ETFI bit to enable the transmission finished interrupt. If active, a status interrupt request is asserted when the UART_STAT.TFI bit is set. The UART_STAT.TFI is the sticky version of the UART_STAT.TEMT bit, indicating that a byte that started transmission has finished. A W1C operation to the UART_STAT.TFI bit clears the interrupt request.

Multi-Drop Bus Events

Several status bits and interrupt features in the UART_STAT and UART_IMSK registers facilitate efficient data handling in multi-drop bus mode. These features include the address (UART_STAT.ADDR) bit, the address sticky (UART_STAT.ASTKY) bit and the enable address word interrupt (UART_IMSK.EAWI). One of the key features of the multi-drop bus protocol is its address bit. The address bit signifies to the completer that the requester is transmitting an address word (all read it) or a data word (only the addressed target reads its). The UART hardware provides an efficient method of handling the situation described with the use of UART_STAT.ASTKY bit.

NOTE: The UART module uses the UART_STAT.ASTKY bit in multi-drop bus mode to indicate when an address operation for a peripheral is occurring. The UART_STAT.ASTKY bit is a sticky version of the UART_STAT.ADDR bit. Hardware sets the bit whenever the UART_STAT.ADDR bit is set. Only software clears it with a W1C operation. With the UART_STAT.ASTKY bit set, words are received regardless of the mode bit or address bit setting. With the UART_STAT.ASTKY bit cleared, only address words (mode bit =1) are received and words with mode bit =0 are ignored in MDB mode. This bit does not affect reception in non-MDB modes. (Words with mode bit =0 are not moved from the UART_RSR register to the receive FIFO.)

UART Programming Model

The following sections provide basic procedures for configuring various UART operations.

Detecting Autobaud

To detect Autobaud:

1. Ensure that the timer is disabled.

2. Configure the following bits: UART_CTL.MOD =00, UART_CTL.LOOP_EN =1, UART_CTL.WLS =11 (8-bit data), and UART_CTL.EN =1
3. Configure the following bits: TIMER_TMR[n]_CFG.TMODE =1101, TIMER_TMR[n]_CFG.OUTDIS =1, TIMER_TMR[n]_CFG.IRQMODE =10 and enable the timer.
4. Send test data through the host device and wait for the timer interrupt and disable the timer.

   The bit rate can be derived from the timer period register value according to the formula provided in the Autobaud Detection section.

Using Common Initialization Steps

When using the core or the DMA to execute transfers, the following steps are common to all UART modes.

1. All UART signals are multiplexed and compete with other functions at pin level. First, program the port registers according to the guidelines in the PORT chapter.
2. Program the UART_CLK register. Refer to Bit Rate Generation topic.
3. Program the UART_CTL register and enable the UART clock.

Using Core Transfers

Write data into the UART_THR register, when the UART_STAT.THRE bit is set, to initiate a core transmit operation. If the UART_STAT.DR bit is set, received data can be read from the UART_RBR register.

Using DMA Transfers

1. Make sure that the UART_IMSK.ETBEI or the UART_IMSK.ERBFI bits are cleared before configuring the DMA.
2. Configure the dedicated DMA channel.
3. Set the UART_IMSK.ETBEI or UART_IMSK.ERBFI bits to start the transfer.

Using Interrupts

Each UART features three interrupt signal outputs.

1. Enable individual interrupts in the system event controller (SEC)
2. Register IRQ handlers.
3. Use the interrupts mask registers to enable specific IRQ events.

Setting Up Hardware Flow Control

1. Configure automatic or manual hardware flow control for the receiver through the UART_CTL.ARTS bit, or the transmitter through the UART_CTL.ACTS bit.
2. Configure $\overline{\text{UART_CTS}}$ and $\overline{\text{UART_RTS}}$ polarity through the UART_CTL.FCPOL bit.

ADSP-2184x UART Register Descriptions

UART (UART) contains the following registers.

Table 22-9: ADSP-2184x UART Register List

Name	Description
UART_CLK	Clock Rate Register
UART_CTL	Control Register
UART_IMSK	Interrupt Mask Register
UART_IMSK_CLR	Interrupt Mask Clear Register
UART_IMSK_SET	Interrupt Mask Set Register
UART_RBR	Receive Buffer Register
UART_RSR	Receive Shift Register
UART_RXCNT	Receive Counter Register
UART_SCR	Scratch Register
UART_STAT	Status Register
UART_TAIP	Transmit Address/Insert Pulse Register
UART_THR	Transmit Hold Register
UART_TSR	Transmit Shift Register
UART_TXCNT	Transmit Counter Register

Clock Rate Register

The UART_CLK register divides the system clock (SCLK) down to the bit clock.

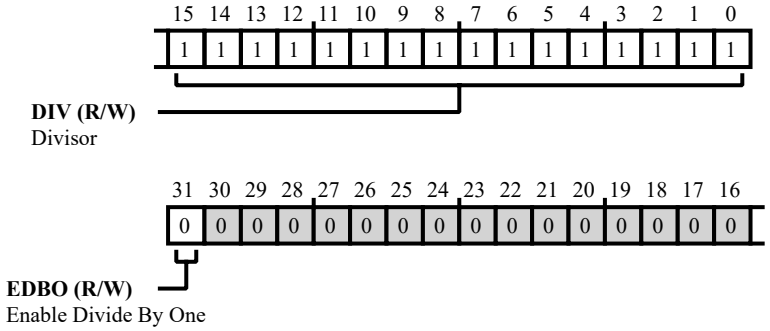


Figure 22-11: UART_CLK Register Diagram

Table 22-10: UART_CLK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EDBO	Enable Divide By One. The UART_CLK.EDBO bit enables the bypassing of the divide-by-16 prescaler in bit clock generation. This functionality improves bit rate granularity, especially at high bit rates. Do not set this bit in IrDA mode. Note: Properly select the SCLK frequencies. Even multiples of bit rate decrease the error percentage. Setting the bit clock equal to the sample clock (UART_CLK.EDBO =1) improves the bit rate granularity and the bit clock matches with the bit rate of the communication partner. The disadvantage is that the power dissipation is higher and sample points are not always accurate. Therefore, use UART_CLK.EDBO=1 mode only when bit rate accuracy is not acceptable in the UART_CLK.EDBO=0 mode. The UART_CLK.EDBO=1 mode is not intended to increase the speed of operation beyond the electrical limitations of the UART transfer protocol.	
		0	Bit clock prescaler = 16
		1	Bit clock prescaler = 1
15:0 (R/W)	DIV	Divisor. The UART_CLK.DIV provides the divisor for the UART's clock bit rate calculation. The bit rate is defined by: Bit Rate = SCLK / ($16^{(1-EDBo)}$ x UART_CLK.DIV)	

Control Register

The UART_CTL register provides enable and disable control for UART and IrDA mode of operation. It also provides UART line control, permitting selection of the format of received and transmitted character frames. Modem feature control also is available from this register, including partial modem functionality to allow for hardware flow control and loopback mode.

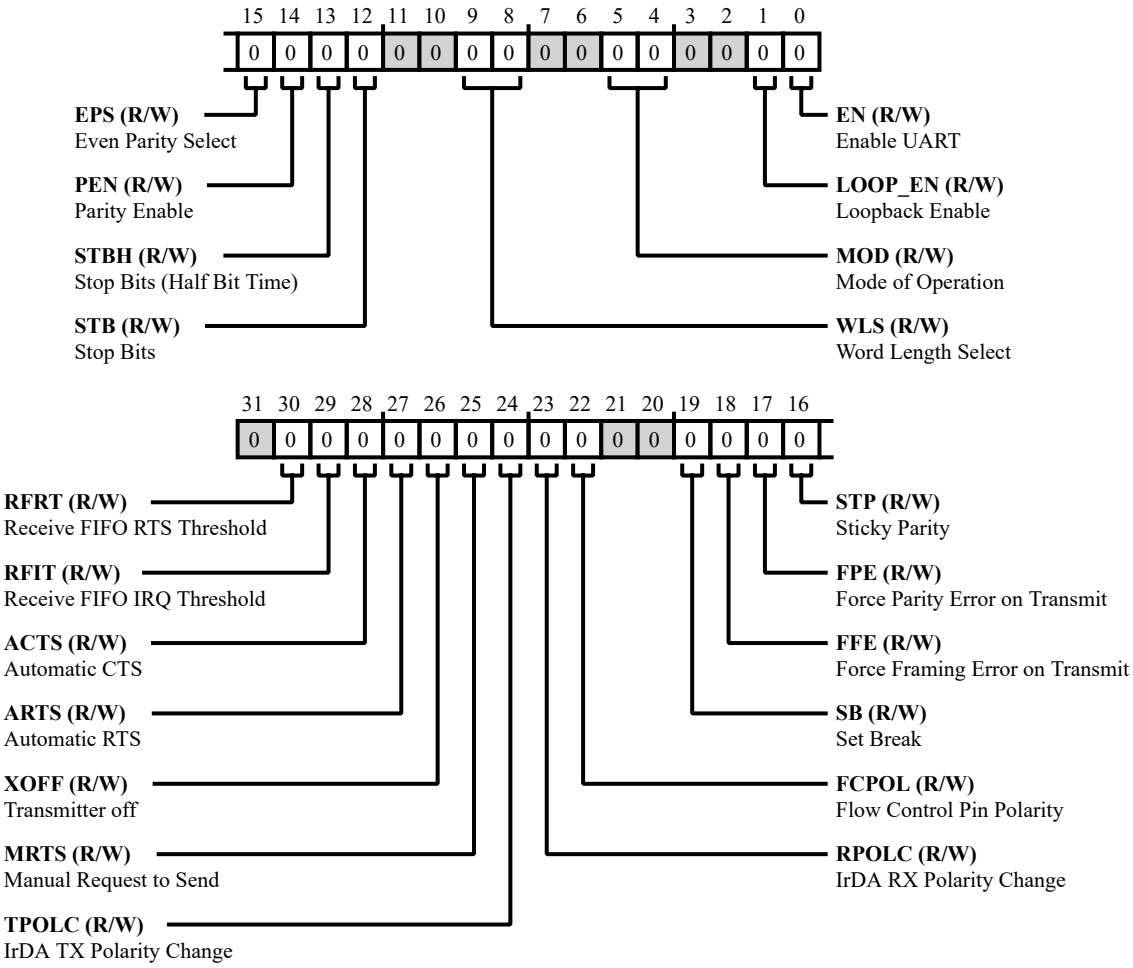


Figure 22-12: UART_CTL Register Diagram

Table 22-11: UART_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	RFRT	Receive FIFO RTS Threshold. The UART_CTL.RFRT bit controls UART_RTS pin assertion and deassertion timing. This bit is ignored if UART_CTL.ARTS is cleared. If set, the UART_RTS pin is deasserted when the receive buffer already holds seven words and an eighth start bit is detected. It is reasserted when the FIFO contains seven words or less. If cleared, the UART_RTS pin is deasserted when the RX buffer already holds four words and a fifth start bit is detected. The UART_RTS pin is reasserted when the RX buffer contains no more than 4 words.	
		0	Deassert RTS if RX FIFO word count > 4; assert if <= 4
		1	Deassert RTS if RX FIFO word count > 7; assert if <= 7
29 (R/W)	RFIT	Receive FIFO IRQ Threshold. The UART_CTL.RFIT bit controls the timing of the UART_STAT.RFCS bit. If UART_CTL.RFIT is cleared, the receive threshold is two. If UART_CTL.RFIT is set, the threshold is four words in the receive buffer.	
		0	Set RFCS=1 if RX FIFO count >= 4
		1	Set RFCS=1 if RX FIFO count >= 7
28 (R/W)	ACTS	Automatic CTS. The UART_CTL.ACTS bit must be set to enable the UART_CTS input pin for UART_TX handshaking. If enabled, the UART_STAT.CTS bit holds the value (if UART_CTL.FCPOL is set) or complement value (if UART_CTL.FCPOL is cleared) of the UART_CTS input pin. The UART_STAT.CTS bit can be used to determine whether the external device is ready to receive data (if UART_STAT.CTS is set) or whether it is busy (if UART_STAT.CTS is cleared). If UART_CTL.ACTS is cleared, the UART_TX handshaking protocol is disabled, and the UART_TX line transmits data whenever there is data to send, regardless of the value of UART_CTS. Software can pause ongoing transmission by setting the UART_CTL.XOFF bit.	
		0	Disable TX handshaking protocol
		1	Enable TX handshaking protocol
27 (R/W)	ARTS	Automatic RTS. The UART_CTL.ARTS bit must be set to enable the UART_RTS input pin for UART_TX handshaking. If set, the hardware guarantees a minimal UART_RTS pin deassertion pulse width of at least the number of data bits defined by the UART_CTL.WLS bit field. If cleared, the UART_RTS pin is not generated automatically by hardware. The UART_RTS pin can still be manually controlled by the UART_CTL.MRTS bit, and software is responsible for UART_RTS pulse width control (if needed).	
		0	Disable RX handshaking protocol.
		1	Enable RX handshaking protocol.

Table 22-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	XOFF	Transmitter off. The UART_CTL.XOFF bit (if set) turns off transmission (XOFF) by preventing the content of THR from being continued to TSR. When set, this bit turns on transmission (XON). The state of the UART_CTL.XOFF bit is ignored if the UART_CTL.ACTS bit is set.	
		0	Transmission ON, if ACTS=0
		1	Transmission OFF, if ACTS=0
25 (R/W)	MRTS	Manual Request to Send. The UART_CTL.MRTS bit controls the state of the UART_RTS output pin when the UART_CTL.ARTS bit is cleared. When UART_CTL.MRTS is cleared, the UART deasserts the UART_RTS pin, signaling to the external device that the UART is not ready to receive. When UART_CTL.MRTS is set, the UART asserts the UART_RTS pin, signaling to the external device that the UART is ready to receive.	
		0	Deassert RTS pin when ARTS=0
		1	Assert RTS pin when ARTS=0
24 (R/W)	TPOLC	IrDA TX Polarity Change. The UART_CTL.TPOLC bit selects the active low or high polarity for IrDA communications. This bit is effective only in IrDA mode. If set, in IrDA mode, the UART_TX pin idles high. In UART or MDB mode, it is inverted-NRZ. If cleared, in IrDA mode, the UART_TX pin idles low. In UART or MDB mode, it is NRZ.	
		0	Active-low TX polarity setting
		1	Active-high TX polarity setting
23 (R/W)	RPOLC	IrDA RX Polarity Change. The UART_CTL.RPOLC bit selects the active low or high polarity for IrDA communications. This bit is effective only in IrDA mode. If set, in IrDA mode, the UART_RX pin idles high. In UART or MDB mode, it is inverted-NRZ. If cleared, in IrDA mode, the UART_RX pin idles low. In UART or MDB mode, it is NRZ.	
		0	Active-low RX polarity setting
		1	Active-high RX polarity setting

Table 22-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	FCPOL	Flow Control Pin Polarity. The UART_CTL.FCPOL bit selects the polarities of the UART_CTS and UART_RTS pins. When the UART_CTL.FCPOL bit is cleared, the UART_RTS and UART_CTS pins are active low, and the UART is halted when the UART_RTS and UART_CTS pin state is high. When UART_CTL.FCPOL bit is set, the UART_RTS and UART_CTS pins are active high, and the UART is halted when the UART_RTS and UART_CTS pin state is low.	
		0	Active low CTS/RTS
		1	Active high CTS/RTS
19 (R/W)	SB	Set Break. If set, the UART_CTL.SB bit forces the UART_TX pin to low asynchronously, regardless of whether or not data is currently transmitted. This bit functions even when the UART clock is disabled. Because the UART_TX pin normally drives high, it can be used as a flag output pin, if the UART is not used. (For example, if UART_CTL.TPOLC is cleared, drive UART_TX pin low; or if UART_CTL.TPOLC is set, drive UART_TX pin high.)	
		0	No force
		1	Force TX pin to 0
18 (R/W)	FFE	Force Framing Error on Transmit. The UART_CTL.FFE bit is intended for test purposes. This bit is useful for debugging software, especially in loopback mode.	
		0	Normal operation
		1	Force error
17 (R/W)	FPE	Force Parity Error on Transmit. The UART_CTL.FPE bit is intended for test purposes. This bit is useful for debugging software, especially in loopback mode.	
		0	Normal operation
		1	Force parity error

Table 22-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	STP	Sticky Parity. The UART_CTL.STP bit controls whether the parity is generated by hardware based on the data bits or whether it is set to a fixed value. If this bit is cleared, the hardware calculates the parity bit value based on the data bits. Then, the EPS bit determines whether odd or even parity mode is chosen. If this bit is set, odd parity is used. That means that the total count of logical-1 data bits including the parity bit must be an odd value. Even parity is chosen by UART_CTL.STP cleared and UART_CTL.EPS set. Then, the count of logical-1 bits must be a even value. If the UART_CTL.STP bit is set, hardware parity calculation is disabled. In this case, the sent and received parity equals the inverted UART_CTL.EPS bit.	
		0	No forced parity
		1	Force (Stick) parity to defined value (if PEN=1)
15 (R/W)	EPS	Even Parity Select.	
		0	Odd parity
		1	Even parity
14 (R/W)	PEN	Parity Enable. The UART_CTL.PEN bit enables parity transmission and parity check. The UART_CTL.PEN bit inserts one additional bit between the most significant data bit and the first stop bit. The polarity of this so-called parity bit depends on data and the UART_CTL.STP and UART_CTL.EPS control bits. Both transmitter and receiver calculate the parity value. The receiver compares the received parity bit with the expected value and issues a parity error if they do not match. If the UART_CTL.PEN bit is cleared, the UART_CTL.STP and the UART_CTL.EPS bits are ignored.	
		0	Disable
		1	Enable parity transmit and check
13 (R/W)	STBH	Stop Bits (Half Bit Time).	
		0	0 half-bit-time stop bit
		1	1 half-bit-time stop bit
12 (R/W)	STB	Stop Bits. The UART_CTL.STB bit controls how many stop bits are appended to transmitted data.	
		0	1 stop bit
		1	2 stop bits

Table 22-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	WLS	Word Length Select. The UART_CTL.WLS field determines whether the transmitted and received UART word consists of 5, 6, 7, or 8 data bits.	
		0	5-bit word
		1	6-bit word
		2	7-bit word
		3	8-bit word
5:4 (R/W)	MOD	Mode of Operation. The UART_CTL.MOD selects the UART operation mode (UMOD).	
		0	UART mode
		1	MDB mode
		2	IrDA SIR mode
1 (R/W)	LOOP_EN	Loopback Enable. The UART_CTL.LOOP_EN bit enables UART loopback mode. When set, this bit disconnects the input of the receiver from the UART_RX pin, and internally redirects the transmit output to the receiver. The UART_TX pin remains active and continues to transmit data externally as well. Loopback mode also forces the UART_RTS pin to its deassertive state, disconnects the UART_CTS bit from the UART_CTS input pin, and directly connects the UART_CTL.MRTS bit to the UART_STAT.CTS bit. In loopback mode, setting the UART_CTL.MRTS bit sets the UART_STAT.CTS bit and enables the transmitter of the UART. Clearing the UART_CTL.MRTS bit clears the UART_STAT.CTS bit and disables the transmitter of the UART.	
		0	Disable
		1	Enable
0 (R/W)	EN	Enable UART. The UART_CTL.EN enables the UART clocks. This bit also resets the state machine and control registers when cleared. Using this bit to disable the UART -- when not used -- reduces power consumption.	
		0	Disable
		1	Enable

Interrupt Mask Register

The UART_IMSK register indicates the interrupt mask status (unmasked, if set, or masked, if cleared) of the UART status interrupt requests. This register is not a data register. Instead, it is controlled by the UART_IMSK_SET and UART_IMSK_CLR register pair. Writing ones to the UART_IMSK_SET register enables (unmasks) interrupt requests, and writing ones to the UART_IMSK_CLR register disables (masks) them. Reads from either register return the enabled bits.

The UART_IMSK register is used to enable requests for system handling of empty or full states of UART data registers. Unless polling is used as a means of action, the UART_IMSK.ERBFI and UART_IMSK.ETBEI bits are normally set. Setting this register without enabling system DMA causes the UART to notify the processor of the data inventory state using interrupts. For proper operation in this mode, system interrupts must be enabled, and appropriate interrupt handling routines must be present.

Each UART features three separate interrupt channels to handle the data transmit, data receive, and line status events independently, regardless of whether DMA is enabled or not. If no DMA channels are assigned to the UART, set the UART_IMSK.ELSI bit to reroute the transmit and receive interrupts to the status interrupt request output.

With system DMA enabled, the UART uses DMA to transfer data to or from the processor. Dedicated DMA channels are available for receive and transmit operations. Line error handling can be configured independently from the receive or transmit setup.

The DMA of the UART is enabled by first setting up the system DMA control registers and then enabling the UART_IMSK.ERBFI and UART_IMSK.ETBEI interrupts. This configuration is because the interrupt request lines double as DMA request lines. Depending on whether DMA is enabled or not, upon receiving these requests, the DMA control unit either generates a direct memory access or passes the UART interrupt on to the system interrupt handling unit(s). However, the error interrupt for the UART goes directly to the system interrupt handling unit(s), bypassing the DMA unit completely.

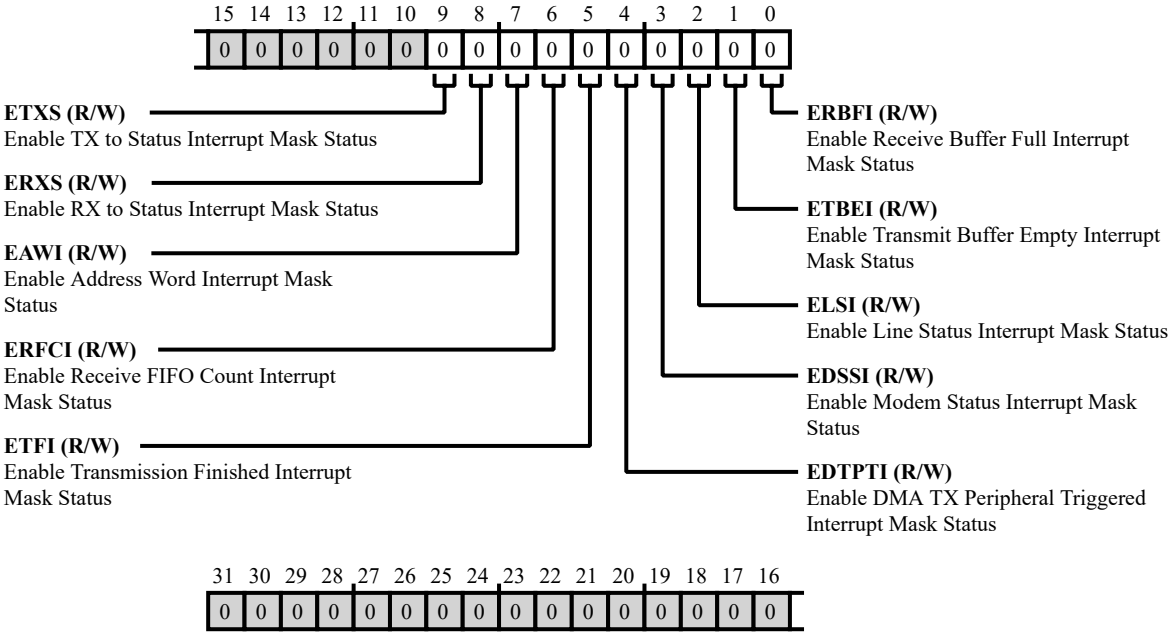


Figure 22-13: UART_IMSK Register Diagram

Table 22-12: UART_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ETXS	Enable TX to Status Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ETXS` bit indicates re-direction of the TX interrupt requests to status interrupt output. If cleared, TX interrupt requests are routed to normal interrupt outputs.	
		0	Interrupt is masked
		1	Interrupt is unmasked
8 (R/W)	ERXS	Enable RX to Status Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ERXS` bit indicates re-direction of RX interrupt requests to status interrupt output. If cleared, RX interrupt requests are routed to normal interrupt outputs.	
		0	Interrupt is masked
		1	Interrupt is unmasked
7 (R/W)	EAWI	Enable Address Word Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.EAWI` bit indicates generation of a status interrupt request when an Address word in MDB-mode is present in the `UART_RBR`. A received word is an address word if the `UART_STAT.ADDR` bit is set.	
		0	Interrupt is masked
		1	Interrupt is unmasked

Table 22-12: UART_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	ERFCI	Enable Receive FIFO Count Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.ERFCI bit indicates enabling of the receive buffer threshold interrupt request if signaled by the UART_STAT.RFCS bit. Read the UART_RBR register sufficient times to clear the interrupt request.	
		0	Interrupt is masked
		1	Interrupt is unmasked
5 (R/W)	ETFI	Enable Transmission Finished Interrupt Mask Status. If set (interrupt unmasked) the UART_IMSK.ETFI bit indicates enabling of interrupt generation on the status interrupt channel when the transmit buffer register, the transmit address register, and the transmit shift register are all empty as indicated by the UART_STAT.TFI. The UART_IMSK.ETFI interrupt can be used to avoid expensive polling of the UART_STAT.TEMT bit, when the UART clock or line drivers should be disabled after transmission has completed. W1C the UART_STAT.TFI bit to clear the interrupt request. In DMA operation, the UART_IMSK.ETFI bits functionality might be preferred.	
		0	Interrupt is masked
		1	Interrupt is unmasked
4 (R/W)	EDTPTI	Enable DMA TX Peripheral Triggered Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.EDTPTI bit indicates enabling of the DMA completion interrupt request to be delayed until the data has left the UART completely. This bit is required for DMA transmit operation only. If set, the UART can generate a DMA interrupt request by the time the UART_STAT.TEMT bit goes high after the last DMA data word is transmitted. When UART_IMSK.EDTPTI is set, usually the DMA_CFG.INT field is cleared to 00 in a STOP mode DMA. This set up suppresses the normal completion interrupt request, and the UART_STAT.TEMT event is signaled through the DMA controller and triggers the DMA interrupt. If both (DMA_CFG.INT not 00 and UART_IMSK.EDTPTI set), two interrupts are requested at the end of a STOP mode DMA.	
		0	Interrupt is masked
		1	Interrupt is unmasked
3 (R/W)	EDSSI	Enable Modem Status Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.EDSSI bit indicates enabling of a modem status interrupt request on the same status interrupt channel when the UART_STAT.SCTS bit is set. This indicates UART_CTS pin re-assertion. Write-1-to-clear (W1C) the UART_STAT.SCTS bit to clear the interrupt request.	
		0	Interrupt is masked
		1	Interrupt is unmasked

Table 22-12: UART_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	ELSI	Enable Line Status Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ELSI` bit indicates that redirection of TX and RX interrupt requests to the status interrupt output of the UART by OR'ing them with the `UART_STAT.OE`, `UART_STAT.PE`, `UART_STAT.FE`, and `UART_STAT.BI` interrupt requests. Set this bit when no DMA channel is associated with the UART. Enabling `UART_IMSK.ELSI` disables the RX/TX interrupt channels and negates the `UART_IMSK.EDTPTI` bit.	
		0	Interrupt is masked
		1	Interrupt is unmasked
1 (R/W)	ETBEI	Enable Transmit Buffer Empty Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ETBEI` bit indicates generation of a TX interrupt request if the `UART_STAT.THRE` bit is set.	
		0	Interrupt is masked
		1	Interrupt is unmasked
0 (R/W)	ERBFI	Enable Receive Buffer Full Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ERBFI` indicates generation of an RX interrupt request if the `UART_STAT.DR` bit is set.	
		0	Interrupt is masked
		1	Interrupt is unmasked

Interrupt Mask Clear Register

The UART_IMSK indicates interrupt mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART_IMSK_SET and UART_IMSK_CLR register pair. Writing ones to UART_IMSK_SET enables (unmasks) interrupt requests, and writing ones to UART_IMSK_CLR disables (masks) them. Reads from either register return the enabled bits. For more information, see the UART_IMSK register description.

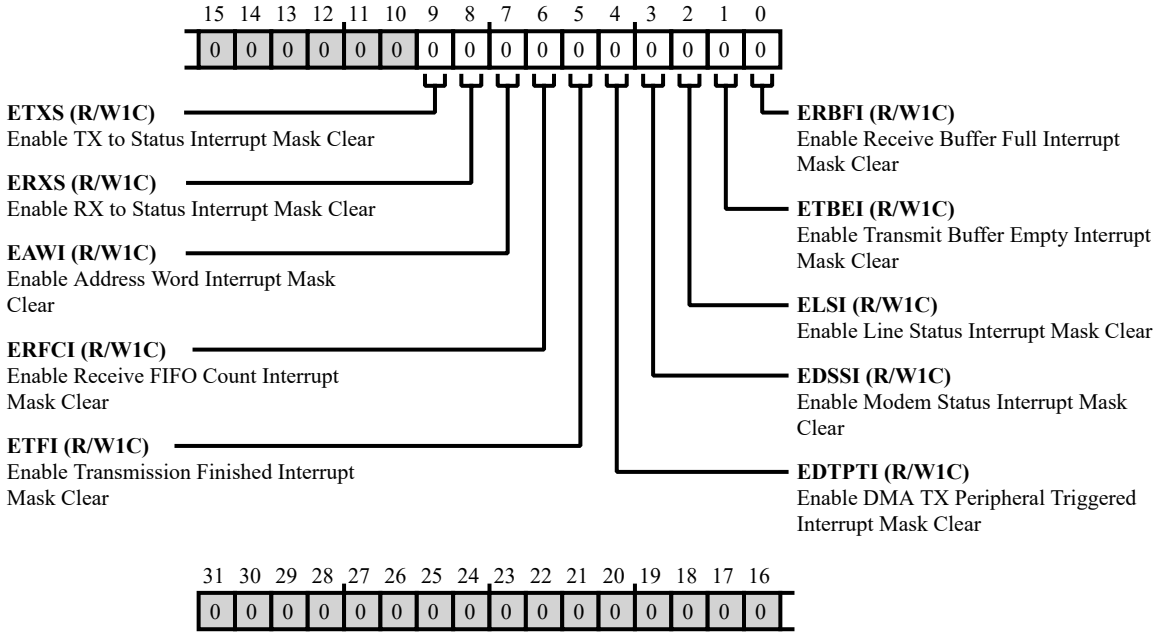


Figure 22-14: UART_IMSK_CLR Register Diagram

Table 22-13: UART_IMSK_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	ETXS	Enable TX to Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
8 (R/W1C)	ERXS	Enable RX to Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
7 (R/W1C)	EAWI	Enable Address Word Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt

Table 22-13: UART_IMSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	ERFCI	Enable Receive FIFO Count Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
5 (R/W1C)	ETFI	Enable Transmission Finished Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
4 (R/W1C)	EDTPTI	Enable DMA TX Peripheral Triggered Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
3 (R/W1C)	EDSSI	Enable Modem Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
2 (R/W1C)	ELSI	Enable Line Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
1 (R/W1C)	ETBEI	Enable Transmit Buffer Empty Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
0 (R/W1C)	ERBFI	Enable Receive Buffer Full Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt

Interrupt Mask Set Register

The UART_IMSK indicates interrupt request mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART_IMSK_SET and UART_IMSK_CLR register pair. Writing ones to UART_IMSK_SET enables (unmasks) interrupt requests, and writing ones to UART_IMSK_CLR disables (masks) them. Reads from either register return the enabled bits. For more information, see the UART_IMSK register description.

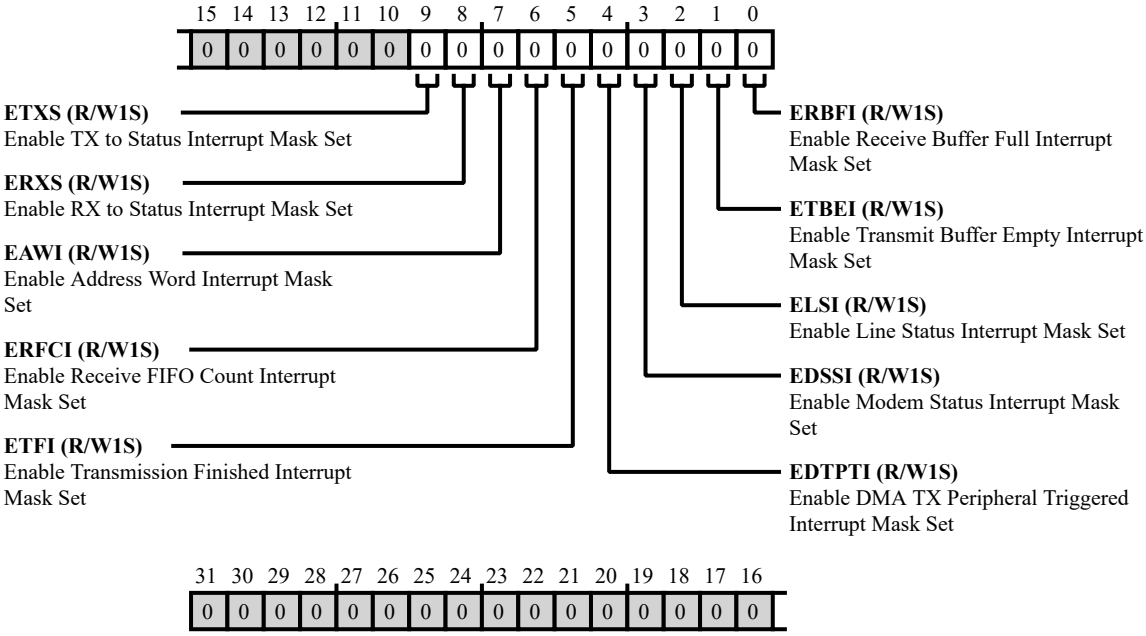


Figure 22-15: UART_IMSK_SET Register Diagram

Table 22-14: UART_IMSK_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1S)	ETXS	Enable TX to Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
8 (R/W1S)	ERXS	Enable RX to Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
7 (R/W1S)	EAWI	Enable Address Word Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt

Table 22-14: UART_IMSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	ERFCI	Enable Receive FIFO Count Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
5 (R/W1S)	ETFI	Enable Transmission Finished Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
4 (R/W1S)	EDTPTI	Enable DMA TX Peripheral Triggered Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
3 (R/W1S)	EDSSI	Enable Modem Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
2 (R/W1S)	ELSI	Enable Line Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
1 (R/W1S)	ETBEI	Enable Transmit Buffer Empty Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
0 (R/W1S)	ERBFI	Enable Receive Buffer Full Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt

Receive Buffer Register

The read-only UART_RBR register is the UART's receive buffer. It is updated when there is pending data in the receive FIFO. Newly available data is signaled by the UART_STAT.DR bit.

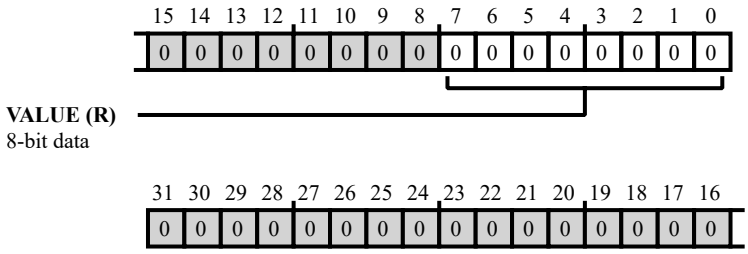


Figure 22-16: UART_RBR Register Diagram

Table 22-15: UART_RBR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VALUE	8-bit data.

Receive Shift Register

The read only UART_RSR register which returns the content of the UART's receive shift register.

The frame data is moved into this shift register after polarity inversion, if any (including the native polarity inversion in the IrDA case).

In the case of the longest frame (MDB, with parity mode, and 8 bit data word-length), the start bit may be shifted out and not available for reading at the end of the frame reception. This register is NOT reset at the start of frame. If read, in the middle of a frame reception, data corresponding the previous frame may not have entirely shifted out (for example, the read data that have been read may NOT correspond entirely to the frame being received).

Because the UART is receiving only 1 stop bit, the UART_RSR contains only 1 stop bit even if more than one stop bit is present in the actual transfer. This register may be considered as storing the 10 most recently received bits (taking into consideration the stop bit receive limitation above).

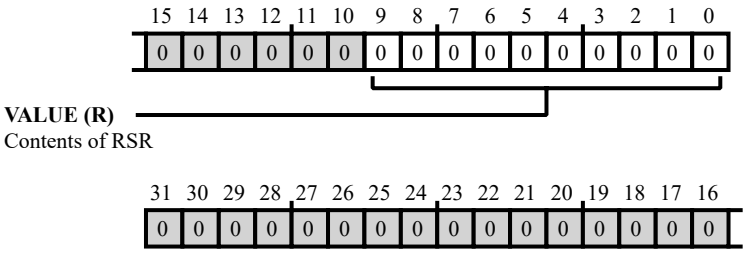


Figure 22-17: UART_RSR Register Diagram

Table 22-16: UART_RSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	VALUE	Contents of RSR.

Receive Counter Register

The UART_RXCNT register returns the content of 16-bit counter in the UART receiver. This count is used for baud rate clock generation (the lower [15:0] is the count data).

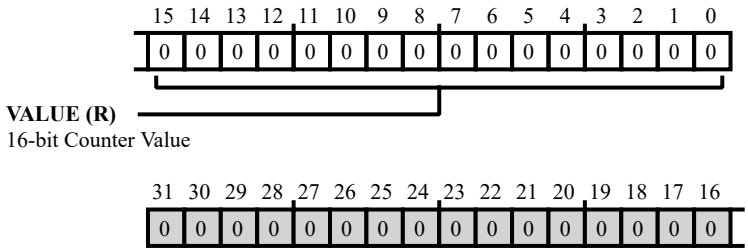


Figure 22-18: UART_RXCNT Register Diagram

Table 22-17: UART_RXCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	16-bit Counter Value.

Scratch Register

The UART_SCR registers contain 8-bit scratch pad data. These registers are used for general purpose data storage and do not control the UART hardware in any way.

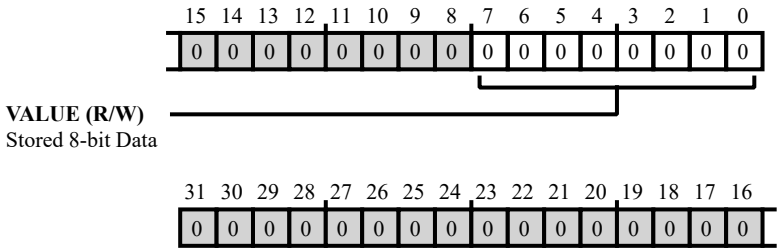


Figure 22-19: UART_SCR Register Diagram

Table 22-18: UART_SCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Stored 8-bit Data.

Status Register

The UART_STAT register contains the UART line status and UART modem status, as indicated by the current states of the UART's UART_CTS pin and internal receive buffers. Writes to this register can perform write-one-to-clear (W1C) operations on most status bits. Reading this register has no side effects.

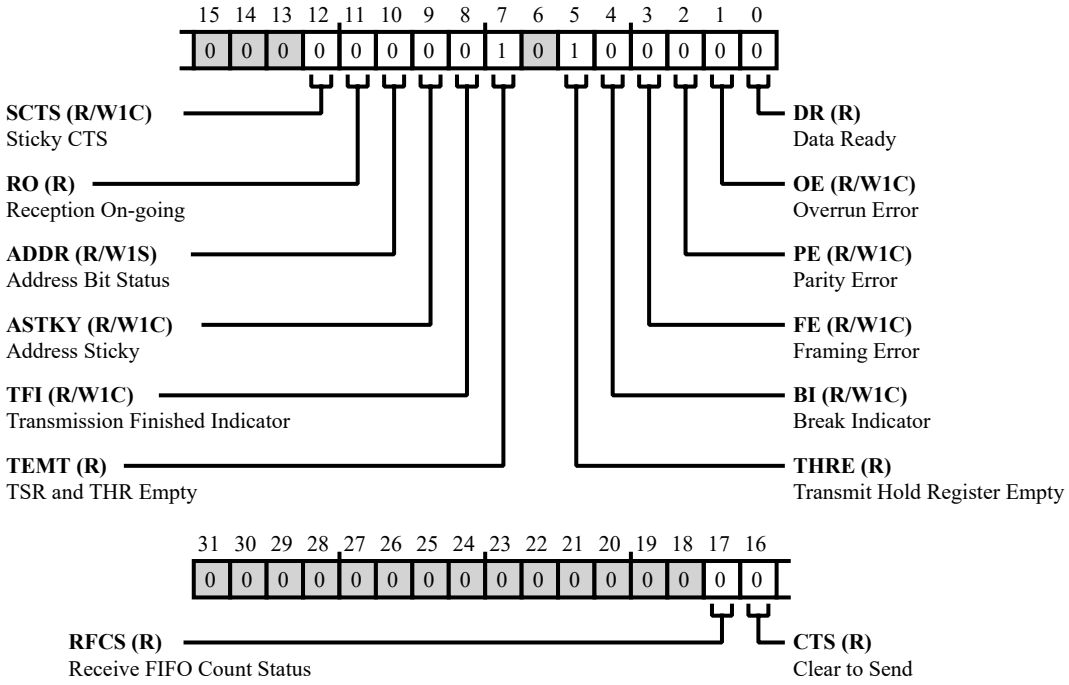


Figure 22-20: UART_STAT Register Diagram

Table 22-19: UART_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	RFCS	Receive FIFO Count Status. The UART_STAT.RFCS bit is set when the receive buffer holds more or equal entries than a certain threshold. The threshold is controlled by the UART_CTL.RFIT bit. If UART_CTL.RFIT is cleared, the threshold is four entries. If UART_CTL.RFIT is set, the threshold is seven entries. The UART_STAT.RFCS bit is cleared when the UART_RBR register is read sufficient times until the buffer is drained below the threshold. The UART_STAT.RFCS bit can trigger a status interrupt request if enabled by the UART_IMSK_SET.ERFCI bit.	
		0	RX FIFO has less than 4 (7) entries when RFIT=0 (1)
		1	RX FIFO has at least 4 (7) entries when RFIT=0 (1)

Table 22-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	CTS	Clear to Send. The UART_STAT.CTS bit holds the value (if UART_CTL.FCPOL set) or the complement value (if UART_CTL.FCPOL cleared) of the UART_CTS input pin. The UART_CTL.ACTS bit must be set to enable this feature. The core can read the value of the UART_STAT.CTS bit to determine whether the external device is ready to receive (UART_STAT.CTS set) or if it is busy (UART_STAT.CTS cleared). If UART_CTL.ACTS is cleared, the UART_TX handshaking protocol is disabled, and the UART transmits data as long as there is data to transmit, regardless of the value of UART_STAT.CTS. When UART_CTL.ACTS is cleared, the software can pause transmission temporarily by setting the XOFF bit. Note that in loopback mode (UART_CTL.LOOP_EN set), the UART_STAT.CTS bit is disconnected from the UART_CTS input pin. Instead, the bit is directly connected to the UART_CTL.MRTS bit.	
		0	Not clear to send (External device not ready to receive)
		1	Clear to send (External device ready to receive)
12 (R/W1C)	SCTS	Sticky CTS. The UART_STAT.SCTS bit is a sticky bit that is set when UART_STAT.CTS transitions from 0 to 1. The UART_STAT.SCTS bit is cleared by software with a W1C operation. This bit can trigger a line status interrupt request if enabled by the UART_IMSK_SET.EDSSI bit.	
		0	CTS has not transitioned from low to high
		1	CTS has transitioned from low to high
11 (R/NW)	RO	Reception On-going.	
		0	No data reception in progress
		1	Data reception in progress
10 (R/W1S)	ADDR	Address Bit Status. The UART_STAT.ADDR bit is used to mirror the address bit of the word in UART_RBR in multi-drop bus protocol, and is enabled only in MDB mode. The UART_STAT.ADDR bit is updated by hardware upon detecting a received word with the address bit in UART_RBR set or cleared. Additionally, software can set the ADDR bit with a write-1-to-set (W1S) operation.	
		0	Address bit is low
		1	Address bit is high

Table 22-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	ASTKY	Address Sticky. The UART_STAT.ASTKY bit is used in multi-drop bus mode to indicate whether a peripheral is currently being addressed. This bit is a sticky version of the UART_STAT.ADDR bit and is set by hardware when setting the UART_STAT.ADDR bit. The UART_STAT.ASTKY bit can only be cleared by software with a write-one-to-clear (W1C) operation. With the UART_STAT.ASTKY bit set, words will be received irrespective of the UART_CTL.MOD bit or UART_STAT.ADDR bit selection. With the UART_STAT.ASTKY bit cleared, only address words (UART_CTL.MOD bit set) will be received and words with UART_CTL.MOD bit cleared are ignored (not moved from the RSR to the RX FIFO) in MDB mode. The UART_STAT.ASTKY bit does not affect reception in non-MDB modes.	
		0	ADDR bit has not been set
		1	ADDR bit has been set
8 (R/W1C)	TFI	Transmission Finished Indicator. The UART_STAT.TFI bit is a sticky version of the UART_STAT.TEMT bit. While UART_STAT.TEMT is automatically cleared by hardware when new data is written to the UART_THR register, the sticky UART_STAT.TFI bit remains set, until it is cleared by software (W1C). The UART_STAT.TFI bit enables more flexible transmit interrupt request timing.	
		0	TEMT did not transition from 0 to 1
		1	TEMT transition from 0 to 1
7 (R/NW)	TEMT	TSR and THR Empty. The UART_STAT.TEMT bit indicates that the UART_THR and UART_TAIP registers and the UART_TSR register are empty. In this case, the program is permitted to write to the UART_THR and UART_TAIP registers twice without losing data. The UART_STAT.TEMT bit can also be used as indicator that pending UART transmission is completed. At that time, it is safe to disable the UART_CTL.EN bit or to three-state the off-chip line driver.	
		0	Not empty TSR/THR
		1	TSR/THR Empty

Table 22-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	THRE	Transmit Hold Register Empty. The UART_STAT.THRE bit indicates that the UART transmit channel is ready for new data and software can write to the UART_THR and UART_TAIP registers. Writes to the UART_THR and UART_TAIP registers clear the UART_STAT.THRE. The bit is set again when the UART_THR and UART_TAIP registers are empty and ready to accept data.	
		0	Not empty THR/TAIP
		1	Empty THR/TAIP
4 (R/W1C)	BI	Break Indicator. The UART_STAT.BI bit indicates that the first stop bit is sampled low and the entire data word, including parity bit, consists of low bits only. (This condition indicates that UART_RX was held low for more than the maximum word length.) The UART_STAT.BI bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The bit is sticky and can be cleared by W1C operations.	
		0	No break interrupt
		1	Break interrupt this indicates UARTxRX was held low(RPOLC=0) / high (RPOLC=1) for more than the maximum word length
3 (R/W1C)	FE	Framing Error. The UART_STAT.FE bit indicates that the first stop bit is sampled. This bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The UART_STAT.FE bit is sticky and can be cleared by W1C operations. Note that invalid stop bits can be simulated by setting the UART_CTL.FFE bit.	
		0	No error
		1	Invalid stop bit error
2 (R/W1C)	PE	Parity Error. The UART_STAT.PE bit indicates that the received parity bit does not match the expected value. This bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The UART_STAT.PE bit is sticky and can be cleared by W1C operations. Note that invalid parity bits can be simulated by setting the UART_CTL.FPE bit.	
		0	No parity error
		1	Parity error

Table 22-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	OE	Overrun Error. The UART_STAT.OE bit indicates that further data is received while the internal receive buffer was full. This bit is set when sampling the stop bit of the sixth data word. To avoid overruns, read the UART_RBR register in time. In DMA receive mode, overruns are very unlikely to happen ever. After an overrun occurs, the UART_RBR and receive FIFO are protected from being overwritten by new data until the UART_STAT.OE bit is cleared by software. The content of the UART_RSR register is lost as soon as the overrun occurs. The UART_STAT.OE bit is sticky and can be cleared by W1C operations.	
		0	No overrun
		1	Overrun error
0 (R/NW)	DR	Data Ready. The UART_STAT.DR bit indicates that data is available in the receiver and can be read from the UART_RBR register. The bit is set by hardware when the receiver detects the first valid stop bit. The bit is cleared by hardware when the UART_RBR register is read.	
		0	No new data
		1	New data in RBR

Transmit Address/Insert Pulse Register

The UART_TAIP register and the UART_THR register share the same physical register, but UART_TAIP has different effect than the UART_THR register when UART_TAIP is written to in MDB and UART modes.

In MDB mode, data written to the UART_TAIP register is transmitted as an address frame (as with the UART_CTL.MOD bit set).

In UART mode, a write to UART_TAIP causes a pulse of value UART_TAIP [7] for a duration of UART_TAIP [6:0] x bit time. (There is additional inversion if the UART_CTL.TPOLC bit is set).

Bit time is defined by the UART_CLK register. The transmission of the pulse is followed by stop bit transmission as specified by the UART_CTL.STB and UART_CTL.STBH bits. This could be used for supporting line break command and inter-frame gap.

In IrDA mode, writes to UART_TAIP is treated the same as writes to UART_THR.

Accesses to the UART_TAIP register have the same affects as the UART_THR register with respect to the UART_STAT.THRE, UART_STAT.TEMT, and UART_STAT.TFI flags.

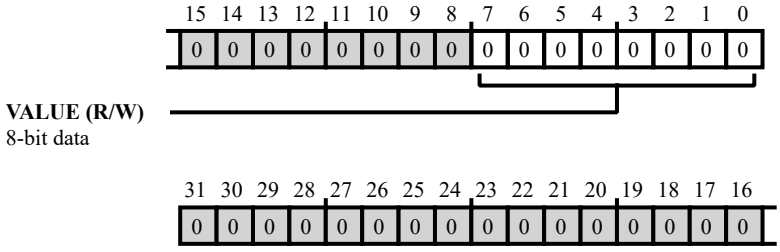


Figure 22-21: UART_TAIP Register Diagram

Table 22-20: UART_TAIP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	8-bit data.

Transmit Hold Register

The write-only UART_THR register is the UART's transmit buffer. The UART_STAT.THRE bit indicates whether data can be written to UART_THR. Writes to this register automatically propagate to the internal UART_TSR register as soon as UART_TSR is ready. Then, transmit operation is initiated immediately.

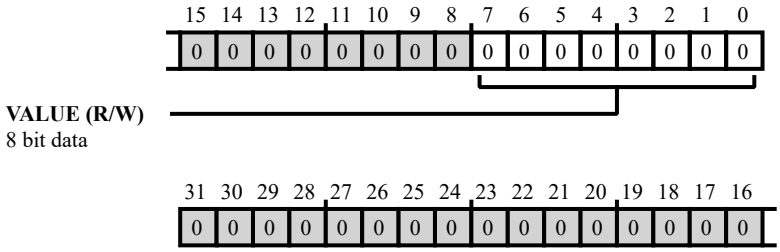


Figure 22-22: UART_THR Register Diagram

Table 22-21: UART_THR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	8 bit data.

Transmit Shift Register

The read only UART_TSR register which returns the content of the UART's transmit shift register.

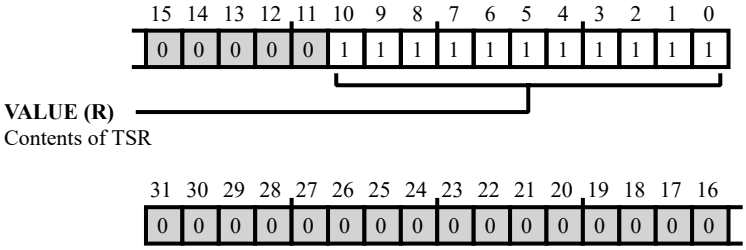


Figure 22-23: UART_TSR Register Diagram

Table 22-22: UART_TSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Contents of TSR.

Transmit Counter Register

The UART_TXCNT read only register returns the content of 16-bit counter in the UART transmitter. This count is used for baud rate clock generation (the lower [15:0] is the count data).

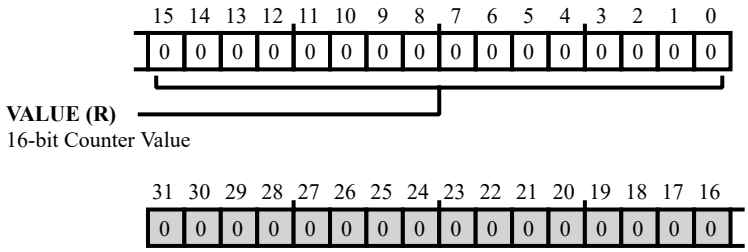


Figure 22-24: UART_TXCNT Register Diagram

Table 22-23: UART_TXCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	16-bit Counter Value.

23 General-Purpose Timer (TIMER)

The general-purpose timer (GP Timer) module serves as a collection of system timers that support various system-level functions. These functions include:

- Synchronized PWM waveform output capability
- External signal capture
- External event count
- General time-base functionality

Additionally, interrupt requests can be generated upon completion of timer events. Moreover, GP timers can act both as trigger requesters and trigger completers.

GP Timer Features

Each timer can be individually configured in any of these modes:

- Pin interrupt capture mode
- Windowed watchdog mode
- Pulse-width count and capture (WDTH_CAP) mode
- External Event (EXT_CLK) mode
- Pulse-width modulation (PWM_OUT) mode

Other features include:

- Synchronous operation
- Consistent management of period and pulse width values
- Autobaud detection for UART module (where available)
- Graceful bit pattern termination when stopping
- Support for center-aligned PWM patterns
- Error detection on implausible pattern values

- All read and write accesses to 32-bit registers are atomic
- Every timer has its dedicated interrupt request output

ADSP-2184x TIMER Register List

The General-Purpose Timer block (TIMER) provides timers that may be used for external event capture and measurement, system timing, and PWM waveform generation. A set of registers governs TIMER operations. For more information on TIMER functionality, see the TIMER register descriptions.

Table 23-1: ADSP-2184x TIMER Register List

Name	Description
TIMER_BCAST_DLY	Broadcast Delay Register
TIMER_BCAST_PER	Broadcast Period Register
TIMER_BCAST_WID	Broadcast Width Register
TIMER_DATA_ILAT	Data Interrupt Latch Register
TIMER_DATA_IMSK	Data Interrupt Mask Register
TIMER_ERR_TYPE	Error Type Status Register
TIMER_RUN	Run Register
TIMER_RUN_CLR	Run Clear Register
TIMER_RUN_SET	Run Set Register
TIMER_STAT_ILAT	Status Interrupt Latch Register
TIMER_STAT_IMSK	Status Interrupt Mask Register
TIMER_STOP_CFG	Stop Configuration Register
TIMER_STOP_CFG_CLR	Stop Configuration Clear Register
TIMER_STOP_CFG_SET	Stop Configuration Set Register
TIMER_TMR[n]_CFG	Timer n Configuration Register
TIMER_TMR[n]_CNT	Timer n Counter Register
TIMER_TMR[n]_DLY	Timer n Delay Register
TIMER_TMR[n]_PER	Timer n Period Register
TIMER_TMR[n]_WID	Timer n Width Register
TIMER_TRG_IE	Trigger Receiver Enable Register
TIMER_TRG_MSK	Trigger Generator Mask Register

ADSP-2184x TIMER Interrupt List

Table 23-2: ADSP-2184x TIMER Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
329	TIMER0_TMR00	TIMER0 Timer 0	Level	
330	TIMER0_TMR01	TIMER0 Timer 1	Level	
331	TIMER0_TMR02	TIMER0 Timer 2	Level	
332	TIMER0_TMR03	TIMER0 Timer 3	Level	
333	TIMER0_TMR04	TIMER0 Timer 4	Level	
334	TIMER0_TMR05	TIMER0 Timer 5	Level	
335	TIMER0_TMR06	TIMER0 Timer 6	Level	
336	TIMER0_TMR07	TIMER0 Timer 7	Level	
337	TIMER0_TMR08	TIMER0 Timer 8	Level	
338	TIMER0_TMR09	TIMER0 Timer 9	Level	
339	TIMER0_TMR10	TIMER0 Timer 10	Level	
340	TIMER0_TMR11	TIMER0 Timer 11	Level	
341	TIMER0_TMR12	TIMER0 Timer 12	Level	
342	TIMER0_TMR13	TIMER0 Timer 13	Level	
343	TIMER0_TMR14	TIMER0 Timer 14	Level	
344	TIMER0_TMR15	TIMER0 Timer 15	Level	
345	TIMER0_STAT	TIMER0 Status	Level	

ADSP-2184x TIMER Trigger List

Table 23-3: ADSP-2184x TIMER Trigger List Generators

Trigger ID	Name	Description	Sensitivity
173	TIMER0_TMR00_GEN	TIMER0 Timer 0 Trigger Initiator	Edge
174	TIMER0_TMR01_GEN	TIMER0 Timer 1 Trigger Initiator	Edge
175	TIMER0_TMR02_GEN	TIMER0 Timer 2 Trigger Initiator	Edge
176	TIMER0_TMR03_GEN	TIMER0 Timer 3 Trigger Initiator	Edge
177	TIMER0_TMR04_GEN	TIMER0 Timer 4 Trigger Initiator	Edge
178	TIMER0_TMR05_GEN	TIMER0 Timer 5 Trigger Initiator	Edge
179	TIMER0_TMR06_GEN	TIMER0 Timer 6 Trigger Initiator	Edge
180	TIMER0_TMR07_GEN	TIMER0 Timer 7 Trigger Initiator	Edge
181	TIMER0_TMR08_GEN	TIMER0 Timer 8 Trigger Initiator	Edge

Table 23-3: ADSP-2184x TIMER Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
182	TIMER0_TMR09_GEN	TIMER0 Timer 9 Trigger Initiator	Edge
183	TIMER0_TMR10_GEN	TIMER0 Timer 10 Trigger Initiator	Edge
184	TIMER0_TMR11_GEN	TIMER0 Timer 11 Trigger Initiator	Edge
185	TIMER0_TMR12_GEN	TIMER0 Timer 12 Trigger Initiator	Edge
186	TIMER0_TMR13_GEN	TIMER0 Timer 13 Trigger Initiator	Edge
187	TIMER0_TMR14_GEN	TIMER0 Timer 14 Trigger Initiator	Edge
188	TIMER0_TMR15_GEN	TIMER0 Timer 15 Trigger Initiator	Edge

Table 23-4: ADSP-2184x TIMER Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
188	TIMER0_TMR00_RCV0	TIMER0 Timer 0 Trigger Responder 0	Pulse
189	TIMER0_TMR00_RCV1	TIMER0 Timer 0 Trigger Responder 1	Pulse
190	TIMER0_TMR01_RCV0	TIMER0 Timer 1 Trigger Responder 0	Pulse
191	TIMER0_TMR01_RCV1	TIMER0 Timer 1 Trigger Responder 1	Pulse
192	TIMER0_TMR02_RCV0	TIMER0 Timer 2 Trigger Responder 0	Pulse
193	TIMER0_TMR02_RCV1	TIMER0 Timer 2 Trigger Responder 1	Pulse
194	TIMER0_TMR03_RCV0	TIMER0 Timer 3 Trigger Responder 0	Pulse
195	TIMER0_TMR03_RCV1	TIMER0 Timer 3 Trigger Responder 1	Pulse
196	TIMER0_TMR04_RCV0	TIMER0 Timer 4 Trigger Responder 0	Pulse
197	TIMER0_TMR04_RCV1	TIMER0 Timer 4 Trigger Responder 1	Pulse
198	TIMER0_TMR05_RCV0	TIMER0 Timer 5 Trigger Responder 0	Pulse
199	TIMER0_TMR05_RCV1	TIMER0 Timer 5 Trigger Responder 1	Pulse
200	TIMER0_TMR06_RCV0	TIMER0 Timer 6 Trigger Responder 0	Pulse
201	TIMER0_TMR06_RCV1	TIMER0 Timer 6 Trigger Responder 1	Pulse
202	TIMER0_TMR07_RCV0	TIMER0 Timer 7 Trigger Responder 0	Pulse
203	TIMER0_TMR07_RCV1	TIMER0 Timer 7 Trigger Responder 1	Pulse
204	TIMER0_TMR08_RCV0	TIMER0 Timer 8 Trigger Responder 0	Pulse
205	TIMER0_TMR08_RCV1	TIMER0 Timer 8 Trigger Responder 1	Pulse
206	TIMER0_TMR09_RCV0	TIMER0 Timer 9 Trigger Responder 0	Pulse
207	TIMER0_TMR09_RCV1	TIMER0 Timer 9 Trigger Responder 1	Pulse
208	TIMER0_TMR10_RCV0	TIMER0 Timer 10 Trigger Responder 0	Pulse

Table 23-4: ADSP-2184x TIMER Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
209	TIMER0_TMR10_RCV1	TIMER0 Timer 10 Trigger Responder 1	Pulse
210	TIMER0_TMR11_RCV0	TIMER0 Timer 11 Trigger Responder 0	Pulse
211	TIMER0_TMR11_RCV1	TIMER0 Timer 11 Trigger Responder 1	Pulse
212	TIMER0_TMR12_RCV0	TIMER0 Timer 12 Trigger Responder 0	Pulse
213	TIMER0_TMR12_RCV1	TIMER0 Timer 12 Trigger Responder 1	Pulse
214	TIMER0_TMR13_RCV0	TIMER0 Timer 13 Trigger Responder 0	Pulse
215	TIMER0_TMR13_RCV1	TIMER0 Timer 13 Trigger Responder 1	Pulse
216	TIMER0_TMR14_RCV0	TIMER0 Timer 14 Trigger Responder 0	Pulse
217	TIMER0_TMR14_RCV1	TIMER0 Timer 14 Trigger Responder 1	Pulse
218	TIMER0_TMR15_RCV0	TIMER0 Timer 15 Trigger Responder 0	Pulse
219	TIMER0_TMR15_RCV1	TIMER0 Timer 15 Trigger Responder 1	Pulse

Timer Block Diagram

The ***Timer Block Diagram*** shows all of the possible clock sources.

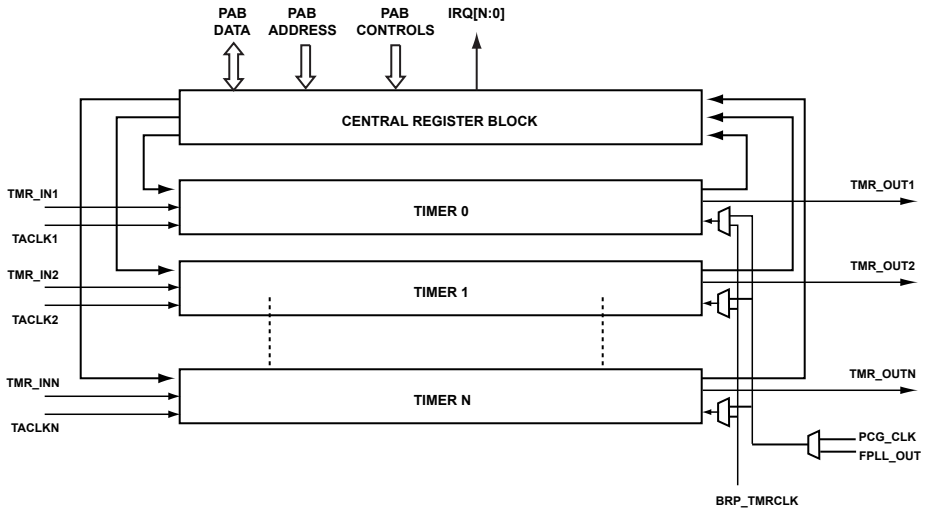


Figure 23-1: Timer Block Diagram

Internal Interface

The processor core always accesses the timer registers through the MMR access bus. Hardware ensures that all read and write operations from and to 32-bit timer registers are atomic. Every timer has a dedicated data interrupt request. There is also one common timer status and error interrupt request output that connects to the system event controller. Whenever a data interrupt request is generated, a data trigger generator pulse is also driven out, if

enabled. Each timer has an individual trigger input line, and each timer can be either started or stopped as a trigger receiver.

In total, the GP timer module can have up to (N + 1) interrupt request output lines and N data trigger lines.

Internal Timer Connections

The Timers support alternate inputs for the clock/capture (see External Interface). Some signals have internal default alternate connections according to the ***Timer Signal Routing*** table.

Table 23-5: Timer Signal Routing

Timer Signal	Connection
TM0_ACLK0	SYS_CLKIN0
TM0_ACI5	DAI0_CRS_PB04
TM0_ACLK5	DAI0_CRS_PB03
TM0_ACI6	DAI1_CRS_PB04
TM0_ACLK6	DAI1_CRS_PB03
TM0_ACI07	CNT0_TO signal
TM0_ACLK7	SYS_CLKIN0
TM0_ACI8	DAI0_PB06
TM0_ACLK8	DAI0_PB05
TM0_ACI9	DAI1_PB06
TM0_ACLK9	DAI1_PB05

External Interface

Each GP timer module can support up to 16 individual timers. However, most processors have less than this number. The exact number of timers available on a given processor is available in the data sheet for the processor.

Every timer has one main input/output signal (`TIMER_TMR[nn]`) and, usually, one auxiliary input pin, used as an alternate capture input (`TIMER_ACI[nn]`). Each timer can either run with a time base of SCLK or can reference an external clock on one of two (`TIMER_ACLK[nn]`) pins. The TMR_ALT_CLK0 signal maps to individual alternate clock ((`TIMER_ACLK[nn]`) pins for one or more timers. For instance, a TM_ACLK3 pin would provide an alternate site to supply an external signal that would serve as reference clock for TMR3. Likewise, the TMR_ALT_CLK1 signal from each timer unit connects internally to provide a single global timer clock pin (`TIMER_CLK`) for the GP timer module. It is used as an additional time base.

GP Timer Operating Modes

The following sections provide information on the various operating modes of the GP timer.

General Operation

The core of every timer is a 32-bit counter that can be interrogated through the read-only TIMER_TMR[n]_CNT register. Once the module enables a timer, it loads the timer TIMER_TMR[n]_CNT register with a starting value.

A timer can operate in one of several different modes, configured through the TIMER_TMR[n]_CFG register for that timer. These modes are: PWMOUT, EXTCLK, WIDCAP, WATCHDOG, PININT, and IDLE. The ***Timer Mode Descriptions*** table summarizes the modes.

Table 23-6: Timer Mode Descriptions

Timer Mode	Description
PWMOUT	Generates single or continuous PWM waveforms with programmable pulse width, period, and delay
EXTCLK	Counts edges of an externally applied waveform
WIDCAP	Captures pulse width or period of an externally applied waveform
WATCHDOG	Monitors pulse width or period of an external signal and compares against a window of acceptable values, optionally generating an interrupt when it falls inside or outside of that window
PININT	Can generate an interrupt request on an active edge applied to a timer pin
IDLE	Idle; no activity

Period, Width and Delay Register Interaction

When the timer is started, writes to the buffer registers are immediately copied through to the double-buffered period, pulse width, and delay registers. These values are then ready for use in the first timer period. When a timer is already running, software can write new values to the TIMER_TMR[n]_PER, TIMER_TMR[n]_WID, and TIMER_TMR[n]_DLY registers. The written values are buffered and do not update into the registers until the end of the current period. (The update occurs when the value in the TIMER_TMR[n]_CNT register equals the value in the TIMER_TMR[n]_PER register.)

If new values are not written to these registers, the value from the previous period is reused. Writes to these registers are atomic; it is not possible for the high word to be written without the low word also being written. Values written to the period, pulse width, and delay registers are always stored in the buffer registers. Reads from the same register always return the current, active value of period, pulse width, or delay value. Written values are not readback until they become active.

The usage of the TIMER_TMR[n]_PER, TIMER_TMR[n]_WID, and TIMER_TMR[n]_DLY registers varies, depending on the mode of the timer specified by the TIMER_TMR[n]_CFG.TMODE bits. See the ***Usage of the Period, Width, and Delay Registers in Different Timer Modes*** table for more information.

Table 23-7: Usage of the Period, Width, and Delay Registers in Different Timer Modes

Timer Mode	TIMER_TMR[n]_PER	TIMER_TMR[n]_WID	TIMER_TMR[n]_DLY
IDLE	Not writable	Not writable	Not writable

Table 23-7: Usage of the Period, Width, and Delay Registers in Different Timer Modes (Continued)

Timer Mode	TIMER_TMR[n]_PER	TIMER_TMR[n]_WID	TIMER_TMR[n]_DLY
WATCHDOG	Update on-the-fly. New value takes effect either at timer start or when an asserting edge on the input signal is sensed.	Read-only. Retains value of last measured width or period of the input signal.	Update on-the-fly. New value takes effect either at timer start or when an asserting edge on the input signal is sensed.
WIDCAP	Read-only. Period value captured at the appropriate time and updated from its buffer register simultaneously with the Width register.	Read-only. Width value captured at the appropriate time and updated from its buffer register simultaneously with the Period register.	Not used
PWMOUT	Update on-the-fly. New value takes effect either at timer start or at the end of the current period. A write followed by immediate read returns the current operating values.	Update on-the-fly. New value takes effect either at timer start or at the end of the current period. A write followed by immediate read returns the current operating values.	Update on-the-fly. New value takes effect either at timer start or at the end of the current period. A write followed by immediate read returns the current operating values.
EXTCLK	Can be updated on-the-fly.	Not used	Not used
PININT	Not used	Not used	Not used

If any of the period, pulse width, and delay registers are not used, then programs cannot write into that register. For example, in WIDCAP mode, the delay registers are not used. So, the program is not allowed to write any value to the TIMER_TMR[n]_DLY register. To prevent undesired operation, program the TIMER_TMR[n]_CFG.TMODE bits before programming the period, width, or delay registers.

If a program changes the TIMER_TMR[n]_CFG.TMODE bits from a status register to writable register (for example in PWMOUT mode), hardware does not clear these registers. These values are automatically overwritten by new values specified by software.

In PWMOUT mode with small periods, there may not be enough time between updates from the buffer registers to write these registers. The next period can use one old value and one new value. To prevent (width + pulse delay) > period errors, write the width and delay registers before the period register when decreasing the values. Write the period register before the width and delay registers when increasing the value.

Single-Pulse PWMOUT Mode

In single-pulse PWMOUT mode, the timer generates a single pulse on the TIMER_TMR[nn] pin. This mode is frequently used to implement a precise delay, often with generating an output trigger. The timer module uses the value in the TIMER_TMR[n]_DLY register to control the assertion of a pulse. The value in the TIMER_TMR[n]_WID register defines the pulse width. The TIMER_TMR[n]_PER is not used and cannot be written in this mode. After completion of the pulse, the timer is automatically stopped, and optionally generates an interrupt. The timer uses the TIMER_TMR[n]_CFG.PULSEHI bit to control pulse polarity.

The timer can be configured to generate a data interrupt request after satisfying various conditions specified by the TIMER_TMR[n]_CFG.IRQMODE bits.

It is not necessary to clear the relevant TIMER_RUN bit to stop the timer cleanly. At the end of the pulse, the timer stops automatically and the corresponding TIMER_RUN bit is cleared. To generate multiple discrete pulses (as opposed to a continuous PWM waveform), write a 1 to the appropriate TIMER_RUN bit, and wait for the timer to stop. Then, write another 1 to the same TIMER_RUN bit.

Continuous PWMOUT Mode

In continuous PWMOUT mode, the timer generates a repetitive pulse with a well-defined period, duty cycle, and pulse position. The TIMER_TMR[n]_DLY, TIMER_TMR[n]_PER, and TIMER_TMR[n]_WID registers are programmed with the values of the required PWM pulse. After the timer is started, the counter counts towards the value programmed in the TIMER_TMR[n]_PER register. Initially, the TIMER_TMR[nn] pin remains in a deasserted state. The pin toggles to an asserted state when the value in the TIMER_TMR[n]_CNT register equals the value in the TIMER_TMR[n]_DLY register.

The timer can control the assertion sense of the TIMER_TMR[nn] pin with the TIMER_TMR[n]_CFG.PULSEHI bit. The TIMER_TMR[nn] pin holds this value for the number of clock cycles specified in the TIMER_TMR[n]_WID register. Then, the pin deasserts and holds this value until the completion of the programmed period. The same waveform is generated repeatedly until the timer is disabled.

The timer can be configured to generate a data interrupt request after satisfying any of various conditions specified by the TIMER_TMR[n]_CFG.IRQMODE bits.

It is important to guarantee that the programmed period is greater than or equal to the sum of width and delay. Similarly, delay must be less than period. Violating either of these criteria results in an unpredictable waveform on the TIMER_TMR[nn] pin until the situation is rectified by writing proper values to these registers.

The maximum frequency possible to generate on the TIMER_TMR[nn] pin is achieved by setting TIMER_TMR[n]_PER to 2 and TIMER_TMR[n]_WID to 1. This operation makes the TIMER_TMR[nn] pin toggle each SCLK clock cycle (assuming the timer is configured to clock internally), producing a duty cycle of 50%.

When the TIMER_STOP_CFG.TMR[nn] bit of a timer is 0, the timer treats a stop operation as a stop-is-pending condition. When terminated with this setting, the timer automatically completes the current waveform and then stops cleanly, remaining in a deasserted state. This functionality prevents truncation of the current pulse and unwanted PWM patterns at the TIMER_TMR[nn] pin. The processor can determine when the timer stops running by polling the corresponding TIMER_RUN.TMR[nn] bit until it reads 0 or by waiting for the last interrupt (if enabled).

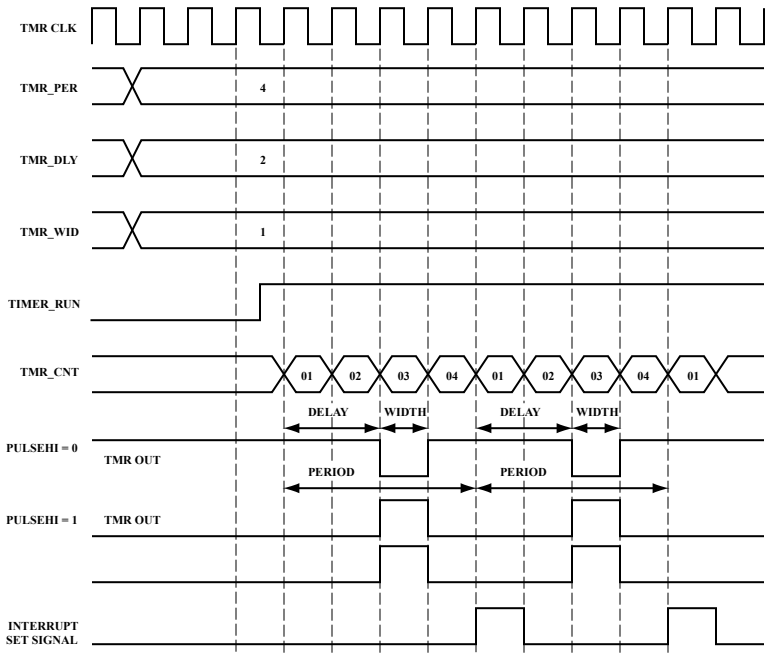


Figure 23-2: Signal Generation in Continuous PWMOUT Mode

The `TIMER_TMR[n]_CFG` register cannot be reconfigured until after the timer stops and the `TIMER_RUN` register reads 0.

Programs can force a timer to stop immediately in PWMOUT mode by writing a 1 to the `TIMER_STOP_CFG` register followed by writing a 1 to the `TIMER_RUN_CLR` register. (Or a program can stop a timer by writing a 0 to the appropriate `TIMER_RUN.TMR[nn]` bit.) This operation stops the timer whether the pending stop is waiting for the end of the current period or the end of the current pulse width. The timer can use this feature to regain immediate control of a timer during an error recovery sequence.

Use this feature carefully, as it can corrupt the PWM pattern generated at the `TIMER_TMR[nn]` pin, though after such a stop the pin deasserts automatically. Each timer samples its `TIMER_RUN.TMR[nn]` bit at the end of each period. It stops cleanly at the end of the first period after the `TIMER_RUN.TMR[nn]` bit is low. A timer that is disabled and then restarted (before the end of the current period), continues to run as if nothing happened. Typically, the program disables a PWMOUT timer and then waits for it to stop itself.

Width Capture (WIDCAP) Mode

The timer uses WIDCAP mode, often called capture mode, to measure pulse widths on the `TIMER_TMR[nn]` or (`TIMER_ACI[nn]`) inputs. The polarity (active high or low) of the input signal can be selected with the `TIMER_TMR[n]_CFG.PULSEHI` bit. The ***Timer Signal Flow in Width Capture Mode*** figure shows the control signal flow for WIDCAP_CAP mode.

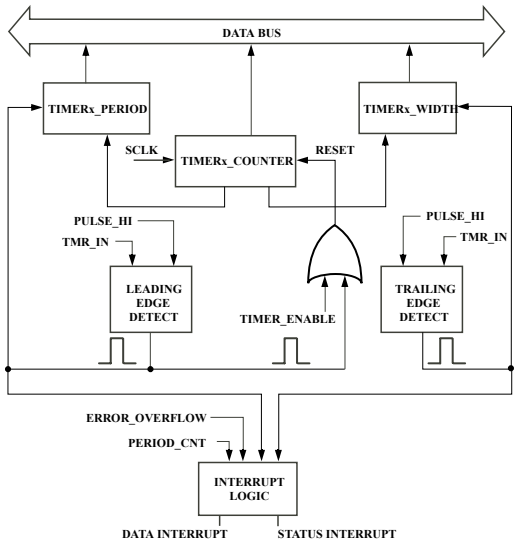


Figure 23-3: Timer Signal Flow in Width Capture Mode

NOTE: SCLK in the ***Timer Signal Flow in Width Capture Mode*** figure is SCLK.

In this mode, the timer uses the `TIMER_TMR[n]_CFG.TINSEL` bit to select between the `TIMER_TMR[nn]` or(`TIMER_ACI[nn]`) input. The internally clocked timer is used to determine the period and pulse width of the externally applied rectangular waveforms.

When a timer is enabled in this mode, the timer resets the count in its `TIMER_TMR[n]_CNT` register to 0x0000 0001. It does not start counting until it detects a leading edge on the selected input pin.

When the timer detects the first leading edge, it starts incrementing. When it detects a trailing edge of a waveform, it captures the current 32-bit value of its `TIMER_TMR[n]_CNT` register into its width buffer register. At the next leading edge, the timer transfers the current 32-bit value of its `TIMER_TMR[n]_CNT` register into its period buffer register. The `TIMER_TMR[n]_CNT` register is reset to 0x0000 0001 again, and the timer continues counting and capturing until it is disabled.

In this mode, programs can measure both the pulse width and the pulse period of a waveform. The timer does not use the `TIMER_TMR[n]_DLY` register in this mode. The timer uses the `TIMER_TMR[n]_CFG.PULSEHI` bit to control the definition of leading edge and trailing edge of the `TIMER_TMR[nn]` /(`TIMER_ACI[nn]`) pin.

In WIDCAP mode, the following events always occur at the same time as one unit:

1. The `TIMER_TMR[n]_PER` register is updated from the period buffer register.
2. The `TIMER_TMR[n]_WID` register is updated from the width buffer register.
3. The `TIMER_DATA_ILAT.TMR[nn]` bit is set (if enabled).
4. A timer data trigger pulse is generated (if enabled).

The `TIMER_TMR[n]_CFG.TMODE` bit 0 controls the point in time at which this set of events is executed. Taken together, these four events are called a measurement report. The `TIMER_STAT_ILAT` register is not

set at a measurement report. A measurement report occurs, at most, once per input signal period. The current TIMER_TMR[n]_CNT value is always copied to the width buffer and period buffer registers at the trailing and leading edges of the input signal, respectively. But these values are not visible to software. A measurement report event samples the captured values into visible registers and sets the timer interrupt request to signal that the TIMER_TMR[n]_PER and the TIMER_TMR[n]_WID registers are ready to be read.

When the TIMER_TMR[n]_CFG.TMODE bit =b#1011, the measurement report occurs just after the width buffer register captures its value at a falling edge. Then, the TIMER_TMR[n]_WID register reports the pulse width measured in the pulse that has ended, but the TIMER_TMR[n]_PER register reports the pulse period measured at the end of the previous period. If only the first trailing edge has occurred, then the first period value has not yet been measured at the first measurement report. So, the period value is not valid. A read of the TIMER_TMR[n]_PER value in this case returns 0. See the ***Example of Width Capture Deasserted Mode (TMODE=b#1011)*** figure for more information.

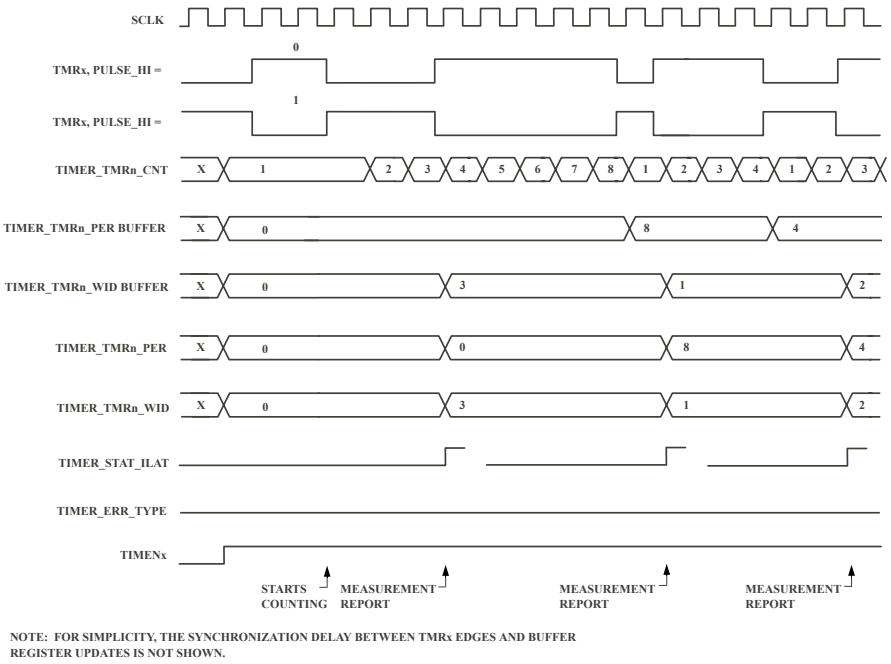


Figure 23-4: Example of Width Capture Deasserted Mode (TMODE=b#1011)

NOTE: SCLK in the ***Example of Width Capture Deasserted Mode (TMODE=b#1011)*** figure is SCLK.

When the TIMER_TMR[n]_CFG.TMODE bit =b#1010, the measurement report occurs just after the period buffer register captures its value at a leading edge. Then, the TIMER_TMR[n]_PER and TIMER_TMR[n]_WID registers report the pulse period and pulse width measured in the period that has ended. Refer to the ***Example of Width Capture Asserted Mode (TMODE=b#1010)*** figure for more information.

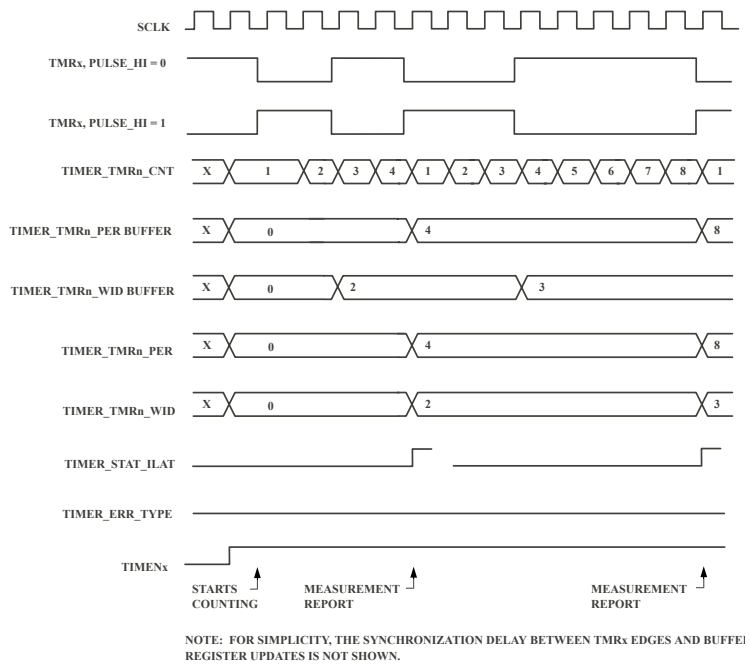


Figure 23-5: Example of Width Capture Asserted Mode (TMODE=b#1010)

NOTE: SCLK in the ***Example of Width Capture Asserted Mode (TMODE=b#1010)*** figure is SCLK.

To measure the pulse width of a waveform that has only one leading edge and one trailing edge, set `TIMER_TMR[n]_CFG.TMODE` = b#1011. When `TIMER_TMR[n]_CFG.TMODE` = b#1010, no period value is captured in the period buffer register. Instead, the timer generates an error report interrupt request (if enabled) when the `TIMER_TMR[n]_CNT` range is exceeded and the counter wraps around. In this case, both the `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers read 0 (because no measurement report occurred to copy the value captured in the width buffer register to the `TIMER_TMR[n]_WID` register).

When using the `TIMER_TMR[n]_CFG.TMODE` bit =b#1010 mode to measure the width of a single pulse, programs can disable the timer after taking the interrupt that ends the measurement interval. When desired, restart the timer as appropriate in preparation for another measurement. This procedure prevents the timer from freely running after the width measurement and logging errors generated by the timer count overflowing.

Width Capture Mode Overflow

A timer status interrupt request (when enabled) is generated when the `TIMER_TMR[n]_CNT` register wraps around from 0xFFFF FFFF to 0 in the absence of a leading edge. At that point, the `TIMER_STAT_ILAT` bit is set and the `TIMER_ERR_TYPE` bits change to indicate a count overflow due to a period greater than the range of the counter. This indication is referred to as an error report. A data interrupt request in WIDCAP mode indicates that a new measurement is ready to be read (a measurement report). Similarly, an interrupt request on the timer status interrupt line (shared interrupt request for all timers) indicates an overflow error when generated in this mode.

The `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers are never updated at the time an overflow error is signaled. If the timer overflows and the `TIMER_TMR[n]_CFG.TMODE` bit =b#1010, the `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers are not updated. If the timer overflows and the

TIMER_TMR[n]_CFG.TMODE bit =b#1011, the TIMER_TMR[n]_PER and TIMER_TMR[n]_WID registers are updated only if a trailing edge is detected at a previous measurement report.

Software can count the number of error reports between measurement report interrupt requests to measure input signal periods longer than 0xFFFF FFFF. Each error report interrupt request adds a full 2^{32}SCLK counts to the total for the period, but the width is ambiguous. Ensure that if software monitors only the status interrupt request, then status interrupt requests from all other timers are masked.

Refer to the ***Example Timing for Width Capture Followed by Period Overflow (TMR_CFG.TMODE=b#1010)*** figure. The period is 0x1 0000 0004, but the pulse width could be either 0x0 0000 0002 or 0x1 0000 0002.

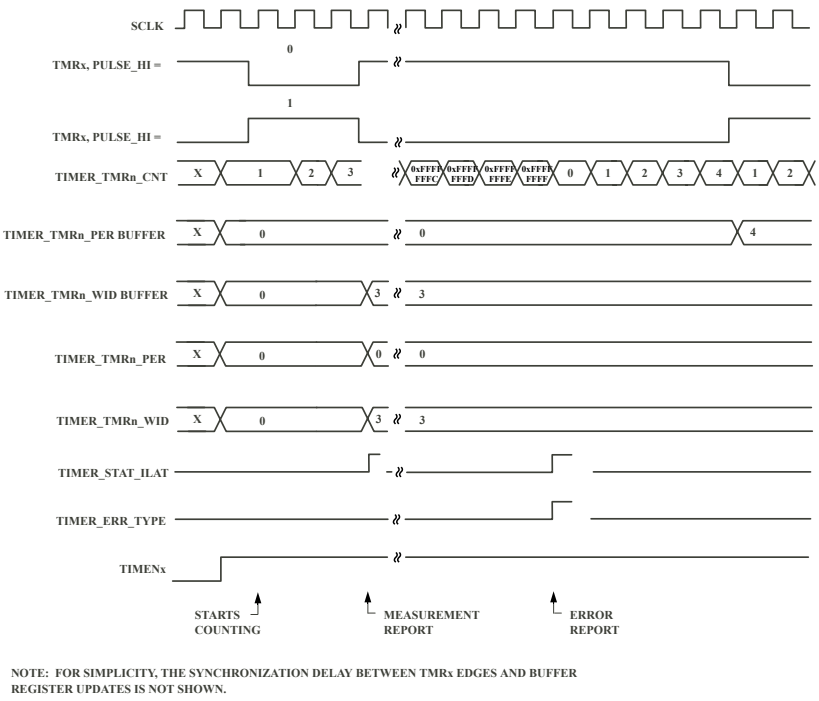


Figure 23-6: Example Timing for Width Capture Followed by Period Overflow (TMR_CFG.TMODE=b#1010)

NOTE: SCLK in the ***Example Timing for Width Capture Followed by Period Overflow*** figure is SCLK.

The waveform applied to the TIMER_TMR[nn] (or (TIMER_ACI[nn])) pin is not required to have a 50% duty cycle. The minimum input low time is little more than one SCLK period. The minimum input high time is a little more than one SCLK period. (Refer to the product data sheet for details.) The maximum TIMER_TMR[nn] input frequency is less than SCLK/2, with a 50% duty cycle. Under these conditions, the WIDCAP mode timer measures: period =2 and pulse width =1.

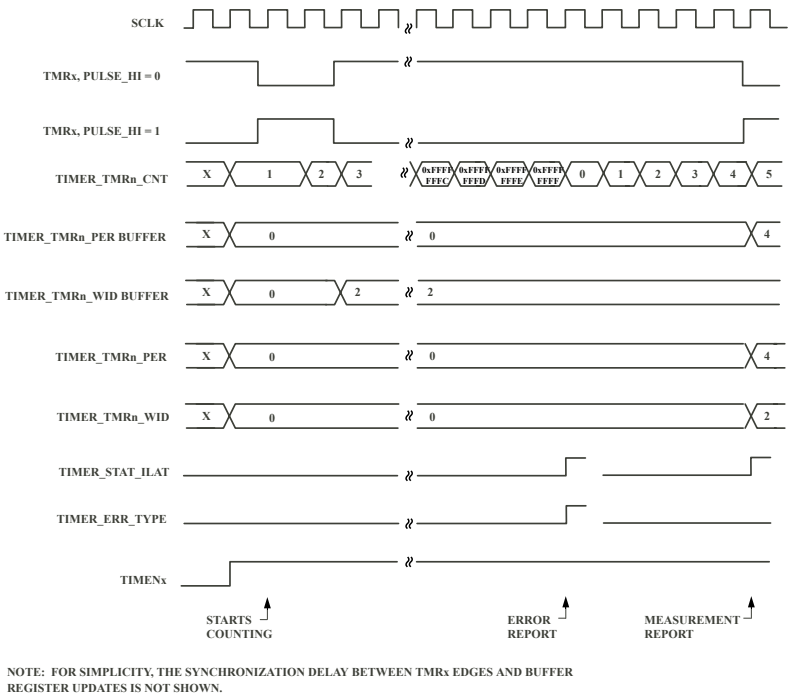


Figure 23-7: Example Timing for Width Capture Followed by Period Overflow (TMR_CFG.TMODE=b#1011)

NOTE: SCLK in the ***Example Timing for Width Capture Followed by Period Overflow*** figure is SCLK.

Windowed Watchdog (WATCHDOG) Modes

In windowed watchdog (WATCHDOG) modes, a timer can take inputs from one of the `TIMER_TMR[nn]` pins or the `TIMER_ACI[nn]` pins. With this mode, the timer can monitor pulse width (width watchdog mode) or pulse period (period watchdog mode) on the input line. It also compares the measured value against a minimum required value and maximum allowed value and generates an interrupt request appropriately. The timer uses the `TIMER_TMR[n]_CFG.PULSEHI` bit to select polarity of the input signal.

The waveform applied to the input pin in watchdog mode is not required to have a 50% duty cycle. The minimum input pulse low time, high time, and total period specifications are available in the product data sheet.

Windowed Watchdog Width Mode

In windowed watchdog width mode, the timer counter monitors the pulse width of an input signal on either the `TIMER_TMR[nn]` pin or one of the alternate clock pins (`TIMER_ACLK[nn]`). Program the minimum pulse width (p_{MIN}) in the `TIMER_TMR[n]_DLY` register and the maximum pulse width (p_{MAX}) in the `TIMER_TMR[n]_PER` register. Both values are programmed in terms of number of clock cycles (SCLK or alternate clock). The timer can generate an interrupt if the deasserting pulse edge occurs:

- Inside the window (p_{MIN} < pulse width ≤ p_{MAX}), or
- Outside the window (pulse width ≤ p_{MIN} or pulse width > p_{MAX})

After enabling the timer in this mode, it always starts counting at the asserting edge of the input signal. Any pulse that is already active when the timer is enabled is ignored.

With the TIMER_TMR[n]_CFG.IRQMODE bit =b#11, the timer generates an interrupt if the timed pulse width exceeds p_{MAX}, or if the pulse width is less than p_{MIN}. After attaining p_{MAX}, the pulse stays at an active level, and the counter keeps on counting until it sees a deasserting edge. When the input pulse is not active, the counter holds its current value until it again sees an asserting edge, or it restarts. An interrupt can also be generated for when the pulse occurs within the specified window condition, by setting TIMER_TMR[n]_CFG.IRQMODE =b#10.

In this mode, a trailing edge on the input pin triggers capturing of pulse width into the TIMER_TMR[n]_WID register. During the inactive portion of the input signal, the internal counter does not increment. The ***Watchdog Width Mode Timing*** figure shows the signal flow in this mode.

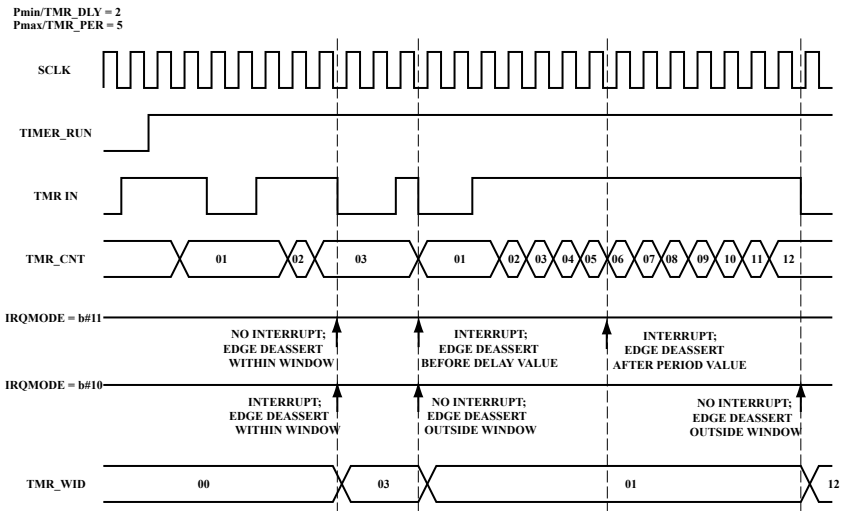


Figure 23-8: Watchdog Width Mode Timing

NOTE: SCLK in the ***Watchdog Width Mode Timing*** figure is SCLK.

To check only the upper limit on pulse width (p_{MAX} but not p_{MIN}), then program p_{MIN} as 0 or 1. In such a case, it is better to use TIMER_TMR[n]_CFG.IRQMODE =b#11. With TIMER_TMR[n]_CFG.IRQMODE = b#10, a pulse width of 1 clock cycle results in an interrupt. For details, see the ***Windowed Watchdog Width Mode Interpretation*** table.

Table 23-8: Windowed Watchdog Width Mode Interpretation

Timer Delay	Timer Period	Incoming Pulse Width	IRQMODE= b#10	IRQMODE= b#11	Error Interrupt?
0 or 1	Anything ≥ 1	PW = 1	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		PW ≤ TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		PW > TMR_PER	No Interrupt	Interrupt when pulse width exceeds Pmax (Period Register) Value	No Error Interrupt
> 1 but ≤ (Period -1)	Anything > 1	PW ≤ TMR_DLY	No Interrupt	Interrupt at deasserting edge of input signal	No Error Interrupt
		TMR_DLY < PW≤ TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		PW > TMR_PER	No Interrupt	Interrupt when pulse width exceeds Pmax (Period Register) Value	No Error Interrupt
≥ Period	-	PW ≤ TMR_PER	Undefined	Undefined	No Error Interrupt
	-	PW > TMR_PER	Undefined	Undefined	b#11 Error Type
-	0	-	Undefined	Undefined	b#10 Error Type

Windowed Watchdog Period Mode

In this mode, the timer monitors the number of clock cycles between two consecutive rising or falling edges of an input signal on one of the `TIMER_TMR[nn]` or `TIMER_ACI[nn]` pins. Program the required minimum number of clock cycles (t_{MIN}) in the `TIMER_TMR[n]_DLY` register and the required maximum allowed number of clock cycles (t_{MAX}) in the `TIMER_TMR[n]_PER` register. Both values are programmed in terms of number of clock cycles (SCLK) or alternate time clock (`TIMER_ACLK[nn]`). The timer can generate an interrupt when two consecutive occurrences of an active edge are:

- Within a specified window (t_{MIN} < Pulse Period ≤ t_{MAX}), or
- Outside a specified window (pulse width ≤ (t_{MIN} or t_{MAX} < pulse width)

When the `TIMER_TMR[n]_CFG.IRQMODE` bit =b#11 and the pulse period > t_{MAX} or is ≤ t_{MIN}, the timer generates an interrupt (if unmasked). After attaining the t_{MAX} value, the counter keeps on counting until it sees an active edge on the input line. An interrupt can also be generated for when the pulse occurs within the specified

window condition, by setting TIMER_TMR[n]_CFG.IRQMODE =b#10. Refer to the ***Watchdog Period Mode Timing*** figure for timer functionality in period watchdog mode.

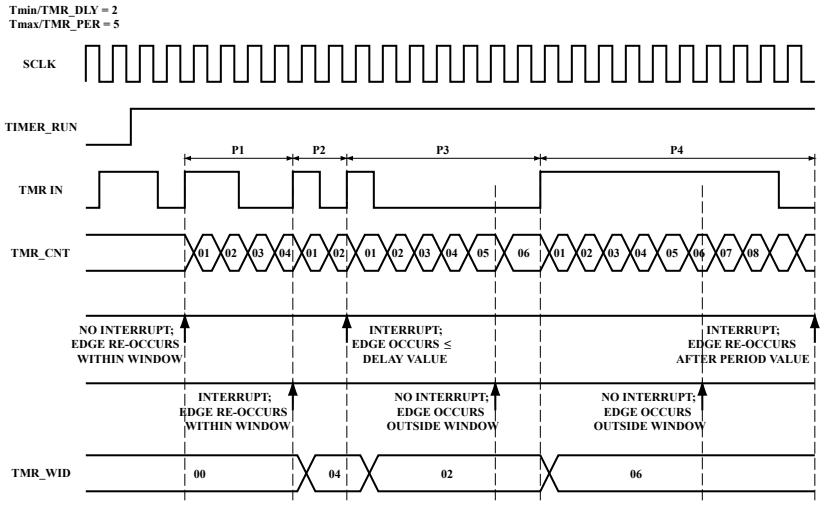


Figure 23-9: Watchdog Period Mode Timing

NOTE: SCLK in the ***Watchdog Period Mode Timing*** figure is SCLK.

To check only the upper limit on period (the t_{MAX} value, not the t_{MIN} value), program t_{MIN} as 0 or 1. For details, refer to the ***Windowed Watchdog Period Mode Interpretation*** table.

Table 23-9: Windowed Watchdog Period Mode Interpretation

Timer Delay	Timer Period	Incoming Pulse Width	IRQMODE=b#10	IRQMODE =b#11	Error Interrupt?
0 or 1	Anything ≥ 2	Pulse Period ≤ TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		Pulse Period > TMR_PER	No Interrupt	Interrupt when pulse period crosses Pmax (Period Register) value	No Error Interrupt
> 1 but ≤ Period -1	Anything ≥ 2	Pulse Period ≤ TMR_DLY	No Interrupt	Interrupt at deasserting edge of input signal	No Error Interrupt
		TMR_DLY < Pulse Period ≤TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		Pulse Period > TMR_PER	No Interrupt	Interrupt when pulse width exceeds Pmax (Period Register) value	No Error Interrupt

Table 23-9: Windowed Watchdog Period Mode Interpretation (Continued)

Timer Delay	Timer Period	Incoming Pulse Width	IRQMODE=b#10	IRQMODE =b#11	Error Interrupt?
≥ Period	-	Pulse Period < TMR_PER	Undefined	Undefined	No Error Interrupt
		Pulse Period ≥ TMR_PER	Undefined	Undefined	b#11 Error Type
-	0 or 1	-	Undefined	Undefined	b#10 Error Type

Pin Interrupt (PININT) Mode

In PININT mode, any active edges on either the TIMER_TMR[nn] pin or the (TIMER_ACI[nn]) pin can cause an edge-based interrupt, if enabled. (The timer uses the TIMER_TMR[n]_CFG.TINSEL register to select the pin). The event on the input pin can set the TIMER_DATA_ILAT.TMR[nn] bit and issue a system interrupt request. Program the TIMER_TMR[n]_CFG.PULSEHI bit to change active edge polarity.

Since the interrupt request is generated in the SCLK clock domain, the width of the input signal must be more than one SCLK period. Along with generating the interrupt request, the timer also generates a trigger pulse (configured using the TIMER_TRG_MSK register). Due to the configuration of polarity, glitches can cause the generation of an undesired interrupt request at the input. To avoid this problem, programs must ensure that interrupt requests are unmasked only after configuring the desired polarity.

External Clock (EXTCLK) Mode

The timer uses EXTCLK mode, sometimes referred to as the counter mode, to count external events (signal edges), on either the TIMER_TMR[nn] or input pin. The timer works as a counter clocked by an external source (the signal at the pin), which can be asynchronous to SCLK. The current count in the TIMER_TMR[n]_CNT register represents the number of leading-edge events detected. The TIMER_TMR[n]_PER register is programmed with the value of the maximum timer external count before stopping or issuing an interrupt request or trigger.

The TIMER_TMR[n]_CFG.PULSEHI bit determines the polarity of the leading edge on the input pin. The timer uses the TIMER_TMR[n]_CFG.TINSEL bit to select whether the event is counted on the TIMER_TMR[nn] or on the pin. The TIMER_STAT_ILAT.TMR[nn] and TIMER_ERR_TYPE bits are set if *one* of these conditions is met:

- TIMER_TMR[n]_CNT wraps around from 0xFFFF FFFF to 0
- The period = 0 at startup
- TIMER_TMR[n]_CNT register rolls over (from count = period to count = 0x1)

The TIMER_TMR[n]_WID and TIMER_TMR[n]_DLY registers are unused in this mode and must not be written.

The ***EXTCLK Mode Control Flow*** figure shows a flow diagram for EXTCLK mode.

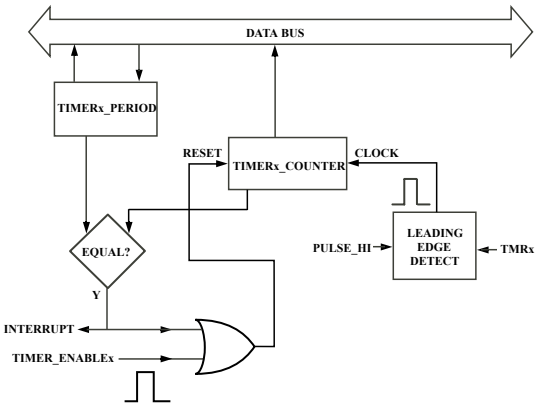


Figure 23-10: EXTCLK Mode Control Flow

The waveform applied to the input pin is not required to have a 50% duty cycle. The minimum input pulse low time, high time, and total period specifications are available in the product data sheet. Program the period to any value from 1 to (2^{32} – 1), inclusive.

After the timer has started, it resets the `TIMER_TMR[n]_CNT` register to 0x0 and then waits for the first leading edge on the input pin. This edge causes `TIMER_TMR[n]_CNT` to be incremented to the value 0x1, and every subsequent leading edge increments it by one. After the `TIMER_TMR[n]_CNT` register reaches the value programmed in the `TIMER_TMR[n]_PER` register, the corresponding `TIMER_DATA_ILAT` bit is set, and an interrupt and trigger are both generated (if enabled). The next leading-edge reloads the `TIMER_TMR[n]_CNT` register with 0x1, and the timer continues counting until it is disabled.

GP Timer Programming Concepts

Using the features, operating modes, and event control for the GP timer to their greatest potential requires an understanding of some GP timer-related concepts.

Setting Up Constantly Changing Timer Conditions

This task shows how to use different period, pulse width, and delay settings for each of the first three timer periods after the timer starts.

1. Program the first set of `TIMER_TMR[n]_PER`, `TIMER_TMR[n]_WID`, and `TIMER_TMR[n]_DLY` register values.
2. Enable the timer using the `TIMER_RUN` register.
3. Immediately program the second set of `TIMER_TMR[n]_PER`, `TIMER_TMR[n]_WID`, and `TIMER_TMR[n]_DLY` register values, as needed.
4. Wait for the first timer interrupt request.
5. Program the third set of `TIMER_TMR[n]_PER`, `TIMER_TMR[n]_WID`, and `TIMER_TMR[n]_DLY` register values.

Each new setting is then programmed when the preceding timer interrupt request is received.

Configuring, Enabling, and Disabling One or More Timers

1. Configure the relevant timers for the operating mode and other properties using the TIMER_TMR[n]_CFG register.
2. Write a 1 to the representative TIMER_RUN.TMR[nn] bit. Or use the TIMER_RUN_SET register to avoid disturbing the settings of other timers that are not going through configuration.

   The timer is enabled and operating.
3. To stop one or more timers, first program the TIMER_STOP_CFG register to determine whether to stop immediately or gracefully upon receiving a stop command.

   ADDITIONAL INFORMATION: PWMOUT modes are the only modes where a timer can be configured for graceful termination.
4. Write a 0 to the representative TIMER_RUN.TMR[nn] bits to stop the timer according to their TIMER_STOP_CFG settings. Alternately, write a 1 to the appropriate TIMER_RUN_CLR.TMR[nn] bits to avoid disturbing the settings of other timers that are not terminating.

   The timers stop.

Configuring Timer Data and Status Interrupts

1. Configure the TIMER_TMR[n]_CFG.IRQMODE bit field with the desired interrupt properties.
2. Unmask the interrupt source at the system event controller.
3. Set the TIMER_TMR[n]_CFG.IRQMODE field but leave the interrupt masked at the system level to poll the TIMER_DATA_ILAT.TMR[nn] bit of the timer without generating an interrupt.
4. Use the TIMER_STAT_IMSK register to generate interrupt requests by overflow or error conditions (incorrect programming values). The timer uses the TIMER_STAT_ILAT.TMR[nn] bits to report interrupt errors, when the timer status interrupt source is unmasked at the system event controller.
5. To poll the TIMER_STAT_ILAT.TMR[nn] bit of the timer without generating an interrupt, unmask the corresponding bit in the TIMER_STAT_IMSK register, but leave the interrupt masked at the system level.

Configuring the Timer as a Trigger Receiver

The timer can be configured to either start or stop or toggle between these two states on the input trigger pulse depending on the configuration of the TIMER_TMR[n]_CFG.TGLTRIG and TIMER_TMR[n]_CFG.RCVTRIG bits.

- If TIMER_TMR[n]_CFG.TGLTRIG bit =0 and TIMER_TMR[n]_CFG.RCVTRIG bit =1 then the trigger pulse starts timer, if it is stopped.

- If TIMER_TMR[n]_CFG.TGLTRIG bit =0 and TIMER_TMR[n]_CFG.RCVTRIG bit =0 then the trigger pulse stops timer, if it is running.
- When TIMER_TMR[n]_CFG.TGLTRIG bit =0, the trigger pulse has no effect when the timer is already in the requested state.

If TIMER_TMR[n]_CFG.TGLTRIG bit is 1, the trigger pulse starts the timer if it is stopped, or stops the timer if it is running. The TIMER_TMR[n]_CFG.RCVTRIG bit has no effect on trigger mechanism. In continuous PWMOUT mode, the timer stops gracefully or abruptly depending on the stop mechanism programmed in the TIMER_STOP_CFG_CLR register. In other modes, the timer stops immediately.

Ordered Trigger Toggle Mode

The timer can be configured to either start or stop the timer on the input trigger pulse depending on the configuration of the TIMER_TMR[n]_CFG.TGLTRIG and TIMER_TMR[n]_CFG.ORDTGLTRIG bits.

- If TIMER_TMR[n]_CFG.TGLTRIG bit is 1 and TIMER_TMR[n]_CFG.ORDTGLTRIG bit is 0, then the input trigger pulse on trigger receiver 0 or trigger receiver 1 starts the timer if it is stopped, or stops the timer if it is running.
- If TIMER_TMR[n]_CFG.TGLTRIG bit is 1 and TIMER_TMR[n]_CFG.ORDTGLTRIG bit is 1, then the input trigger pulse on trigger receiver 0 starts timer if it is stopped. If the timer is running, it stops when the input trigger pulse on trigger receiver 1 is detected.

  Trigger receiver 0 cannot toggle the timer from the run to halt state. Trigger receiver 1 cannot toggle the timer from the halt to run state.

NOTE: The timer state is not toggled by writes to the TIMER_RUN, TIMER_RUN_SET, and TIMER_RUN_CLR registers in this mode.

Using the Timer Broadcast Feature

The broadcast feature provides a means to update period, width, and delay registers simultaneously across more than one timer.

1. Enable the appropriate broadcast bits (TIMER_TMR[n]_CFG.BPEREN, TIMER_TMR[n]_CFG.BWIDEN are TIMER_TMR[n]_CFG.BDLYEN) for the timers involved in the broadcast. The use of these bits depends on which broadcast registers the timer uses (TIMER_BCAST_PER, TIMER_BCAST_WID, or TIMER_BCAST_DLY).
2. Program the TIMER_BCAST_PER register (for example), to broadcast the period setting across the multiple timers enabled.

   The enabled timers load their TIMER_TMR[n]_PER registers with the value specified in the TIMER_BCAST_PER register.
3. Repeat Step 2 as needed for the TIMER_BCAST_WID and TIMER_BCAST_DLY register settings.

Timer Illegal States

The following sections use these definitions:

- Startup. The first clock period during which the timer counter is running after the timer is started by writing the TIMER_RUN register.
- Rollover. The time when the current count in TIMER_TMR[n]_CNT matches the value in TIMER_TMR[n]_PER and the counter is reloaded with the value 1.
- Overflow. The timer counter was incremented instead of doing a rollover when it was holding the maximum count value of 0xFFFF FFFF. The counter does not have a large enough range to express the next greater value and so it erroneously loads a new value of 0x0000 0000.
- Unchanged. No new error.

When the TIMER_ERR_TYPE register is designated unchanged, it displays the previously reported error code orb# 00 when there has been no error since this timer was enabled.

When the TIMER_STAT_ILAT register is unchanged, it reads 0 when there has been no error or overflow since this timer was enabled. Or, it reads 0 if software has performed a W1C to clear any previous error. If software has not acknowledged a previous error, the TIMER_STAT_ILAT register reads 1. Software can read the TIMER_STAT_ILAT register to check for errors. If a particular bit of a timer is set in this register, software can then read the TIMER_ERR_TYPE register for more information. Once detected, software can W1C the appropriate TIMER_STAT_ILAT bit to acknowledge the error.

Read the following tables as:

- In mode __ at event __,
- if TIMER_TMR[n]_PER is __ and TIMER_TMR[n]_WID is __ and TIMER_TMR[n]_DLY is __,
- then TIMER_ERR_TYPE is __ and TIMER_STAT_ILAT is __.

Startup error conditions do not prevent the timer from starting. Similarly, overflow and rollover error conditions do not stop the timer. Illegal cases can cause unwanted behavior of the TIMER_TMR[nn] pins.

NOTE: For PININT mode, the timer does not use error functionality.

Continuous PWMOUT Mode

Table 23-10: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≤ 1	Anything other than period [*1]	Anything	Anything	b#10	Set

Table 23-10: Startup Event (Continued)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≥ 2	Anything including 0, excluding TMR_PER value	Anything including 0	≤ PERIOD	Unchanged	Unchanged
	Anything including 0	Anything including 0	> PERIOD	Unchanged *2(Detected at rollover)	Unchanged (Detected at rollover)
	Anything	Anything	> 2^{32} – 1	b#11	Set
	=Period	=0	=Period	No error	Unchanged (Detected at rollover)

*1 If delay = period, the #10 error is rolled to a #11 error.

*2 This case is not detected at startup, but when the timer counts to the complete period.

Table 23-11: Rollover Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≥ 1	Anything	Anything	Anything	b#10[timer running at SCLK] b#11 [timer running at ALT_CLKx]	Set
≥ 2	Anything including 0, excluding TMR_PER value	Anything including 0	≤PERIOD	Unchanged	Unchanged
	Anything including 0, excluding TMR_PER value	Anything >0	>PERIOD	b#11	Set
	Anything	Anything	> 2^{32}– 1	b#11	Set
	= Period*1	=0	=Period	b#11	Set
	>Period	=0	>Period	Unchanged	Unchanged

*1 In case of graceful termination, this error is not generated.

Table 23-12: Overflow Event (On TMR_PER Register Programming Error Only)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	Anything	Anything	Anything	b#01	Set

Single Pulse PWMOUT Mode

For single pulse PWMOUT mode, there are no rollover events.

Table 23-13: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_STAT_ILAT (if enabled)	TIMER_STAT_ILAT (if enabled)
N/A	Anything	== 0	Anything	b#11*1	Set
N/A	Anything including 0	≥1	≤ 2^{32}- 1	Unchanged	Unchanged
N/A	Anything including 0	≥1	> 2^{32}- 1	b#11	Set

*1 The timer does not run in this case.

Table 23-14: Overflow Event (On another error, such as DELAY + WIDTH ≥ 2^{32}- 1)

	TIMER_TMR[n]_DLY		TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY		TIMER_STAT_ILAT (if enabled)
Anything	Anything	Anything	Anything	b#01	Set

WIDCAP Mode

For WIDCAP mode, the TIMER_TMR[n]_PER and TIMER_TMR[n]_WID registers are read-only and the TIMER_TMR[n]_DLY register is not used. Therefore, no startup or rollover errors are possible.

Table 23-15: Overflow Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	N/A	Anything	N/A	b#01	Set

EXTCLK Mode

Table 23-16: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
=0	N/A	N/A	N/A	b#01	Set

Table 23-16: Startup Event (Continued)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≥1	N/A	N/A	N/A	Unchanged	Unchanged

Table 23-17: Rollover Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
=0	N/A	N/A	N/A	b#01	Set
≥1	N/A	N/A	N/A	Unchanged	Unchanged

Table 23-18: Overflow Event (On TMR_PER Register = 0 Only)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	N/A	N/A	N/A	b#01	Set

WATCHDOG Events

Table 23-19: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≤ Allowed MIN*1	Anything < PERIOD	N/A	N/A	b#01	Set
> Allowed MIN	Anything < PERIOD	N/A	N/A	Unchanged	Unchanged
> Allowed MIN	Anything ≥ PERIOD	Refer to WATCHDOG Mode tables			

*1 Allowed MIN=0 for width watchdog mode. Allowed MIN=1 for period watchdog mode.

Table 23-20: Rollover Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≤ Allowed MIN*1	Anything < PERIOD	N/A	N/A	b#01	Set

Table 23-20: Rollover Event (Continued)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
> Allowed MIN	Anything	N/A	N/A	Unchanged	Unchanged
> Allowed MIN	Anything ≥ PERIOD	Refer to WATCHDOG Mode tables			

*1 In case of graceful termination, this error is not generated.

Table 23-21: Overflow Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	Anything	N/A	N/A	b#01	Set

ADSP-2184x TIMER Register Descriptions

General-Purpose Timer Block (TIMER) contains the following registers.

Table 23-22: ADSP-2184x TIMER Register List

Name	Description
TIMER_BCAST_DLY	Broadcast Delay Register
TIMER_BCAST_PER	Broadcast Period Register
TIMER_BCAST_WID	Broadcast Width Register
TIMER_DATA_ILAT	Data Interrupt Latch Register
TIMER_DATA_IMSK	Data Interrupt Mask Register
TIMER_ERR_TYPE	Error Type Status Register
TIMER_RUN	Run Register
TIMER_RUN_CLR	Run Clear Register
TIMER_RUN_SET	Run Set Register
TIMER_STAT_ILAT	Status Interrupt Latch Register
TIMER_STAT_IMSK	Status Interrupt Mask Register
TIMER_STOP_CFG	Stop Configuration Register
TIMER_STOP_CFG_CLR	Stop Configuration Clear Register
TIMER_STOP_CFG_SET	Stop Configuration Set Register
TIMER_TMR[n]_CFG	Timer n Configuration Register

Table 23-22: ADSP-2184x TIMER Register List (Continued)

Name	Description
TIMER_TMR[n]_CNT	Timer n Counter Register
TIMER_TMR[n]_DLY	Timer n Delay Register
TIMER_TMR[n]_PER	Timer n Period Register
TIMER_TMR[n]_WID	Timer n Width Register
TIMER_TRG_IE	Trigger Receiver Enable Register
TIMER_TRG_MSK	Trigger Generator Mask Register

Broadcast Delay Register

For timers with TIMER_TMR[n]_CFG.BDLYEN enabled, a write to the TIMER_BCAST_DLY register concurrently updates the delay (TIMER_TMR[n]_DLY) registers of only those timers. A read of the TIMER_BCAST_DLY register returns 0x00000000, and no bus error is generated. To read back a written value, read that TMR's TIMER_TMR[n]_DLY register.

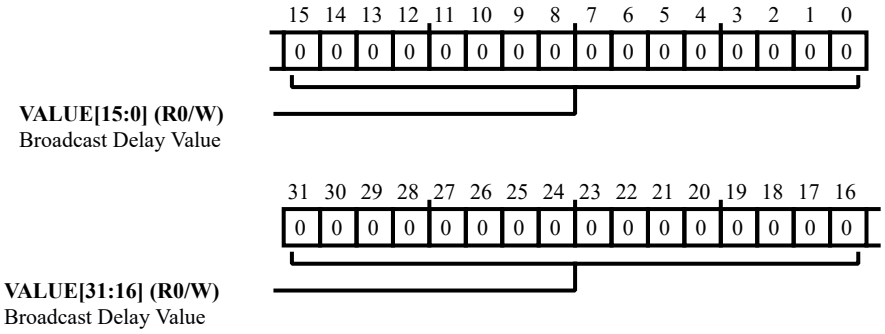


Figure 23-11: TIMER_BCAST_DLY Register Diagram

Table 23-23: TIMER_BCAST_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	VALUE	Broadcast Delay Value. A write to the TIMER_BCAST_DLY.VALUE bit field concurrently updates the delay (TIMER_TMR[n]_DLY) registers of only those timers. A read of the TIMER_BCAST_DLY.VALUE bit field returns 0x0000 0000, and no bus error is generated.

Broadcast Period Register

For timers with TIMER_TMR[n]_CFG.BPEREN enabled, a write to the TIMER_BCAST_PER register concurrently updates the period (TIMER_TMR[n]_PER) registers of only those timers. A read of TIMER_BCAST_PER returns 0x00000000, and no bus error is generated. To read back a written value, read that TMR's TIMER_TMR[n]_PER register.

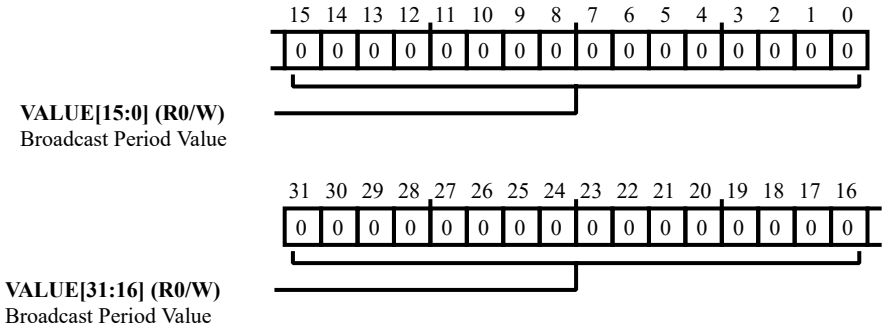


Figure 23-12: TIMER_BCAST_PER Register Diagram

Table 23-24: TIMER_BCAST_PER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	VALUE	Broadcast Period Value. A write to the TIMER_BCAST_PER.VALUE bit field concurrently updates the period (TIMER_TMR[n]_PER) registers of only those timers. A read of the TIMER_BCAST_PER.VALUE bit fields returns 0x0000 0000, and no bus error is generated.

Broadcast Width Register

For timers with TIMER_TMR[n]_CFG.BWIDEN enabled, a write to the TIMER_BCAST_WID register concurrently updates the width (TIMER_TMR[n]_WID) registers of only those timers. A read of the TIMER_BCAST_WID register returns 0x00000000, and no bus error is generated. To read back a written value, read that TMR's TIMER_TMR[n]_WID register.

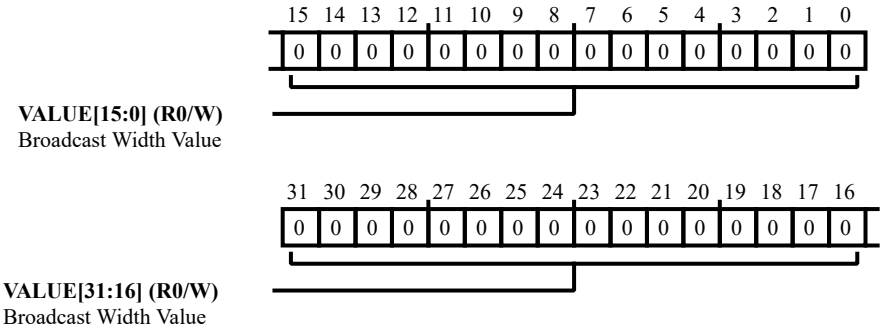


Figure 23-13: TIMER_BCAST_WID Register Diagram

Table 23-25: TIMER_BCAST_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	VALUE	Broadcast Width Value. A write to the TIMER_BCAST_WID.VALUE bit field concurrently updates the width (TIMER_TMR[n]_WID) registers of only those timers. A read of the TIMER_BCAST_WID.VALUE bit field returns 0x0000 0000, and no bus error is generated.

Data Interrupt Latch Register

The TIMER_DATA_ILAT holds the latched interrupt status for interrupt requests that have been unmasked (enabled) by the TIMER_DATA_IMSK register and generated according to the conditions selected by the TIMER_TMR[n]_CFG.IRQMODE bits. If a bit in TIMER_DATA_ILAT is already set and the corresponding interrupt is masked in TIMER_DATA_IMSK, the latch holds its old value, leaving the interrupt request asserted until it is reset by software with a W1C operation.

Note that interrupt service routines (ISRs) should clear the appropriate bits in TIMER_DATA_ILAT before returning from the ISR, to ensure that the interrupt is not re-issued. To make sure that no timer event is missed, the latch should be reset at the very beginning of the ISR when in EXTCLK mode.

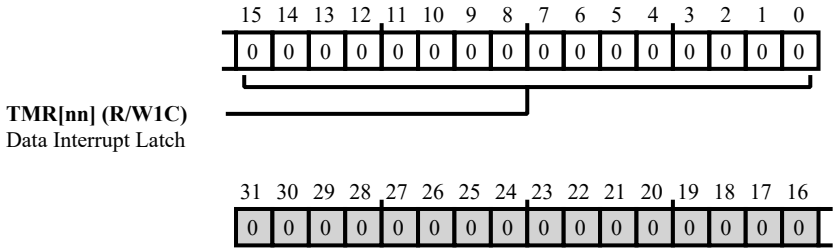


Figure 23-14: TIMER_DATA_ILAT Register Diagram

Table 23-26: TIMER_DATA_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W1C)	TMR[nn]	Data Interrupt Latch. For all TIMER_DATA_ILAT.TMR[nn] bits, status of =0 indicates no interrupt is latched, and status of =1 indicates a latched interrupt (indicating an unmasked interrupt request from a timer with a condition matching the one selected with corresponding TIMER_TMR[n]_CFG.IRQMODE bit has occurred).

Data Interrupt Mask Register

Each timer may generate a unique processor data interrupt request signal. The `TIMER_DATA_IMSK` register contains an interrupt mask for these requests, masking (disabling) or unmasking (enabling) the interrupts as programmed. The reset value of the `TIMER_DATA_IMSK` register is 0xFFFF, masking these interrupts after reset.

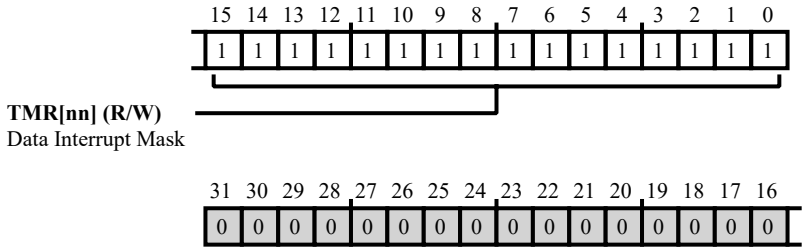


Figure 23-15: TIMER_DATA_IMSK Register Diagram

Table 23-27: TIMER_DATA_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Data Interrupt Mask. For all `TIMER_DATA_IMSK.TMR[nn]` bits, write =0 unmasks (enables) the corresponding data interrupt request, and write =1 masks (disables) the corresponding data interrupt request.

Error Type Status Register

The TIMER_ERR_TYPE register contains error type status bits for each timer. These bits indicate the type of error (if any) in a running timer. This register is read-only. These status bits are cleared at reset and when a particular timer is enabled.

Each time an error request interrupt is latched in the TIMER_STAT_ILAT register, the corresponding TERRx bits in the TIMER_ERR_TYPE register are loaded with a code that identifies the type of error that was detected. This status value is held until the next error or until a particular timer is restarted. No bus error is generated if a write is performed on the TIMER_ERR_TYPE register.

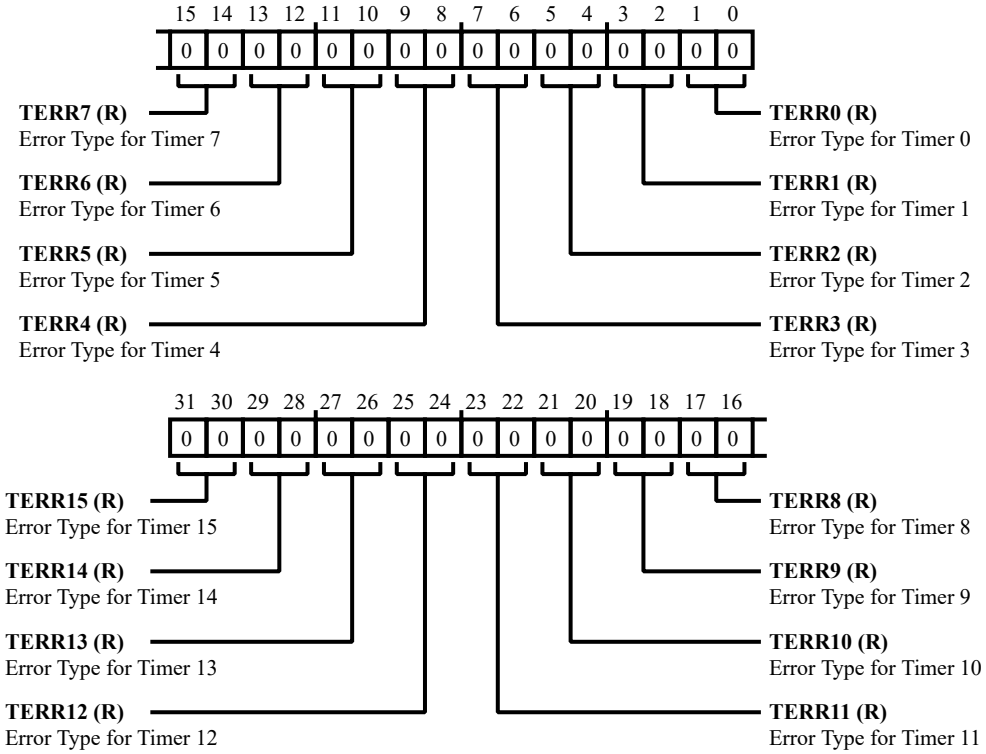


Figure 23-16: TIMER_ERR_TYPE Register Diagram

Table 23-28: TIMER_ERR_TYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	TERR15	Error Type for Timer 15.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Table 23-28: TIMER_ERR_TYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/NW)	TERR14	Error Type for Timer 14.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
27:26 (R/NW)	TERR13	Error Type for Timer 13.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
25:24 (R/NW)	TERR12	Error Type for Timer 12.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
23:22 (R/NW)	TERR11	Error Type for Timer 11.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
21:20 (R/NW)	TERR10	Error Type for Timer 10.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
19:18 (R/NW)	TERR9	Error Type for Timer 9.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Table 23-28: TIMER_ERR_TYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/NW)	TERR8	Error Type for Timer 8.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
15:14 (R/NW)	TERR7	Error Type for Timer 7.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
13:12 (R/NW)	TERR6	Error Type for Timer 6.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
11:10 (R/NW)	TERR5	Error Type for Timer 5.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
9:8 (R/NW)	TERR4	Error Type for Timer 4.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
7:6 (R/NW)	TERR3	Error Type for Timer 3.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Table 23-28: TIMER_ERR_TYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	TERR2	Error Type for Timer 2.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
3:2 (R/NW)	TERR1	Error Type for Timer 1.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
1:0 (R/NW)	TERR0	Error Type for Timer 0.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Run Register

The TIMER_RUN allows all timers to be enabled simultaneously, permitting them to run synchronously. For each timer, there is a single start/stop control bit. Writing a 1 to this bit starts the corresponding timer; writing a 0 stops the timer with mechanism specified in the timer stop configuration TIMER_STOP_CFG register.

The start/stop control bits can be set/reset individually or in any combination. While starting or stopping one particular timer directly with this register, software must perform a read-modify write, so the bits corresponding to other timers remain unchanged. To avoid this need, software can use the TIMER_RUN_CLR register.

Reading the TIMER_RUN register shows the start status for the corresponding timer. A 1 indicates that the timer is running.

If a timer is in run state (corresponding run bit is =1), a software write of 1 in this bit does not have any effect on the timer state. The write does not result in restarting the timer.

Note that the TIMER_RUN register is not used in PININT mode. PININT mode starts as soon as the TIMER_TMR[n]_CFG.TMODE bits are set to 111.

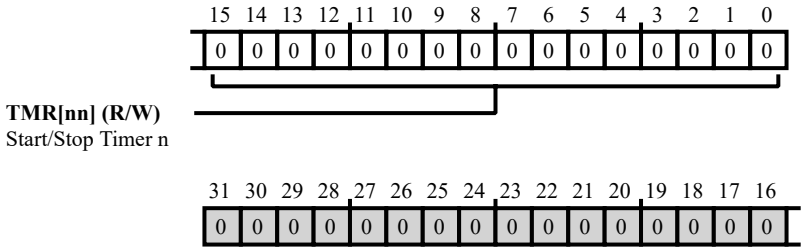


Figure 23-17: TIMER_RUN Register Diagram

Table 23-29: TIMER_RUN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Start/Stop Timer n. For all TIMER_RUN.TMR[nn] bits, write =0 for stop, and write =1 for start. Read =1 when timer is running.

Run Clear Register

The TIMER_RUN_CLR register is an alias register, providing a mechanism to clear a specific start/stop bit in the TIMER_RUN register without affecting other bits in TIMER_RUN. To stop a particular timer, software must write a 1 into the corresponding TIMER_RUN_CLR bit. Writing a 0 has no effect. Because TIMER_RUN_CLR is a write-only register, the result of any write to this register must be checked by reading the TIMER_RUN register. A read of the TIMER_RUN_CLR returns 0x0000.

Note that the stopping mechanism of a timer may be abrupt or graceful (after completion of current waveform period) depending on the selection in the TIMER_STOP_CFG register.

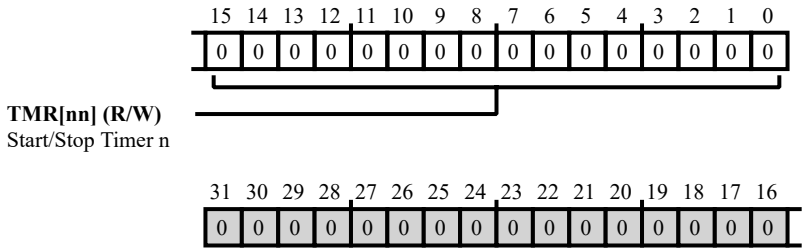


Figure 23-18: TIMER_RUN_CLR Register Diagram

Table 23-30: TIMER_RUN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1C)	TMR[nn]	RUN Clear Alias. For all TIMER_RUN_CLR.TMR[nn] bits, write =0 has no effect, and write =1 for stop (clearing the corresponding in start/stop bit in the TIMER_RUN register). Using TIMER_RUN_CLR to clear start/stop bits permits stopping specific timers without influencing run status of other timers.

Run Set Register

The TIMER_RUN_SET register is an alias register, providing a mechanism to set a specific start/stop bit in the TIMER_RUN register without affecting other bits in TIMER_RUN. To start a particular timer, software must write a 1 into the corresponding TIMER_RUN_SET bit. Writing a zero has no effect. For an example, to start timer 3 without affecting any other timer, write 0x0008 into TIMER_RUN_SET. Because TIMER_RUN_SET is a write-only register, the result of any write to this register must be checked by reading the TIMER_RUN register. A read of the TIMER_RUN_SET returns 0x0000.

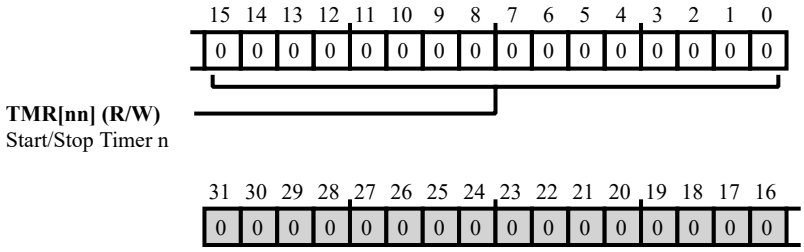


Figure 23-19: TIMER_RUN_SET Register Diagram

Table 23-31: TIMER_RUN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1S)	TMR[nn]	RUN Set Alias. For all TIMER_RUN_SET.TMR[nn] bits, write =0 has no effect, and write =1 for start (setting the corresponding start/stop bit in the TIMER_RUN register). Using TIMER_RUN_SET to set start/stop bits permits starting specific timers without influencing the run status of other timers.

Status Interrupt Latch Register

The TIMER_STAT_ILAT holds the latched interrupt status for error interrupt requests, indicating a timer overflow condition or indicating that prohibited programming has occurred for a timer. These interrupt status bits are sticky and are W1C. The bits in the TIMER_STAT_ILAT register provide information regarding each timer interrupt request source.

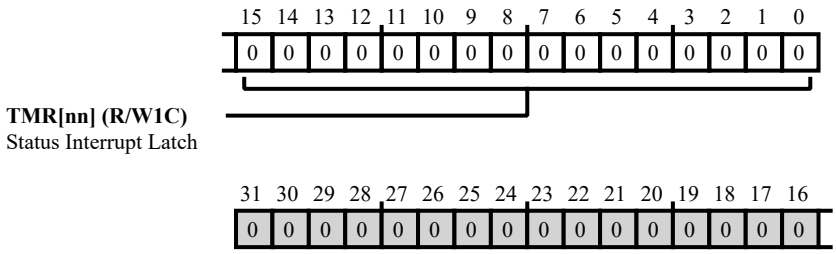


Figure 23-20: TIMER_STAT_ILAT Register Diagram

Table 23-32: TIMER_STAT_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W1C)	TMR[nn]	Status Interrupt Latch. For all TIMER_STAT_ILAT.TMR[nn] bits, status of 0 indicates no error interrupt request is latched, and status of 1 indicates a timer counter overflow or programming error interrupt request is latched.

Status Interrupt Mask Register

While each timer may generate a status interrupt request, these requests are OR'ed to generate a single status interrupt signal to the system event controller. The TIMER_STAT_IMSK register contains an interrupt mask for these requests, masking (disabling) or unmasking (enabling) the interrupts as programmed. The reset value of the TIMER_STAT_IMSK register is 0xFFFF, masking these interrupts after reset.

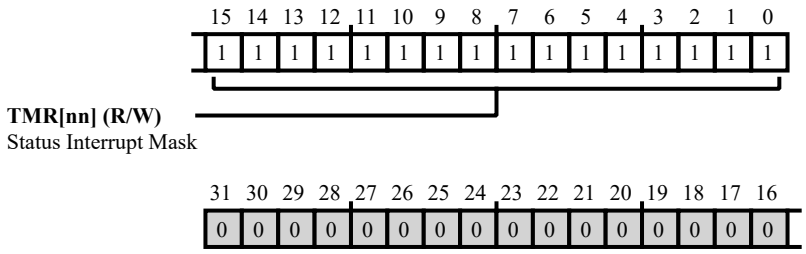


Figure 23-21: TIMER_STAT_IMSK Register Diagram

Table 23-33: TIMER_STAT_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Status Interrupt Mask. For all TIMER_STAT_IMSK.TMR[nn] bits, write =0 unmasks (enables) the corresponding status interrupt request, and write =1 masks (disables) the corresponding status interrupt request.

Stop Configuration Register

The TIMER_STOP_CFG register selects the stop mode for each timer. Timers may be stopped abruptly (immediate halt - all modes) or gracefully in PWMOUT modes (single pulse and continuous). The halt is achieved through either a write =0 to the corresponding bit in TIMER_RUN or a write =1 to the corresponding bit in TIMER_RUN_CLR. A read of TIMER_STOP_CFG returns the last value written.

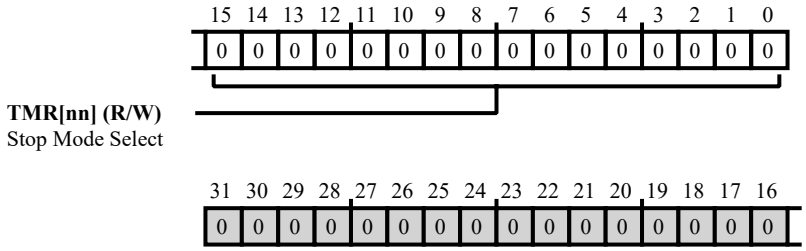


Figure 23-22: TIMER_STOP_CFG Register Diagram

Table 23-34: TIMER_STOP_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Stop Mode Select. For all TIMER_STOP_CFG.TMR[nn] bits, write =0 for graceful termination (PWMOUT modes only), and write =1 for abrupt (immediate halt) on stop.

Stop Configuration Clear Register

This is an alias register, providing a mechanism to clear a specific bit in the TIMER_STOP_CFG register without affecting other bits in TIMER_STOP_CFG. To clear a bit in TIMER_STOP_CFG, software must write a 1 to the corresponding bit of TIMER_STOP_CFG_CLR register. Writing a zero has no effect. Because the TIMER_STOP_CFG_CLR register is a write-only register, the result of any write to this register must be checked by reading the TIMER_STOP_CFG register. A read of the TIMER_STOP_CFG_CLR register returns 0x0000.

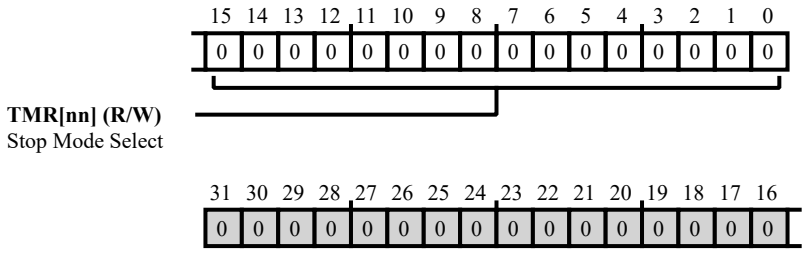


Figure 23-23: TIMER_STOP_CFG_CLR Register Diagram

Table 23-35: TIMER_STOP_CFG_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1C)	TMR[nn]	STOP_CFG Clear Alias. For all TIMER_STOP_CFG_CLR.TMR[nn] bits, write =0 has no effect, and write =1 for graceful stop in PWMOUT modes (clearing the corresponding stop mode select bit in the TIMER_STOP_CFG register). Using TIMER_STOP_CFG_CLR to clear stop mode bits permits configuring specific timers without influencing the stop mode configuration of other timers.

Stop Configuration Set Register

This is an alias register, providing a mechanism to set a specific bit in the TIMER_STOP_CFG register without affecting other bits in TIMER_STOP_CFG. To set a bit in the TIMER_STOP_CFG register, software must write a 1 to the corresponding bit of the TIMER_STOP_CFG_SET register. Writing a zero has no effect. Because the TIMER_STOP_CFG_SET register is a write-only register, the result of any write to this register must be checked by reading the TIMER_STOP_CFG register. A read of the TIMER_STOP_CFG_SET register returns 0x0000.

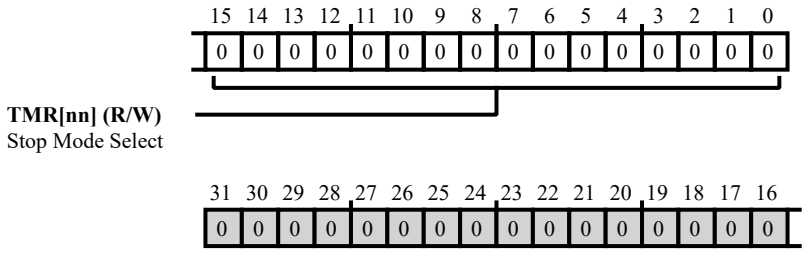


Figure 23-24: TIMER_STOP_CFG_SET Register Diagram

Table 23-36: TIMER_STOP_CFG_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1S)	TMR[nn]	STOP_CFG Set Alias. For all TIMER_STOP_CFG_SET.TMR[nn] bits, write =0 has no effect, and write =1 for abrupt stop (setting the corresponding stop mode select bit in the TIMER_STOP_CFG register). Using TIMER_STOP_CFG_SET to set stop mode bits permits configuring specific timers without influencing the stop mode configuration of other timers.

Timer n Configuration Register

Each timer has a TIMER_TMR[n]_CFG register that specifies its operating mode. Only write to a TIMER_TMR[n]_CFG register when the corresponding timer is not running.

After disabling a timer operating in PWMOUT mode, verify that the timer has stopped running by checking the start/stop status of the timer in the TIMER_RUN register before writing to the timer's TIMER_TMR[n]_CFG register.

Note that a timer's TIMER_TMR[n]_CFG register may be read at any time.

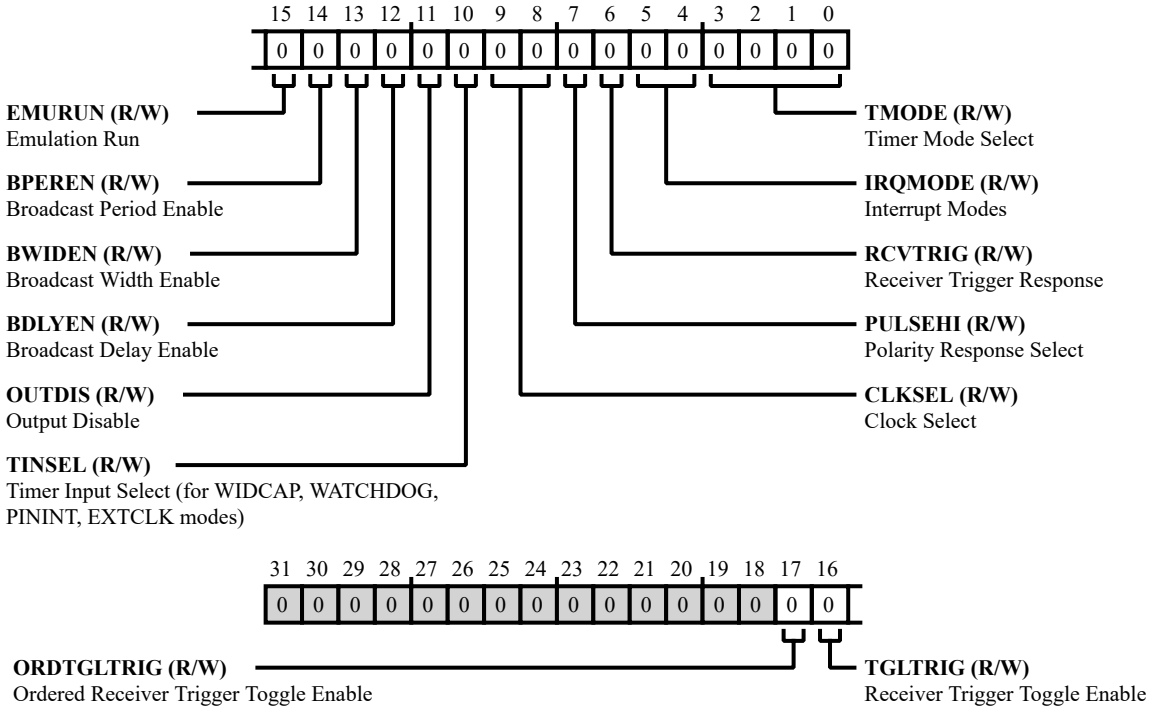


Figure 23-25: TIMER_TMR[n]_CFG Register Diagram

Table 23-37: TIMER_TMR[n]_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	ORDTGLTRIG	Ordered Receiver Trigger Toggle Enable. The TIMER_TMR[n]_CFG.ORDTGLTRIG bit controls ordered receive trigger toggle mode. The timer(receiver) state is not toggled by writes to the TIMER_RUN, TIMER_RUN_SET and TIMER_RUN_CLR registers when this bit is enabled.	
		0	No effect
		1	Trigger pulse on trigger receiver 0 starts the timer if it is in the halt state. Trigger pulse in trigger receiver 1 stops the timer if it is running.
16 (R/W)	TGLTRIG	Receiver Trigger Toggle Enable. The TIMER_TMR[n]_CFG.TGLTRIG bit stops the timer if it is running and starts the timer if it is halted (in the stop state). If the TIMER_TMR[n]_CFG.TGLTRIG bit is set, then the setting of the TIMER_TMR[n]_CFG.RCVTRIG bit is ignored.	
		0	Receiver trigger response depends on RCVTRIG bit
		1	Receiver trigger toggles timer state
15 (R/W)	EMURUN	Emulation Run. The TIMER_TMR[n]_CFG.EMURUN bit causes the timer to run (count) during emulation.	
		0	Stop Timer During Emulation
		1	Run Timer During Emulation
14 (R/W)	BPEREN	Broadcast Period Enable. The TIMER_TMR[n]_CFG.BPEREN bit enables updates to the TIMER_TMR[n]_PER register simultaneously across more than one timer.	
		0	Disable Broadcast to PER Register
		1	Enable Broadcast to PER Register
13 (R/W)	BWIDEN	Broadcast Width Enable. The TIMER_TMR[n]_CFG.BWIDEN bit enables updates to the TIMER_TMR[n]_WID register simultaneously across more than one timer.	
		0	Disable Broadcast to WID Register
		1	Enable Broadcast to WID Register

Table 23-37: TIMER_TMR[n]_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	BDLYEN	Broadcast Delay Enable. The `TIMER_TMR[n]_CFG.BDLYEN` bit enables updates to the `TIMER_TMR[n]_DLY` register simultaneously across more than one timer.	
		0	Disable Broadcast to DLY Register
		1	Enable Broadcast to DLY Register
11 (R/W)	OUTDIS	Output Disable. The `TIMER_TMR[n]_CFG.OUTDIS` bit enables or disables the timer pin output buffer.	
		0	Enable TMR Pin Output Buffer
		1	Disable TMR Pin Output Buffer
10 (R/W)	TINSEL	Timer Input Select (for WIDCAP, WATCHDOG, PININT, EXTCLK modes).	
		0	Use TMR Pin Input
		1	Use TMR Alternate Capture Input
9:8 (R/W)	CLKSEL	Clock Select. The `TIMER_TMR[n]_CFG.CLKSEL` bit field selects the TIMER clock to use.	
		0	Use SCLK
		1	Use TMR_ALT_CLK0 as TMR Clock
		3	Use TMR_ALT_CLK1 as TMR Clock
7 (R/W)	PULSEHI	Polarity Response Select. The `TIMER_TMR[n]_CFG.PULSEHI` bit defines specific behaviors of the timer based on the operating mode. For more information, see the specific operating mode in the Programming Concepts section.	
		0	Negative Response or Pulse. A Negative Edge Response or Negative Action Pulse on the TMR pin.
		1	Positive Response or Pulse. A Positive Edge Response or Positive Action Pulse on the TMR pin.
6 (R/W)	RCVTRIG	Receiver Trigger Response. The `TIMER_TMR[n]_CFG.RCVTRIG` bit controls the trigger response. The trigger pulse has no effect (to stop or start the timer) if the timer is already in the requested state.	
		0	Pulse Stops Timer if it is Running
		1	Pulse Starts Timer if it is Stopped

Table 23-37: TIMER_TMR[n]_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/W)	IRQMODE	Interrupt Modes. The TIMER_TMR[n]_CFG.IRQMODE bit field selects the interrupt request mode. Note that any mismatched combination of the TIMER_TMR[n]_CFG.IRQMODE and the TIMER_TMR[n]_CFG.TMODE bits results in no interrupt being generated. In WIDCAP modes, the position of the interrupt is controlled with the TIMER_TMR[n]_CFG.TMODE bit, and the TIMER_TMR[n]_CFG.IRQMODE bit is ignored. Whenever an interrupt is generated, a trigger generator pulse is also occurs, when enabled in the TIMER_TRG_MSK register.	
		0	Active Edge Mode. The timer generates an interrupt at every active edge. The active edge polarity depends on the state of the TIMER_TMR[n]_CFG.PULSEHI bit. Valid for PININT mode only.
		1	Delay Expired Mode. The timer generates an interrupt when the TIMER_TMR[n]_CNT value reaches the value in the TIMER_TMR[n]_DLY register. This mode is valid for all PWMOUT modes.
		2	Width Plus Delay Expired Mode. The timer generates an interrupt when the TIMER_TMR[n]_CNT value reaches the value in the TIMER_TMR[n]_WID register plus the value in the TIMER_TMR[n]_DLY register. (PWMOUT modes only) The timer generates an interrupt if the deasserting edge is within the specified window. (WATCHDOG modes only)
		3	Period Expired Mode. The timer generates an interrupt when the TIMER_TMR[n]_CNT value reaches the value in the TIMER_TMR[n]_PER register. (Continuous PWMOUT and EXTCLK modes only) The timer generates an interrupt if the deasserting edge is outside the specified window.(WATCHDOG modes only)
3:0 (R/W)	TMODE	Timer Mode Select. The TIMER_TMR[n]_CFG.TMODE bit field selects the operating mode of each timer.	
		0-7	Idle Mode
		8	Period Watchdog Mode
		9	Width Watchdog Mode

Table 23-37: TIMER_TMR[n]_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Measurement Report at Asserting Edge of Waveform
		11	Measurement Report at Deasserting Edge of Waveform
		12	Continuous PWMOUT Mode
		13	Single Pulse PWMOUT Mode
		14	EXTCLK Mode
		15	PININT (pin interrupt) Mode

Timer n Counter Register

The TIMER_TMR[n]_CNT register holds the current timer count. After enabling, the count is re-initialized to either 0x0 or 0x1, depending on the configuration and mode. The TIMER_TMR[n]_CNT register is read-only and may be read at any time (whether the timer is running or stopped). Reading the TIMER_TMR[n]_CNT register returns an atomic 32-bit value.

Depending on the timer operation mode, the counter increment can be clocked by a number of sources, including SCLK, the TMR or alternate capture input pins, ACLK[n]. The counter retains its value after the timer is disabled.

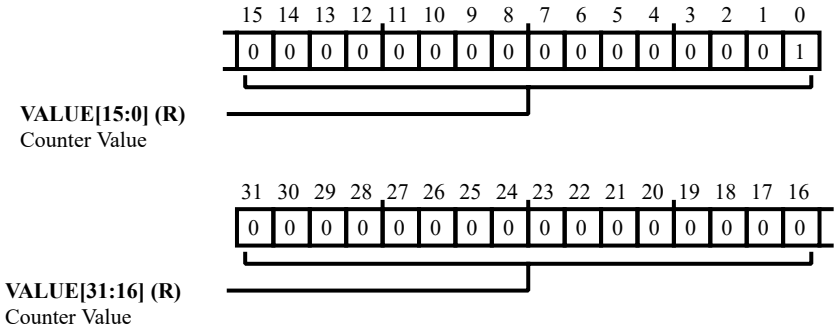


Figure 23-26: TIMER_TMR[n]_CNT Register Diagram

Table 23-38: TIMER_TMR[n]_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Counter Value. The TIMER_TMR[n]_CNT.VALUE bit field holds the current timer count.

Timer n Delay Register

The TIMER_TMR[n]_DLY register holds the delay value for the corresponding timer. This register's use is based on the selected timer mode.

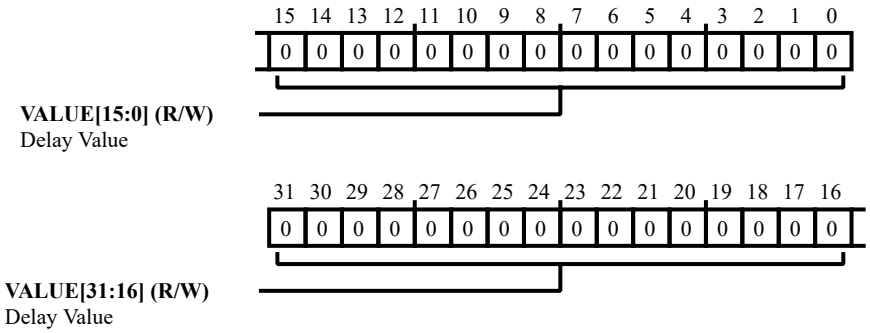


Figure 23-27: TIMER_TMR[n]_DLY Register Diagram

Table 23-39: TIMER_TMR[n]_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Delay Value. The TIMER_TMR[n]_DLY.VALUE bit field holds the delay value for the corresponding timer.

Timer n Period Register

The TIMER_TMR[n]_PER register holds the period value for the corresponding timer. This register's use is based on the selected timer mode.

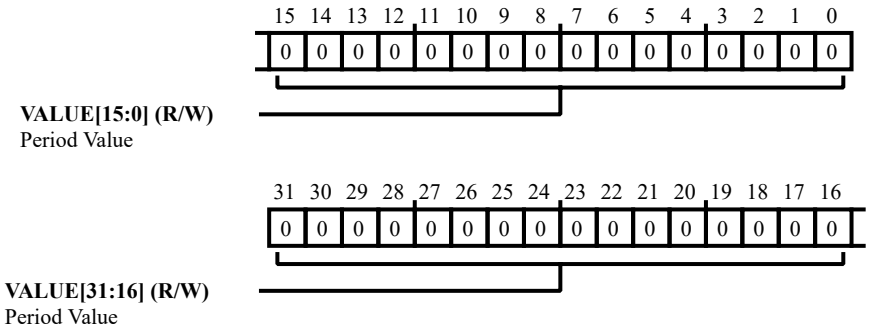


Figure 23-28: TIMER_TMR[n]_PER Register Diagram

Table 23-40: TIMER_TMR[n]_PER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Period Value. The TIMER_TMR[n]_PER.VALUE bit field holds the period value for the corresponding timer.

Timer n Width Register

The TIMER_TMR[n]_WID register holds the width value for the corresponding timer. This register's use is based on the selected timer mode.

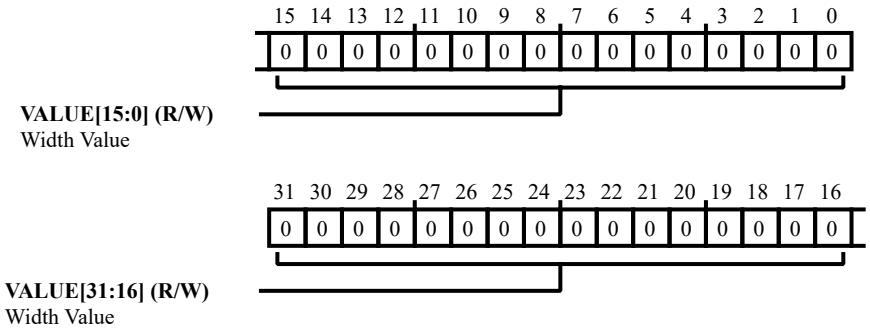


Figure 23-29: TIMER_TMR[n]_WID Register Diagram

Table 23-41: TIMER_TMR[n]_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Width Value. The TIMER_TMR[n]_WID.VALUE bit field holds the width value for the corresponding timer.

Trigger Receiver Enable Register

As a trigger receiver, each timer can generate a unique data trigger pulse signal. The TIMER_TRG_IE contains trigger input enable bits for these signals, disabling or enabling the triggers as programmed. The reset value of the TIMER_TRG_IE register is 0xFFFF, masking these triggers after reset.

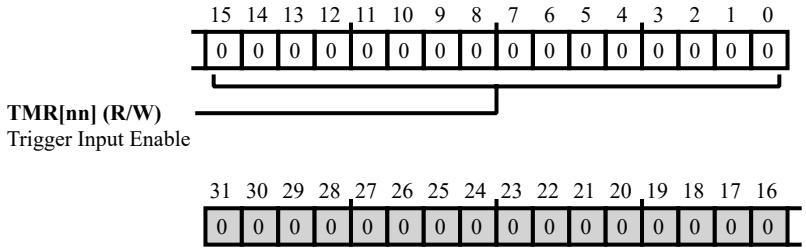


Figure 23-30: TIMER_TRG_IE Register Diagram

Table 23-42: TIMER_TRG_IE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Trigger Input Enable. For all TIMER_TRG_IE.TMR[nn] bits, write =0 disables the corresponding trigger input, and write =1 enables the corresponding trigger input.

Trigger Generator Mask Register

As a trigger generator, each timer can generate a unique data trigger pulse signal. The TIMER_TRG_MSK register contains a trigger mask for these outputs, masking (disabling) or unmasking (enabling) the triggers as programmed. The reset value of the TIMER_TRG_MSK register is 0xFFFF, masking these triggers after reset.

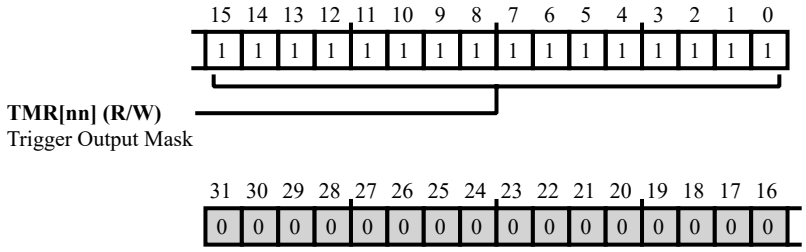


Figure 23-31: TIMER_TRG_MSK Register Diagram

Table 23-43: TIMER_TRG_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Trigger Output Mask. For all TIMER_TRG_MSK.TMR[nn] bits, write =0 unmasks (enables) the corresponding data trigger output, and write =1 masks (disables) the corresponding data trigger output.

24 Pulse-Width Modulator (PWM)

The pulse-width modulator (PWM) module is a flexible and programmable waveform generator. With minimal CPU intervention, the PWM peripheral can generate complex waveforms for:

- Motor control
- Pulse-coded modulation (PCM)
- Digital-to-analog conversion (DAC)
- Power switching
- Power conversion

The PWM module has four PWM pairs capable of three-phase PWM generation for source inverters for AC induction and DC brushless motors.

PWM Features

Each PWM generation unit features:

- 16-bit center-based PWM generation unit
- Programmable PWM pulse width
- Single or double update modes
- Programmable dead time and switching frequency
- Two's-complement implementation that permits smooth transition to full-on and full-off states
- Dedicated asynchronous PWM trigger shutdown signal
- Debounce filter option on trip inputs that allows the system programmer to filter out short transient signals
- Expanded clocking system with the PCG and the fractional PLL that can be fed to the PWM using the CDU multiplexer
- The PWM trip 1b input can be conditionally sourced from the TRU generator output. The PWM sync input can also be conditionally sourced using the TRU generator output

Functional Description

The following sections provide details on the functionality of the PWM.

- Architectural Concepts
- Timer Units
- Channel Timing Control Unit
- Output Disable and Cross-Over Modes
- Sync Operation Modes

ADSP-2184x PWM Register List

The Pulse-Width Modulator unit (PWM) includes multiple timers (providing period flexibility) and channels (providing mode, interrupt, and pulse shape flexibility), permitting a wide variety of PWM output options for motor control and other applications. A set of registers governs PWM operations. For more information on PWM functionality, see the PWM register descriptions.

Table 24-1: ADSP-2184x PWM Register List

Name	Description
PWM_ACTL	Channel A Control Register
PWM_AH0	Channel A-High Duty-0 Register
PWM_AH1	Channel A-High Duty-1 Register
PWM_AH_DUTY0	Channel A-High Full Duty0 Register
PWM_AH_DUTY1	Channel A-High Full Duty1 Register
PWM_AL0	Channel A-Low Duty-0 Register
PWM_AL1	Channel A-Low Duty-1 Register
PWM_AL_DUTY0	Channel A-Low Full Duty0 Register
PWM_AL_DUTY1	Channel A-Low Full Duty1 Register
PWM_BCTL	Channel B Control Register
PWM_BH0	Channel B-High Duty-0 Register
PWM_BH1	Channel B-High Duty-1 Register
PWM_BH_DUTY0	Channel B-High Full Duty0 Register
PWM_BH_DUTY1	Channel B-High Full Duty1 Register
PWM_BL0	Channel B-Low Duty-0 Register
PWM_BL1	Channel B-Low Duty-1 Register
PWM_BL_DUTY0	Channel B-Low Full Duty0 Register
PWM_BL_DUTY1	Channel B-Low Full Duty1 Register

Table 24-1: ADSP-2184x PWM Register List (Continued)

Name	Description
PWM_CCTL	Channel C Control Register
PWM_CH0	Channel C-High Pulse Duty Register 0
PWM_CH1	Channel C-High Pulse Duty Register 1
PWM_CHANCFG	Channel Configuration Register
PWM_CHA_DT	Channel A Dead-time Register
PWM_CHB_DT	Channel B Dead-time Register
PWM_CHC_DT	Channel C Dead-time Register
PWM_CHD_DT	Channel D Dead-time Register
PWM_CHOPCFG	Chop Configuration Register
PWM_CH_DUTY0	Channel C-High Full Duty0 Register
PWM_CH_DUTY1	Channel C-High Full Duty1 Register
PWM_CL0	Channel C-Low Pulse Duty Register 0
PWM_CL1	Channel C-Low Duty-1 Register
PWM_CL_DUTY0	Channel C-Low Full Duty0 Register
PWM_CL_DUTY1	Channel C-Low Full Duty1 Register
PWM_CTL	Control Register
PWM_DCTL	Channel D Control Register
PWM_DH0	Channel D-High Duty-0 Register
PWM_DH1	Channel D-High Pulse Duty Register 1
PWM_DH_DUTY0	Channel D-High Full Duty0 Register
PWM_DH_DUTY1	Channel D-High Full Duty1 Register
PWM_DL0	Channel D-Low Pulse Duty Register 0
PWM_DL1	Channel D-Low Pulse Duty Register 1
PWM_DLYA	Channel A Delay Register
PWM_DLYB	Channel B Delay Register
PWM_DLYC	Channel C Delay Register
PWM_DLYD	Channel D Delay Register
PWM_DL_DUTY0	Channel D-Low Full Duty0 Register
PWM_DL_DUTY1	Channel D-Low Full Duty1 Register
PWM_ILAT	Interrupt Latch Register
PWM_IMSK	Interrupt Mask Register

Table 24-1: ADSP-2184x PWM Register List (Continued)

Name	Description
PWM_STAT	Status Register
PWM_SWTRIP	Software Trip Register
PWM_SYNC_WID	Sync Pulse Width Register
PWM_TM0	Timer 0 Period Register
PWM_TM1	Timer 1 Period Register
PWM_TM2	Timer 2 Period Register
PWM_TM3	Timer 3 Period Register
PWM_TM4	Timer 4 Period Register
PWM_TRIPCFG	Trip Configuration Register
PWM_TRIP_POL	Trip Polarity Register

ADSP-2184x PWM Interrupt List

Table 24-2: ADSP-2184x PWM Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
227	PWM0_TRIP	PWM0 Trip	Level	
228	PWM0_SYNC	PWM0 PWMTMR Grouped	Edge	

ADSP-2184x PWM Trigger List

Table 24-3: ADSP-2184x PWM Trigger List Generators

Trigger ID	Name	Description	Sensitivity
None			

Table 24-4: ADSP-2184x PWM Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
145	PWM0_TRIP	PWM0 PWM Trip1B Trigger In	Pulse
146	PWM0_SYNC	PWM0 PWMTMR Grouped	Pulse

PWM Definitions

The following definitions are helpful when using the PWM module.

Chopping

Used to simplify the design of isolated gate drive circuits for PWM inverters. If using a transformer coupled power device gate drive amplifier, then the active PWM signal must be chopped at a high frequency.

Dead-Time

A short delay introduced between turning off one PWM signal (for example, AH) and turning on the complementary signal (for example, AL). This short time delay permits turning off a power switch (AH in this case) to completely recover its blocking capability before the complementary switch is turned on. This time delay prevents a potentially destructive short-circuit condition from developing across the dc link capacitor of a typical voltage source inverter.

Duty Cycle

The proportion of on time to the regular interval or period of time (expressed in percent, 100% being fully on). A low duty cycle corresponds to low power, because the power is off for most of the time.

Switching Frequency

The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load.

Architectural Concepts

A clock, whose period is t_{PCLK}, drives the PWM controller. The PWM generator produces four pairs (four high-side and four low-side) of PWM signals on the eight PWM output pins. Each high and low pair signal constitutes a channel. For example, the `PWM_AL` and `PWM_AH` signals make up channel A, and the `PWM_BL` and `PWM_BH` signals make up channel B, and so on.

Each pair of channel outputs references either a main timer or an independent timer. These timers operate on a switching frequency determined by the `PWM_TM0` through `PWM_TM4` registers. There are two duty registers for every PWM output. The registers enable generation of symmetrical or asymmetrical waveforms. The waveforms produce lower harmonic distortion in three-phase PWM inverters, with minimal CPU intervention.

Block Diagram

The ***PWM Block Diagram*** figure shows a block diagram that represents the main functional blocks of the PWM controller.

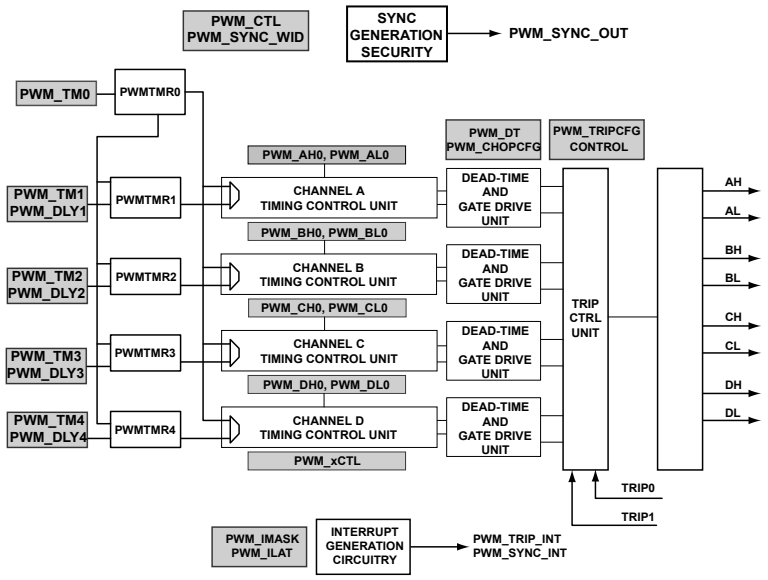


Figure 24-1: PWM Block Diagram

The following list describes the primary diagram blocks.

- Each pair of PWM signals references either the main timer or the independent timer.
- PWMTMR0 is the main timer and can trigger the delayed start of the other timers.
- Timing control units, one for each channel, together form the core of the PWM. The unit generates the required complex waveforms on the high-side and low-side outputs for the respective channel.
- Dead-time insertion occurs after generating the ideal PWM output pair.
- The gate drive unit generates the high-frequency chopping signal and then mixes it with the required PWM output signals.
- The PWM shutdown and interrupt controller manage the various PWM shutdown modes for the timing unit and generates the needed interrupt signals.
- The PWM sync pulse control unit generates the internal `PWM_SYNC` pulse and also controls whether the external `PWM_SYNC` input pulse is used.

Enhanced PWM Clocking

To support low jitter and high precision audio clock-based PWM generation for audio applications, an expanded clocking scheme is used for the Precision Clock Generator (PCG) and Fractional PLL (Frac-N PLL) clocks. The clocks input the signals to the PWM clock using the chip CDU multiplexer on the CDU clock output, which is connected to the main PWM input clock PCLK. The default clock for CDU clock output remains the system clock, SYSCLK.

The alternate clock sources are 16 PCG clocks: eight from each of the individual digital audio interface instances (DAI0 and DAI), from each PCG output, and one Frac-N PLL clock output. The 16 PCG clock sources are multiplexed and the resulting output is again multiplexed with the

Frac-N PLL clock output (FPLL_OUT) using a 2:1 multiplexer. The multiplexer control bits are derived from the MISCREG_CLK_MUX_SEL.PCG_CLKSELFMISCREG_CLK_MUX_SEL.PCG_CLKSEL and MISCREG_CLK_MUX_SEL.PCG_CLKSELF bit fields.

Synchronizers between SYSCLK and PCLK are provided for all system-connected PWM signals using MMRG, SEC, TRU and CTI. The *Enhanced PWM Clocking Diagram* illustrates the enhanced clocking scheme.

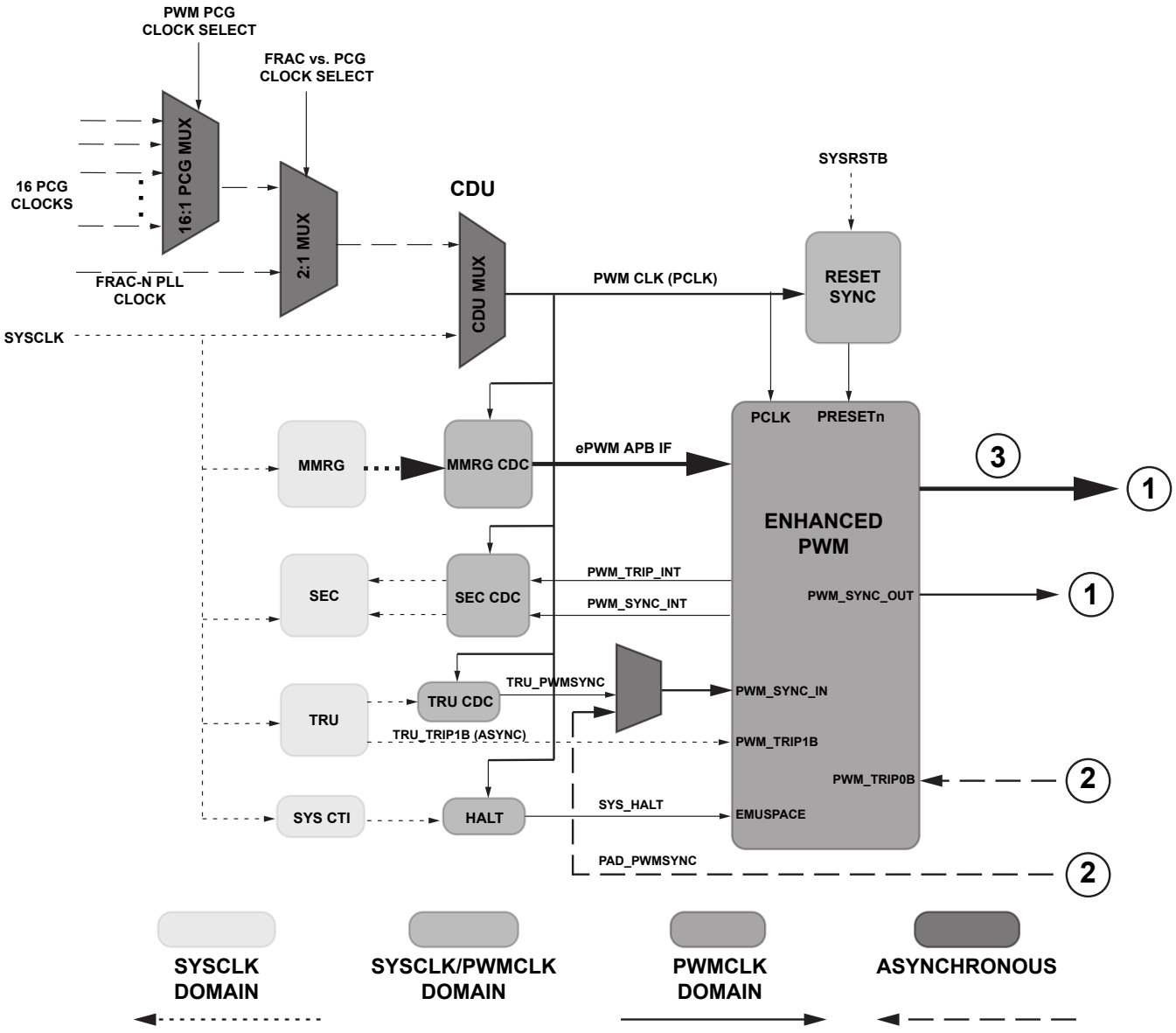


Figure 24-2: Enhanced PWM Clocking Diagram

1	To PINMUX and PADS (Combo path)
2	From PADS (Asynchronous)
3	PWM CHANNELS (xH, xL, x = A, B, C, and D)

CAUTION: Engineers should not configure the PWM_TMx [Timer (0-4) Period Register] to less than two. This configuration results in PWM waveform periods that exceed 125 MHz, having a minimum period of 2 ns, and a total period of 2*TMx (2 * 2 * 2 ns = 8 ns). The processor has a pad limitation of 125 MHz.

Timer Units

Five timers make up the time base for the PWM module. The main timer, PWMTMR0 operates at a switching frequency determined by the period register PWM_TM0. The four remaining timers (PWMTMR1 through PWMTMR4) can operate at independent switching frequencies determined by their respective registers.

The respective time registers (PWM_TM1 through PWM_TM4) can be programmed to work at a multiple of the main timer frequency. In this case, the PWM_DLYA through PWM_DLYD registers control the lead-lag phase of a given timer based on the main timer PWMTMR0.

NOTE: The delayed operation of a timer requires one of the following:

- The register value of the timer must be equal to the PWM_TM0 register value.
- The PWM_TM0 value must be an integer multiple of each register of the timer. Non-integer multiples are not allowed.

PWM Timer Period (PWM_TM) Registers

The 16-bit read/write PWM period registers (PWM_TM0 through PWM_TM4) control the PWM switching frequency. The fundamental timing unit of the PWM controller is t_{PCLK}. Therefore, the time increment (t_{PCLK}) is 10 ns for a 100-MHz system clock (PCLK) frequency, f_{PCLK}. The value written to the register of a timer is effectively the number of t_{PCLK} clock increments in one half of a PWM period. The following equation describes the required timer register value as a function of the desired PWM switching frequency (f_{PWM}):

$$PWM_TM = f_{PCLK}/2 \times f_{PWM}$$

Therefore, the PWM switching period (T_s) is:

$$T_s = 2 \times PWM_TM \times t_{PCLK}$$

For example, for an f_{PCLK} of 100 MHz and a desired PWM switching frequency (f_{PWM}) of 10 kHz (T_s = 100 ms), the correct value to load into the timer register is:

$$PWM_TM = 100 \times 10^6 \div 2 \times 10 \times 10^3 = 5000$$

The largest value that can be written to the 16-bit timer register is 0xFFFF = 65,535. For an f_{PCLK} of 100 MHz, this value corresponds to a minimum PWM switching frequency of:

$$f_{PWM(min)} = 100 \times 10^6 \div 2 \text{ x } 65535 = 762 \text{ Hz}$$

NOTE: Timer register values of 0 and 1 are not defined. Do not use these values when the PWM outputs or PWM is enabled.

Timer Unit Operation

The PWM timers are up-down counters, and they operate on the peripheral clock with a period of t_{CK}. The period of the PWM timer is divided into two halves. In the first half, the timer roughly counts down from PWM_TMx/2 to –PWM_TMx/2. During this half, the PWM_STAT.TMR0PHASE through PWM_STAT.TMR4PHASE bits are held at 0. In the second half of the period, the timer roughly counts up from –PWM_TMx/2 to PWM_TMx/2. The PWM_STAT.TMR0PHASE through PWM_STAT.TMR4PHASE bits indicates a 1 during this half.

The actual partition of the periods varies slightly between odd and even values of the half-period, in the PWM_TM1 - PWM_TM4 registers.

If a timer register value is odd, for example 11, then that timer loads +5 at the beginning of the period. The timer counts down from +5 to –5 in the first half, reloads –5 at the midpoint and counts up from –5 to +5 in the second half. The reload values at the period and mid-period boundaries are the same as the previous count. The timer counts 2 × 11 half-periods = 22 total counts in the entire period as shown in the ***Operation of Timer for Odd Value of PWM_TM*** figure.

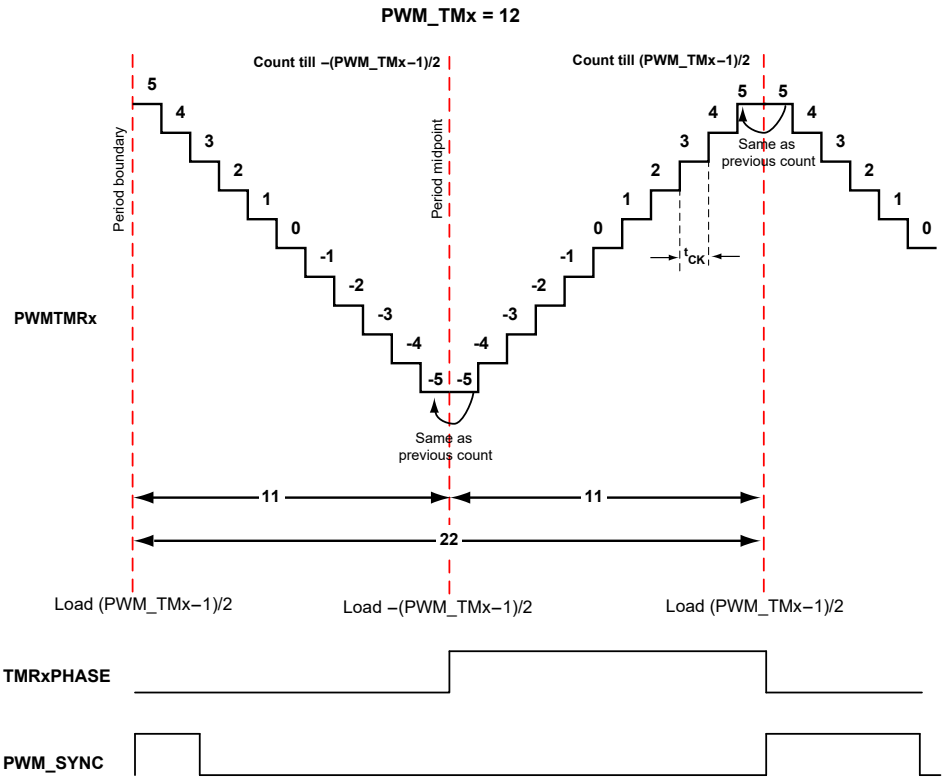


Figure 24-3: Operation of Timer for Odd Value of PWM_TM

When the timer register value is even, for example 12, then that timer loads +5 at the beginning of the period. The timer counts from +5 to –6 in the first half, reloads –5 at the midpoint and counts up from –5 to +6 in the second half. The reload values at the period and mid-period boundaries are different from the previous count. It counts 2 × 12 half-periods = 24 total counts in the entire period as shown in the ***Operation of Timer for Even Value of PWM_TM*** figure.

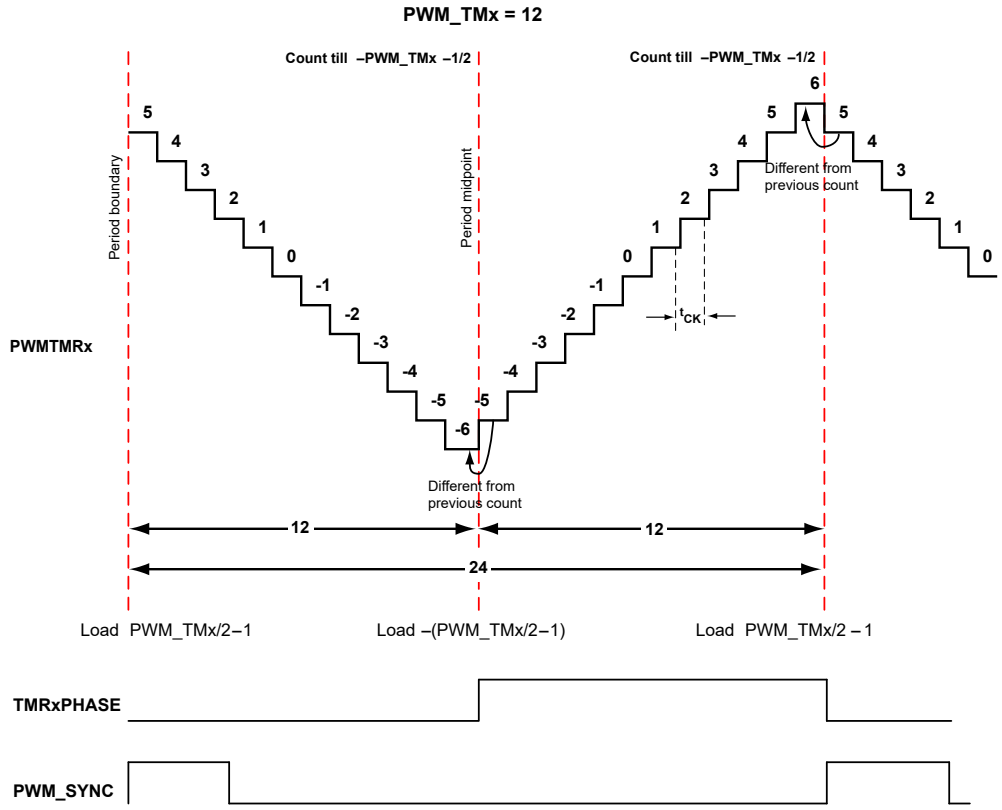


Figure 24-4: Operation of Timer for Even Value of PWM_TM

NOTE: In the operation discussed in this section, double-buffering of all channel registers and the timer registers takes place at the period boundary of the respective timers.

Phase Offset Control

The PWM timers (PWMTMR1 through PWMTMR4) can operate with a programmable phase lag relative to the main timer, PWMTMR0. To implement phase offset for a channel, use the counter-registers for channel delay (`PWM_DLYA` - `PWM_DLYD`) with the PWMTMR0 and set the `PWM_CTL.DLYAEN` bit to l.

Phase offset works as follows.

1. If phase lag is used for channel A (and channel A uses PWMTMR1 to generate a duty cycle), when PWMTMR0 reaches its period boundary, it triggers the `PWM_DLYA` register. The register counts out the number PCLK cycles that are equal to the value programmed in the `PWM_DLYA` register.
2. At the end of this count, the `PWM_DLYA` register sends out a trigger to PWMTMR1. It receives a synchronization pulse in every period of PWMTMR0 at a point delayed from its period boundary by the value in the `PWM_DLYA` register.

For more information on how channels can reference different timers for their outputs, see Channel Timing Control Unit.

NOTE: Satisfy the following conditions when using this feature on timer y for channel Y relative to PWMTMRx.

- Program the PWM_DLY[n] register to a value less than 2 × PWM_TM[n].
- PWM_TM0 = N × PWM_TM[n], where N is an integer.

The function of PWM_TM[n] (PWMTMR1 in the example) differs in cases where PWM_TM0 = PWM_TM1 (Case 1) to cases where PWM_TM0 = N × PWM_TM1 (Case 2). The following examples describe both cases.

Case 1: PWM_TM0 = PWM_TMy

When PWM_TM0 = PWM_TMy, PWMTMRy restarts its period after receiving the synchronization pulse from the channel delay register (PWM_DLY[n]). If the trigger from the PWM_DLY[n] register is late, PWMTMRy holds its count until the trigger occurs. If the trigger is a bit early, PWMTMRy reloads without regard to whether it has completed its current period. As a result, PWMTMRy resyncs with PWMTMR0 with the phase lag programmed in the PWM_DLYA register in every one of its periods.

In this case, the expiration of the delay registers (PWM_DLY[n]) is the period boundary of PWMTMRy. Now, all the double buffered registers related to the given channel update (except the delay registers which are double buffered at the period boundary of PWMTMR0).

The *Phase Offset Control Using DELAY* figure shows an example where:

- PWM_TM0, PWM_TM1, and PWM_TM2 are programmed with the same value.
- PWM_DLYA and PWM_DLYB are programmed with values DELAY1 and DELAY2 respectively, such that DELAY2 > DELAY1.
- The outputs of Channel A are referenced to PWMTMR1. The outputs of channel B are referenced to PWMTMR2.

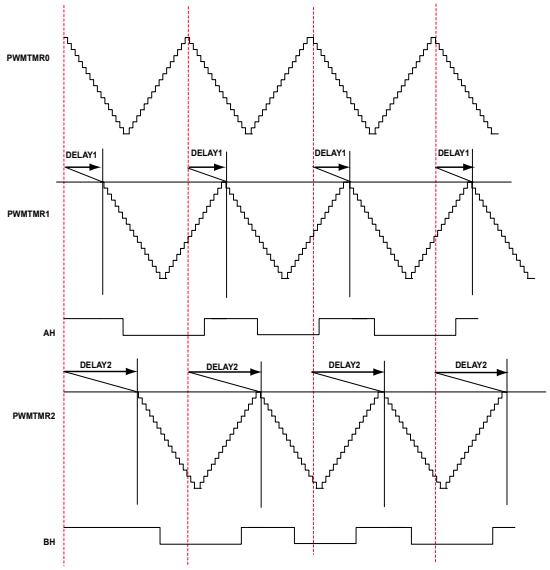


Figure 24-5: Phase Offset Control Using DELAY

The delay registers are double buffered and the new value of DELAY reloads at the period boundary of PWMTMR0. The two options exist when the new value is different from the older one. The behavior of

PWMTMRy in both these cases is discussed. The ***Impact of New DELAY Value on Timer Count for Equal Timer Periods*** figure shows the behavior in the two cases. It is assumed that channel B references its outputs to PWMTMR0 and channel A references its outputs to PWMTMR1.

1. The new delay value is higher than the previous value. Here the corresponding PWMTMRy allows more than one time period between consecutive triggers from the channel delay (`PWM_DLYA` - `PWM_DLYD`) registers. In this case, after reaching its period boundary, PWMTMRy holds its count at the period boundary and waits for the trigger from the channel delay register. The ***Impact of New DELAY Value on Timer Count for Equal Timer Periods*** figure shows case A functionality.
2. The next delay value programmed is smaller than the previous value. Here, the corresponding PWMTMRy allows only less than one time period between consecutive triggers from the channel delay register. Though the trigger comes earlier in this case, before PWMTMRy has counted out one full period, it reloads and starts its period again. The ***Impact of New DELAY Value on Timer Count for Equal Timer Periods*** figure shows case B functionality.

Therefore, PWMTMR1 waits and obeys a synchronization pulse from the `PWM_DLYA` register in every one of its periods.

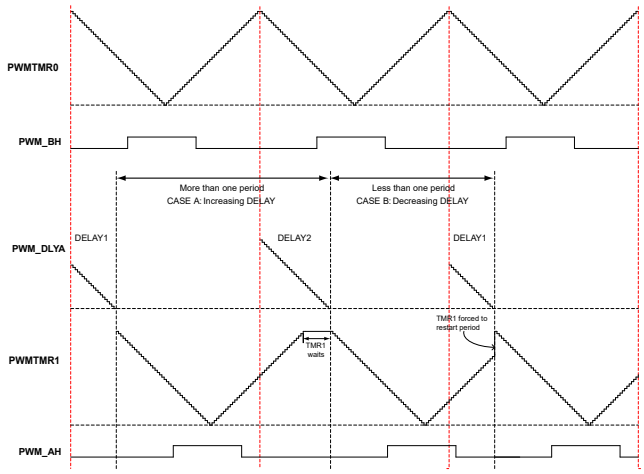


Figure 24-6: Impact of New DELAY Value on Timer Count for Equal Timer Periods

Case 2: PWM_TM0 = N x PWM_TMy

In this case, within a single period of PWMTMR0 a program can fit multiple periods (N) of PWMTMRy. Additionally, the channel delay register is triggered only once every N periods of PWMTMRy.

The operation is as follows: Every Nth period of PWMTMRy, PWMTMRy expects a synchronization pulse from the `PWM_DLY[n]` register. When this register counts out that period and the trigger has not yet arrived, PWMTMRy waits at the end of the period for the trigger. PWMTMRy starts counting down once the trigger arrives. If the trigger comes earlier, PWMTMRy restarts immediately without waiting to complete the period count.

In the intervening periods, PWMTMRy operates independently. As the period ends, PWMTMRy reloads and starts the next period without intervention from the channel delay register.

The *Impact of DELAY Value Change for the Multiple Timer Periods* shows an example with N = 2. PWMTMRy syncs up with PWMTMR0 every second period, and is free running across every odd period boundary.

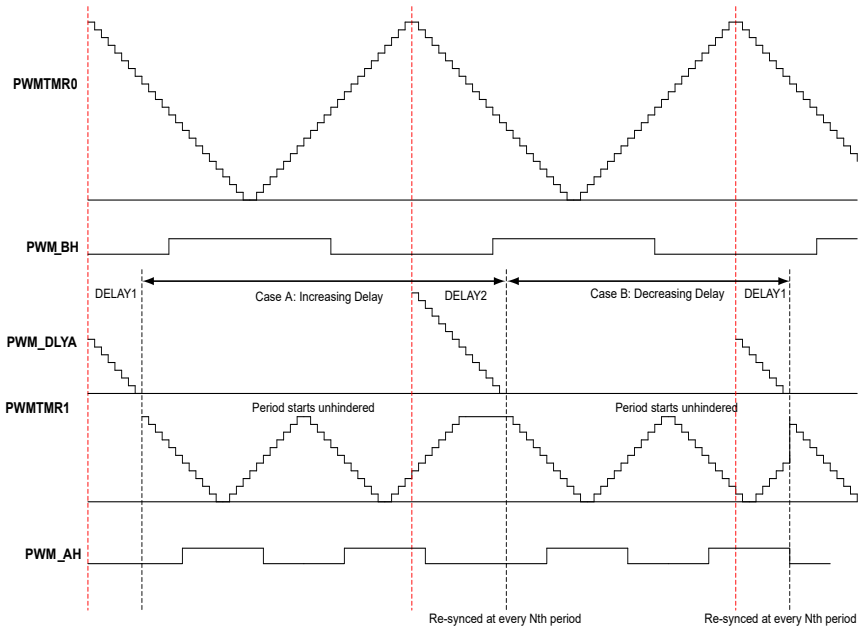


Figure 24-7: Impact of DELAY Value Change for Multiple Timer Periods

Channel Timing Control Unit

The channel timing control unit is the core of the PWM. There are four separate channels, each channel controlling a pair of output signals – the high-side output and the low-side output.

Channel Control

The `PWM_CHANCFG` register controls the static configuration of all the channels and is initialized once before the beginning of a PWM operation.

NOTE: The `PWM_CHANCFG` register is not double buffered. Do not change the contents of the register once the PWM is enabled.

Each channel works with a reference timer base. The time base can be either the main timer PWMTMR0 or the appropriate PWMTMRx. Configure the time base with the `PWM_CHANCFG.REFTMRA` bit field as follows.

- Channel A works with PWMTMR0 or PWMTMR1
- Channel B works with PWMTMR0 or PWMTMR2
- Channel C works with PWMTMR0 or PWMTMR3
- Channel D works with PWMTMR0 or PWMTMR4

The double-buffered channel control registers (`PWM_ACTL` through `PWM_DCTL`) contain bits that control the dynamic pulse behavior of the channel outputs. These registers have bits that enable or disable outputs and select the pulse position of outputs (explained in the following section).

Pulse Positioning and Duty Cycle Registers

The PWM uses the PULSEMODEHI and PULSEMODELO bit fields of the PWM_ACTL - PWM_DCTL registers to define the region within the timer period where the output pulses are positioned.

- When the PWM_CHANCFG.MODELSC bit is 0, the PWM uses the PULSEMODEHI field to specify the pulse positioning for both the high-side and low-side outputs of the channel.
- When the bit is 1, the PWM uses PWM_ACTL.PULSEMODELO to define the pulse positioning for the low-side output of the channel. It uses the PWM_ACTL.PULSEMODEHI to define the pulse positioning for the high-side output of the channel.

Each channel output has two duty-cycle registers: PWM_AH0 and PWM_AH1 for the high-side output, and PWM_AL0 and PWM_AL1 for the low-side output. These registers determine the width of the output pulses. When the PWM_CHANCFG.MODELSC bit is 0, the high-side duty-cycle registers are used to determine the width of the output pulse for the low side. The duty cycle range that can be programmed into these registers is between – PWM_TM[n]/2 and + PWM_TM[n]/2 , when ignoring dead time.

When including dead time for channel A, for PULSEMODEs 00 and 01, the programmed duty cycle is modified. The range is limited between the values [–PWM_TM[n]/2 + PWM_CHA_DT] and [+PWM_TM[n]/2 + PWM_CHA_DT] considering the high-side output. For PULSEMODEs 10 and 11, the high-side duty cycle registers range is limited between values [PWM_TM[n]/2 + PWM_CHA_DT] and [–PWM_TM[n]/2 – PWM_CHA_DT].

The following section explains dead time in detail.

- Switching Dead Time (PWM_DT) Register

NOTE: Values programmed into these registers that fall outside these limits result in over or under modulation.

Duty Cycle and Pulse Positioning Control

The PWM_ACTL.PULSEMODEHI and PWM_ACTL.PULSEMODELO fields control how the duty cycle registers modify the waveform of the high and low-side outputs. (The PWM_ACTL.PULSEMODEHI and PWM_ACTL.PULSEMODELO fields are referred to as *pulse mode* in the subsequent discussion.)

- Pulse mode = 00 – Produce a symmetrical pulse waveform around the center of the PWM period. In this mode, PWM uses only one of the duty cycle registers for an output. For example, for the AH output, PWM uses only the PWM_AH0 register. In this mode, the values in the duty cycle registers are scaled such that a value of 0 produces a 50% duty cycle.
- Pulse mode = 01 – Produce an asymmetrical pulse waveform around the center of the PWM period. In this mode, PWM uses both duty cycle registers. For example, for the PWM_AH output, PWM uses the PWM_AH0 and PWM_AH1 registers. In this mode, if the PWM_AH1 register is programmed with the same value as the PWM_AH0 register, the output is identical to the output when pulse mode =00.
- Pulse mode = 10 or 11 – Produce pulse waveforms either on the first half or the second half of the PWM period respectively. PWM uses both PWM_AH0 and PWM_AH1 registers.

  Pulse mode = 10. If the low side works from the low-side duty-cycle registers, strictly adhere to the condition PWM_AL0> PWM_AL1.

In pulse mode = 11. If the low side works from the low-side duty-cycle registers, strictly adhere to the condition PWM_AL0< PWM_AL1.

The ***Pulse Positioning Modes*** figure shows the pulse positioning modes as previously described for PWM_AH. In the figure, DUTY0 is the value in the PWM_AL0 register and DUTY1 is the value in the PWM_AH1 register. The step signal, count, indicates the output of the timer for channel A. In the example, the signal is configured as active high and dead time is zero.

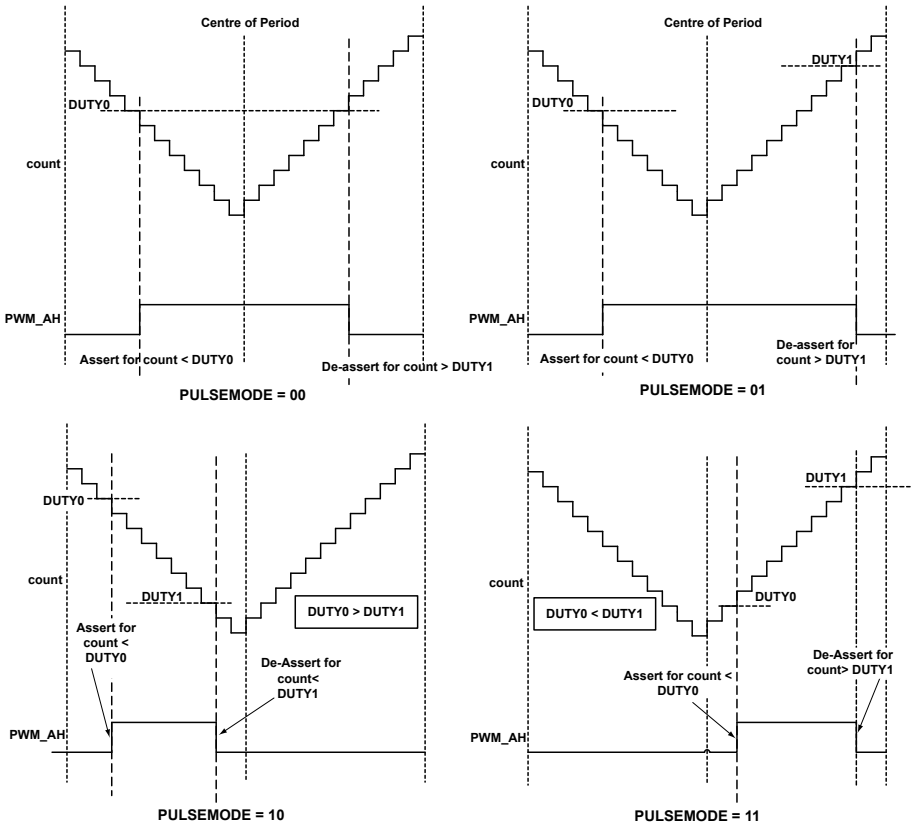


Figure 24-8: Pulse Positioning Modes

Channel Low Side Output Dependent Operation Mode and Dead Time

The low-side output waveform can be programmed to depend on the waveform of the high-side output or to be independent. The PWM uses the PWM_CHANCFG.MODELSC bit to control this functionality.

For example, channel A produces the high-side output PWM_AH and the low-side output PWM_AL. When the PWM_CHANCFG.MODELSC bit =0, the low-side output is also generated using the high-side duty-cycle registers for pulse width, the PWM_ACTL.PULSEMODEHI bits for pulse positioning and the PWM_CHANCFG.POLAH bit for polarity. If the PWM_CH[n]_DT register is 0, the low-side output is an inverted version of the high-side output.

When the PWM_CH[n]_DTregister is programmed with a non-zero value, both the high-side and low-side outputs are scaled symmetrically about the points of transition in the zero dead time case. The PWM scales the output by the value programmed in the PWM_CH[n]_DT register.

The ***Channel Outputs in Dependent Mode for Pulse Mode*** figures show the high and low-side outputs for the case with zero and non-zero dead time for PWM_ACTL.PULSEMODEHI =00 and 01. DUTY0 is the value programmed into the PWM_AH0 register. DUTY1 is the value programmed into the PWM_AH1 register. The PWM_CHANCFG.POLAH bit =1, indicates that both signals are active high. The PWM_CHA_DT register holds the value DT.

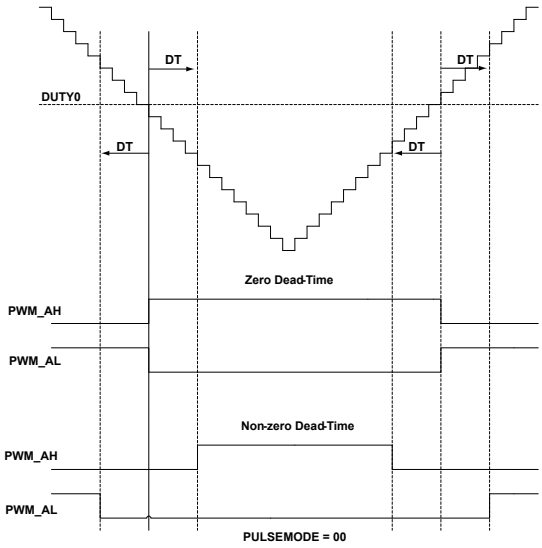


Figure 24-9: Channel Outputs in Dependent Mode for Pulse Mode = 00

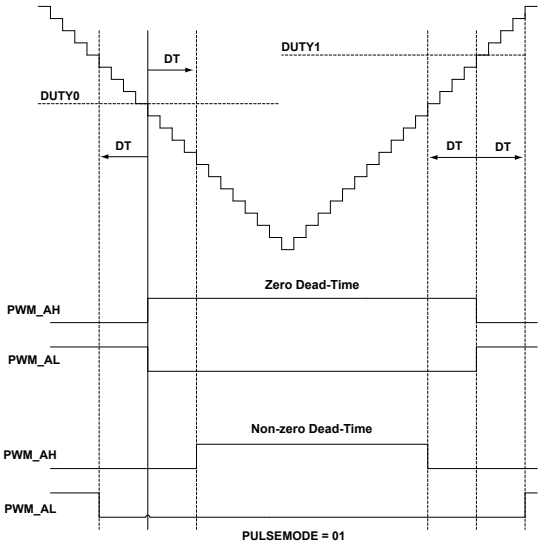


Figure 24-10: Channel Outputs in Dependent Mode for Pulse Mode = 01

The following pair of figures shows the high and low-side outputs for the case with zero and non-zero dead-time for PWM_ACTL.PULSEMODEHI =10 and 11. In the figures, DUTY0 is the value programmed into PWM_AH0 register and DUTY1 is the value programmed into the PWM_AH1 register. PWM_CHANCFG.POLAH is 1 indicating that both signals are active high. The channel dead-time registers hold the value DT.

NOTE: Using dead time, the guidelines for programming the duty-cycle registers in pulse modes 10 and 11 given in Duty Cycle and Pulse Positioning Control are modified as follows:

Pulse mode 10: PWM_xH0 – DT > PWM_xH1 + DT

Pulse mode 11: PWM_xH0 + DT < PWM_xH1 – DT

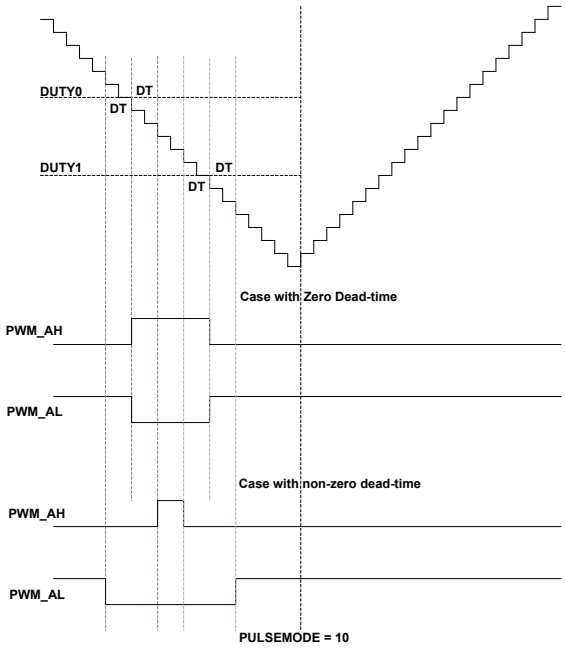


Figure 24-11: Channel Outputs in Dependent Mode for Pulse Mode = 10

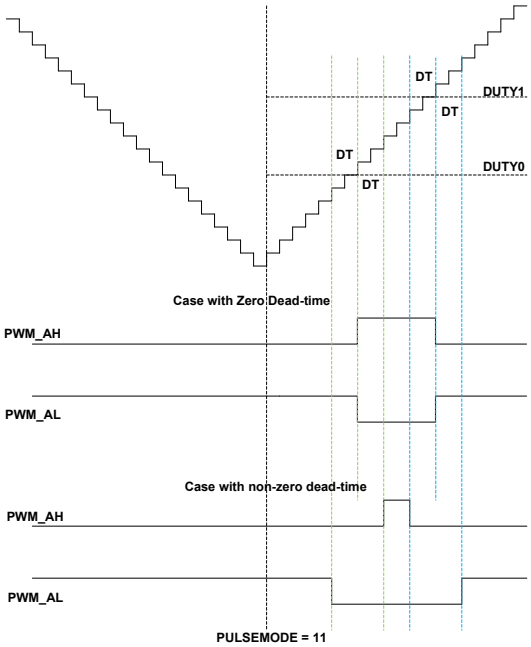


Figure 24-12: Channel Outputs in Dependent Mode for Pulse Mode = 11

Channel High Side and Low Side Outputs, Independent Operation Mode

Independent control of the PWM_AH0 and PWM_AL0 channel outputs is possible by setting the PWM_CHANCFG.MODELSA bit to 1. In this case, the PWM module:

- Generates PWM_AH using the PWM_AH0 register
- Uses the PWM_AH1 register to configure pulse width
- Uses the PWM_ACTL.PULSEMODEHI bit to configure pulse position
- Uses the PWM_CHANCFG.POLAH bit to configure polarity
- Generates PWM_AL using PWM_AL0
- Uses the PWM_AL1 register to configure pulse width
- Uses the PWM_ACTL.PULSEMODELO bit to configure pulse position
- Uses the PWM_CHANCFG.POLAL bit to configure polarity

NOTE: In independent mode, the dead-time insertion is not applicable. The hardware forces dead time to zero.

The ***PWM_AH and PWM_AL in Independent Operation Mode*** figure shows an example of the independent mode of operation where PWM_AH and PWM_AL work from different register bits.

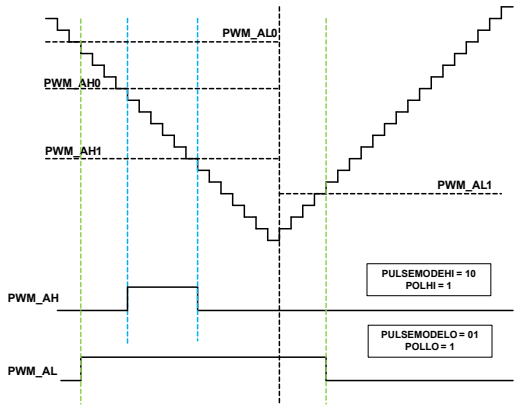


Figure 24-13: PWM_AH and PWM_AL in Independent Operation Mode

PWM_AH and PWM_AL can be positioned in the timer period with a given phase difference between them. Program the PWM_ACTL.PULSEMODEHI and PWM_ACTL.PULSEMODELO bits to different values to achieve this positioning as shown in the ***Channel Outputs Controlled Independently*** figure.

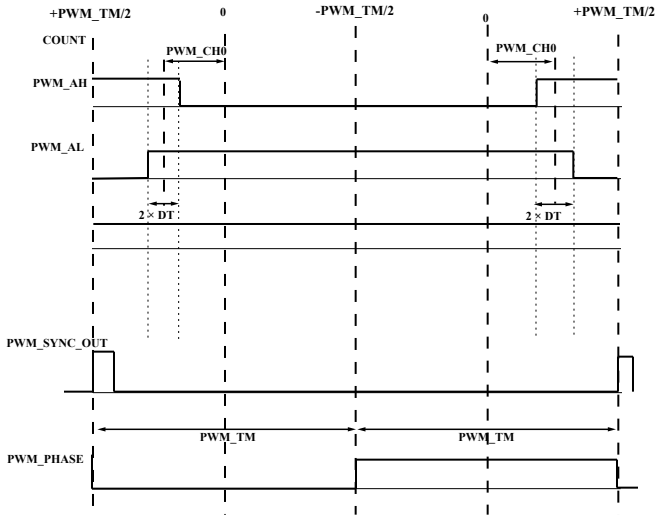


Figure 24-14: Channel Outputs Controlled Independently

Switched Reluctance Motors Application

In typical power converter configurations for switched or variable reluctance motors, motor winding is connected between the two power switches of a given inverter leg. To allow for a complete circuit in the motor winding, turn on both switches at the same time.

PWM uses switched reluctance motors in the following configurations: hard chop, alternate chop, soft chop—bottom on, and soft chop— top on.

The ***Four SR Mode Types, Active High PWM Output Signals*** figure shows the four SR mode types as active high PWM output signals.

Hard chop mode contains independently programmed rising edges of the high and low signals of a channel in the same PWM half cycle. Both signals contain independently programmed falling edges in the next PWM half cycle. The `PWM_CHANCFG.POLAH` and `PWM_CHANCFG.POLAL` bits are programmed to same values.

Alternate chop mode is similar to normal PWM operation except that the PWM channel high and low signal edges are always opposite and are independently programmed. The `PWM_CHANCFG.POLAH` and `PWM_CHANCFG.POLAL` bits are programmed to opposite values. The low-side invert is the only difference between hard chop mode and alternate chop mode.

Soft chop - bottom on uses a 100% duty on the low side of the channel. Soft chop - top on uses a 100% duty on the high side of the channel. Similar to hard chop mode, PWM uses the `PWM_AH0` duty register for the high channel and the `PWM_AL0` duty register for the low channel.

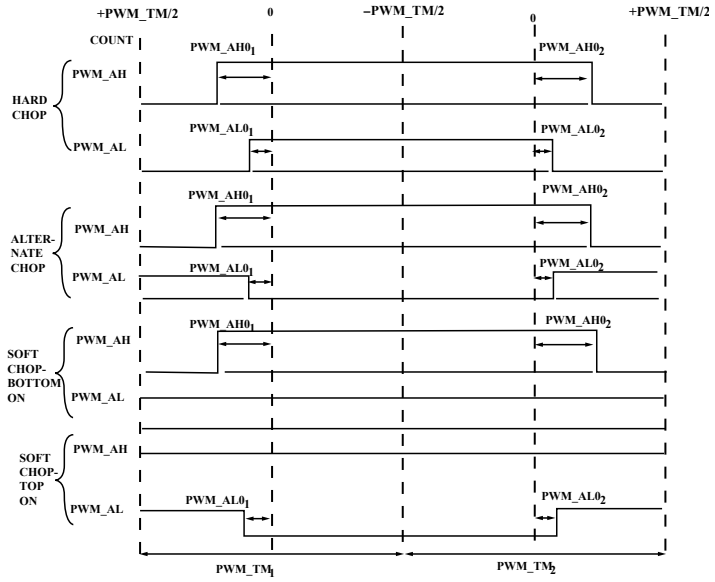


Figure 24-15: Four SR Mode Types, Active High PWM Output Signals

Switching Dead Time (PWM_DT) Register

The second important parameter that must be set up in the initial configuration of the PWM controller is the switching dead time. Dead time is a short delay introduced between turning off one PWM signal (for example, AH) and turning on the complementary signal (for example, AL). This short time delay permits turning off a power switch (AH in this case) to completely recover its blocking capability before the complementary switch is turned on. This time delay prevents a potentially destructive short-circuit condition from developing across the DC link capacitor of a typical voltage source inverter.

The 10-bit, read/write channel A through channel D dead-time registers (`PWM_CHA_DT` through `PWM_CHD_DT`) control the dead time for channel-x. If the value carried by any dead-time register is PWMDT, the dead time, Td, for that channel is:

T_d = PWM_CH[n]_DT × 2 × t_{PCLK}

Therefore, a dead-time value of 0x00A introduces a 200-ns delay (for an PCLK of 100 MHz). The delay occurs between turning off any PWM signal (for example, AH) and then turning on its complementary signal (for example, AL). The length of dead time can be programmed in increments of 2 × t_{PCLK} (or 20 ns for an PCLK of 100 MHz). The channel A through channel D dead-time registers have a maximum value of 0x3FF (1023 decimal) and correspond to a maximum programmed dead time of:

$T_{d(max)}$ = 1023 × 2 × t_{PCLK} = 1023 × 2 × 10 × 10^{-9} = 20.5 µs for an f_{PCLK} rate of 100 MHz.

Write 0 to the `PWM_CHA_DT` through `PWM_CHD_DT` registers to program the dead time.

Duty Cycle with Dead Time Control: Calculations for PULSEMODE 00

The duty cycle registers are scaled so that a value of 0 represents a 50% PWM duty cycle. The switching signals produced are also adjusted to incorporate the programmed dead-time value using the channel dead-time registers

(PWM_CHA_DT through PWM_CHD_DT). The unit in this case produces active low signals so that a low level corresponds to a command to turn-on the associated power device.

The ***Dead Time Between Outputs in Dependent Mode*** figure shows a typical pair of PWM outputs, PWM_AH and PWM_AL. The time values in the figure indicate the integer value in the associated register and can be converted to time by multiplying by the fundamental time increment, t_{CK}. In the example, channel A is working from PWM_TM0.

In the example, the pulse mode is set to 00 so that the switching patterns are perfectly symmetrical about the mid-point of the switching period. The dead time is incorporated by moving the switching instants of both PWM signals away from the instant set by the PWM_AH0 register. Both switching edges are moved by an equal amount (DT × t_{CK}) to preserve the symmetrical output patterns. Also shown is the PWM_SYNC output pulse whose rising edge denotes the beginning of the switching period, and the PWM_STAT.TMR0PHASE bit.

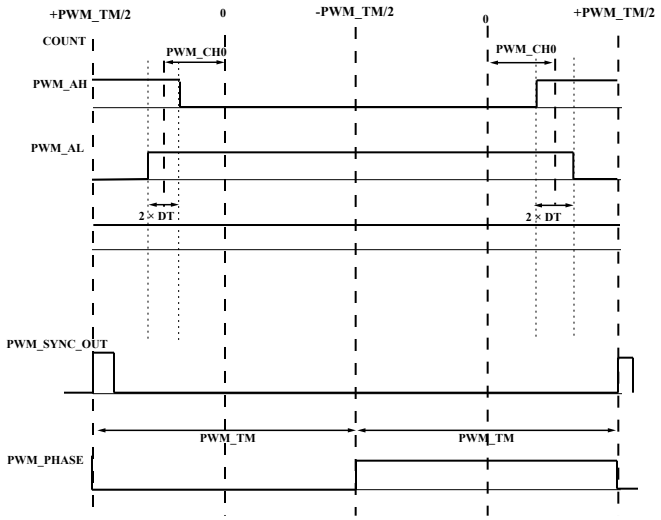


Figure 24-16: Dead Time Between Outputs in Dependent Mode

The PWM timing unit produces the resulting on-times (active low) of the PWM signals over the full PWM period (two half-periods). The figure illustrates this timing. The timing can be written per the following equation.

T_{AH} = (PWM_TM0 + 2 × (PWM_AH0 – PWM_CH[n]_DT)) × t_{CK};

Range of T_{AH} is [0:2 × PWM_TM0 × t_{CK}]

T_{AL} = (PWM_TM0 – 2 × (PWM_AH0 + PWM_CH[n]_DT)) × t_{CK};

Range of T_{AL} is [0:2 × PWM_TM0 × t_{CK}]

$$d_{AH} = \frac{T_{AH}}{Ts} = \frac{1}{2} + \frac{PWM_AH0 - DT}{PWM_TM}$$

$$d_{AL} = \frac{T_{AL}}{Ts} = \frac{1}{2} - \frac{PWM_AH0 + DT}{PWM_TM0}$$

The negative values of T_{AH} and T_{AL} are not permitted and the minimum permissible value is zero, corresponding to a 0% duty cycle. In a similar fashion, the maximum value is T_s, the PWM switching period, corresponding to a 100% duty cycle. Calculation of duty for other pulse modes can be similarly executed.

Special Consideration for PWM Operation in Over-Modulation

The PWM timing unit can produce PWM signals with variable duty cycle values at the PWM output pins. In pulse modes 00 and 01, at the extremities of the modulation process, duty cycles of 0% and 100% occur. In pulse modes 01 and 10, at the extremities of the modulation process, duty cycles of 0% and 50% occur. The modulation is called *full off* when the lower extremity of modulation is set for any PWM timer period for the corresponding channel. The modulation is called *full on* when the higher extremity of modulation is set for any PWM timer period for the corresponding channel. In between, for other duty cycle values, the operation is termed *normal modulation*.

Full On Modulation

In pulse modes 00 and 01, a PWM channel is in full on modulation if the high-side output of that channel is asserted. The output is asserted for the whole duration of the period of the PWM timer that channel is referencing. The conditions for full on modulation are:

- PWM_xH0 - DT > PWM_TMy/2 for pulse mode 00
- PWM_xH1 - DT > PWM_TMy/2 for pulse modes 00 and 01

In pulse mode 10, a PWM channel is in full on modulation if the high-side output of that channel is asserted. The output is asserted for the whole duration of the first half period of the PWM timer that the channel is referencing. The conditions for full on modulation are:

- PWM_xH0 - DT > PWM_TMy/2 for pulse mode 10
- PWM_xH1 + DT < PWM_TMy/2 for pulse mode 10

In pulse mode 11, a PWM channel is in full on modulation if the high-side output of that channel is asserted. The output is asserted for the whole duration of the second half period of the PWM timer that the channel is referencing. The conditions for full on modulation are:

- PWM_xH0 + DT < PWM_TMy/2 for pulse mode 11
- PWM_xH1 - DT > PWM_TMy/2 for pulse mode 11

Full Off Modulation

In pulse modes 00 and 01, a PWM channel is in full off modulation if the high-side output of that channel is deasserted. The output is deasserted for the whole duration of the period of the PWM timer that channel is referencing. The conditions for full off modulation are:

- PWM_xH0 - DT < PWM_TMy/2 for pulse mode 00
- PWM_xH1 - DT < PWM_TMy/2 for pulse modes 00 and 01

In pulse mode 10, a PWM channel is in full off modulation if the high-side output of that channel is deasserted. The output is deasserted for the whole duration of the first half period of the PWM timer that the channel is referencing. In the second half-period, it is deasserted anyway. The conditions for full off modulation are:

- PWM_xH0 - DT < PWM_TMy/2 for pulse mode 10
- PWM_xH1 + DT < PWM_xH0 DT for pulse mode 10

In pulse mode 11, a PWM channel is in full off modulation if the high-side output of that channel is deasserted. The output is deasserted for the whole duration of the second half period of the PWM timer that the channel is referencing. In the first half of the period, it is deasserted anyway. The conditions for full off modulation are:

- PWM_xH0 + DT > PWM_TMy/2 for pulse mode 11
- PWM_xH1 - DT > PWM_xH0 + DT for pulse mode 11

Normal Modulation

All other cases of modulation fall under this category.

Emergency Dead-Time Delays

Sometimes, during modulation transition, it is necessary to insert more emergency dead-time delays to prevent potential shoot through conditions in the inverter. (For example, when the PWM transitions into or out of full on or full off modulation.) Disabling and enabling usage (related to the `PWM_ACTL.DISHI` and `PWM_ACTL.DISLO` bits) also can potentially cause outputs to violate shoot-through condition criteria. Another case is when large values vary the phase delay of a PWM timer. These transitions are detected automatically. If appropriate for safety, an emergency dead-time is inserted to prevent shoot through conditions.

There is another atypical case for the insertion of the additional emergency dead time. It occurs when both PWM signals do not toggle within a dead time of each other. In this case, insert more emergency dead time into one of the PWM signals of a given pair during these transitions. The dead-time delay is inserted into the PWM signal that is toggling into the on-state. In effect, an amount ($2 \times DT \times t_{CK}$) from the rising edge of the opposite output delays the turn-on of this signal. After this delay, the PWM signal is allowed to turn-on provided the desired output is still scheduled to be in the on-state after the emergency dead-time delay.

The ***Over Modulation Transition Example*** figure illustrates two examples of such a transition. In the figure, `PWM_ACTL.PULSEMODEHI` is kept at 1. The `PWM_AH` signal has been in full on modulation for some time and, during the current period, its pulse mode is changed to 10, keeping the full on condition. At the half-period boundary, `PWM_AH` is forced to transition to a deasserted state because pulse mode is 10. An emergency dead-time is inserted on the low-side output.

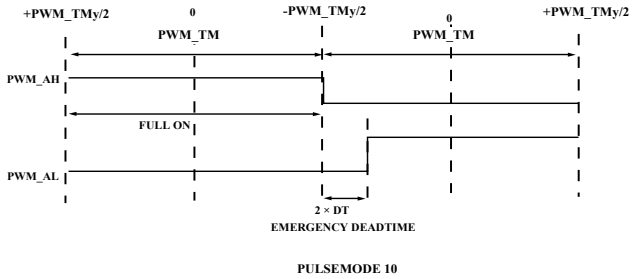


Figure 24-17: Over Modulation Transition Example

Gate Drive Unit

The gate drive unit of the PWM adds features that simplify the design of isolated gate drive circuits for PWM inverters. When using a transformer coupled power device gate drive amplifier, the active PWM signal must be chopped at a high frequency. The `PWM_CHOPCFG` register allows the programming of this chopping mode for high frequency. The chopped active PWM signals can be required for the high-side drivers only, for the low-side drivers only, or for both the high-side and low-side switches. Therefore, independent control of this mode for both high and low-side switches is included with two separate control bits in the `PWM_CHANCFG` register.

The ***High-Side and Low-Side Outputs With Gate Chop Enabled*** figure shows the typical PWM output signals with high-frequency chopping enabled on both high-side and low-side signals. Chopping of the PWM outputs is enabled by setting bits in `PWM_CHANCFG` register. The 8-bit `PWM_CHOPCFG.VALUE` value controls the high frequency chopping. The following equation gives the period of this high frequency carrier.

$T_{chop} = [4 \times (CHOPDIV + 1)] \times t_{CK}$

and the chopping frequency is therefore an integral subdivision of the peripheral clock frequency:

$f_{chop} = f_{CK}/[4 \times (CHOPDIV + 1)]$

The `PWM_CHOPCFG.VALUE` value can range from 0 to 255, corresponding to a programmable chopping frequency rate from 122 kHz to 31.25 MHz for a 125 MHz, f_{CK} rate. Program the gate drive features before enabling the PWM controller. Do not change the gate drive features during normal operation of the controller. Following a reset, clear all bits of the `PWM_CHANCFG` register so that high frequency chopping is disabled, by default.

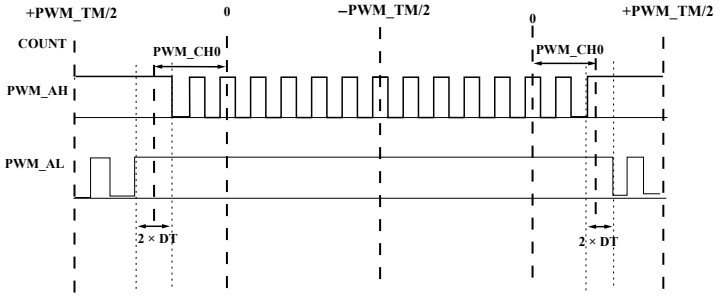


Figure 24-18: High-Side and Low-Side Outputs With Gate Chop Enabled

Output Control Feature Precedence

The order of applying output control features to the PWM signal is important and significant. Use the following order for applying the signal features to the PWM output signal.

1. Duty generation
2. Cross-over
3. High-side or low-side disable
4. Emergency dead-time insertion
5. Gate-drive chopping
6. Polarity

Operating Modes

The PWM generator is capable of operating in the following modes:

- Sync Operation Modes
- Output Disable and Cross-Over Modes
- Emulation Mode

Sync Operation Modes

The PWMx_SYNC pins can operate as internally generated or externally generated. If its internally generated, PWMs can drive the signal to synchronize other PWMs and other devices. If externally generated, the PWMs can be synchronized externally.

External (Triggered) PWM Sync Generation

By setting the PWM_CTL.EXTSYNC bit, the PWM is configured in a mode to expect an external PWM_SYNC signal on the PWM_SYNC_IN pin through the TRU. The trigger source can be any TRU1 trigger generator. Multiple PWM units can be precisely synchronized by selecting the same trigger generator as the sources for each PWM sync trigger receiver. Examples of useful trigger generators include:

- A PWM_SYNC GPIO controller. This option allows synchronizing the PWMs to an off-chip timing source. (The PWM_CTL.EXTSYNC bit must be programmed to 1 to enable the GPIO input.) The PWM_SYNC GPIO pads are connected through the pin mux to configurable trigger generators using PWM generator IDs TRGM_SYS_PWMn_SYNC_IN. Using the TRU, these generators can be connected to the PWM_SYNC trigger recievers in any desired combination.
- A TTU trigger output. This option can include a member of a TTU trigger group, which can also control the timing of other devices such as ADCs, SINC filter inputs, or GP timers. The TTU supports relative trigger delays so that timing offsets can be applied to manage system latencies with precision.

- A general-purpose timer trigger generator
- A software trigger generator. This option allows starting one or more PWM units simultaneously by an MMR write.

The external PWM_SYNC signal only determines the operation of the main timer PWMTMR0.

Synchronize the external sync by setting the PWM_CTL.EXTSYNCSEL bit to 0 (assumes the external PWM_SYNC selected is asynchronous).

The external PWM_SYNC period is expected to be an integer multiple of the value of the PWM_TM0 period register. When the rising edge of the external PWM_SYNC is detected, the PWMTMR0 timer is restarted at the beginning of its period. If the external PWM_SYNC period is not exactly an integer multiple of the internal PWM_SYNC, the behavior of the PWM channel outputs which are referenced to PWMTMR0 are clipped.

CAUTION: Do not change the value of the PWM_CTL.EXTSYNC bit while the PWM is enabled (PWM_CTL.GLOBEN =1).

Output Disable and Cross-Over Modes

Each PWM_ACTL channel control register contains separate enable bits for the high and low-side signals. The PWM module uses the PWM_ACTL.DISHI and PWM_ACTL.DISLO bits in the channel A control register to enable or disable the PWM_AH and PWM_AL outputs respectively. If the disable bit is set (=1), then the corresponding PWM output is disabled, irrespective of the value of the corresponding duty cycle register. This PWM output signal remains in the OFF state as long as the corresponding enable or disable bit is set.

The cross-over bit (PWM_ACTL.XOVR) allows programs to send the low-side output through the high-side output pin and the high-side output through the low-side output pin.

One example uses the following configuration:

- The PWM_AH0 register =0
- The PWM_CHANCFG.MODELSC bit =0
- The PWM_ACTL.DISLO bit =1
- The PWM_ACTL.XOVR bit =1

The low-side output remains off, as in the case without crossover. The difference in cross-over is that the high-side output changes character and becomes like the low-side. What actually occurs is that the low-side duty cycle is sent to the high-side output pins, and the high-side duty cycle is sent to the low side pins. Because the PWM_ACTL.DISLO bit =1, the low-side pin remains off (see Output Control Feature Precedence).

The ***XOVR and DISHI/DISLO Functionality*** figure shows this example. In case 1, PWM_ACTL.XOVR =0; and in case 2, PWM_ACTL.XOVR =1.

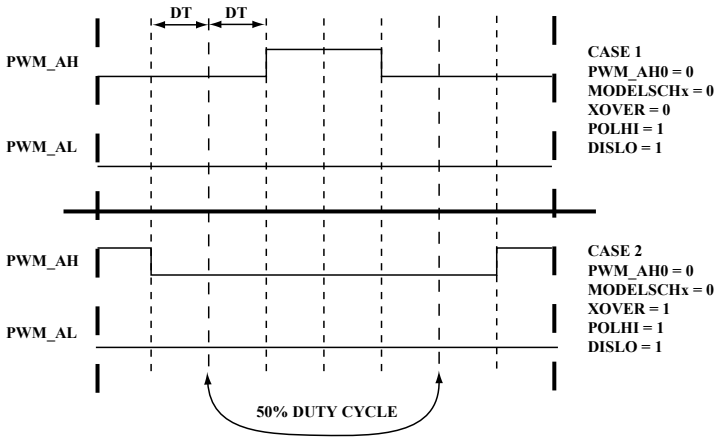


Figure 24-19: XOVR and DISHI/DISLO Functionality

Brushless DC Motor (Electronically Commutated Motor) Control

In the control of an electronically commutated motor (ECM), only two inverter legs are switched at any time. Often, the high-side device in one leg must be switched on at the same time as the low-side driver in a second leg. It is possible to turn on the high-side switch of phase A and the low-side switch of phase B at the same time by:

- Programming identical values for the duty cycles for two PWM channels (for example, `PWM_CH0` = `PWM_CH1`), and
- Setting the `PWM_BCTL.XOVR` bit to crossover the BH and BL pair of PWM signals

To control ECM, normally the third inverter leg (phase C in this example) is disabled for a number of PWM cycles. To implement this function, both the `PWM_CH` and `PWM_CL` outputs are disabled by setting the `PWM_CCTL.DISHI` and `PWM_CCTL.DISLO` bits.

In normal ECM operation, each inverter leg is disabled for certain time periods so that the PWM channel registers change based on the position of the rotor shaft (motor commutation).

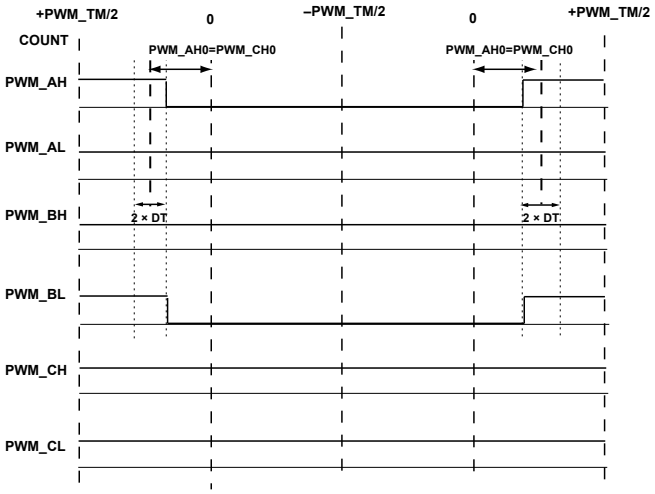


Figure 24-20: ECM Control

Emulation Mode

Depending upon system configuration, a debug halt may be presented to the PWM unit. The PWM has local control of its response to an emulator halt based on the PWM_CTL.EMURUN bit setting.

- PWM_CTL.EMURUN =1. When the processor is halted in emulation mode, the outputs continue to toggle and be driven out of the PWM block. The counters and status register bits are set or reset according to the PWM timer count or period settings.
- PWM_CTL.EMURUN =0. When the processor is halted in emulation mode, the outputs are shut down (enter their inactive state based on polarity), and all counters that affect status register bits are paused.

  The PWM_STAT.EMU bit is set.
- At restart, the PWM counters resume from their paused value. The outputs are still held in their inactive state.

  To reactivate the outputs, clear the PWM_STAT.EMU bit with a W1C operation.

NOTE: The PWM_STAT.EMU bit is not cleared by disabling the PWM or writing 0 to the PWM_CTL.EMURUN bit.

Emergency dead time is not ensured on re-enabling the outputs by a W1C operation to the PWM_STAT.EMU bit.

Event Control

The PWM uses bits in the PWM_IMSK and PWM_ILAT registers for event control. These registers allow masking and show masked interrupt request status bits, respectively. The interrupt status bits are latched and held on the interrupt event. The software must write a 1 to clear the interrupt status bit, usually during the interrupt service routine.

The timer period (TMRxPER) interrupt requests are configured using the PWM_ILAT.TMR0PER - PWM_ILAT.TMR4PER bits. The PWM uses the interrupts to execute an interrupt service routine (ISR) periodically. The ISR updates the PWM channel control and duty registers (according to a control algorithm based on expected operation and sampled existing operation). The TMRxPER interrupts also can trigger an ADC to sample data for use during the ISR.

The PWM uses the PWM_CTL.INTSYNCREF bit field to control the PWM_SYNC interrupt request. The PWM uses the bit field to assign the interrupt request to a system user interrupt of the core. The PWM_SYNC can be configured to be either internal or externally driven using the PWM_CTL.EXTSYNC bit. When configured as an external sync, the signal can be further configured as synchronous or asynchronous using the PWM_CTL.EXTSYNCSEL bit.

As an example, when the PWM_SYNC interrupt is serviced:

- The ADC samples data
- The data is algorithmically interpreted

- New PWM channel duties are calculated and written to the PWM

More sophisticated implementations include different startup, run time, and shutdown algorithms to determine PWM channel duties based on expected behavior and further features.

During the PWM_SYNC interrupt driven control loop, only the channel delay registers, the duty registers, and the channel C high pulse duty register values are typically updated. To see programming limitations on the PWM registers, see the "Register Descriptions".

Status information about the PWM is available in the PWM_STAT register, which stores all status bits, including raw interrupt status bits. In particular, the period boundary of each timer is available, as well as status bits. The PWM uses the status bits to indicate whether the operation is in the first half or the second half of the timer. Additionally, the TRIP status is also available. For more information on TRIP interrupts, see Trip Control Unit .

Trip Control Unit

The PWM output signals can be shut-off in a number of different ways. The trip inputs ($\overline{\texttt{PWM_TRIP[n]}}$) can be mapped to provide either a temporary or permanent shutdown on any channel outputs (hi/lo/pair). This shutdown mechanism is asynchronous, so that the associated PWM output disable circuitry does not go through any clocked logic. This functionality ensures correct PWM shutdown even in the event of a loss of the processor clock. In addition to the hardware shutdown features, the PWM system can be shut down in software with the PWM_CTL.SWTRIP bit.

During any external trip event (if not disabled), the PWM outputs are turned off. When a PWM output is turned off, it means that the output level is held at a polarity opposite that given in the PWM_CHANCFG.POLAH through PWM_CHANCFG.POLDH (channel high side polarity) bits. The PWM sync pulse continues to operate, when it is already enabled. A PWMTRIP interrupt occurs if unmasked, to notify the software of this event. In dependent mode of operation, both high and low-side outputs refer to the channel high side polarity bits.

Even if the clock to the PWM is damaged, an external trip event turns off the PWM outputs. In this case, the PWMTRIP interrupt request may not occur.

The PWM trip unit processes hardware or software fault conditions and shuts down the PWM channel outputs immediately on the occurrence of these conditions. The PWM can enable the shutdown mechanism separately for each channel. The design also allows for a self-restart mechanism to be enabled on a channel. Self-restart reenables the channel outputs following the fault condition (allowed only on hardware trips) when the PWMTMRy that the channel is using reaches its period boundary.

Program the PWM_TRIPCFG.MODE0A bits to specify the restart mechanism for a channel that has been tripped.

1. If the PWM_TRIPCFG.MODE0A bit =0, once tripped, a trip condition is registered on this channel in the PWM_STAT.FLTTRIPA bit and the outputs of that channel are immediately shut down. This condition is called a *fault trip* condition. To resume channel output when a fault trip occurs, write a 1 to clear the PWM_STAT.FLTTRIPA bit. A processor write cannot clear the bit when the trip condition is still active. The raw trip status is available for both pins in the PWM_STAT.RAWTRIP0 register bits.
2. If the PWM_TRIPCFG.MODE0A bit =1, once tripped, a trip condition is registered on this channel in the PWM_STAT.SRTRIPA bit and the outputs of that channel are immediately shut down. This condition

is called a *self-restart trip* condition. If the trip condition is not active at the next period boundary of the PWMTMRy that the channel is using, the status register bit is cleared. The outputs are restored.

In addition to the hardware trip conditions, a global software trip bit in the PWM_CTL register allows for a software-forced fault trip condition. When the global software trip bit is set to 1, irrespective of the values in the PWM_TRIPCFG register, it sets all the PWM_STAT.FLTTRIPA bits and also gates the channel outputs. To remove the trip condition from the channel, perform a W1C on the PWM_STAT.FLTTRIPA bit of the particular channel.

If the PWM_TRIPCFG.EN0A bit is set to 1 to, for any channel, then the occurrence of a fault condition on the PWMTRIPy bit is logged in the PWM_STAT.FLTTRIPA register bit. If the corresponding PWM_IMSK.TRIP0 bit = 1, then an interrupt request is generated. Tripping a channel output does not interfere with PWM_SYNC generation.

The ***Operation Under Hardware Fault Conditions*** figure shows an example where PWMTRIP0 is enabled on channel A as self-restart trip. Channel A works with the PWM_CHANCFG.POLAH bit =1. In period 2, the PWM_AH signal is full on modulated, and tries to rise at the period boundary where the self-restart occurs for the channel. However, since the low-side output of the channel was only recently removed due to a trip, the rise edge on PWM_AH is delayed until the emergency dead-time period is over. PWMTRIP1 is enabled on channel B as a fault trip. Channel B works with the PWM_CHANCFG.POLAH bit =0. PWMTRIP1 stays low for an extended time period. The first processor write to reenable the channel output fails. The second processor write passes since the fault condition has gone away.

NOTE: Dead time is ensured on re-enabling the channel outputs after trip.

NOTE: Programs must not allow changes in the configuration or enable bits of PWM_TRIPCFG register within ± 10 clock cycles of when the external trip pulse toggles. (The configuration or enable bits of PWM_TRIPCFG register select between trip enable and disable). If this time frame is not followed, then unexpected behavior occurs.

Programming Model

The following sections provide general (and some application-specific) programming steps for configuring and using the PWM module.

- Programming Model for Three-Phase AC Motor Control

Programming Model for Three-Phase AC Motor Control

The ***PWM Module and Interaction with System*** figure shows how the PWM unit (green) interfaces to both software (blue) and hardware (yellow). The software configures the unit, calculates duty cycles (Duty A, Duty B, Duty C), and services the interrupts generated by the module (PWM SYNC IRQ, TRIP IRQ). The hardware applies the gate signals (AH, AL, BH, BL, CH, CL) to the inverter and provides an over-current trip signal back to the unit (TRIP0).

The typical three-phase AC motor configuration shown in the ***PWM Module and Interaction with System*** figure applies for both permanent magnet and induction motor types.

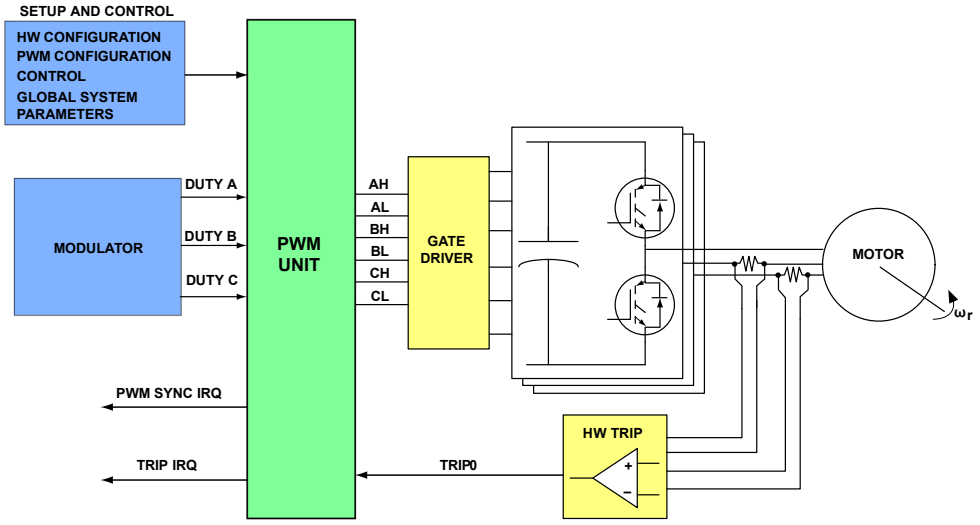


Figure 24-21: PWM Module and Interaction with System

System Parameters

The following system parameters (characteristics) influence the module configuration for this application. This example system features:

- One three-phase AC machine
- B6 inverter
- SVPWM, including both linear- and over-modulation
- Switching frequency of 20 kHz
- Dead time of 1us
- Trip signal generated by hardware
- Active high-level gate drive
- Core frequency of 200 MHz
- Peripheral clock of 100 MHz

System State Sequencing

Managing the system state and sequence of states is critically important when programming the PWM module. The ***PWM System States*** figure provides an overview of these states.

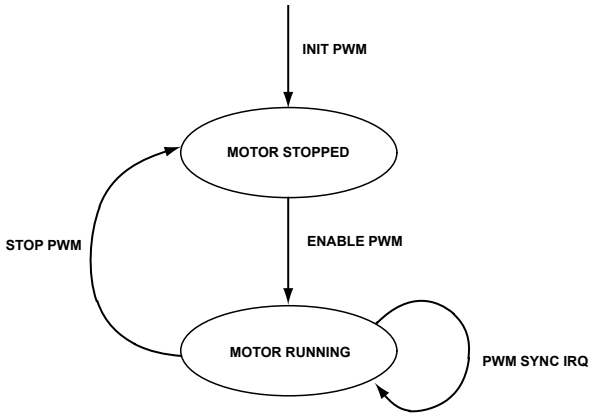


Figure 24-22: PWM System States

As shown in the state diagram, the module configuration is updated on state transitions (indicated by the arrows). The transitions are initialization, motor start, PWM sync interrupt request (on each), and motor stop. The following sections discuss the transitions in detail.

- PWM Initialization for Motor Control
- PWM Enable for Motor Control
- PWM Response to Sync Interrupt for Motor Control
- PWM Disable (and Stop the Motor) for Motor Control

PWM Initialization for Motor Control

The processor must program the PWM at power-up as follows and repeat this programming to bring the PWM and the system into a known (safe) state.

1. Place the PWM module in a safe state and set up synchronization of the module using the following bitwise operations on the `PWM_CTL` and `PWM_CHANCFG` registers:

```
PWM_CTL &= 0xFFE0FF08
PWM_CTL |= 0x20000
PWM_CHANCFG &= 0x80808080
PWM_CHANCFG |= 0x24242424
```

   These operations result in the following bit settings:

   - Disable PWM (`PWM_CTL.GLOBEN` =0)
   - Disable delay for channels A, B, C, D (`PWM_CTL.DLYAEN` through `PWM_CTL.DLYDEN` =0). All phases must run with same phase.
   - Use internal synchronization by timer TMR0 (`PWM_CTL.EXTSYNC` =0, `PWM_CTL.EXTSYNCSEL` =1)
   - The same timer, TMR0 (`PWM_CTL.INTSYNCREF` = b#000) synchronizes all phases.

- Low-side is always the inverse of high-side (PWM_CHANCFG.POLAL through PWM_CHANCFG.POLDL = 1).
- System uses active high gate driver (PWM_CHANCFG.ENCHOPAH through PWM_CHANCFG.ENCHOPDH =1).
- Disable gate chopping (PWM_CHANCFG.ENCHOPAL through PWM_CHANCFG.ENCHOPDL =0). PWM does not use the pulse transformer.

2. Set up the trip and associated interrupt requests using the following bitwise operations on the PWM_TRIPCFG and PWM_ILAT registers:

```
PWM_TRIPCFG &= 0xF0F0F0F0
PWM_TRIPCFG |= 0x1010101
PWM_ILAT &= 0xFFE0FFFC
PWM_ILAT |= 0x1
```

These operations result in the following bit settings:

- All phases must shut down simultaneously in case of fault: (PWM_TRIPCFG.EN0A through PWM_TRIPCFG.EN0D =0, PWM_TRIPCFG.MODE0A through PWM_TRIPCFG.MODE0D =0, PWM_TRIPCFG.EN1A through PWM_TRIPCFG.EN1D =0, PWM_TRIPCFG.MODE1A through PWM_TRIPCFG.MODE1D =0)
- Enable TRIP0 as fault trigger for all channels (PWM_TRIPCFG.EN0A through PWM_TRIPCFG.MODE1D =1).
- For thermal control and synchronization, SW intervention is needed at trip. Do not use automatic restart of any channels.
- Generate an interrupt at trip on TRIP0 (PWM_ILAT.TMR0PER = 1).

3. Configure the PWM channels using the following bitwise operations on the PWM_TRIPCFG and PWM_ILAT registers:

```
PWM_CHA_DT=0x32 PWM_CHB_DT=0x32 PWM_CHC_DT=0x32
PWM_TM0 = 0x9C4
PWM_ACTL = 0xFFFFF0000
PWM_BCTL = 0xFFFFF0000
PWM_CCTL = 0xFFFFF0000
PWM_AH0 = 0x0
PWM_BH0 = 0x0
PWM_CH0 = 0x0
```

These operations result in the following bit settings:

- Configure a dead time of 1 µs (DT =0x32 =0x32)
- Configure a PWM frequency of 20 kHz (assuming a system clock frequency of 100Mhz, so a clock divisor of 5000:1) (PWM_TM0 =0x9C4)

- Disable all outputs (PWM_ACTL.DISHI through PWM_CCTL.DISHI =0, PWM_ACTL.DISLO through PWM_CCTL.DISLO =0)
- Use conventional PWM, disable crossover (PWM_ACTL.XOVR through PWM_CCTL.XOVR =0)
- Use symmetrical pulse position on all outputs (PWM_ACTL.PULSEMODEHI through PWM_CCTL.PULSEMODEHI =0, PWM_ACTL.PULSEMODELO through PWM_CCTL.PULSEMODELO =0)
- Set an initial duty-cycle of 50% (PWM_AH0 through PWM_CH0 =0x0)

PWM Enable for Motor Control

The processor must do the following programming to enable the PWM before starting the motor.

1. Start the PWM timer TMR0 using the PWM_CTL |=0x1 bitwise operation on the PWM_CTL register.

   ADDITIONAL INFORMATION: This operation has the same effect as setting the PWM_CTL.GLOBEN bit =1.

2. Enable six PWM outputs using the following bitwise operations on the PWM_ACTL through PWM_CCTL registers.

   ```
   PWM_ACTL|=0x3
   PWM_BCTL|=0x3
   PWM_CCTL|=0x3
   ```

   ADDITIONAL INFORMATION: These operations have the same effect as enabling high and low-side channel outputs by setting the PWM_ACTL.DISHI through PWM_CCTL.DISHI bits =1 and the PWM_ACTL.DISLO through PWM_CCTL.DISLO bits =1.

3. Enable the PWM TRIP0 interrupt using the PWM_ILAT |= 0x1 bitwise operation on the PWM_ILAT register.

   ADDITIONAL INFORMATION: This operation has the same effect as setting the PWM_ILAT.TRIP0 bit =1

PWM Response to Sync Interrupt for Motor Control

When the PWM sync interrupt occurs, the processor could need to update to the PWM duty cycle with a value calculated by the motor control algorithm. This application uses symmetric pulses position and uses dependent high and low-side output. So, the PWM updates only one register for each phase.

1. Write the new duty cycle value (calculated by motor control algorithm) to the timer when the sync interrupt occurs.

   The following bitwise operations on the PWM_AH0 through PWM_CH0 registers accomplish this task:

```
PWM_AH0 = Duty_A_mc_algorithm_current_value
PWM_BH0 = Duty_B_mc_algorithm_current_value
PWM_CH0 = Duty_C_mc_algorithm_current_value
```

PWM Disable (and Stop the Motor) for Motor Control

The processor must program the PWM as follows to stop the motor, disable the PWM, and disable PWM interrupts. These actions place the PWM and system in a safe, passive state.

1. Disable the PWM timer using the `PWM_CTL &= 0xFFFFFFFE` bitwise operation on the `PWM_CTL` register.

   ADDITIONAL INFORMATION: This operation has the same effect as clearing the `PWM_CTL.GLOBEN` bit =0.

2. Disable all PWM outputs using the following bitwise operations on the `PWM_ACTL` through `PWM_CCTL` registers.

   ```
   PWM_ACTL &= 0xFFFFFFFC
   PWM_BCTL &= 0xFFFFFFFC
   PWM_CCTL &= 0xFFFFFFFC
   ```

   These operations disable PWM outputs where the `PWM_ACTL.DISHI` through `PWM_CCTL.DISHI` bits =1 and the `PWM_ACTL.DISLO` through `PWM_CCTL.DISLO` bits =0)

3. Set the PWM duty-cycle to 50% using the following bitwise operations on the `PWM_AH0` through `PWM_CH0` registers.

   ```
   PWM_AH0 =0x0
   PWM_BH0 =0x0
   PWM_CH0 =0x0
   ```

   These operations have the same effect as clearing the `PWM_AH0.DUTY` through `PWM_CH0.DUTY` bit =0.

4. Disable the PWM TRIP0 interrupt request using the `PWM_ILAT &= 0xFFFFFFFE` bitwise operation on the `PWM_ILAT` register.

   ADDITIONAL INFORMATION: This operation has the same effect as clearing the PWM TRIP0 interrupt request `PWM_ILAT.TRIP0` =0.

ADSP-2184x PWM Register Descriptions

Pulse-Width Modulator (PWM) contains the following registers.

Table 24-5: ADSP-2184x PWM Register List

Name	Description
`PWM_ACTL`	Channel A Control Register

Table 24-5: ADSP-2184x PWM Register List (Continued)

Name	Description
PWM_AH0	Channel A-High Duty-0 Register
PWM_AH1	Channel A-High Duty-1 Register
PWM_AH_DUTY0	Channel A-High Full Duty0 Register
PWM_AH_DUTY1	Channel A-High Full Duty1 Register
PWM_AL0	Channel A-Low Duty-0 Register
PWM_AL1	Channel A-Low Duty-1 Register
PWM_AL_DUTY0	Channel A-Low Full Duty0 Register
PWM_AL_DUTY1	Channel A-Low Full Duty1 Register
PWM_BCTL	Channel B Control Register
PWM_BH0	Channel B-High Duty-0 Register
PWM_BH1	Channel B-High Duty-1 Register
PWM_BH_DUTY0	Channel B-High Full Duty0 Register
PWM_BH_DUTY1	Channel B-High Full Duty1 Register
PWM_BL0	Channel B-Low Duty-0 Register
PWM_BL1	Channel B-Low Duty-1 Register
PWM_BL_DUTY0	Channel B-Low Full Duty0 Register
PWM_BL_DUTY1	Channel B-Low Full Duty1 Register
PWM_CCTL	Channel C Control Register
PWM_CH0	Channel C-High Pulse Duty Register 0
PWM_CH1	Channel C-High Pulse Duty Register 1
PWM_CHANCFG	Channel Configuration Register
PWM_CHA_DT	Channel A Dead-time Register
PWM_CHB_DT	Channel B Dead-time Register
PWM_CHC_DT	Channel C Dead-time Register
PWM_CHD_DT	Channel D Dead-time Register
PWM_CHOPCFG	Chop Configuration Register
PWM_CH_DUTY0	Channel C-High Full Duty0 Register
PWM_CH_DUTY1	Channel C-High Full Duty1 Register
PWM_CL0	Channel C-Low Pulse Duty Register 0
PWM_CL1	Channel C-Low Duty-1 Register
PWM_CL_DUTY0	Channel C-Low Full Duty0 Register

Table 24-5: ADSP-2184x PWM Register List (Continued)

Name	Description
PWM_CL_DUTY1	Channel C-Low Full Duty1 Register
PWM_CTL	Control Register
PWM_DCTL	Channel D Control Register
PWM_DH0	Channel D-High Duty-0 Register
PWM_DH1	Channel D-High Pulse Duty Register 1
PWM_DH_DUTY0	Channel D-High Full Duty0 Register
PWM_DH_DUTY1	Channel D-High Full Duty1 Register
PWM_DL0	Channel D-Low Pulse Duty Register 0
PWM_DL1	Channel D-Low Pulse Duty Register 1
PWM_DLYA	Channel A Delay Register
PWM_DLYB	Channel B Delay Register
PWM_DLYC	Channel C Delay Register
PWM_DLYD	Channel D Delay Register
PWM_DL_DUTY0	Channel D-Low Full Duty0 Register
PWM_DL_DUTY1	Channel D-Low Full Duty1 Register
PWM_ILAT	Interrupt Latch Register
PWM_IMSK	Interrupt Mask Register
PWM_STAT	Status Register
PWM_SWTRIP	Software Trip Register
PWM_SYNC_WID	Sync Pulse Width Register
PWM_TM0	Timer 0 Period Register
PWM_TM1	Timer 1 Period Register
PWM_TM2	Timer 2 Period Register
PWM_TM3	Timer 3 Period Register
PWM_TM4	Timer 4 Period Register
PWM_TRIPCFG	Trip Configuration Register
PWM_TRIP_POL	Trip Polarity Register

Channel A Control Register

The PWM_ACTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

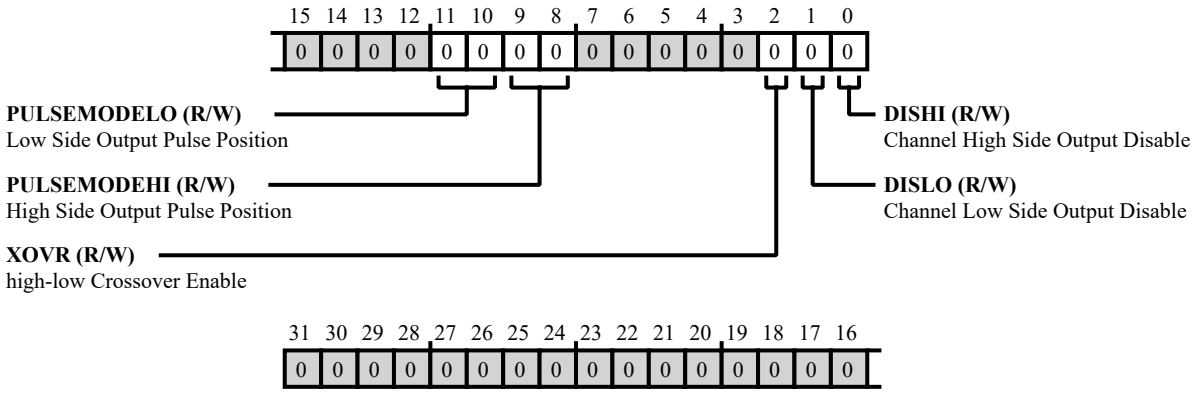


Figure 24-23: PWM_ACTL Register Diagram

Table 24-6: PWM_ACTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	PULSEMODELO	Low Side Output Pulse Position. The PWM_ACTL.PULSEMODELO bits select the pulse position for Channel A low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_AL0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_AL0 and PWM_AL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_AL0 and PWM_AL1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half

Table 24-6: PWM_ACTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	PULSEMODEHI	High Side Output Pulse Position. The PWM_ACTL.PULSEMODEHI bits select the pulse position for Channel A high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_AH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_AH0 and PWM_AH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_AH0 and PWM_AH1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half
2 (R/W)	XOVR	high-low Crossover Enable. The PWM_ACTL.XOVR bit enables crossover between the channels high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.	
		0	Disable Crossover
		1	Enable Crossover
1 (R/W)	DISLO	Channel Low Side Output Disable. The PWM_ACTL.DISLO bit enables the channels low side output.	
		0	Enable Low Side Output
		1	Disable Low Side Output
0 (R/W)	DISHI	Channel High Side Output Disable. The PWM_ACTL.DISHI bit enables the channels high side output.	
		0	Enable High Side Output
		1	Disable High Side Output

Channel A-High Duty-0 Register

The PWM_AH0 and PWM_AH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the high side output pulses for the channel A duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_ACTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_AH0 register to determine the assertion and deassertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel A high pulse output for a count less than PWM_AH0 and deasserts this output for a count greater than PWM_AH1.

The value range for the PWM_AH0 and PWM_AH1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHA_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_AH0 and PWM_AH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHA_DT (two's complement) and +PWM_TM0/2 + PWM_CHA_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHA_DT (two's complement) and -PWM_TM0/2 - PWM_CHA_DT .

Note that using values in the PWM_AH0 or PWM_AH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

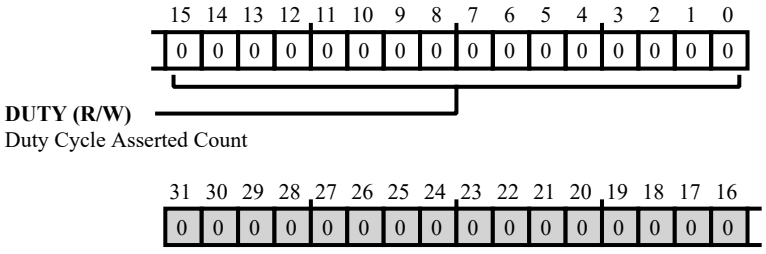


Figure 24-24: PWM_AH0 Register Diagram

Table 24-7: PWM_AH0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_AH0.DUTY bits select the duty cycle asserted count for Channel A high side output.

Channel A-High Duty-1 Register

The PWM_AH0 and PWM_AH1 registers determine the width for the high side output pulses. For more information, see the PWM_AH0 register description.

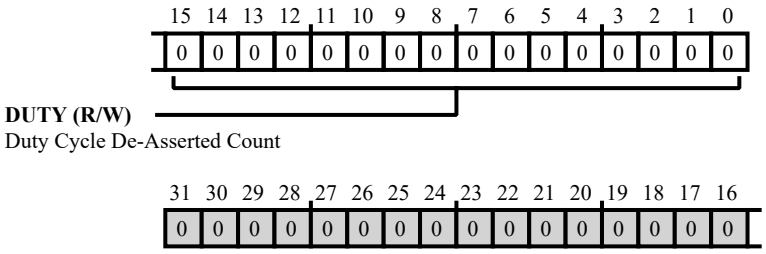


Figure 24-25: PWM_AH1 Register Diagram

Table 24-8: PWM_AH1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count. The PWM_AH1.DUTY bits select the duty cycle de-asserted count for Channel A high side output.

Channel A-High Full Duty0 Register

The PWM_AH_DUTY0 register contains the PWM_AH_DUTY0.DUTY bit field from the PWM_AH0 register.

Note that the PWM_AH_DUTY0 register reads the PWM_AH0 register values and visa-versa.

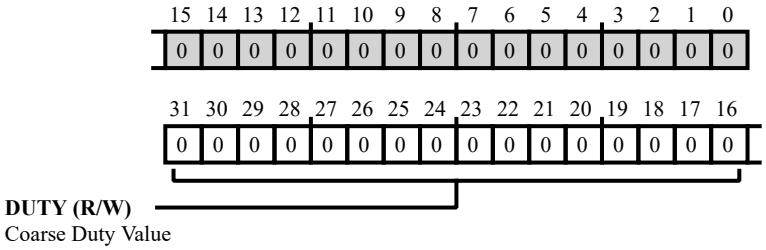


Figure 24-26: PWM_AH_DUTY0 Register Diagram

Table 24-9: PWM_AH_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_AH_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel A-High Full Duty1 Register

The PWM_AH_DUTY1 register contains the PWM_AH_DUTY1.DUTY bit field from the PWM_AH1 register.

Note that the PWM_AH_DUTY1 register reads the PWM_AH1 register values and visa-versa.

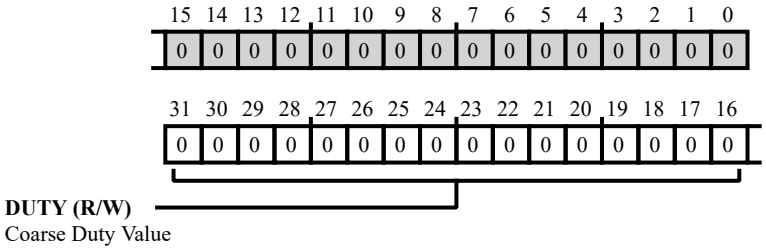


Figure 24-27: PWM_AH_DUTY1 Register Diagram

Table 24-10: PWM_AH_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_AH_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel A-Low Duty-0 Register

The PWM_AL0 and PWM_AL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the low side output pulses for the channel A duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_ACTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_AL0 register to determine the assertion and deassertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel A low pulse output for count less than PWM_AL0 and deasserts this output for count greater than PWM_AL1.

The value range for the PWM_AL0 and PWM_AL1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHA_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_AL0 and PWM_AL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHA_DT (two's complement) and +PWM_TM0/2 + PWM_CHA_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHA_DT (two's complement) and -PWM_TM0/2 - PWM_CHA_DT.

Note that using values in the PWM_AL0 or PWM_AL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

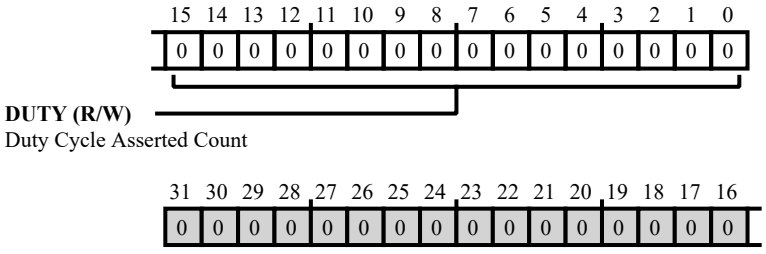


Figure 24-28: PWM_AL0 Register Diagram

Table 24-11: PWM_AL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_AL0.DUTY bits select the duty cycle asserted count for Channel A low side output.

Channel A-Low Duty-1 Register

The PWM_AL0 and PWM_AL1 registers determine the width for the low side output pulses. For more information, see the PWM_AL0 register description.

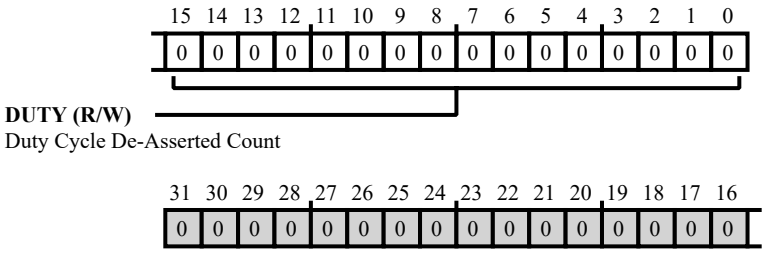


Figure 24-29: PWM_AL1 Register Diagram

Table 24-12: PWM_AL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count. The PWM_AL1.DUTY bits select the duty cycle de-asserted count for Channel A low side output.

Channel A-Low Full Duty0 Register

The PWM_AL_DUTY0 register contains the PWM_AL_DUTY0.DUTY bit field from the PWM_AL0 register.

Note that the PWM_AL_DUTY0 register reads the PWM_AL0 and visa-versa.

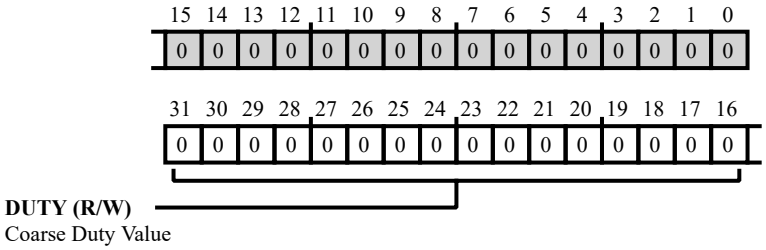


Figure 24-30: PWM_AL_DUTY0 Register Diagram

Table 24-13: PWM_AL_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_AL_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel A-Low Full Duty1 Register

The PWM_AL_DUTY1 register contains the PWM_AL_DUTY1.DUTY bit field from the PWM_AL1 register.

Note that the PWM_AL_DUTY1 register reads the PWM_AL1 and visa-versa.

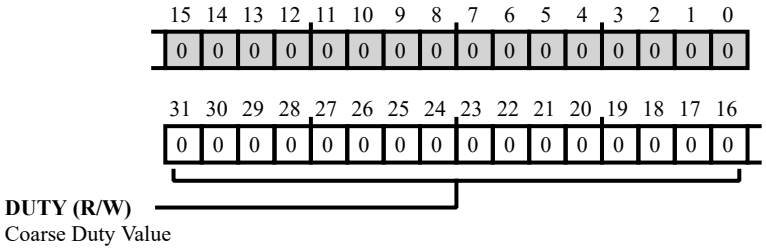


Figure 24-31: PWM_AL_DUTY1 Register Diagram

Table 24-14: PWM_AL_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_AL_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel B Control Register

The PWM_BCTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

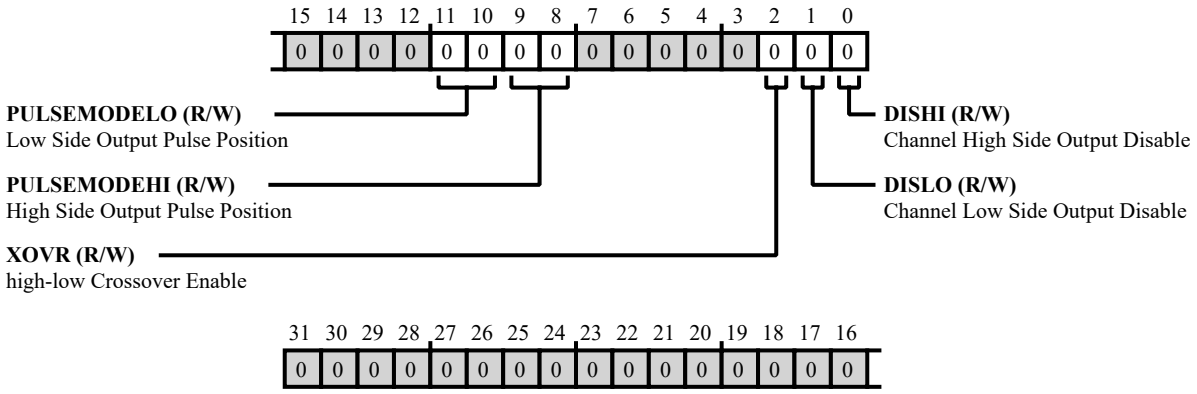


Figure 24-32: PWM_BCTL Register Diagram

Table 24-15: PWM_BCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	PULSEMODELO	Low Side Output Pulse Position. The PWM_BCTL.PULSEMODELO bits select the pulse position for Channel B low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_BL0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_BL0 and PWM_BL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_BL0 and PWM_BL1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half

Table 24-15: PWM_BCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	PULSEMODEHI	High Side Output Pulse Position. The PWM_BCTL.PULSEMODEHI bits select the pulse position for Channel B high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_BH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_BH0 and PWM_BH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_BH0 and PWM_BH1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half
2 (R/W)	XOVR	high-low Crossover Enable. The PWM_BCTL.XOVR bit enables crossover between the channels high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.	
		0	Disable Crossover
		1	Enable Crossover
1 (R/W)	DISLO	Channel Low Side Output Disable. The PWM_BCTL.DISLO bit enables the channels low side output.	
		0	Enable Low Side Output
		1	Disable Low Side Output
0 (R/W)	DISHI	Channel High Side Output Disable. The PWM_BCTL.DISHI bit enables the channels high side output.	
		0	Enable High Side Output
		1	Disable High Side Output

Channel B-High Duty-0 Register

The PWM_BH0 and PWM_BH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the high side output pulses for the channel B duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_BCTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_BH0 register to determine the assertion and deassertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel B high pulse output for count less than PWM_BH0 and deasserts this output for count greater than PWM_BH1.

The value range for the PWM_BH0 and PWM_BH1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHB_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_BH0 and PWM_BH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHB_DT (two's complement) and +PWM_TM0/2 + PWM_CHB_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHB_DT (two's complement) and -PWM_TM0/2 - PWM_CHB_DT.

Note that using values in the PWM_BH0 or PWM_BH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

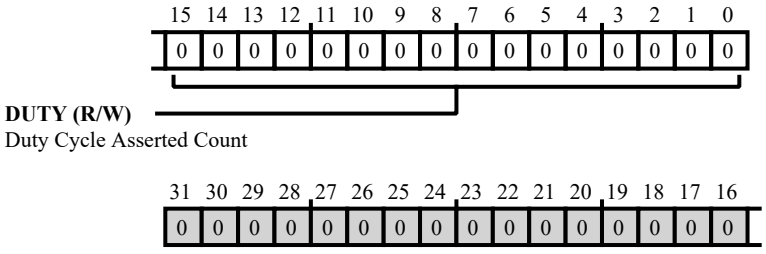


Figure 24-33: PWM_BH0 Register Diagram

Table 24-16: PWM_BH0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count.

Channel B-High Duty-1 Register

The PWM_BH0 and PWM_BH1 registers determine the width for the high side output pulses. For more information, see the PWM_BH0 register description.

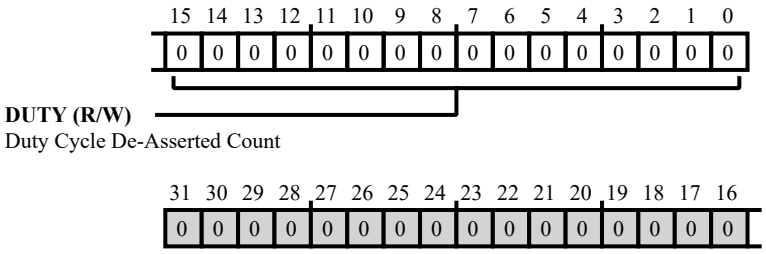


Figure 24-34: PWM_BH1 Register Diagram

Table 24-17: PWM_BH1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count.

Channel B-High Full Duty0 Register

The PWM_BH_DUTY0 register contains the PWM_BH_DUTY0.DUTY bit field from the PWM_BH0 register.

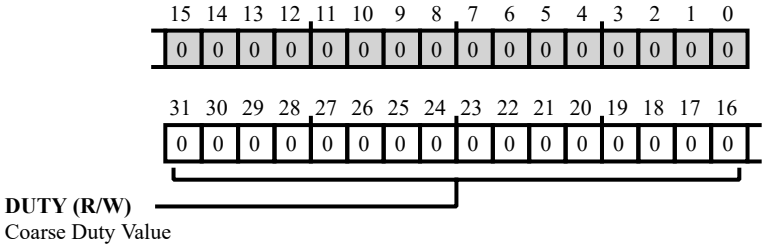


Figure 24-35: PWM_BH_DUTY0 Register Diagram

Table 24-18: PWM_BH_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_BH_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel B-High Full Duty1 Register

The PWM_BH_DUTY1 register contains the PWM_BH_DUTY1.DUTY bit field from the PWM_BH1 register.

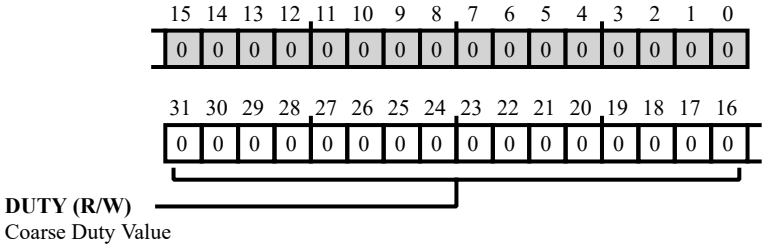


Figure 24-36: PWM_BH_DUTY1 Register Diagram

Table 24-19: PWM_BH_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_BH_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel B-Low Duty-0 Register

The PWM_BL0 and PWM_BL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the low side output pulses for the channel B duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_BCTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_BL0 register to determine the assertion and deassertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel B low pulse output for count less than PWM_BL0 and deasserts this output for count greater than PWM_BL1.

The value range for the PWM_BL0 and PWM_BL1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHB_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_BL0 and PWM_BL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHB_DT (two's complement) and +PWM_TM0/2 + PWM_CHB_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHB_DT (two's complement) and -PWM_TM0/2 - PWM_CHB_DT.

Note that using values in the PWM_BL0 or PWM_BL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

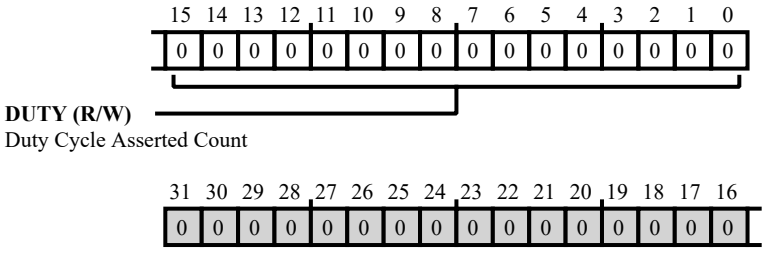


Figure 24-37: PWM_BL0 Register Diagram

Table 24-20: PWM_BL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_BL0.DUTY bits select the duty cycle asserted count for Channel B low side output.

Channel B-Low Duty-1 Register

The PWM_BL0 and PWM_BL1 registers determine the width for the low side output pulses. For more information, see the PWM_BL0 register description.

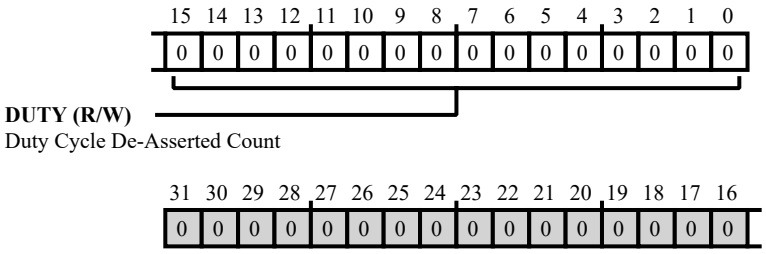


Figure 24-38: PWM_BL1 Register Diagram

Table 24-21: PWM_BL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count. The PWM_BL1.DUTY bits select the duty cycle de-asserted count for Channel B low side output.

Channel B-Low Full Duty0 Register

The PWM_BL_DUTY0 register contains the PWM_BL_DUTY0.DUTY bit field from the PWM_BL0 register.

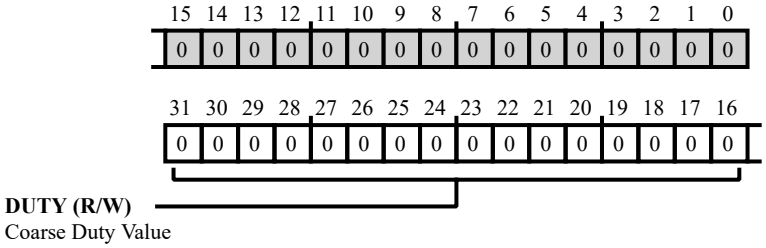


Figure 24-39: PWM_BL_DUTY0 Register Diagram

Table 24-22: PWM_BL_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_BL_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel B-Low Full Duty1 Register

The PWM_BL_DUTY1 register contains the PWM_BL_DUTY1.DUTY bit field from the PWM_BL1 register.

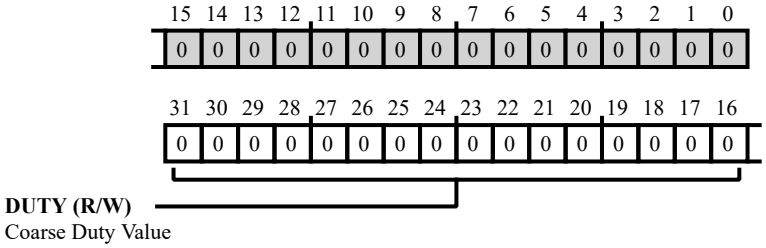


Figure 24-40: PWM_BL_DUTY1 Register Diagram

Table 24-23: PWM_BL_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_BL_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel C Control Register

The PWM_CCTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

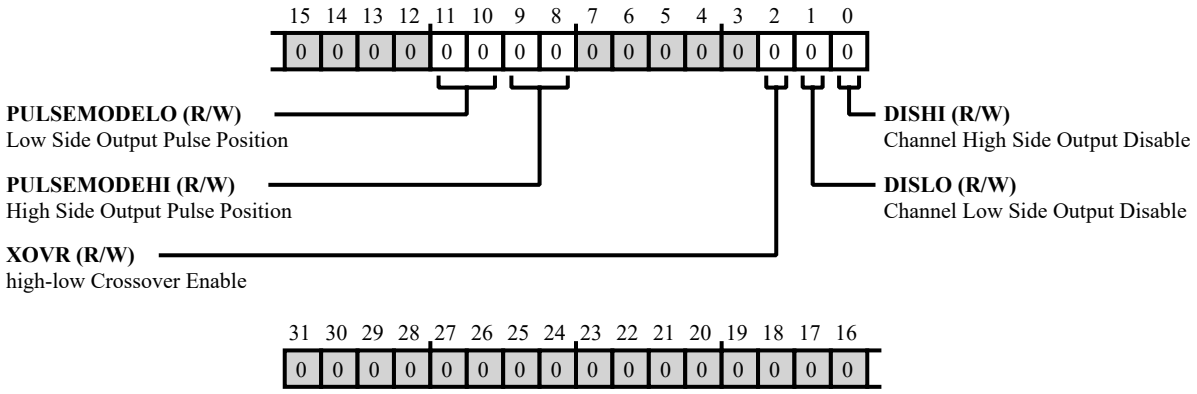


Figure 24-41: PWM_CCTL Register Diagram

Table 24-24: PWM_CCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	PULSEMODELO	Low Side Output Pulse Position. The PWM_CCTL.PULSEMODELO bits select the pulse position for Channel C low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_CL0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_CL0 and PWM_CL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_CL0 and PWM_CL1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half

Table 24-24: PWM_CCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	PULSEMODEHI	High Side Output Pulse Position. The PWM_CCTL.PULSEMODEHI bits select the pulse position for Channel C high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_CH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_CH0 and PWM_CH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_CH0 and PWM_CH1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half
2 (R/W)	XOVR	high-low Crossover Enable. The PWM_CCTL.XOVR bit enables crossover between the channels high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.	
		0	Disable Crossover
		1	Enable Crossover
1 (R/W)	DISLO	Channel Low Side Output Disable. The PWM_CCTL.DISLO bit enables the channels low side output.	
		0	Enable Low Side Output
		1	Disable Low Side Output
0 (R/W)	DISHI	Channel High Side Output Disable. The PWM_CCTL.DISHI bit enables the channels high side output.	
		0	Enable High Side Output
		1	Disable High Side Output

Channel C-High Pulse Duty Register 0

The PWM_CH0 and PWM_CH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the high side output pulses for the channel C duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_CCTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_CH0 register to determine the assertion and deassertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel C high pulse output for count less than PWM_CH0 and deasserts this output for count greater than PWM_CH1.

The value range for the PWM_CH0 and PWM_CH1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHC_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_CH0 and PWM_CH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHC_DT (two's complement) and +PWM_TM0/2 + PWM_CHC_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHC_DT (two's complement) and -PWM_TM0/2 - PWM_CHC_DT.

Note that using values in the PWM_CH0 or PWM_CH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

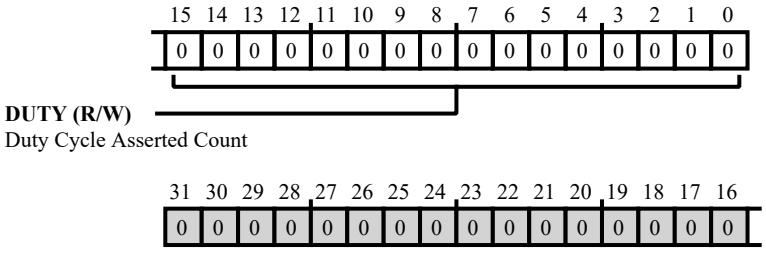


Figure 24-42: PWM_CH0 Register Diagram

Table 24-25: PWM_CH0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_CH0.DUTY bits select the duty cycle asserted count for Channel C high side output.

Channel C-High Pulse Duty Register 1

The PWM_CH0 and PWM_CH1 registers determine the width for the high side output pulses. For more information, see the PWM_CH0 register description.

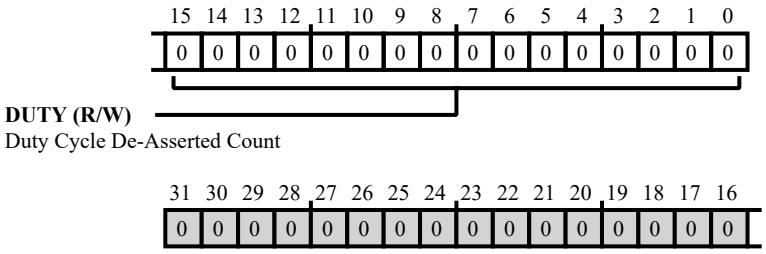


Figure 24-43: PWM_CH1 Register Diagram

Table 24-26: PWM_CH1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count. The PWM_CH1.DUTY bits select the duty cycle de-asserted count for Channel C high side output.

Channel Configuration Register

The PWM_CHANCFG register configures Channel A, B, C, and D reference timer selection, high and low side output features, and enables high frequency chopping operation. Do not change the value of any bits in the PWM register while the PWM is enabled (PWM_CTL.GLOBEN =1).

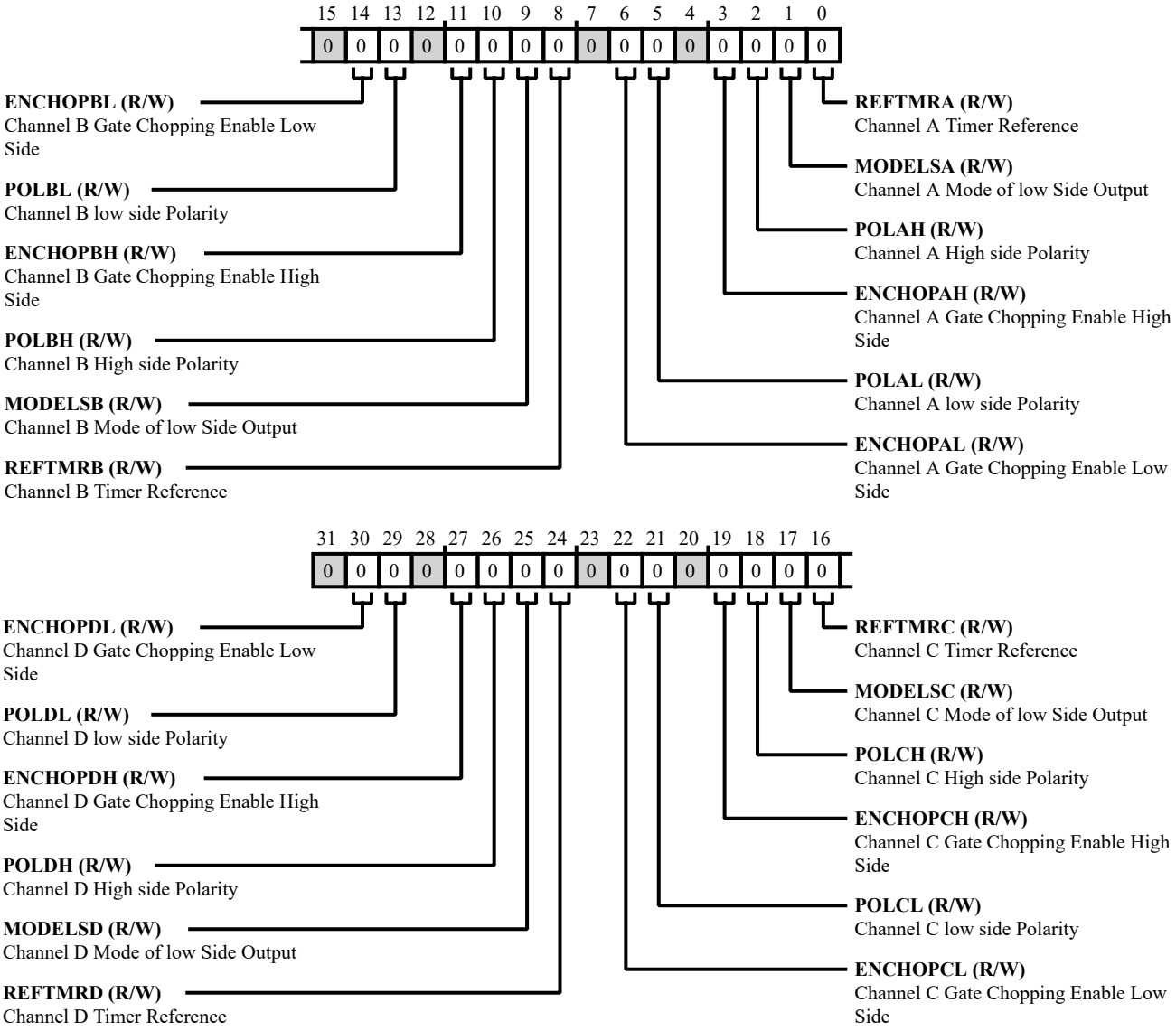


Figure 24-44: PWM_CHANCFG Register Diagram

Table 24-27: PWM_CHANCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	ENCHOPDL	Channel D Gate Chopping Enable Low Side. The PWM_CHANCFG.ENCHOPDL bit enables mixing of the Channel D low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel D Low Side
		1	Enable Chopping Channel D Low Side
29 (R/W)	POLDL	Channel D low side Polarity. The PWM_CHANCFG.POLDL bit selects the Channel D low side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High
27 (R/W)	ENCHOPDH	Channel D Gate Chopping Enable High Side. The PWM_CHANCFG.ENCHOPDH bit enables mixing of the Channel D high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel D High Side
		1	Enable Chopping Channel D High Side
26 (R/W)	POLDH	Channel D High side Polarity. The PWM_CHANCFG.POLDH bit selects the Channel D high side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High
25 (R/W)	MODELSD	Channel D Mode of low Side Output. The PWM_CHANCFG.MODELSD bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSD =0, the low side output is an inverted form of the high side output, which is generated using the PWM_BH0 and PWM_BH1 registers for pulse width, using the PWM_BCTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLBH bits for polarity.	
		0	Invert of high output
		1	Independent control

Table 24-27: PWM_CHANCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	REFTMRD	Channel D Timer Reference. The PWM_CHANCFG.REFTMRD bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel D operation.	
		0	PWMTMR0 is Channel D reference
		1	PWMTMR4 is Channel D reference
22 (R/W)	ENCHOPCL	Channel C Gate Chopping Enable Low Side. The PWM_CHANCFG.ENCHOPCL bit enables mixing of the Channel C low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel C Low Side
		1	Enable Chopping Channel C Low Side
21 (R/W)	POLCL	Channel C low side Polarity. The PWM_CHANCFG.POLCL bit selects the Channel C low side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High
19 (R/W)	ENCHOPCH	Channel C Gate Chopping Enable High Side. The PWM_CHANCFG.ENCHOPCH bit enables mixing of the Channel C high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel C High Side
		1	Enable Chopping Channel C High Side
18 (R/W)	POLCH	Channel C High side Polarity. The PWM_CHANCFG.POLCH bit selects the Channel C high side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High

Table 24-27: PWM_CHANCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	MODELSC	Channel C Mode of low Side Output. The PWM_CHANCFG.MODELSC bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSC =0, the low side output is an inverted form of the high side output, which is generated using the PWM_BH0 and PWM_BH1 registers for pulse width, using the PWM_BCTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLBH bits for polarity.	
		0	Invert of high output
		1	Independent control
16 (R/W)	REFTMRC	Channel C Timer Reference. The PWM_CHANCFG.REFTMRC bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel C operation.	
		0	PWMTMR0 is Channel C reference
		1	PWMTMR3 is Channel C reference
14 (R/W)	ENCHOPBL	Channel B Gate Chopping Enable Low Side. The PWM_CHANCFG.ENCHOPBL bit enables mixing of the Channel B low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel B Low Side
		1	Enable Chopping Channel B Low Side
13 (R/W)	POLBL	Channel B low side Polarity. The PWM_CHANCFG.POLBL bit selects the Channel B low side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High
11 (R/W)	ENCHOPBH	Channel B Gate Chopping Enable High Side. The PWM_CHANCFG.ENCHOPBH bit enables mixing of the Channel B high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel B High Side
		1	Enable Chopping Channel B High Side

Table 24-27: PWM_CHANCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	POLBH	Channel B High side Polarity. The PWM_CHANCFG.POLBH bit selects the Channel B high side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High
9 (R/W)	MODELSB	Channel B Mode of low Side Output. The PWM_CHANCFG.MODELSB bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSB =0, the low side output is an inverted form of the high side output, which is generated using the PWM_BH0 and PWM_BH1 registers for pulse width, using the PWM_BCTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLBH bits for polarity.	
		0	Invert of high output
		1	Independent control
8 (R/W)	REFTMRB	Channel B Timer Reference. The PWM_CHANCFG.REFTMRB bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel B operation.	
		0	PWMTMR0 is Channel B reference
		1	PWMTMR2 is Channel B reference
6 (R/W)	ENCHOPAL	Channel A Gate Chopping Enable Low Side. The PWM_CHANCFG.ENCHOPAL bit enables mixing of the Channel A low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel A Low Side
		1	Enable Chopping Channel A Low Side
5 (R/W)	POLAL	Channel A low side Polarity. The PWM_CHANCFG.POLAL bit selects the Channel A low side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High

Table 24-27: PWM_CHANCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	ENCHOPAH	Channel A Gate Chopping Enable High Side. The PWM_CHANCFG.ENCHOPAH bit enables mixing of the Channel A high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.	
		0	Disable Chopping Channel A High Side
		1	Enable Chopping Channel A High Side
2 (R/W)	POLAH	Channel A High side Polarity. The PWM_CHANCFG.POLAH bit selects the Channel A high side output polarity (active-high or active-low).	
		0	Active Low
		1	Active High
1 (R/W)	MODELSA	Channel A Mode of low Side Output. The PWM_CHANCFG.MODELSA bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSA =0, the low side output is an inverted form of the high side output, which is generated using the PWM_AH0 and PWM_AH1 registers for pulse width, using the PWM_ACTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLAH bits for polarity.	
		0	Invert of high output
		1	Independent control
0 (R/W)	REFTMRA	Channel A Timer Reference. The PWM_CHANCFG.REFTMRA bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel A operation.	
		0	PWMTMR0 is Channel A reference
		1	PWMTMR1 is Channel A reference

Channel A Dead-time Register

The PWM_CHA_DT register controls the value of dead-time for channel A independently.

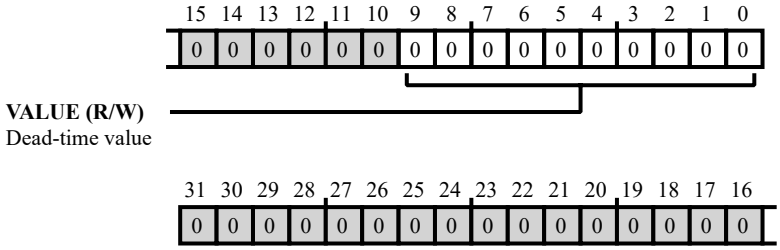


Figure 24-45: PWM_CHA_DT Register Diagram

Table 24-28: PWM_CHA_DT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Dead-time value.

Channel B Dead-time Register

The PWM_CHB_DT register controls the value of dead-time for channel B independently.

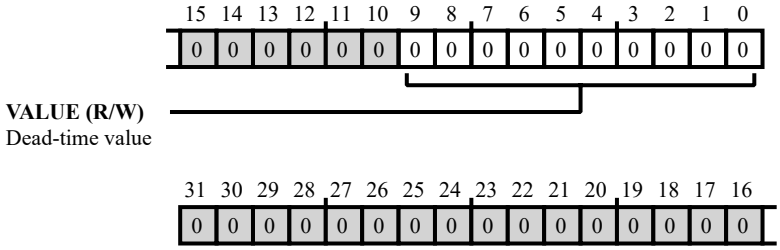


Figure 24-46: PWM_CHB_DT Register Diagram

Table 24-29: PWM_CHB_DT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Dead-time value. The PWM_CHB_DT.VALUE bit field contains the dead-time value.

Channel C Dead-time Register

The PWM_CHC_DT register controls the value of dead-time for channel C independently.

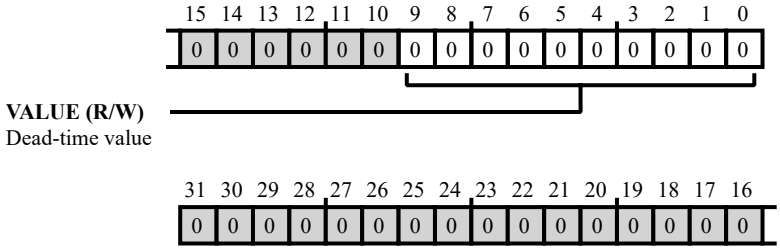


Figure 24-47: PWM_CHC_DT Register Diagram

Table 24-30: PWM_CHC_DT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Dead-time value. The PWM_CHC_DT.VALUE bit field contains the dead-time value.

Channel D Dead-time Register

The PWM_CHD_DT register controls the value of dead-time for channel D independently.

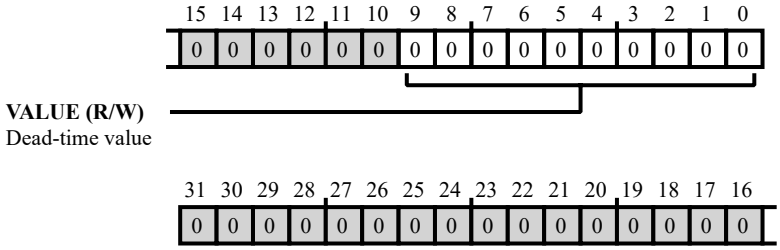


Figure 24-48: PWM_CHD_DT Register Diagram

Table 24-31: PWM_CHD_DT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Dead-time value. The PWM_CHD_DT.VALUE bit field contains the dead-time value.

Chop Configuration Register

The PWM_CHOPCFG register holds a divisor value that controls the chopping frequency. The PWM permits a mixing of the output signals with a high-frequency chopping signal to aid with interfacing to pulse transformers. Also note that high-frequency chopping may be independently enabled for each channel's high-side and the low-side outputs using channel control bits. (For example, control chopping for Channel A with the PWM_CHANCFG.ENCHOPAH and PWM_CHANCFG.ENCHOPAH bits.)

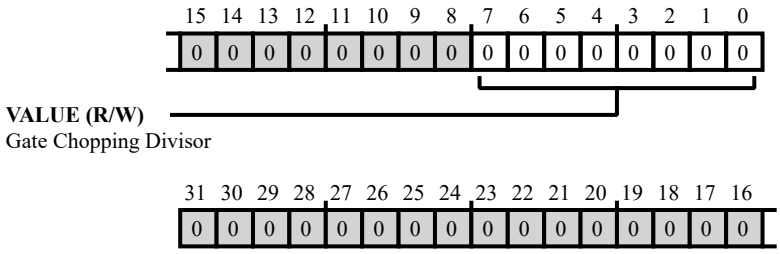


Figure 24-49: PWM_CHOPCFG Register Diagram

Table 24-32: PWM_CHOPCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Gate Chopping Divisor. The PWM_CHOPCFG.VALUE bits provide the high frequency chopping divisor. When the divisor value is changed, the new period takes effect from the next edge of the chopping signal. The PWM_CHOPCFG.VALUE value may be calculated using either of the following formulas: CHOPDIV = $[(T_{CHOP}/T_{CK}) / 4] - 1$ CHOPDIV = $[(f_{CK} / f_{CHOP}) / 4] - 1$

Channel C-High Full Duty0 Register

The PWM_CH_DUTY0 register contains the PWM_CH_DUTY0.DUTY bit field from the PWM_CH0 register.

Note that the PWM_CH_DUTY0 register reads the PWM_CH0 register values and visa-versa.

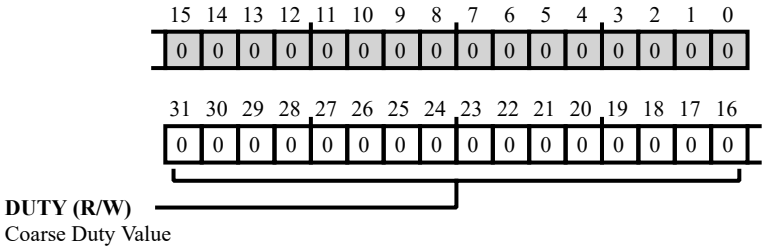


Figure 24-50: PWM_CH_DUTY0 Register Diagram

Table 24-33: PWM_CH_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_CH_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel C-High Full Duty1 Register

The PWM_CH_DUTY1 register contains the PWM_CH_DUTY1.DUTY bit field from the PWM_CH1 register.

Note that the PWM_CH_DUTY1 register reads the PWM_CH1 register values and visa-versa.

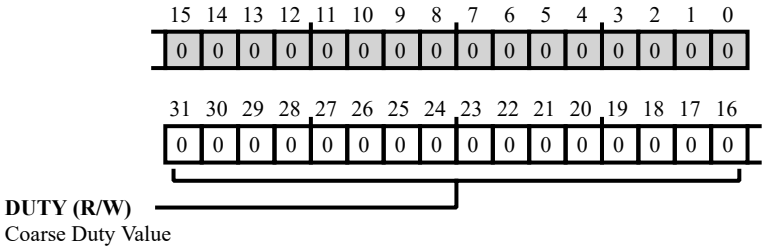


Figure 24-51: PWM_CH_DUTY1 Register Diagram

Table 24-34: PWM_CH_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_CH_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel C-Low Pulse Duty Register 0

The PWM_CL0 and PWM_CL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the low side output pulses for the channel C duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_CCTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_CL0 register to determine the assertion and deassertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel C low pulse output for count less than PWM_CL0 and deasserts this output for count greater than PWM_CL1.

The value range for the PWM_CL0 and PWM_CL1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHC_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_CL0 and PWM_CL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHC_DT (two's complement) and +PWM_TM0/2 + PWM_CHC_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHC_DT (two's complement) to -PWM_TM0/2 - PWM_CHC_DT.

Note that using values in the PWM_CL0 or PWM_CL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

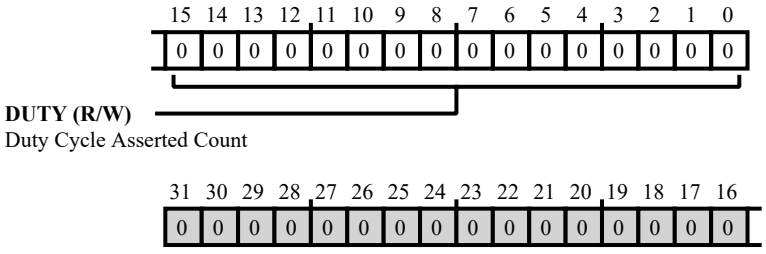


Figure 24-52: PWM_CL0 Register Diagram

Table 24-35: PWM_CL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_CL0.DUTY bits select the duty cycle asserted count for Channel C low side output.

Channel C-Low Duty-1 Register

The PWM_CL0 and PWM_CL1 registers determine the width for the low side output pulses. For more information, see the PWM_CL0 register description.

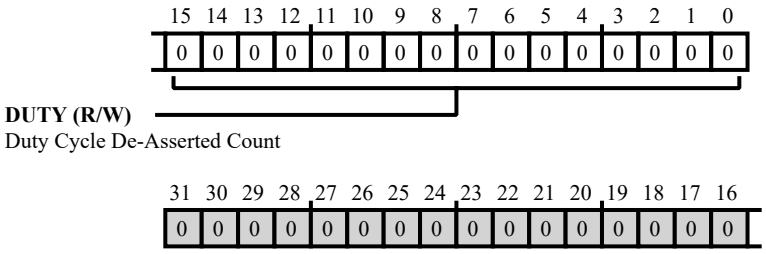


Figure 24-53: PWM_CL1 Register Diagram

Table 24-36: PWM_CL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count.

Channel C-Low Full Duty0 Register

The PWM_CL_DUTY0 register contains the PWM_CL_DUTY0.DUTY bit field from the PWM_CL0 register.

Note that the PWM_CL_DUTY0 register reads the PWM_CL0 register values and visa-versa.

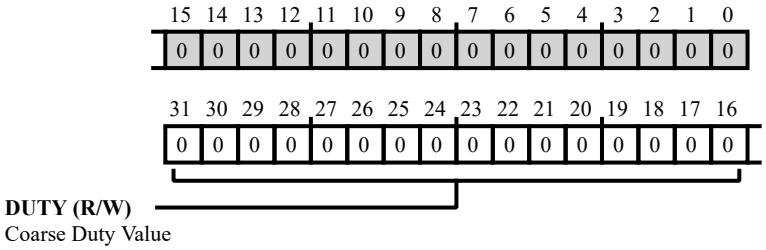


Figure 24-54: PWM_CL_DUTY0 Register Diagram

Table 24-37: PWM_CL_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_CL_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel C-Low Full Duty1 Register

The PWM_CL_DUTY1 register contains the PWM_CL_DUTY1.DUTY bit field from the PWM_CL1 register.

Note that the PWM_CL_DUTY1 register reads the PWM_CL1 and visa-versa.

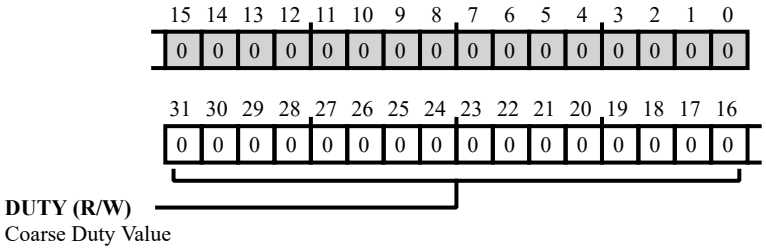


Figure 24-55: PWM_CL_DUTY1 Register Diagram

Table 24-38: PWM_CL_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_CL_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Control Register

The PWM_CTL register enables the PWM, enables delay counters for the channels, and configures sync features. This register also provides support for tripping a PWM fault condition through software.

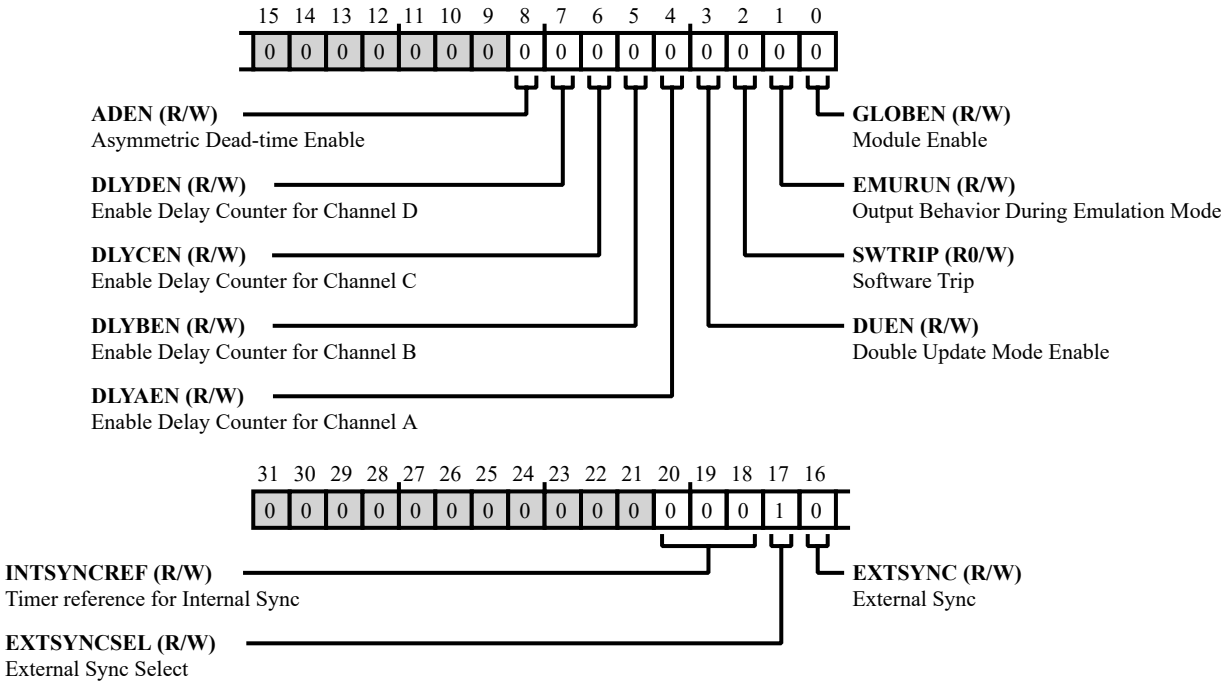


Figure 24-56: PWM_CTL Register Diagram

Table 24-39: PWM_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:18 (R/W)	INTSYNCREF	Timer reference for Internal Sync. The PWM_CTL.INTSYNCREF bits select the timer reference for the internal sync. Note that all other combinations reserved.	
		0	PWMTMR0 provides sync reference
		1	PWMTMR1 provides sync reference
		2	PWMTMR2 provides sync reference
		3	PWMTMR3 provides sync reference
		4	PWMTMR4 provides sync reference

Table 24-39: PWM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	EXTSYNCSEL	External Sync Select. The PWM_CTL.EXTSYNCSEL bit selects whether the external sync signal is synchronous or asynchronous. Note that latency in PWM sync response differs between asynchronous and synchronous external sync modes. For more information, see the PWM functional description.	
		0	Asynchronous External Sync
		1	Synchronous External Sync
16 (R/W)	EXTSYNC	External Sync. The PWM_CTL.EXTSYNC bit selects whether the PWM uses an external or internal sync signal for the main timer (PWMTMR0). Do not change the value of the PWM_CTL.EXTSYNC bit while the PWM is enabled (PWM_CTL.GLOBEN =1).	
		0	Internal sync used
		1	External sync used
8 (R/W)	ADEN	Asymmetric Dead-time Enable. When symmetric dead-time is enabled (PWM_CTL.ADEN = 0), in the dependent mode, both the high-side and low-side outputs are reduced by DT cycles on both the assertion and de-assertion edges. When symmetric dead-time is disabled (PWM_CTL.ADEN = 1), the falling edges of both high and low-side outputs occur at the programmed duty value, but their rise-times are delayed by 2 x DT.	
		0	Dead-time is symmetric
		1	Dead-time is asymmetric
7 (R/W)	DLYDEN	Enable Delay Counter for Channel D. The PWM_CTL.DLYDEN bit enables the Channel D delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYDEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).	
		0	Disable
		1	Enable
6 (R/W)	DLYCEN	Enable Delay Counter for Channel C. The PWM_CTL.DLYCEN bit enables the Channel C delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYCEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).	
		0	Disable
		1	Enable

Table 24-39: PWM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DLYBEN	Enable Delay Counter for Channel B. The PWM_CTL.DLYBEN bit enables the Channel B delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYBEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).	
		0	Disable
		1	Enable
4 (R/W)	DLYAEN	Enable Delay Counter for Channel A. The PWM_CTL.DLYAEN bit enables the Channel A delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYAEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).	
		0	Disable
		1	Enable
3 (R/W)	DUEN	Double Update Mode Enable. In Single Update Mode, double buffering of all registers happens at the period boundary of the PWM timer. In Double Update Mode, double buffering of all registers (except delay counter registers) happens at the middle of the period as well as the beginning of the period.	
		0	Single Update Mode
		1	Double Update Mode
2 (R0/W)	SWTRIP	Software Trip. The PWM_CTL.SWTRIP bit permits tripping a fault condition through software, shutting down PWM output. This bit always read as 0. If the PWM_CTL.SWTRIP bit and PWM_CTL.GLOBEN bit are set in the same write, the write does not trip the fault condition.	
		0	Don't force fault
		1	Force a Fault Trip Condition
1 (R/W)	EMURUN	Output Behavior During Emulation Mode. The PWM_CTL.EMURUN bit selects PWM output behavior during emulation mode.	
		0	Disable Outputs
		1	Enable Outputs

Table 24-39: PWM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	GLOBEN	Module Enable. The PWM_CTL.GLOBEN bit enables the PWM, enabling all timers and outputs. While this bit is enabled, processor code should not change the value of the PWM_CTL.DLYAEN bit, PWM_CTL.DLYBEN bit, PWM_CTL.DLYCEN bit, PWM_CTL.DLYDEN bit, PWM_CTL.EXTSYNCSEL bit, or any bits in the PWM_CHANCFG register. Note that there is a latency between PWM disable and the cessation of output waveforms. There is also a latency between PWM enable and start of output waveforms. For the latency description, see the PWM functional description.	
		0	Disable
		1	Enable

Channel D Control Register

The PWM_DCTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

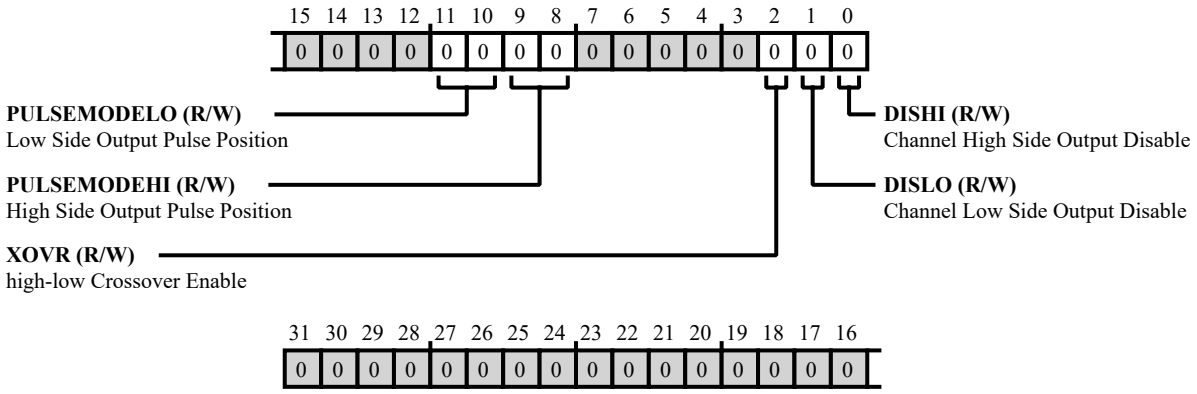


Figure 24-57: PWM_DCTL Register Diagram

Table 24-40: PWM_DCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	PULSEMODELO	Low Side Output Pulse Position. The PWM_DCTL.PULSEMODELO bits select the pulse position for Channel D low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_DL0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_DL0 and PWM_DL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_DL0 and PWM_DL1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half

Table 24-40: PWM_DCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	PULSEMODEHI	High Side Output Pulse Position. The PWM_DCTL.PULSEMODEHI bits select the pulse position for Channel D high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the center of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_DH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the center of the PWM period. This mode uses both the duty cycle registers (PWM_DH0 and PWM_DH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty cycle registers (PWM_DH0 and PWM_DH1).	
		0	Symmetrical
		1	Asymmetrical
		2	Left Half
		3	Right Half
2 (R/W)	XOVR	high-low Crossover Enable. The PWM_DCTL.XOVR bit enables crossover between the channels high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.	
		0	Disable Crossover
		1	Enable Crossover
1 (R/W)	DISLO	Channel Low Side Output Disable. The PWM_DCTL.DISLO bit enables the channels low side output.	
		0	Enable Low Side Output
		1	Disable Low Side Output
0 (R/W)	DISHI	Channel High Side Output Disable. The PWM_DCTL.DISHI bit enables the channels high side output.	
		0	Enable High Side Output
		1	Disable High Side Output

Channel D-High Duty-0 Register

The PWM_DH0 and PWM_DH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the high side output pulses for the channel D duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_DCTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_DH0 register to determine the assertion and deassertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel D high pulse output for count less than PWM_DH0 and deasserts this output for count greater than PWM_DH1.

The value range for the PWM_DH0 and PWM_DH1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHD_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_DH0 and PWM_DH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHD_DT (two's complement) to +PWM_TM0/2 + PWM_CHD_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHD_DT (two's complement) to -PWM_TM0/2 - PWM_CHD_DT.

Note that using values in the PWM_DH0 or PWM_DH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

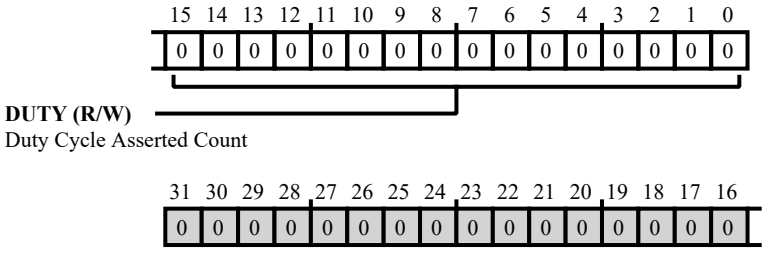


Figure 24-58: PWM_DH0 Register Diagram

Table 24-41: PWM_DH0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_DH0.DUTY bits select the duty cycle asserted count for Channel D high side output.

Channel D-High Pulse Duty Register 1

The PWM_DH0 and PWM_DH1 registers determine the width for the high side output pulses. For more information, see the PWM_DH0 register description.

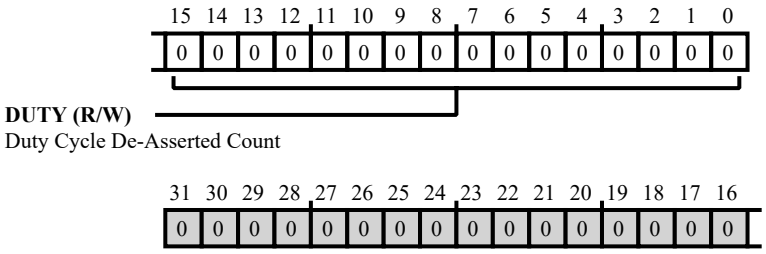


Figure 24-59: PWM_DH1 Register Diagram

Table 24-42: PWM_DH1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count. The PWM_DH1.DUTY bits select the duty cycle de-asserted count for Channel D high side output.

Channel D-High Full Duty0 Register

The PWM_DH_DUTY0 register contains the PWM_DH_DUTY0.DUTY bit field from the PWM_DH0 register.

Note that the PWM_DH_DUTY0 register reads the PWM_DH0 register values and visa-versa.

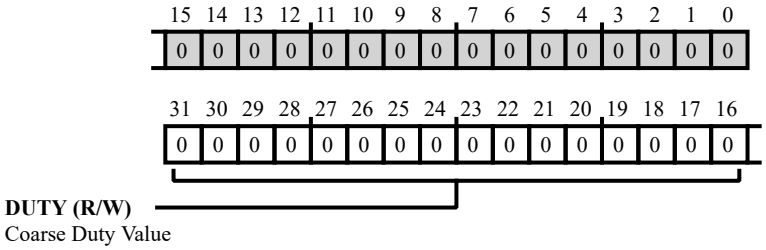


Figure 24-60: PWM_DH_DUTY0 Register Diagram

Table 24-43: PWM_DH_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_DH_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel D-High Full Duty1 Register

The PWM_DH_DUTY1 register contains the PWM_DH_DUTY1.DUTY bit field from the PWM_DH1 register.

Note that the PWM_DH_DUTY1 register reads the PWM_DH1 register values and vice-versa.

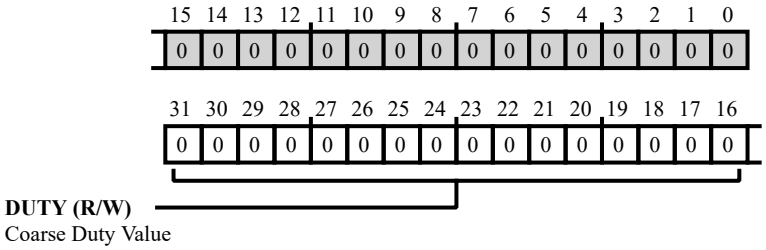


Figure 24-61: PWM_DH_DUTY1 Register Diagram

Table 24-44: PWM_DH_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_DH_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel D-Low Pulse Duty Register 0

The PWM_DL0 and PWM_DL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of t_{CK}) of the low side output pulses for the channel D duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM_DCTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM_DL0 register to determine the assertion and deassertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts channel D low pulse output for count less than PWM_DL0 and deasserts this output for count greater than PWM_DL1.

The value range for the PWM_DL0 and PWM_DL1 registers depends on the period of the timer being used by the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 (two's complement) and +PWM_TM0/2, when dead time (PWM_CHD_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM_DL0 and PWM_DL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM_TM0 is used, the duty cycle values may be between -PWM_TM0/2 + PWM_CHD_DT (two's complement) and +PWM_TM0/2 + PWM_CHD_DT.

When dead time is considered for left half or right half pulse modes, if PWM_TM0 is used, the duty cycle values may be between PWM_TM0/2 + PWM_CHD_DT (two's complement) and -PWM_TM0/2 - PWM_CHD_DT.

Note that using values in the PWM_DL0 or PWM_DL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

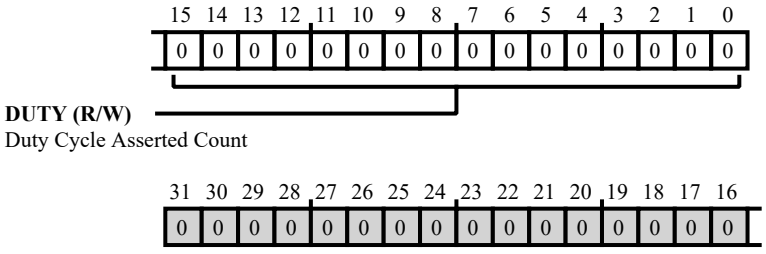


Figure 24-62: PWM_DL0 Register Diagram

Table 24-45: PWM_DL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle Asserted Count. The PWM_DL0.DUTY bits select the duty cycle asserted count for Channel D low side output.

Channel D-Low Pulse Duty Register 1

The PWM_DL0 and PWM_DL1 registers determine the width for the low side output pulses. For more information, see the PWM_DL0 register description.

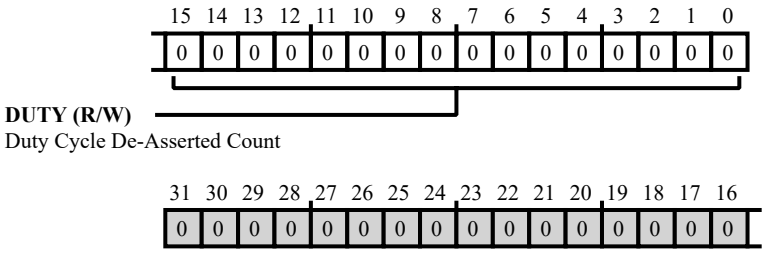


Figure 24-63: PWM_DL1 Register Diagram

Table 24-46: PWM_DL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	DUTY	Duty Cycle De-Asserted Count. The PWM_DL1.DUTY bits select the duty cycle de-asserted count for Channel D low side output.

Channel A Delay Register

The PWM_DLYA register controls a delay for the Channel A timer (only PWMTMR1, PWMTMR2, PWMTMR3 or PWMTMR4) with reference to the main timer (PWMTMR0). To use apply this delay, the delay must be enabled (PWM_CTL.DLYAEN =1). For more information about applying the delay, see the PWM Functional Description section. Note that the PWM_DLYA delay value must be less that less that twice the period value of the timer being used for the channel (for example, if PWMTMR1 is used, PWM_DLYA must be less than 2xPWMTMR1). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if PWMTMR1 is used, PWMTMR0 = NxPWMTMR1, where N is an integer).

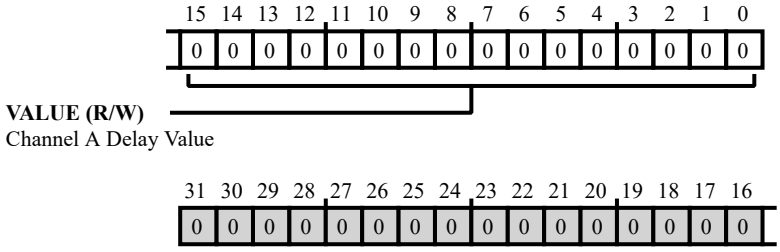


Figure 24-64: PWM_DLYA Register Diagram

Table 24-47: PWM_DLYA Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Channel A Delay Value. The PWM_DLYA.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel A.

Channel B Delay Register

The PWM_DLYB register controls a delay for the Channel B timer (only PWMTMR1, PWMTMR2, PWMTMR3 or PWMTMR4) with reference to the main timer (PWMTMR0). To use apply this delay, the delay must be enabled (PWM_CTL.DLYBEN =1). For more information about applying the delay, see the PWM Functional Description section. Note that the PWM_DLYB delay value must be less that less that twice the period value of the timer being used for the channel (for example, if PWMTMR1 is used, PWM_DLYB must be less than 2xPWMTMR1). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if PWMTMR1 is used, PWMTMR0 = NxPWMTMR1, where N is an integer).

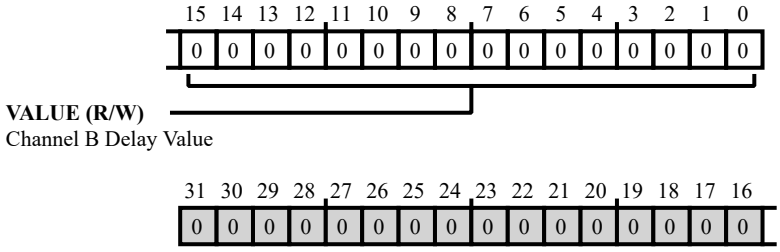


Figure 24-65: PWM_DLYB Register Diagram

Table 24-48: PWM_DLYB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Channel B Delay Value. The PWM_DLYB.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel B.

Channel C Delay Register

The PWM_DLYC register controls a delay for the Channel C timer (only PWMTMR1, PWMTMR2, PWMTMR3 or PWMTMR4) with reference to the main timer (PWMTMR0). To use apply this delay, the delay must be enabled (PWM_CTL.DLYCEN =1). For more information about applying the delay, see the PWM Functional Description section. Note that the PWM_DLYC delay value must be less that less that twice the period value of the timer being used for the channel (for example, if PWMTMR1 is used, PWM_DLYC must be less than 2xPWMTMR1). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if PWMTMR1 is used, PWMTMR0 = NxPWMTMR1, where N is an integer).

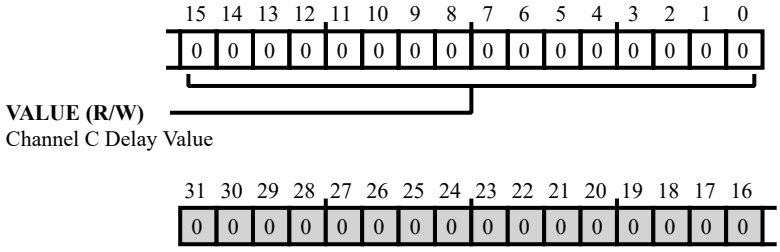


Figure 24-66: PWM_DLYC Register Diagram

Table 24-49: PWM_DLYC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Channel C Delay Value. The PWM_DLYC.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel C.

Channel D Delay Register

The PWM_DLYD register controls a delay for the Channel D timer (only PWMTMR1, PWMTMR2, PWMTMR3 or PWMTMR4) with reference to the main timer (PWMTMR0). To use apply this delay, the delay must be enabled (PWM_CTL.DLYDEN =1). For more information about applying the delay, see the PWM Functional Description section. Note that the PWM_DLYD delay value must be less that less that twice the period value of the timer being used for the channel (for example, if PWMTMR1 is used, PWM_DLYD must be less than 2xPWMTMR1). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if PWMTMR1 is used, PWMTMR0 = NxPWMTMR1, where N is an integer).

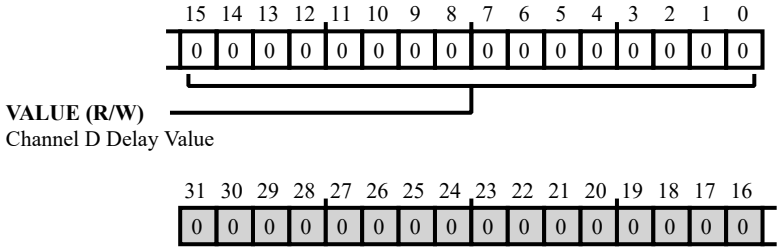


Figure 24-67: PWM_DLYD Register Diagram

Table 24-50: PWM_DLYD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Channel D Delay Value. The PWM_DLYD.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel D.

Channel D-Low Full Duty0 Register

The PWM_DL_DUTY0 register contains the PWM_DL_DUTY0.DUTY bit field from the PWM_DL0 register.

Note that the PWM_DL_DUTY0 register reads the PWM_DL0 register values and visa-versa.

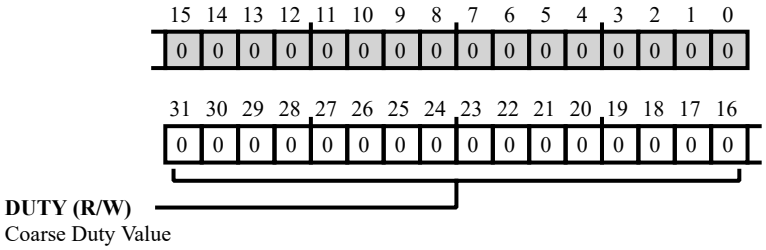


Figure 24-68: PWM_DL_DUTY0 Register Diagram

Table 24-51: PWM_DL_DUTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_DL_DUTY0.DUTY bits determine the output pulse-widths in the normal PWM operation.

Channel D-Low Full Duty1 Register

The PWM_DL_DUTY1 register contains the PWM_DL_DUTY1.DUTY bit field from the PWM_DL1 register.

Note that the PWM_DL_DUTY1 register reads the PWM_DL1 register values and visa-versa.

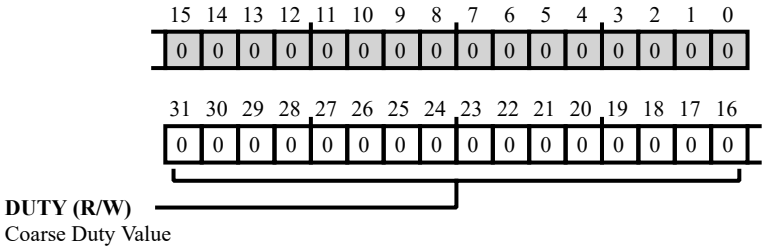


Figure 24-69: PWM_DL_DUTY1 Register Diagram

Table 24-52: PWM_DL_DUTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	DUTY	Coarse Duty Value. The PWM_DL_DUTY1.DUTY bits determine the output pulse-widths in the normal PWM operation.

Interrupt Latch Register

The PWM_ILAT register latches the occurrence of unmasked (enabled) PWM interrupt requests. These interrupt requests are unmasked or masked with the PWM_IMSK register.

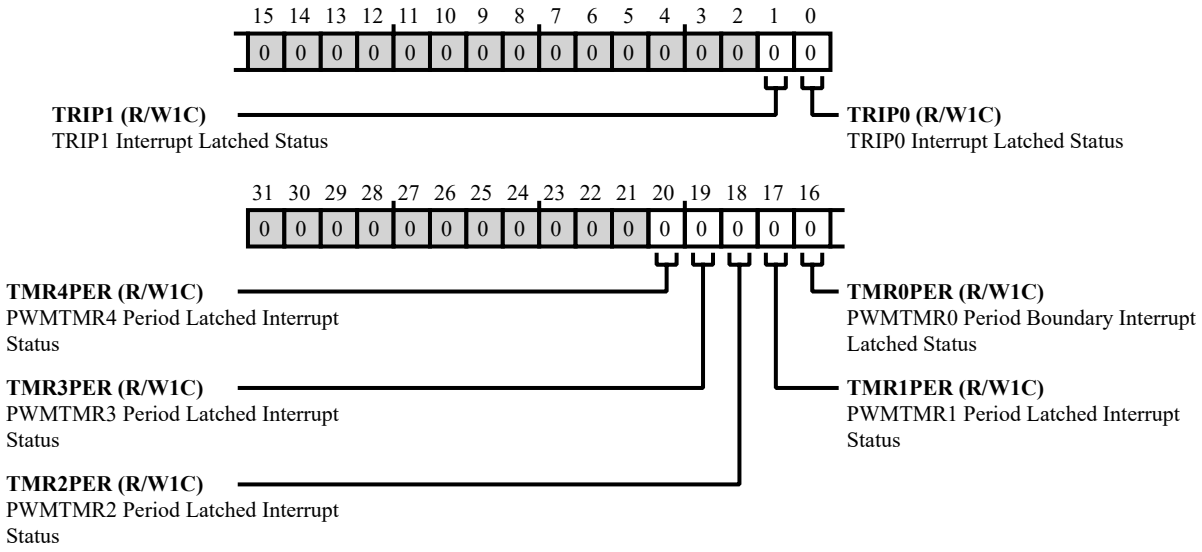


Figure 24-70: PWM_ILAT Register Diagram

Table 24-53: PWM_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W1C)	TMR4PER	PWMTMR4 Period Latched Interrupt Status. The PWM_ILAT.TMR4PER bit indicates the latched status of the PWMTMR4 period boundary interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched
19 (R/W1C)	TMR3PER	PWMTMR3 Period Latched Interrupt Status. The PWM_ILAT.TMR3PER bit indicates the latched status of the PWMTMR3 period boundary interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched
18 (R/W1C)	TMR2PER	PWMTMR2 Period Latched Interrupt Status. The PWM_ILAT.TMR2PER bit indicates the latched status of the PWMTMR2 period boundary interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched

Table 24-53: PWM_ILAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	TMR1PER	PWMTMR1 Period Latched Interrupt Status. The `PWM_ILAT.TMR1PER` bit indicates the latched status of the PWMTMR1 period boundary interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched
16 (R/W1C)	TMR0PER	PWMTMR0 Period Boundary Interrupt Latched Status. The `PWM_ILAT.TMR0PER` bit indicates the latched status of the PWMTMR0 period boundary interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched
1 (R/W1C)	TRIP1	TRIP1 Interrupt Latched Status. The `PWM_ILAT.TRIP1` bit indicates the latched status of the TRIP1 interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched
0 (R/W1C)	TRIP0	TRIP0 Interrupt Latched Status. The `PWM_ILAT.TRIP0` bit indicates the latched status of the TRIP0 interrupt request.	
		0	No Interrupt Latched
		1	Interrupt Latched

Interrupt Mask Register

The PWM_IMSK register masks (disables) or unmasks (enables) PWM interrupt requests. When an unmasked interrupt occurs, the PWM latches the interrupt status in the PWM_ILAT register.

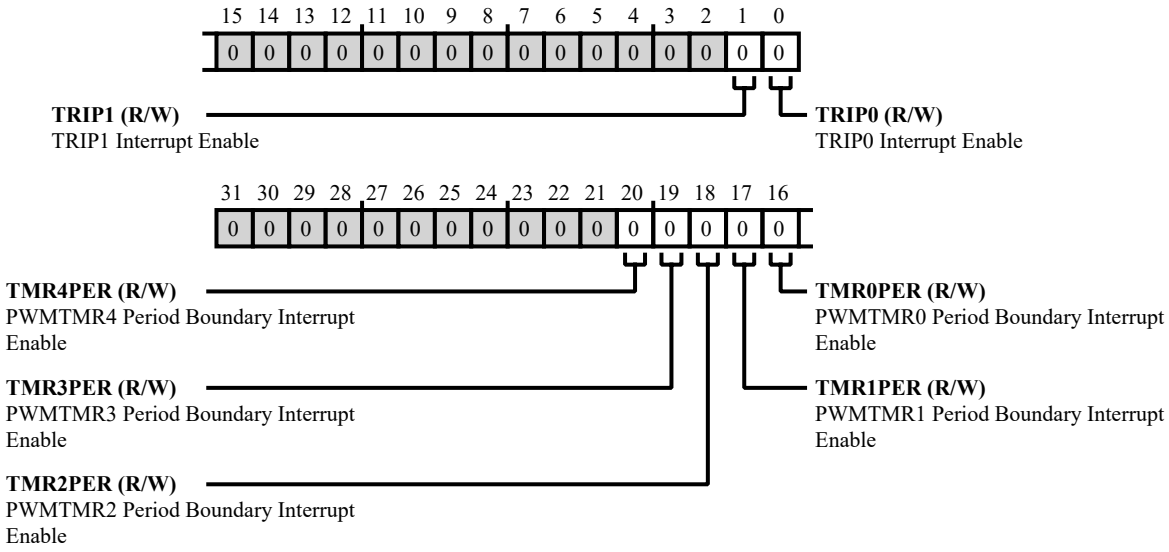


Figure 24-71: PWM_IMSK Register Diagram

Table 24-54: PWM_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	TMR4PER	PWMTMR4 Period Boundary Interrupt Enable. The PWM_IMSK.TMR4PER bit enables (unmasks) the PWMTMR4 period boundary interrupt request. This condition occurs when the timers period boundary is reached (PWM_STAT.TMR4PER =1).	
		0	Mask PWMTMR4 Period Interrupt
		1	Unmask PWMTMR4 Period Interrupt
19 (R/W)	TMR3PER	PWMTMR3 Period Boundary Interrupt Enable. The PWM_IMSK.TMR3PER bit enables (unmasks) the PWMTMR3 period boundary interrupt request. This condition occurs when the timers period boundary is reached (PWM_STAT.TMR3PER =1).	
		0	Mask PWMTMR3 Period Interrupt
		1	Unmask PWMTMR3 Period Interrupt

Table 24-54: PWM_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	TMR2PER	PWMTMR2 Period Boundary Interrupt Enable. The PWM_IMSK.TMR2PER bit enables (unmasks) the PWMTMR2 period boundary interrupt request. This condition occurs when the timers period boundary is reached (PWM_STAT.TMR2PER =1).	
		0	Mask PWMTMR2 Period Interrupt
		1	Unmask PWMTMR2 Period Interrupt
17 (R/W)	TMR1PER	PWMTMR1 Period Boundary Interrupt Enable. The PWM_IMSK.TMR1PER bit enables (unmasks) the PWMTMR1 period boundary interrupt request. This condition occurs when the timers period boundary is reached (PWM_STAT.TMR1PER =1).	
		0	Mask PWMTMR1 Period Interrupt
		1	Unmask PWMTMR1 Period Interrupt
16 (R/W)	TMR0PER	PWMTMR0 Period Boundary Interrupt Enable. The PWM_IMSK.TMR0PER bit enables (unmasks) the PWMTMR0 period boundary interrupt request. This condition occurs when the timers period boundary is reached (PWM_STAT.TMR0PER =1).	
		0	Mask PWMTMR0 Period Interrupt
		1	Unmask PWMTMR0 Period Interrupt
1 (R/W)	TRIP1	TRIP1 Interrupt Enable. The PWM_IMSK.TRIP1 bit enables (unmasks) the TRIP1 interrupt request. This condition occurs when fault input is tripped (PWM_STAT.TRIP1 =1).	
		0	Mask TRIP1 Interrupt
		1	Unmask TRIP1 Interrupt
0 (R/W)	TRIP0	TRIP0 Interrupt Enable. The PWM_IMSK.TRIP0 bit enables (unmasks) the TRIP0 interrupt request. This condition occurs when fault input is tripped (PWM_STAT.TRIP0 =1).	
		0	Mask TRIP0 Interrupt
		1	Unmask TRIP0 Interrupt

Status Register

The PWM_STAT register indicates the PWM PWMTRIP1-0 fault and input level status, indicates the Channel A-D fault and self-restart status, and indicates the PWMTMR4-0 phase.

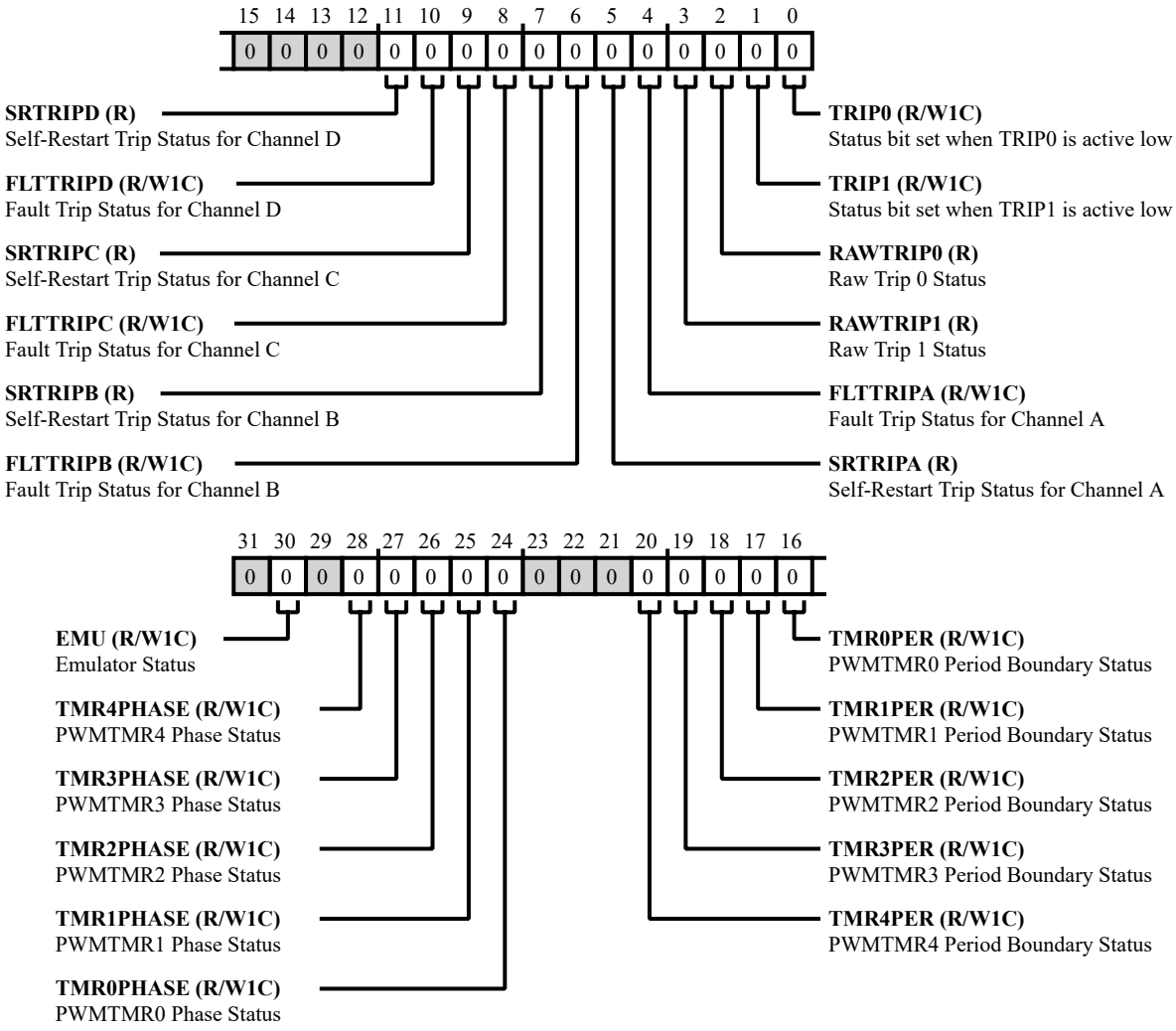


Figure 24-72: PWM_STAT Register Diagram

Table 24-55: PWM_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W1C)	EMU	Emulator Status.

Table 24-55: PWM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W1C)	TMR4PHASE	PWMTMR4 Phase Status. The PWM_STAT.TMR4PHASE bit indicates the current phase for the PWMTMR4 waveform.	
		0	1st Half Phase
		1	2nd Half Phase
27 (R/W1C)	TMR3PHASE	PWMTMR3 Phase Status. The PWM_STAT.TMR3PHASE bit indicates the current phase for the PWMTMR3 waveform.	
		0	1st Half Phase
		1	2nd Half Phase
26 (R/W1C)	TMR2PHASE	PWMTMR2 Phase Status. The PWM_STAT.TMR2PHASE bit indicates the current phase for the PWMTMR2 waveform.	
		0	1st Half Phase
		1	2nd Half Phase
25 (R/W1C)	TMR1PHASE	PWMTMR1 Phase Status. The PWM_STAT.TMR1PHASE bit indicates the current phase for the PWMTMR1 waveform.	
		0	1st Half Phase
		1	2nd Half Phase
24 (R/W1C)	TMR0PHASE	PWMTMR0 Phase Status. The PWM_STAT.TMR0PHASE bit indicates the current phase for the PWMTMR0 waveform.	
		0	1st Half Phase
		1	2nd Half Phase
20 (R/W1C)	TMR4PER	PWMTMR4 Period Boundary Status. The PWM_STAT.TMR4PER bit indicates whether or not the PWMTMR4 period boundary has been reached.	
		0	PWMTMR4 period boundary not reached
		1	PWMTMR4 period boundary reached

Table 24-55: PWM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W1C)	TMR3PER	PWMTMR3 Period Boundary Status. The PWM_STAT.TMR3PER bit indicates whether or not the PWMTMR3 period boundary has been reached.	
		0	PWMTMR3 period boundary not reached
		1	PWMTMR3 period boundary reached
18 (R/W1C)	TMR2PER	PWMTMR2 Period Boundary Status. The PWM_STAT.TMR2PER bit indicates whether or not the PWMTMR2 period boundary has been reached.	
		0	PWMTMR2 period boundary not reached
		1	PWMTMR2 period boundary reached
17 (R/W1C)	TMR1PER	PWMTMR1 Period Boundary Status. The PWM_STAT.TMR1PER bit indicates whether or not the PWMTMR1 period boundary has been reached.	
		0	PWMTMR1 period boundary not reached
		1	PWMTMR1 period boundary reached
16 (R/W1C)	TMR0PER	PWMTMR0 Period Boundary Status. The PWM_STAT.TMR0PER bit indicates whether or not the PWMTMR0 period boundary has been reached.	
		0	PWMTMR0 period boundary not reached
		1	PWMTMR0 period boundary reached
11 (R/NW)	SRTRIPD	Self-Restart Trip Status for Channel D. The PWM_STAT.SRTRIPD bit indicates whether the PWM Channel D self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel D Self-Restart Trip Status is "not tripped"
		1	Channel D Self-Restart Trip Status is "tripped"
10 (R/W1C)	FLTTRIPD	Fault Trip Status for Channel D. The PWM_STAT.FLTTRIPD bit indicates whether the PWM Channel D fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel D Fault Trip Status is "not tripped"
		1	Channel D Fault Trip Status is "tripped"

Table 24-55: PWM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	SRTRIPC	Self-Restart Trip Status for Channel C. The PWM_STAT.SRTRIPC bit indicates whether the PWM Channel C self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel C Self-Restart Trip Status is "not tripped"
		1	Channel C Self-Restart Trip Status is "tripped"
8 (R/W1C)	FLTTRIPC	Fault Trip Status for Channel C. The PWM_STAT.FLTTRIPC bit indicates whether the PWM Channel C fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel C Fault Trip Status is "not tripped"
		1	Channel C Fault Trip Status is "tripped"
7 (R/NW)	SRTRIPB	Self-Restart Trip Status for Channel B. The PWM_STAT.SRTRIPB bit indicates whether the PWM Channel B self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel B Self-Restart Trip Status is "not tripped"
		1	Channel B Self-Restart Trip Status is "tripped"
6 (R/W1C)	FLTTRIPB	Fault Trip Status for Channel B. The PWM_STAT.FLTTRIPB bit indicates whether the PWM Channel B fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel B Fault Trip Status is "not tripped"
		1	Channel A Fault Trip Status is "tripped"
5 (R/NW)	SRTRIPA	Self-Restart Trip Status for Channel A. The PWM_STAT.SRTRIPA bit indicates whether the PWM Channel A self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel A Self-Restart Trip Status is "not tripped"
		1	Channel A Self-Restart Trip Status is "tripped"

Table 24-55: PWM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1C)	FLTTRIPA	Fault Trip Status for Channel A. The PWM_STAT.FLTTRIPA bit indicates whether the PWM Channel A fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Channel A Fault Trip Status is "not tripped"
		1	Channel A Fault Trip Status is "tripped"
3 (R/NW)	RAWTRIP1	Raw Trip 1 Status. The PWM_STAT.RAWTRIP1 bit indicates the raw input level for the PWM TRIP1 input.	
		0	TRIP1 Level is Low
		1	TRIP1 Level is High
2 (R/NW)	RAWTRIP0	Raw Trip 0 Status. The PWM_STAT.RAWTRIP0 bit indicates the raw input level for the PWM TRIP0 input.	
		0	TRIP0 Level is Low
		1	TRIP0 Level is High
1 (R/W1C)	TRIP1	Status bit set when TRIP1 is active low. The PWM_STAT.TRIP1 bit indicates whether the PWM TRIP1 fault has been tripped with an active-low input.	
		0	TRIP1 status is "not tripped"
		1	TRIP1 status is "tripped" (active low)
0 (R/W1C)	TRIP0	Status bit set when TRIP0 is active low. The PWM_STAT.TRIP0 bit indicates whether the PWM TRIP0 fault has been tripped with an active-low input.	
		0	TRIP0 status is "not tripped"
		1	TRIP0 status is "tripped" (active low)

Software Trip Register

The PWM_SWTRIP register has individual, independent control for software trip of all PWM Channels. Unlike the Global SWTRIP bit, each channel (High and Low) has its own SWTRIP bit.

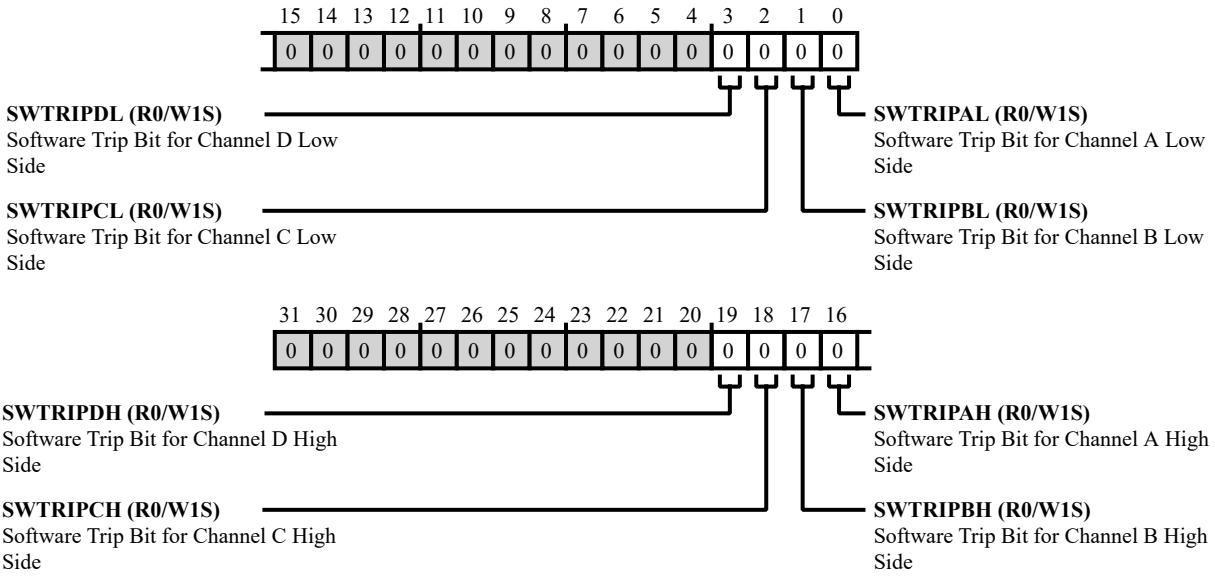


Figure 24-73: PWM_SWTRIP Register Diagram

Table 24-56: PWM_SWTRIP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R0/W1S)	SWTRIPDH	Software Trip Bit for Channel D High Side. Writing 1 forces Channel D High Side to trip.
18 (R0/W1S)	SWTRIPCH	Software Trip Bit for Channel C High Side. Writing 1 forces Channel C High Side to trip.
17 (R0/W1S)	SWTRIPBH	Software Trip Bit for Channel B High Side. Writing 1 forces Channel B High Side to trip.
16 (R0/W1S)	SWTRIPAH	Software Trip Bit for Channel A High Side. Writing 1 forces Channel A High Side to trip.
3 (R0/W1S)	SWTRIPDL	Software Trip Bit for Channel D Low Side. Writing 1 forces Channel D Low Side to trip.
2 (R0/W1S)	SWTRIPCL	Software Trip Bit for Channel C Low Side. Writing 1 forces Channel C Low Side to trip.
1 (R0/W1S)	SWTRIPBL	Software Trip Bit for Channel B Low Side. Writing 1 forces Channel B Low Side to trip.

Table 24-56: PWM_SWTRIP Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R0/W1S)	SWTRIPAL	Software Trip Bit for Channel A Low Side. Writing 1 forces Channel A Low Side to trip.

Sync Pulse Width Register

The PWM_SYNC_WID register selects the pulse width for the external sync pulse available on the PWM_SYNC pin. The relation between the PWM_SYNC_WID register value and the pulse width (T_{PWM_SYNC}) is give by the formula:

PWM_SYNC_WID = (T_{PWM_SYNC} / t_{CK}) -1

For more information about applying the sync pulse width, see the PWM Functional Description section. Note that if the pulse width is changed in between sync pulses, the PWM applies the changed width on the next internal sync pulse. If, while the sync pulse is active, the chosen timer reaches its period boundary, the changed pulse width takes effect on that period boundary.

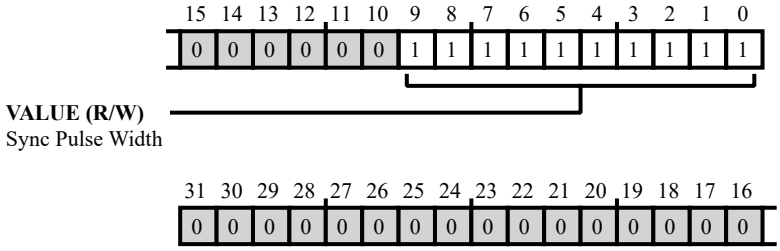


Figure 24-74: PWM_SYNC_WID Register Diagram

Table 24-57: PWM_SYNC_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Sync Pulse Width. The PWM_SYNC_WID.VALUE bits select the pulse width for the external sync pulse available on the PWM_SYNC pin.

Timer 0 Period Register

The PWM_TM0 register controls the switch period T_{SP} of the PWMTMR0 timer. The PWM_TM0 value is in units of t_{CK} (the period of the peripheral clock) and the and is given by the formula:

PWM_TM0= (T_{SP}) / 2 x t_{CK}

The value written to the register is effectively the number of t_{CK} clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM_TM0 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

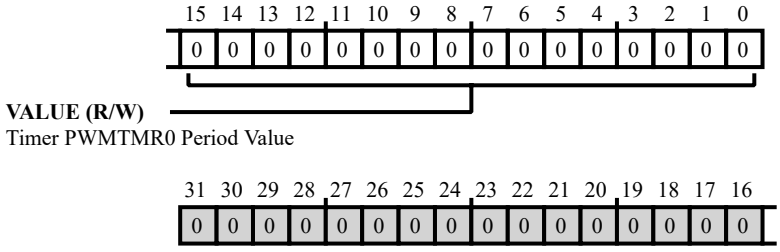


Figure 24-75: PWM_TM0 Register Diagram

Table 24-58: PWM_TM0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer PWMTMR0 Period Value. The PWM_TM0.VALUE bits select the period for the PWMTMR0 timer.

Timer 1 Period Register

The PWM_TM1 register controls the switch period (T_{SP} of the PWMTMR1 timer. The PWM_TM1 value is in units of t_{CK} (the period of the peripheral clock) and the and is given by the formula:

PWM_TM1= (T_{SP}) / 2 x t_{CK}

The value written to the register is effectively the number of t_{CK} clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM_TM1 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

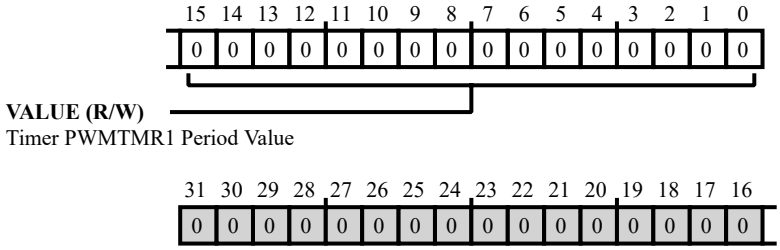


Figure 24-76: PWM_TM1 Register Diagram

Table 24-59: PWM_TM1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer PWMTMR1 Period Value. The PWM_TM1.VALUE bits select the period for the PWMTMR1 timer.

Timer 2 Period Register

The PWM_TM2 register controls the switch period (T_{SP} of the PWMTMR2 timer. The PWM_TM2 value is in units of t_{CK} (the period of the peripheral clock) and the and is given by the formula:

PWM_TM1= (T_{SP}) / 2 x t_{CK}

The value written to the register is effectively the number of t_{CK} clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM_TM2 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

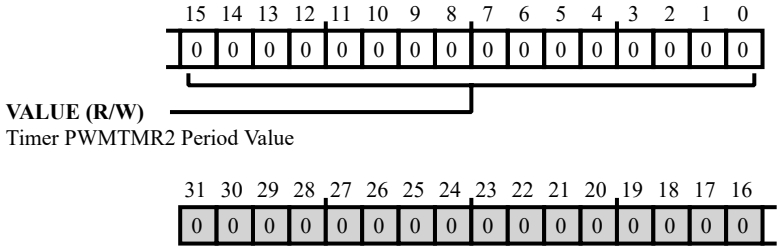


Figure 24-77: PWM_TM2 Register Diagram

Table 24-60: PWM_TM2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer PWMTMR2 Period Value. The PWM_TM2.VALUE bits select the period for the PWMTMR2 timer.

Timer 3 Period Register

The PWM_TM3 register controls the switch period (T_{SP} of the PWMTMR3 timer. The PWM_TM3 value is in units of t_{CK} (the period of the peripheral clock) and the and is given by the formula:

PWM_TM3= (T_{SP}) / 2 x t_{CK}

The value written to the register is effectively the number of t_{CK} clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM_TM3 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

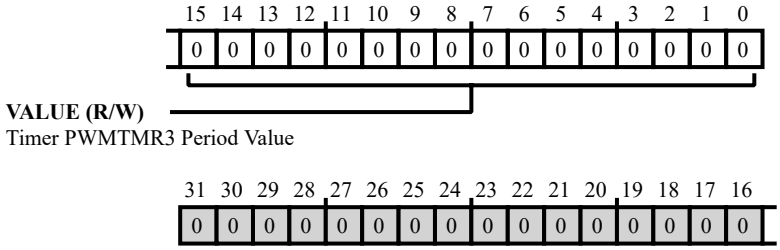


Figure 24-78: PWM_TM3 Register Diagram

Table 24-61: PWM_TM3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer PWMTMR3 Period Value. The PWM_TM3.VALUE bits select the period for the PWMTMR3 timer.

Timer 4 Period Register

The PWM_TM4 register controls the switch period (T_{SP} of the PWMTMR4 timer. The PWM_TM4 value is in units of t_{CK} (the period of the peripheral clock) and the and is given by the formula:

PWM_TM4= (T_{SP}) / 2 x t_{CK}

The value written to the register is effectively the number of t_{CK} clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM_TM4 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

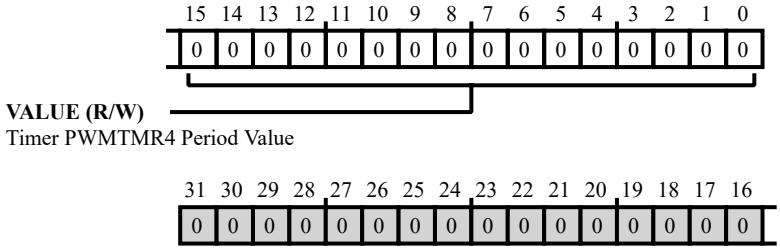


Figure 24-79: PWM_TM4 Register Diagram

Table 24-62: PWM_TM4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer PWMTMR4 Period Value. The PWM_TM4.VALUE bits select the period for the PWMTMR4 timer.

Trip Configuration Register

The PWM_TRIPCFG register configures Channel A, B, C, and D trip operation for trip inputs TRIP0 and TRIP1.

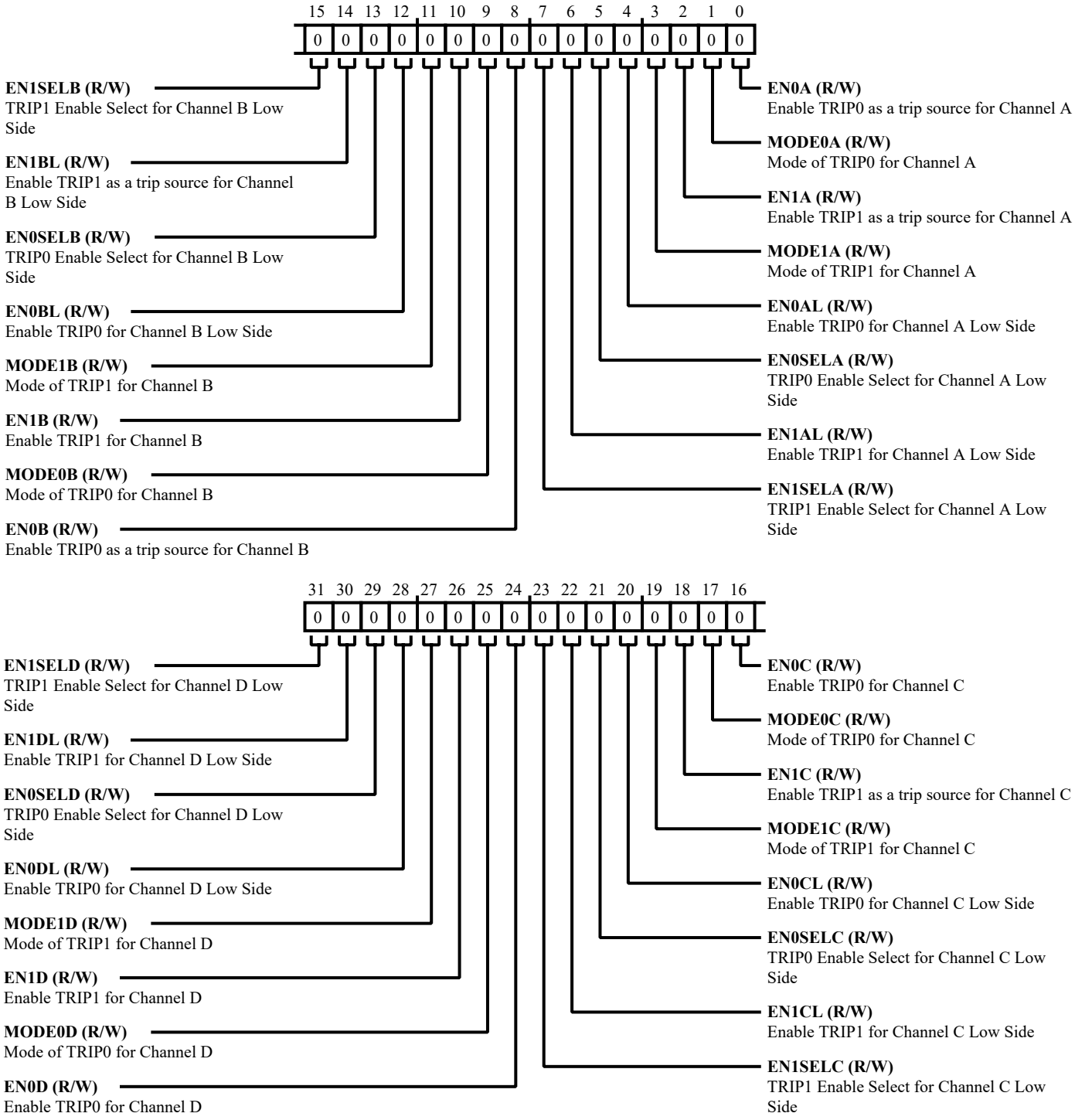


Figure 24-80: PWM_TRIPCFG Register Diagram

Table 24-63: PWM_TRIPCFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN1SELD	TRIP1 Enable Select for Channel D Low Side. The PWM_TRIPCFG.EN1SELD bit selects the TRIP1 Enable bit for Channel D Low Side. If cleared (=0), Channel D Low Side TRIP1 Enable depends on the PWM_TRIPCFG.EN0D bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0DL bit setting.	
		0	Select EN0D for Channel D Low Side Trip1 Enable
		1	Select EN0DL for Channel D Low Side Trip1 Enable
30 (R/W)	EN1DL	Enable TRIP1 for Channel D Low Side. The PWM_TRIPCFG.EN1DL bit independently enables TRIP1 as a source for Channel D Low Side.	
		0	Disable Trip1 for Channel D Low Side
		1	Enable Trip1 for Channel D Low Side
29 (R/W)	EN0SELD	TRIP0 Enable Select for Channel D Low Side. The PWM_TRIPCFG.EN0SELD bit selects the TRIP0 Enable bit for Channel D Low Side. If cleared (=0), Channel D Low Side TRIP0 Enable depends on the PWM_TRIPCFG.EN0D bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0DL bit setting.	
		0	Select EN0D for Channel D Low Side Trip0 Enable
		1	Select EN0DL for Channel D Low Side Trip0 Enable
28 (R/W)	EN0DL	Enable TRIP0 for Channel D Low Side. The PWM_TRIPCFG.EN0DL bit independently enables TRIP0 as a source for Channel D Low Side.	
		0	Disable Trip0 for Channel D Low Side
		1	Enable Trip0 for Channel D Low Side
27 (R/W)	MODE1D	Mode of TRIP1 for Channel D. The PWM_TRIPCFG.MODE1D bit selects the trip mode of TRIP1 for Channel D. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP1 Input
		1	Self Restart on TRIP1 Input

Table 24-63: PWM_TRIPCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	EN1D	Enable TRIP1 for Channel D. The PWM_TRIPCFG.EN1D bit enables TRIP1 as a trip source for Channel D.	
		0	Disable TRIP1 for Channel D
		1	Enable TRIP1 for Channel D
25 (R/W)	MODE0D	Mode of TRIP0 for Channel D. The PWM_TRIPCFG.MODE0D bit selects the trip mode of TRIP0 for Channel D. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP0 Input
		1	Self Restart on TRIP0 Input
24 (R/W)	EN0D	Enable TRIP0 for Channel D. The PWM_TRIPCFG.EN0D bit enables TRIP0 as a trip source for Channel D.	
		0	Disable TRIP0 for Channel D
		1	Enable TRIP0 for Channel D
23 (R/W)	EN1SELC	TRIP1 Enable Select for Channel C Low Side. The PWM_TRIPCFG.EN1SELC bit selects the TRIP1 Enable bit for Channel C Low Side. If cleared (=0), Channel C Low Side TRIP1 Enable depends on the PWM_TRIPCFG.EN0C bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0CL bit setting.	
		0	Select EN0C for Channel C Low Side Trip1 Enable
		1	Select EN0CL for Channel C Low Side Trip1 Enable
22 (R/W)	EN1CL	Enable TRIP1 for Channel C Low Side. The PWM_TRIPCFG.EN1CL bit independently enables TRIP0 as a source for Channel C Low Side.	
		0	Disable Trip1 for Channel C Low Side
		1	Enable Trip1 for Channel C Low Side

Table 24-63: PWM_TRIPCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	EN0SELC	TRIP0 Enable Select for Channel C Low Side. The PWM_TRIPCFG.EN0SELC bit selects the TRIP0 Enable bit for Channel C Low Side. If cleared (=0), Channel C Low Side TRIP0 Enable depends on the PWM_TRIPCFG.EN0C bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0CL bit setting.	
		0	Select EN0C for Channel C Low Side Trip0 Enable
		1	Select EN0CL for Channel C Low Side Trip0 Enable
20 (R/W)	EN0CL	Enable TRIP0 for Channel C Low Side. The PWM_TRIPCFG.EN0CL bit independently enables TRIP0 as a source for Channel C Low Side.	
		0	Disable Trip0 for Channel C Low Side
		1	Enable Trip0 for Channel C Low Side
19 (R/W)	MODE1C	Mode of TRIP1 for Channel C. The PWM_TRIPCFG.MODE1C bit selects the trip mode of TRIP1 for Channel C. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP1 Input
		1	Self Restart on TRIP1 Input
18 (R/W)	EN1C	Enable TRIP1 as a trip source for Channel C. The PWM_TRIPCFG.EN1C bit enables TRIP1 as a trip source for Channel C.	
		0	Disable TRIP1 for Channel C
		1	Enable TRIP1 for Channel C
17 (R/W)	MODE0C	Mode of TRIP0 for Channel C. The PWM_TRIPCFG.MODE0C bit selects the trip mode of TRIP0 for Channel C. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP0 Input
		1	Self Restart on TRIP0 Input
16 (R/W)	EN0C	Enable TRIP0 for Channel C. The PWM_TRIPCFG.EN0C bit enables TRIP0 as a trip source for Channel C.	
		0	Disable TRIP0 for Channel C
		1	Enable TRIP0 for Channel C

Table 24-63: PWM_TRIPCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN1SELB	TRIP1 Enable Select for Channel B Low Side. The PWM_TRIPCFG.EN1SELB bit selects the TRIP1 Enable bit for Channel B Low Side. If cleared (=0), Channel B Low Side TRIP1 Enable depends on the PWM_TRIPCFG.EN0B bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0BL bit setting.	
		0	Select EN0B for Channel B Low Side Trip1 Enable
		1	Select EN0BL for Channel B Low Side Trip1 Enable
14 (R/W)	EN1BL	Enable TRIP1 as a trip source for Channel B Low Side. The PWM_TRIPCFG.EN1BL bit enables TRIP1 as a source for Channel B Low Side independently	
		0	Disable Trip1 for Channel B Low Side
		1	Enable Trip1 for Channel B Low Side
13 (R/W)	EN0SELB	TRIP0 Enable Select for Channel B Low Side. The PWM_TRIPCFG.EN0SELB bit selects the TRIP0 Enable bit for Channel B Low Side. If cleared (=0), Channel B Low Side TRIP0 Enable depends on the PWM_TRIPCFG.EN0B bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0BL bit setting.	
		0	Select EN0B for Channel B Low Side Trip0 Enable
		1	Select EN0BL for Channel B Low Side Trip0 Enable
12 (R/W)	EN0BL	Enable TRIP0 for Channel B Low Side. The PWM_TRIPCFG.EN0BL bit independently enables TRIP0 as a source for Channel B Low Side.	
		0	Disable Trip0 for Channel B Low Side
		1	Enable Trip0 for Channel B Low Side
11 (R/W)	MODE1B	Mode of TRIP1 for Channel B. The PWM_TRIPCFG.MODE1B bit selects the trip mode of TRIP1 for Channel B. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP1 Input
		1	Self Restart on TRIP1 Input

Table 24-63: PWM_TRIPCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	EN1B	Enable TRIP1 for Channel B. The PWM_TRIPCFG.EN1B bit enables TRIP1 as a trip source for Channel B.	
		0	Disable TRIP1 for Channel B
		1	Enable TRIP1 for Channel B
9 (R/W)	MODE0B	Mode of TRIP0 for Channel B. The PWM_TRIPCFG.MODE0B bit selects the trip mode of TRIP0 for Channel B. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP0 Input
		1	Self Restart on TRIP0 Input
8 (R/W)	EN0B	Enable TRIP0 as a trip source for Channel B. The PWM_TRIPCFG.EN0B bit enables TRIP0 as a trip source for Channel B.	
		0	Disable TRIP0 for Channel B
		1	Enable TRIP0 for Channel B
7 (R/W)	EN1SELA	TRIP1 Enable Select for Channel A Low Side. The PWM_TRIPCFG.EN1SELA bit selects the TRIP1 Enable bit for Channel A Low Side. If cleared (=0), Channel A Low Side TRIP1 Enable depends on the PWM_TRIPCFG.EN0A bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0AL bit setting.	
		0	Select EN0A for Channel A Low Side Trip1 Enable
		1	Select EN0AL for Channel A Low Side Trip1 Enable
6 (R/W)	EN1AL	Enable TRIP1 for Channel A Low Side. The PWM_TRIPCFG.EN1AL bit independently enables TRIP1 as a source for Channel A Low Side.	
		0	Disable Trip1 for Channel A Low Side
		1	Enable Trip1 for Channel A Low Side

Table 24-63: PWM_TRIPCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	EN0SELA	TRIP0 Enable Select for Channel A Low Side. The PWM_TRIPCFG.EN0SELA bit selects the TRIP0 Enable bit for Channel A Low Side. If cleared (=0), Channel A Low Side TRIP0 Enable depends on the PWM_TRIPCFG.EN0A bit setting. If set (=1), it depends on the PWM_TRIPCFG.EN0AL bit setting.	
		0	Select EN0A for Channel A Low Side Trip0 Enable
		1	Select EN0AL for Channel A Low Side Trip0 Enable
4 (R/W)	EN0AL	Enable TRIP0 for Channel A Low Side. The PWM_TRIPCFG.EN0AL bit independently enables TRIP0 as a source for Channel A Low Side.	
		0	Disable Trip0 for Channel A Low Side
		1	Enable Trip0 for Channel A Low Side
3 (R/W)	MODE1A	Mode of TRIP1 for Channel A. The PWM_TRIPCFG.MODE1A bit selects the trip mode of TRIP1 for Channel A. For more information, see the PWM_TRIPCFG.MODE0A bit description.	
		0	Fault Trip on TRIP1 Input
		1	Self Restart on TRIP1 Input
2 (R/W)	EN1A	Enable TRIP1 as a trip source for Channel A. The PWM_TRIPCFG.EN1A bit enables TRIP1 as a trip source for Channel A.	
		0	Disable TRIP1 for Channel A
		1	Enable TRIP1 for Channel A

Table 24-63: PWM_TRIPCFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	MODE0A	Mode of TRIP0 for Channel A. The PWM_TRIPCFG.MODE0A bit selects the trip mode of TRIP0 for Channel A. In fault-trip mode (PWM_TRIPCFG.MODE0A =0), after the input is tripped, the trip status appears in the corresponding channels fault-trip status bit (for example, PWM_STAT.FLTTRIPA), and the PWM immediately shuts down outputs of that channel. After a fault trip occurs, when the trip condition is no longer active, the processor may cause channel outputs to resume by completing a write-1-to-clear the corresponding fault-trip status bit. The raw (input level) trip input state is available from the PWM_STAT.RAWTRIP0 and PWM_STAT.RAWTRIP0 bits. In self-restart mode (PWM_TRIPCFG.MODE0A =1), after the input is tripped, the trip status appears in the corresponding channels self-restart status bit (for example, PWM_STAT.SRTRIPA), and the PWM immediately shuts down outputs of that channel. On the next timer period boundary (of the PWMTMRx used by that channel), if the trip condition is not active, the PWM clears the status and restarts the channels output.	
		0	Fault Trip on TRIP0 Input
		1	Self Restart on TRIP0 Input
0 (R/W)	EN0A	Enable TRIP0 as a trip source for Channel A. The PWM_TRIPCFG.EN0A bit enables TRIP0 as a trip source for Channel A.	
		0	Disable TRIP0 for Channel A
		1	Enable TRIP0 for Channel A

Trip Polarity Register

The PWM_TRIP_POL register controls the Channel Polarity upon tripping for each PWM Channel independently.

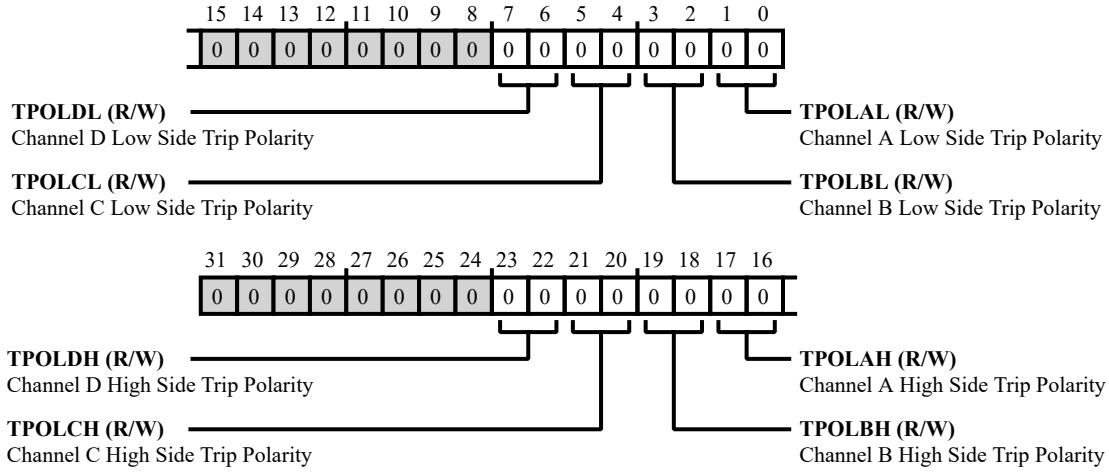


Figure 24-81: PWM_TRIP_POL Register Diagram

Table 24-64: PWM_TRIP_POL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/W)	TPOLDH	Channel D High Side Trip Polarity.	
		0	Set Channel D High Side to Inactive Polarity upon Trip
		1	Set Channel D High Side to Active Polarity upon Trip
		2	Set Channel D High Side to High Impedance upon Trip
21:20 (R/W)	TPOLCH	Channel C High Side Trip Polarity.	
		0	Set Channel C High Side to Inactive Polarity upon Trip
		1	Set Channel C High Side to Active Polarity upon Trip
		2	Set Channel C High Side to High Impedance upon Trip
19:18 (R/W)	TPOLBH	Channel B High Side Trip Polarity.	
		0	Set Channel B High Side to Inactive Polarity upon Trip
		1	Set Channel B High Side to Active Polarity upon Trip
		2	Set Channel B High Side to High Impedance upon Trip

Table 24-64: PWM_TRIP_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	TPOLAH	Channel A High Side Trip Polarity.	
		0	Set Channel A High Side to Inactive Polarity upon Trip
		1	Set Channel A High Side to Active Polarity upon Trip
		2	Set Channel A High Side to High Impedance upon Trip
7:6 (R/W)	TPOLDL	Channel D Low Side Trip Polarity.	
		0	Set Channel D Low Side to Inactive Polarity upon Trip
		1	Set Channel D Low Side to Active Polarity upon Trip
		2	Set Channel D Low Side to High Impedance upon Trip
5:4 (R/W)	TPOLCL	Channel C Low Side Trip Polarity.	
		0	Set Channel C Low Side to Inactive Polarity upon Trip
		1	Set Channel C Low Side to Active Polarity upon Trip
		2	Set Channel C Low Side to High Impedance upon Trip
3:2 (R/W)	TPOLBL	Channel B Low Side Trip Polarity.	
		0	Set Channel B Low Side to Inactive Polarity upon Trip
		1	Set Channel B Low Side to Active Polarity upon Trip
		2	Set Channel B Low Side to High Impedance upon Trip
1:0 (R/W)	TPOLAL	Channel A Low Side Trip Polarity.	
		0	Set Channel A Low Side to Inactive Polarity upon Trip
		1	Set Channel A Low Side to Active Polarity upon Trip
		2	Set Channel A Low Side to High Impedance upon Trip

25 General-Purpose Counter (CNT)

The GP counter converts pulses from incremental position encoders into data that is representative of the actual position of the pulse. This conversion is done by integrating (counting) pulses on one or two inputs. Since integration provides relative position, some devices also feature a zero-position input (zero marker). The GP counter can use the zero position input feature to establish a reference point for verifying that the acquired position does not drift over time. In addition, the GP counter can use the incremental position information to determine speed, if the time intervals are measured.

The GP counter provides flexible ways to establish position information. When used with the GP timer block, the GP counter can allow for the acquisition of coherent position or time stamp information that enables speed calculation.

GP Counter Features

The GP counter includes the following features:

- 32-bit up or down counter
- Quadrature encode mode (Gray code)
- Binary encoder mode
- Alternative frequency-direction mode
- Timed direction and up or down counting modes
- Zero marker or push-button support
- Capture event timing in association with GP Timer
- Boundary comparison and boundary setting features

GP Counter Functional Description

The `CNT_UD` and `CNT_DG` pins accept various forms of incremental inputs. The 32-bit counter processes the inputs. The GP counter uses the `CNT_ZM` pin to sense the pressing of a push button.

NOTE: When enabled, the GP counter requires 3 SCLK cycles of initialization before recognizing valid toggles on its input pins.

The three input pins can be filtered (debounced) before the GP counter evaluates them.

The GP counter features a flexible boundary comparison. In all of the operating modes, the counter can be compared to an upper and lower limit. It takes various actions when reaching these limits.

The GP counter has a flexible input selection. Apart from accepting inputs from the CNT0_UD and CNT0_DG and PORT pins, the counter can be configured to accept trigger inputs by setting PADS_PCFG0.CNT0UDSEL and PADS_PCFG0.CNT0DGSEL bits. Refer to the PADS_PCFG0 register description in General-Purpose Ports (PORT) chapter for details.

The module can optionally generate an interrupt request to the system through its IRQ line. On many processors, a GP timer module can use an output to generate time stamps on certain events.

ADSP-2184x CNT Register List

The GP Counter (CNT) provides support for manually controlled rotary controllers, such as the volume wheel on a radio device. This unit also supports industrial encoders.

The CNT converts pulses from incremental position encoders into data that is representative of the actual position. To complete this task, the CNT integrates (counts) pulses on one or two inputs. Because integration provides relative position, a zero position input (zero marker) is usually provided that establishes a reference point, verifying that the acquired position does not drift over time. The incremental position information may also be used to determine speed, if the relevant time intervals are measured. The CNT provides flexible ways to establish position information. When used in conjunction with the General-purpose Timer (TIMER), the CNT allows acquisition of coherent position/time stamp information, enabling speed calculation.

A set of registers govern CNT operations. For more information on CNT functionality, see the CNT register descriptions.

Table 25-1: ADSP-2184x CNT Register List

Name	Description
CNT_CFG	Configuration Register
CNT_CMD	Command Register
CNT_CNTR	Counter Register
CNT_DEBNCE	Debounce Register
CNT_IMSK	Interrupt Mask Register
CNT_MAX	Maximum Count Register
CNT_MIN	Minimum Count Register
CNT_STAT	Status Register

ADSP-2184x CNT Interrupt List

Table 25-2: ADSP-2184x CNT Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
9	CNT0_STAT	CNT0 Status	Level	

ADSP-2184x CNT Trigger List

Table 25-3: ADSP-2184x CNT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
5	CNT0_STAT	CNT0 Status	Level
6	CNT0_UD	CNT0 Count Up and Direction	Level
7	CNT0_DG	CNT0 Count Down and Gate	Level
8	CNT0_TO	CNT0 Output to Timer Block	Level

Table 25-4: ADSP-2184x CNT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
2	CNT0_UD	CNT0 Count Up and Direction	Pulse
3	CNT0_DG	CNT0 Count Down and Gate	Pulse

GP Counter Operating Modes

The GP counter has the following modes of operation.

1. Quadrature Encoder
2. Binary Encoder
3. Up/Down Counter
4. Direction Counter
5. Timed Direction

With the exception of the timed direction mode, the GP counter can operate with the GP timer block to capture additional timing information (time stamps) associated with events detected by this block.

Quadrature Encoder Mode

In this mode, the CNT_UD and CNT_DG inputs expect a quadrature-encoded signal that is interpreted as a two-bit Gray code. The order of transitions of the CNT_UD and CNT_DG inputs determines whether the counter increments or decrements. The CNT_CNTR register contains the number of transitions that have occurred as shown in the ***Quadrature Events and Counting Mechanism*** table. Optionally, an interrupt is generated when both inputs

change within one SCLK cycle. Gray coding prohibits such transitions. Therefore, the CNT_CNTR register remains unchanged, and an error condition is signaled.

Table 25-5: Quadrature Events and Counting Mechanism

CNT_COUNTER Register Value	-4	-3	-2	-1	0	+1	+2	+3	+4
CDG, CUD Inputs	00	01	11	10	00	01	11	10	00

It is possible to reverse the count direction of the Gray-coded signal by enabling the polarity inverter of either the CNT_UD pin or the CNT_DG pin. Inverting both pins does not alter the behavior. The GP counter can enable this feature with the CNT_CFG.CDGINV and CNT_CFG.CUDINV bits.

As an example, the CNT_DG and CNT_UD inputs are 00 and the next transition is to 01. These inputs normally change the counter in increments as shown in the table. If the CNT_UD polarity is inverted, this condition generates a received input of 01 followed by 00. The results is a decrement of the counter, altering the behavior of the connected hardware.

Binary Encoder Mode

This mode is almost identical to quadrature encoder mode, with the exception that the CNT_UD: CNT_DG inputs expect a binary-encoded signal. The order of transitions of the CNT_UD and CNT_DG inputs determines whether the counter increments or decrements. The CNT_CNTR register contains the number of transitions that have occurred as shown in the ***Binary Events and Counting Mechanism*** table. Optionally, an interrupt is generated when the detected code steps by more than 1 (in binary arithmetic) within one SCLK cycle. Such transitions are erroneous. Therefore, the CNT_CNTR register remains unchanged, and an error condition is signaled.

Table 25-6: Binary Events and Counting Mechanism

CNT_COUNTER Register Value	-4	-3	-2	-1	0	+1	+2	+3	+4
CDG:CUD Inputs	00	01	10	11	00	01	10	11	00

Reversing the CNT_UD and CNT_DG pin polarity has a different effect in binary encoder mode than for the quadrature encoder mode. Inverting the polarity of the CNT_UD pin only or inverting both the CNT_UD and CNT_DG pins, results in reversing the count direction.

Up/Down Counter Mode

In this mode, the counter increments or decrements at every active edge of the input pins. The GP counter uses the CNT_CFG.CUDINV bit to select an active edge and has the following results.

- If the GP counter module detects an active edge at the CNT_UD input, the counter increments.
- If the GP counter module detects an active edge at the CNT_DG input, the counter decrements.
- If simultaneous edges occur on the CNT_DG and CNT_UD pins, the counter remains unchanged, and both up-count and down-count events are signaled in the CNT_STAT register.

Direction Counter Mode

In this mode, the counter is incremented or decremented at every active edge of the CNT_DG input pin. The state of the CNT_UD input determines whether the counter increments or decrements. The GP counter uses the CNT_CFG.CUDINV bit to select the polarity.

If the GP counter detects an active edge at the CNT_DG input, the counter value changes by one in the selected direction.

Timed Direction Mode

In this mode, the counter is incremented or decremented at each SCLK cycle. The state of the CNT_UD input determines whether the counter increments or decrements. The GP counter uses the CNT_CFG.CUDINV bit to select the polarity. The CNT_DG pin can be used to gate the clock. The GP counter uses the CNT_CFG.CDGINV bit to select the polarity.

GP Counter Programming Model

The following sections provide information for programming the interface.

GP Counter General Programming Flow

The following are general guidelines for configuring and enabling the GP counter.

1. Initialize (but do not enable) the GP counter for the desired mode and settings through the CNT_CFG register.
2. Usually, events of interest are processed using interrupts rather than by polling status bits. In this case, clear all status bits and activate the interrupt generation requests with the CNT_IMSK register.
3. Configure interrupts at the system level to insure desired interrupt signaling to the system.
4. If timing information is required, set up the relevant GP Timer in width capture mode.
5. Finally, enable interrupt requests and the GP Counter itself using the CNT_IMSK and CNT_CFG registers, respectively.

GP Counter Mode Configuration

The GP counter can use the CNT_ZM input pin to sense the zero marker output of a rotary device or to detect the pressing of a push button. There are four programming schemes, which are functional in all counter modes:

- Push-button mode
- Zero-marker-zeros-counter mode
- Zero-marker-error mode
- Zero-once mode

Configuring GP Counter Push-Button Operation

Use the following procedure to configure push-button operation:

1. Set CNT_IMSK.CZM to enable (unmask) the zero marker interrupt.
2. Select the active edge polarity through the CNT_CFG.CZMINV bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

An active edge at the CNT_ZM input sets the CNT_IMSK.CZM bit.

Configuring Zero-Marker-Zeros-Counter Mode

The following provides information on configuring zero-marker-zeros-counter mode for the GP counter.

1. Set CNT_IMSK.CZMZ to enable CNT_CNTR. The zero marker interrupt zeroes the counter.
2. Set CNT_CFG.ZMZC to enable ZMZC mode.
3. Select the active edge polarity through the CNT_CFG.CZMINV bit.
4. Proceed with any other desired configuration steps and enable the peripheral.

This configuration causes an active level at the CNT_ZM pin to clear the CNT_CNTR register and keep it cleared until the CNT_ZM pin is deactivated. In addition, the CNT_STAT.CZMZ bit is set.

Configuring Zero-Marker-Error Mode

The GP counter uses this mode to detect discrepancies between counter-value and the zero marker output of certain rotary encoder devices.

1. Set the CNT_STAT.CZME bit to enable this mode.
2. Select the active edge of the CNT_ZM pin through the CNT_CFG.CZMINV bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

When the GP counter detects an active edge at the CNT_ZM input pin, it compares the four LSBs of the CNT_CNTR register to zero. If they are not zero, the GP counter uses CNT_STAT.CZME bit to signal a mismatch.

Configuring Zero-Once Mode

The GP counter uses this mode to perform an initial reset of the counter-value when it detects an active zero marker. After that, the zero marker is ignored (the counter is no longer reset).

1. Set the CNT_CMD.W1ZMONCE bit to enable this mode.
2. Select the active edge of the CNT_ZM pin through the CNT_CFG.CZMINV bit.
3. Ensure that at least one of the following bits is enabled: CNT_IMSK.CZM, CNT_IMSK.CZME, CNT_IMSK.CZMZ.

4. Proceed with any other desired configuration steps and enable the peripheral.

The CNT_CNTR register and the CNT_CMD.W1ZMONCE bit are cleared on the next active edge of the CNT_ZM pin. Now the CNT_CMD.W1ZMONCE bit can be read to check whether the event has already occurred.

Configuring Boundary Auto-Extend Mode

In this mode, hardware modifies the boundary registers (CNT_MIN and CNT_MAX) whenever the CNT_CNTR value reaches either of them. The GP counter uses this mode to monitor the widest angle a thumb wheel even if the software did not generate interrupts.

1. Initialize CNT_CNTR with the desired value.
2. Set both CNT_MIN and CNT_MAX to this same value.
3. Configure the CNT_CFG.BNDMODE field for auto extend mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

The CNT_MAX register is loaded with the current CNT_CNTR value when the latter increments beyond the CNT_MAX value. Similarly, the CNT_MIN register is loaded with the CNT_CNTR value when the latter decrements below the CNT_MIN value. The CNT_STAT.MAXC and CNT_STAT.MINC status bits are set when the CNT_CNTR value matches the respective boundary register value.

Configuring Boundary Capture Mode

In this mode, the CNT_CNTR value is latched into the CNT_MIN register at one detected edge of the CNT_ZM input pin and latched into the CNT_MAX boundary register at the opposite edge.

1. To capture the CNT_ZM pin rising edge into CNT_MIN and the falling edge into CNT_MAX, program CNT_CFG.CZMINV for active high polarity. Conversely, to capture the CNT_ZM pin falling edge into CNT_MIN and the rising edge into CNT_MAX, program CNT_CFG.CZMINV for active low polarity.
2. Program the CNT_IMSK.MAXC and CNT_IMSK.MINC interrupt mask bits according to interrupt generation requirements.
3. Configure the CNT_CFG.BNDMODE field for boundary capture mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

The CNT_STAT.MAXC and CNT_STAT.MINC status bits report the capture event, depending on how interrupt masks are configured.

Configuring Boundary Compare and Boundary Zero Modes

In these modes, the two boundary registers (CNT_MAX and CNT_MIN) are compared to the value in the CNT_CNTR register.

1. Program CNT_MAX and CNT_MIN registers with appropriate upper and lower range values.

2. Program the CNT_IMSK.MAXC and CNT_IMSK.MINC interrupt mask bits according to interrupt generation requirements.
3. Configure the CNT_CFG.BNDMODE field for boundary compare mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

If after incrementing, CNT_CNTR = CNT_MAX, then the CNT_STAT.MAXC bit is set. Similarly if after decrementing, CNT_CNTR = CNT_MIN, then the CNT_STAT.MINC bit is set.

Additionally, for boundary zero mode, the counter-value in CNT_CNTR is set to zero. The CNT_STAT.MAXC and CNT_STAT.MINC bits are not set when software updates the CNT_MAX or CNT_MIN registers.

Configuring GP Counter Push-Button Operation

Use the following procedure to configure push-button operation:

1. Set CNT_IMSK.CZM to enable (unmask) the zero marker interrupt.
2. Select the active edge polarity through the CNT_CFG.CZMINV bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

An active edge at the CNT_ZM input sets the CNT_IMSK.CZM bit.

GP Counter Programming Concepts

Using the features, operating modes, and event control for the GP counter to their greatest potential requires an understanding of some GP counter-related concepts. Some key aspects to consider are input noise filtering and capturing timing information.

CNT Input Noise Filtering

In all modes, the three input pins can be filtered to present clean signals to the GP counter logic. The GP counter uses the CNT_CFG.DEBEN bit to enable or disable this filtering. The ***Programmable Noise Filtering*** figure shows the filtering operation for the CNT_UD pin.

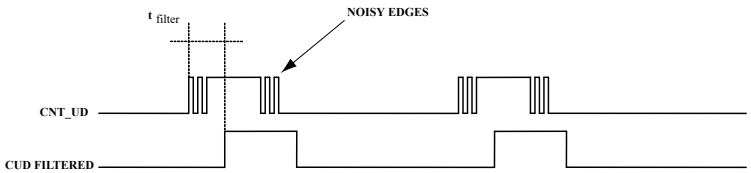


Figure 25-1: Programmable Noise Filtering

The CNT module implements the filtering mechanism using counters for each GP counter pin, where each counter is initialized from the CNT_DEBNCE.DPRESCALE field. When a transition is detected on a pin, the corresponding counter starts counting up to the programmed number of SCLK cycles. The state of the pin is latched after time t_{filter} and passed on to the GP counter logic.

The following formula determines the time t_{filter}, given SCLK and the CNT_DEBNCE.DPRESCALE value, where lower values of CNT_DEBNCE.DPRESCALE result in shorter debounce delays:

$t_{filter} = 128 \times (2^{DPRESCALE} \times SCLK)$

Capturing Counter Interval and CNT_CNTR Read Timing

When the count speed is low, it is often useful to capture the time elapsed since the last count event. Program the TIMER_TMR[n]_CFG register of the associated GP timer in a width capture mode with the following bit settings.

- TIMER_TMR[n]_CFG.PULSEHI = 0
- TIMER_TMR[n]_CFG.TMODE = b#1011
- TIMER_TMR[n]_CFG.TINSEL = 1

The ***Capture Intervals*** figure shows and the following list describes the operation of the GP counter and the GP timer in this mode.

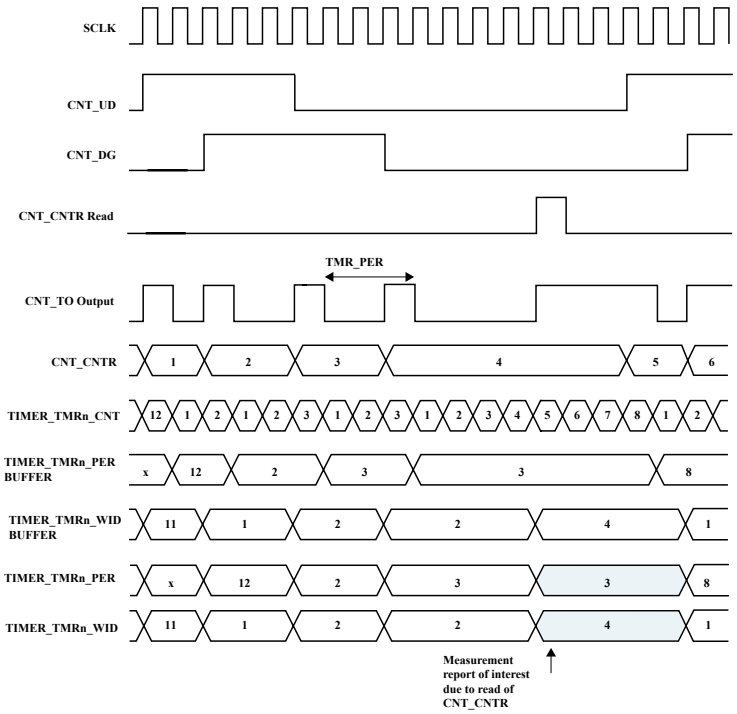


Figure 25-2: Capture Intervals

NOTE: SCLK in the ***Capture Intervals*** figure is SCLK.

1. The CNT_TO signal generates a pulse every time a count event occurs. In addition, when the processor reads the CNT_CNTR register, the CNT_TO signal presents a pulse which is extended (high) until the next count event.
2. The GP timer updates its TIMER_TMR[n]_PER register with the period (measured from falling edge to falling edge, because TIMER_TMR[n]_CFG.PULSEHI = 0) of the CNT_TO signal.

3. The TIMER_TMR[n]_WID register is updated with the pulse width (the portion where CNT_TO is low, again because TIMER_TMR[n]_CFG.PULSEHI = 0).
4. Both registers are updated at every rising edge of the CNT_TO signal (because TIMER_TMR[n]_CFG.TMODE= b#011).

The TIMER_TMR[n]_PER register contains the period between the last two count events. The TIMER_TMR[n]_WID register contains the time since the last count event and the read of the CNT_CNTR register, both measured in SCLK cycles.

Read the CNT_CNTR register to latch the two time measurements, providing a coherent triplet of information to calculate speed and position.

NOTE: Speed restrictions apply to the use of the CNT_TO signal. Therefore, programs must not operate at count event rates that are high. For instance, if CNT_CNTR is incremented or decremented every SCLK cycle (timed direction mode), the CNT_TO signal is not valid.

Capturing Time Interval Between Successive Counter Events

When the required timing information is the interval between successive count events, program the associated timer in a width capture mode. Set the TIMER_TMR[n]_CFG bit of TIMER_TMR[n]_CFG.PULSEHI = 1, TIMER_TMR[n]_CFG.TMODE = b#1010 and TIMER_TMR[n]_CFG.TINSEL = 1. Typically, this information is sufficient if the speed of GP counter events does not to reach low values.

The ***Period Register Timing*** figure shows the operation of the GP counter and the GP timer in this mode.

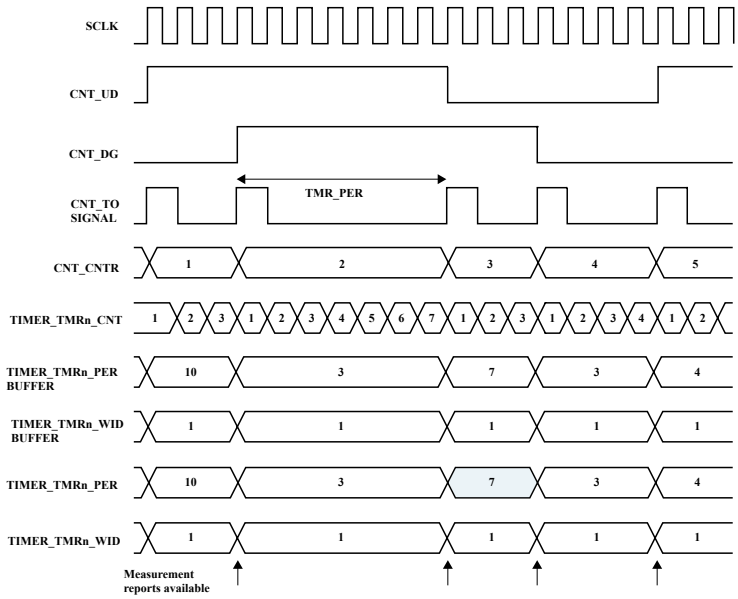


Figure 25-3: Period Register Timing

NOTE: SCLK in the ***Period Register Timing*** figure is SCLK.

The CNT_TO signal generates a pulse every time a count event occurs. The GP timer updates its TIMER_TMR[n]_PER register with the period (measured from rising edge to rising edge) of the CNT_TO signal. The TIMER_TMR[n]_PER register is updated at every rising edge of the CNT_TO signal and contains the number of SCLK cycles that have elapsed since the previous rising edge.

Incidentally, the TIMER_TMR[n]_WID register is also updated at the same time, but is of no interest in this mode of operation. If no reads of the CNT_CNTR register occur between counter events, the TIMER_TMR[n]_WID register only contains the width of the CNT_TO pulse. If a read of CNT_CNTR has occurred between events, the TIMER_TMR[n]_WID register contains the time between the read of CNT_CNTR and the next event.

This mode can also be used with TIMER_TMR[n]_CFG.PULSEHI = 0. In this case, the period of CNT_TO is measured between falling edges. It results in the same values as in the previous case, only the latching occurs one SCLK cycle later.

GP Counter Event Control

Eleven events can be signaled to the processor using status information and optional interrupt requests. The GP counter uses the respective bits in the CNT_IMSK register to enable the interrupt requests. It uses dedicated bits in the CNT_STAT register to report events. When an interrupt request from the GP counter is serviced, the application software is responsible for correct interpretation of the events. It is recommended to logically AND the content of the CNT_IMSK and CNT_STAT registers to identify pending interrupt requests.

Perform a write-one-to-clear (W1C) operation to the CNT_STAT register to clear the interrupt requests. Hardware does not clear the status bits automatically unless the counter module is disabled.

The following sections describe the events associated with the GP counter.

Illegal Gray and Binary Code Events

When illegal transitions occur in quadrature encoder or binary encoder modes, the CNT_STAT.IC bit is set. If enabled by the CNT_STAT.IC bit, the counter module generates an interrupt request. Set the CNT_STAT.IC bit only in the quadrature encoder or binary encoder modes.

Up/Down Count Events

The GP counter uses the CNT_STAT.UC bit to indicate whether the counter has been incremented. Similarly, the CNT_STAT.DC bit reports decrements. The two events are independent. For instance, if the counter increments by one and then decrements by two, both bits remain set, even though the resulting counter-value shows a decrement by one.

In up/down counter mode, hardware can detect simultaneous active edges on the CNT_UD and CNT_DG inputs. In that case, the CNT_CNTR remains unchanged, but both the CNT_STAT.UC and CNT_STAT.DC bits are set. Interrupt requests for these events can be enabled through the CNT_IMSK.UC and CNT_IMSK.DC bits. Use this feature carefully when the counter is clocked at high rates. This suggestion is especially critical when the counter operates in DIR_TMR mode, as interrupts are generated every SCLK cycle.

These events can also be used for more push buttons, when GP counter features are unnecessary. When up/down counter mode is enabled, the GP counter can use these count events to report interrupts from push buttons that connect to the CNT_UD and CNT_DG inputs.

Zero-Count Events

The CNT_STAT.CZERO status bit indicates that the CNT_CNTR has reached a value equal to 0x0000 0000 after an increment or decrement. This bit is not set when the counter value is set to zero by a write to CNT_CNTR or by setting the CNT_CMD.W1LCNTZERO bit. If enabled by the CNT_IMSK.CZERO bit, the GP counter module generates an interrupt request.

Overflow Events

There are two status bits that indicate whether the signed counter-register has overflowed from a positive to a negative value or conversely. The CNT_STAT.COV31 bit reports that the 32-bit CNT_CNTR register has either incremented from 0x7FFF FFFF to 0x8000 0000, or decremented from 0x8000 0000 to 0x7FFF FFFF.

If enabled by the CNT_IMSK.COV31 bit, an interrupt request is generated. Similarly, in applications where only the lower 16 bits of the counter are of interest, the CNT_STAT.COV15 status bit reports counter transitions from 0xXXXX 7FFF to 0xXXXX 8000, or from 0xXXXX 8000 to 0xXXXX 7FFF. If enabled by the CNT_IMSK.COV15 bit, an interrupt request is generated.

Boundary Match Events

The CNT_STAT.MINC and CNT_STAT.MAXC status bits report boundary events as described in Configuring Boundary Capture Mode. These bits are not set if the software updates the CNT_CNTR, CNT_MAX, or CNT_MIN registers or writes to the CNT_CMD register. The CNT_IMSK.MINC and CNT_IMSK.MAXC bits enable interrupt request generation on boundary events.

Zero Marker Events

The CNT_STAT.CZM, CNT_STAT.CZME, and CNT_STAT.CZMZ bits are associated with zero marker events, as described in Configuring GP Counter Push-Button Operation. Each of these events can optionally generate an interrupt request, when enabled by the corresponding CNT_IMSK.CZM, CNT_IMSK.CZME and CNT_IMSK.CZMZ bits.

ADSP-2184x CNT Register Descriptions

General-Purpose Counter (CNT) contains the following registers.

Table 25-7: ADSP-2184x CNT Register List

Name	Description
CNT_CFG	Configuration Register
CNT_CMD	Command Register

Table 25-7: ADSP-2184x CNT Register List (Continued)

Name	Description
CNT_CNTR	Counter Register
CNT_DEBNCE	Debounce Register
CNT_IMSK	Interrupt Mask Register
CNT_MAX	Maximum Count Register
CNT_MIN	Minimum Count Register
CNT_STAT	Status Register

Configuration Register

The CNT_CFG register configures counter modes, configures input pins, and enables the CNT.

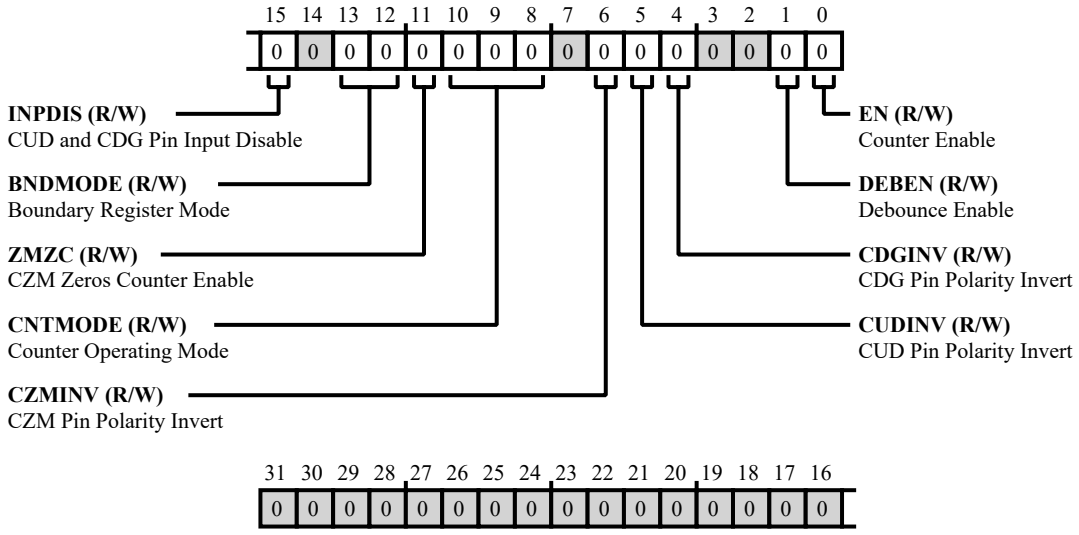


Figure 25-4: CNT_CFG Register Diagram

Table 25-8: CNT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	INPDIS	CUD and CDG Pin Input Disable. The CNT_CFG.INPDIS disables or enables the CNT_UD input pin and the CNT_DG pin.	
		0	Enable
		1	Pin Input Disable
13:12 (R/W)	BNDMODE	Boundary Register Mode. The CNT_CFG.BNDMODE bit field selects the mode for the CNT_MIN and CNT_MAX boundary registers.	
		0	BND_COMP. Boundary Compare Mode
		1	BND_ZERO. Boundary Zero Mode
		2	BND_CAPT. Boundary Capture Mode
		3	BND_AEXT. Boundary Auto-extend Mode

Table 25-8: CNT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	ZMZC	CZM Zeros Counter Enable. The CNT_CFG.ZMZC bit enables or disables level sensitive - active CNT_ZM pin operation to zero the CNT_CNTR register.	
		0	Disable
		1	Enable
10:8 (R/W)	CNTMODE	Counter Operating Mode. The CNT_CFG.CNTMODE bit field selects the operating mode for the CNT_UD input pin and the CNT_DG pin.	
		0	QUAD_ENC. Quadrature Encoder Mode
		1	BIN_ENC. Binary Encoder Mode
		2	UD_CNT. Rotary Counter Mode
		4	DIR_CNT. Direction Counter Mode
		5	DIR_TMR. Direction Timer Mode
6 (R/W)	CZMINV	CZM Pin Polarity Invert. The CNT_CFG.CZMINV bit selects the polarity for the CNT_ZM pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled.	
		0	Active High, Rising Edge
		1	Active Low, Falling Edge
5 (R/W)	CUDINV	CUD Pin Polarity Invert. The CNT_CFG.CUDINV bit selects the polarity for the CNT_UD pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled.	
		0	Active High, Rising Edge
		1	Active Low, Falling Edge
4 (R/W)	CDGINV	CDG Pin Polarity Invert. The CNT_CFG.CDGINV bit selects the polarity for the CNT_DG pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled.	
		0	Active High, Rising Edge
		1	Active Low, Falling Edge

Table 25-8: CNT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	DEBEN	Debounce Enable. The `CNT_CFG.DEBEN` bit enables or disables CNT input debounce filtering operation selected with the `CNT_DEBNCE` register.	
		0	Disable
		1	Enable
0 (R/W)	EN	Counter Enable. The `CNT_CFG.EN` bit enables or disables CNT operation.	
		0	Counter Disable
		1	Counter Enable

Command Register

The CNT_CMD register configures the CNT, enabling operations such as zeroing a counter register and copying or swapping boundary registers. These actions are taken by setting the appropriate bit.

Read operations from this register do not return meaningful values, with the exception of the CNT_CMD.W1ZMONCE bit, where a set bit indicates that the bit has been set by software before, but a zero marker event has not yet been detected on the CNT_ZM pin yet. For more information, see the CNT functional description.

The CNT_CNTR, CNT_MIN, and CNT_MAX registers can be initialized to zero by setting the CNT_CMD.W1LCNTZERO, CNT_CMD.W1LMINZERO, and CNT_CMD.W1LMAXZERO bits. In addition to clearing registers, the CNT_CMD register permits modifying the CNT_MIN and CNT_MAX boundary registers in a number of ways. The current counter value in the CNT_CNTR register can be captured and loaded into either of the two boundary registers to create new boundary limits. This operation is performed by setting the CNT_CMD.W1LMAXCNT and CNT_CMD.W1LMINCNT bits. Alternatively, the counter can be loaded from CNT_MAX or CNT_MIN using the CNT_CMD.W1LCNTMAX and CNT_CMD.W1LCNTMIN bits. It is also possible to transfer the current CNT_MAX value into CNT_MIN (or conversely) through the CNT_CMD.W1LMINMAX and CNT_CMD.W1LMAXMIN bits.

Another counter operation is the ability to only have the zero marker clear the CNT_CNTR register once. For more information, see the CNT functional description.

It is possible for multiple actions to be performed simultaneously by setting multiple bits in the CNT_CMD register. However, there are restrictions. The bits associated with each command have been grouped together such that all bits that involve a write to the CNT_CNTR, CNT_MAX, or CNT_MIN registers are located within bits 4-bit groups of the CNT_CMD register.

Note that a maximum of three commands can be issued at any one time, excluding the CNT_CMD.W1ZMONCE command. Also, note that CNT_CMD.W1LCNTMIN, CNT_CMD.W1LCNTMAX, and CNT_CMD.W1LCNTZERO bits have to be used exclusively. Never set more than one of them at the same time.

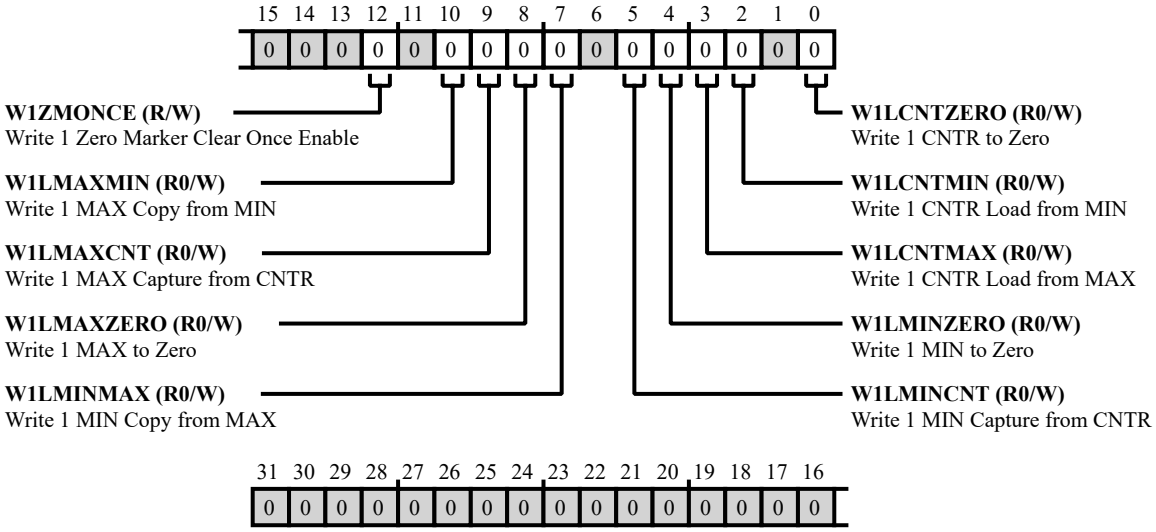


Figure 25-5: CNT_CMD Register Diagram

Table 25-9: CNT_CMD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12 (R/W)	W1ZMONCE	Write 1 Zero Marker Clear Once Enable. The `CNT_CMD.W1ZMONCE` enables a single zero marker clear of the `CNT_CNTR` register. Reading a 1 in this bit indicates that the bit has been set by software before, but no zero marker event has been detected on the `CNT_ZM` pin yet.
10 (R0/W)	W1LMAXMIN	Write 1 MAX Copy from MIN. The `CNT_CMD.W1LMAXMIN` bit transfers the current `CNT_MIN` register value into `CNT_MAX` register.
9 (R0/W)	W1LMAXCNT	Write 1 MAX Capture from CNTR. The `CNT_CMD.W1LMAXCNT` bit loads the current value in the `CNT_CNTR` register into the `CNT_MAX` register to create a new boundary limit.
8 (R0/W)	W1LMAXZERO	Write 1 MAX to Zero. Writing a 1 to the `CNT_CMD.W1LMAXZERO` bit clears the `CNT_MAX` register.
7 (R0/W)	W1LMINMAX	Write 1 MIN Copy from MAX. The `CNT_CMD.W1LMINMAX` bit transfers the current `CNT_MAX` register value into `CNT_MIN` register.
5 (R0/W)	W1LMINCNT	Write 1 MIN Capture from CNTR. The `CNT_CMD.W1LMINCNT` bit loads the current value in the `CNT_CNTR` register into the `CNT_MIN` register to create a new boundary limit.
4 (R0/W)	W1LMINZERO	Write 1 MIN to Zero. Writing a 1 to the `CNT_CMD.W1LMINZERO` bit clears the `CNT_MIN` register.

Table 25-9: CNT_CMD Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R0/W)	W1LCNTMAX	Write 1 CNTR Load from MAX. The CNT_CMD.W1LCNTMAX bit loads the current value in the CNT_MAX register into the CNT_CNTR register to create a new boundary limit.
2 (R0/W)	W1LCNTMIN	Write 1 CNTR Load from MIN. The CNT_CMD.W1LCNTMIN bit loads the current value in the CNT_MIN register into the CNT_CNTR register to create a new boundary limit.
0 (R0/W)	W1LCNTZERO	Write 1 CNTR to Zero. Writing a 1 to the CNT_CMD.W1LCNTZERO bit clears the CNT_CNTR register.

Counter Register

The CNT_CNTR register holds the 32-bit, two's-complement count value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows use of the CNT as a 16-bit counter if sufficient for the application.

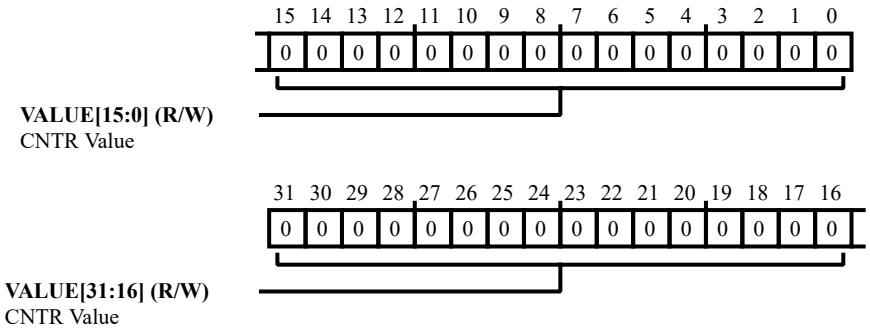


Figure 25-6: CNT_CNTR Register Diagram

Table 25-10: CNT_CNTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	CNTR Value. The CNT_CNTR.VALUE bit field holds the 32-bit, two's-complement count value.

Debounce Register

The CNT_DEBNCE register selects the noise filtering characteristic of the three input pins according to the formula:

t_{filter} = 128 x ($2^{DPRESCALE}$/ SCLK)

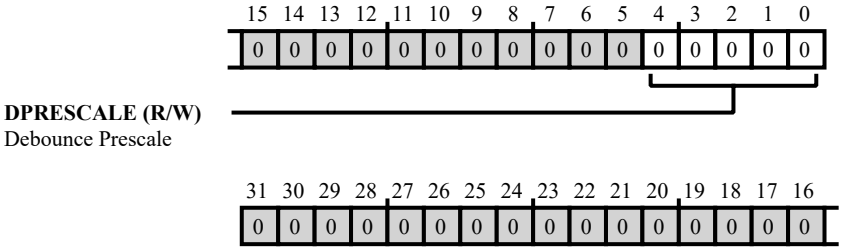


Figure 25-7: CNT_DEBNCE Register Diagram

Table 25-11: CNT_DEBNCE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4:0 (R/W)	DPRESCALE	Debounce Prescale. The CNT_DEBNCE.DPRESCALE selects the desired number of input filtering cycles (and resulting input debounce time) in multiples of SCLK.	
		0	1x Cycles = 128 SCLK Cycles
		1	2x Cycles
		2	4x Cycles
		3	8x Cycles
		4	16x Cycles
		5	32x Cycles
		6	64x Cycles
		7	128x Cycles
		8	256x Cycles
		9	512x Cycles
		10	1024x Cycles
		11	2048x Cycles
		12	4096x Cycles
		13	8192x Cycles
		14	16384x Cycles
		15	32768x Cycles
		16	65536x Cycles

Table 25-11: CNT_DEBNCE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		17	131072x Cycles
		18	Reserved from This Value. The values 10010 - 11111 are reserved.
		31	Reserved Until This Value

Interrupt Mask Register

The CNT_IMSK register supports enabling (unmasking) interrupt request generation from each of the CNT events.

All bits in CNT_IMSK either disable/mask an interrupt request (if bit cleared) or enable/unmask an interrupt request (if bit set).

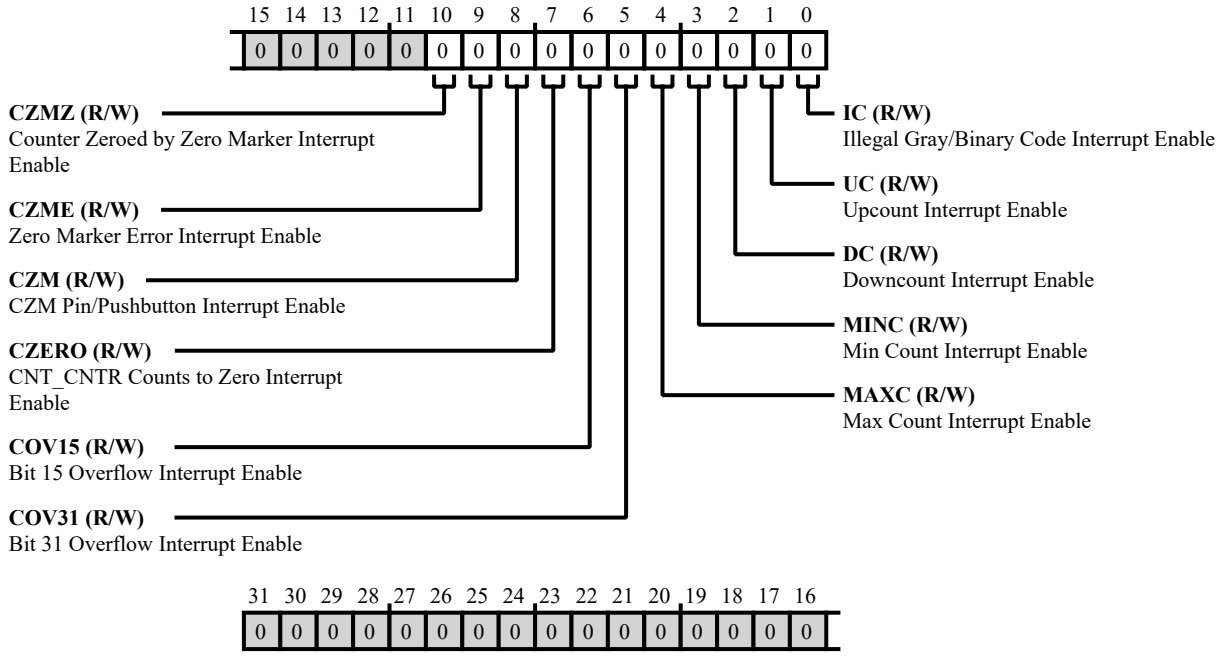


Figure 25-8: CNT_IMSK Register Diagram

Table 25-12: CNT_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	CZMZ	Counter Zeroed by Zero Marker Interrupt Enable. The CNT_IMSK.CZMZ bit enables (unmasks) the counter zeroed by zero marker interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
9 (R/W)	CZME	Zero Marker Error Interrupt Enable. The CNT_IMSK.CZME bit enables (unmasks) the zero marker error interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt

Table 25-12: CNT_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	CZM	CZM Pin/Pushbutton Interrupt Enable. The CNT_IMSK.CZM bit enables (unmasks) the CZM pin/pushbutton interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
7 (R/W)	CZERO	CNT_CNTR Counts to Zero Interrupt Enable. The CNT_IMSK.CZERO bit enables (unmasks) the counts to zero interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
6 (R/W)	COV15	Bit 15 Overflow Interrupt Enable. The CNT_IMSK.COV15 bit enables (unmasks) the bit 15 overflow interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
5 (R/W)	COV31	Bit 31 Overflow Interrupt Enable. The CNT_IMSK.COV31 bit enables (unmasks) the bit 31 overflow interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
4 (R/W)	MAXC	Max Count Interrupt Enable. The CNT_IMSK.MAXC bit enables (unmasks) the max count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
3 (R/W)	MINC	Min Count Interrupt Enable. The CNT_IMSK.MINC bit enables (unmasks) the min count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
2 (R/W)	DC	Downcount Interrupt Enable. The CNT_IMSK.DC bit enables (unmasks) the down count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt

Table 25-12: CNT_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	UC	Upcount Interrupt Enable. The `CNT_IMSK.UC` bit enables (unmasks) the up count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
0 (R/W)	IC	Illegal Gray/Binary Code Interrupt Enable. The `CNT_IMSK.IC` bit enables (unmasks) the illegal Gray/Binary Code interrupt request and should only be used in these modes.	
		0	Mask Interrupt
		1	Unmask Interrupt

Maximum Count Register

The CNT_MAX register holds the 32-bit, two's-complement, higher boundary value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows for using the CNT as a 16-bit counter if sufficient for the application.

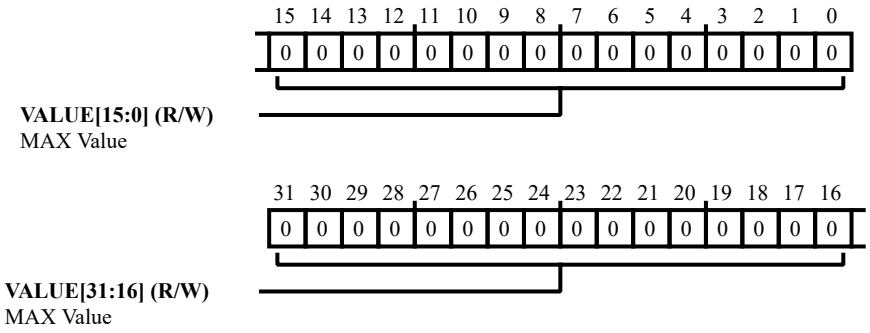


Figure 25-9: CNT_MAX Register Diagram

Table 25-13: CNT_MAX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	MAX Value. The CNT_MAX.VALUE bit field holds the 32-bit, two's-complement, higher boundary value.

Minimum Count Register

The CNT_MIN register holds the 32-bit, two's-complement, lower boundary value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows for using the CNT as a 16-bit counter if sufficient for the application.

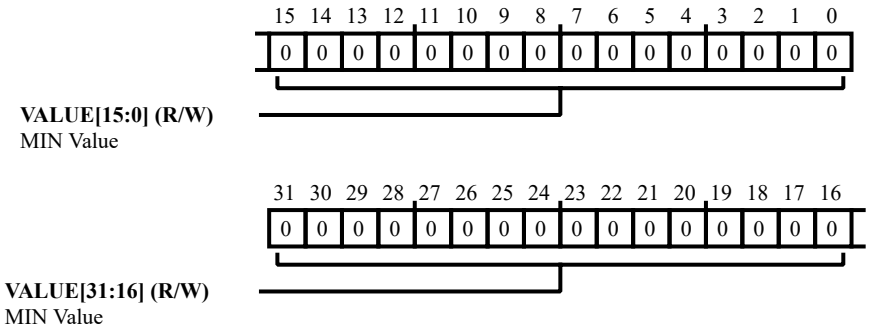


Figure 25-10: CNT_MIN Register Diagram

Table 25-14: CNT_MIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	MIN Value. The CNT_MIN.VALUE bit field holds the 32-bit, two's-complement, lower boundary value.

Status Register

The CNT_STAT register provides status information for each of the CNT events as configured in the CNT_IMSK register. When a CNT event is detected, the corresponding bit in this register is set. It remains set until either software writes a 1 to the bit (write-1-to-clear) or the CNT is disabled.

All bits in the CNT_STAT register indicate either no interrupt request pending (if bit cleared) or an interrupt request pending (if bit set).

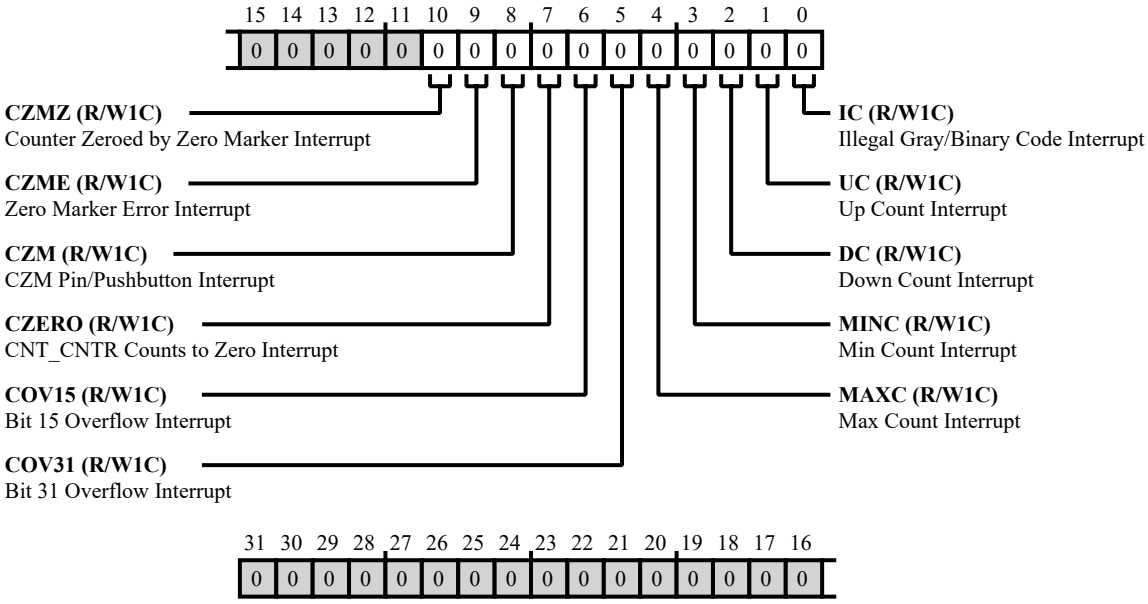


Figure 25-11: CNT_STAT Register Diagram

Table 25-15: CNT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W1C)	CZMZ	Counter Zeroed by Zero Marker Interrupt. The CNT_STAT.CZMZ bit indicates a zero marker error. If the CNT_CFG.ZMZC bit =1, this interrupt request is generated when the CZMII latch reports a significant edge on the CZM input. Once cleared by software the CNT_STAT.CZM bit is not set again when the CZM input remains active without pulsing.	
		0	No Error
		1	Error Occurred

Table 25-15: CNT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	CZME	Zero Marker Error Interrupt. The CNT_STAT.CZME bit behaves similarly to the CNT_STAT.CZM bit, with the exception that CNT_STAT.CZME is not set on the CZM edge when the lower four bits of the CNT_CNTR are not zero. In many applications this indicates an error condition, as the zero marker might be out of sync with the counter.	
		0	No Error
		1	Error Occurred
8 (R/W1C)	CZM	CZM Pin/Pushbutton Interrupt. The CNT_STAT.CZM bit indicates a CZM pin/pushbutton error. This interrupt request is generated when a significant edge is seen on the CZM pin, regardless what mode the counter is operating in. This is often used to sense push buttons (especially with the debouncing circuit enabled).	
		0	No Error
		1	Error Occurred
7 (R/W1C)	CZERO	CNT_CNTR Counts to Zero Interrupt. The CNT_STAT.CZERO bit indicates a counts to zero error. This error is generated when the CNT_CNTR register has incremented or decremented toward 0x0000.0000. The latch is not set when software writes to the CNT_CNTR register directly or when the counter is zeroed by writes to the CNT_CMD register.	
		0	No Error
		1	Error Occurred
6 (R/W1C)	COV15	Bit 15 Overflow Interrupt. The CNT_STAT.COV15 bit indicates a bit 15 overflow error. This error is generated when the 16-bit twos-complement CNT_CNTR register has incremented from 0xxxxx.7FFF to 0xxxxx.8000 or decremented from 0xxxxx.8000 to 0xxxxx.7FFF.	
		0	No Error
		1	Error Occurred
5 (R/W1C)	COV31	Bit 31 Overflow Interrupt. The CNT_STAT.COV31 bit indicates a bit 31 overflow error. This error is generated when the 32-bit twos-complement CNT_CNTR register has incremented from 0x7FFF.FFFF to 0x8000.0000 or decremented from 0x8000.0000 to 0x7FFF.FFFF.	
		0	No Error
		1	Error Occurred

Table 25-15: CNT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1C)	MAXC	Max Count Interrupt. The CNT_STAT.MAXC bit indicates a max count error. This interrupt is used in boundary compare (BND_COMP) mode. If after incrementing the CNT_CNTR register equals CNT_MAX, the CNT_STAT.MAXC bit is set.	
		0	No Error
		1	Error Occurred
3 (R/W1C)	MINC	Min Count Interrupt. The CNT_STAT.MINC bit indicates a minimum count error. This interrupt is used in boundary compare (BND_COMP) mode. If, after decrementing, the CNT_CNTR register equals CNT_MIN, the CNT_STAT.MINC bit is set.	
		0	No Error
		1	Error Occurred
2 (R/W1C)	DC	Down Count Interrupt. The CNT_STAT.DC bit indicates a down count error. This interrupt is generated when the CNT_CNTR register decrements.	
		0	No Error
		1	Error Occurred
1 (R/W1C)	UC	Up Count Interrupt. The CNT_STAT.UC bit indicates an up count interrupt. This interrupt is generated when the CNT_CNTR register increments.	
0 (R/W1C)	IC	Illegal Gray/Binary Code Interrupt. The CNT_STAT.IC bit indicates a illegal Gray/Binary Code interrupt and should only be used in these modes. In normal operation those codes can increment or decrement the CNT_CNTR register by one at a time. If the sensed inputs instruct the counter to increment or decrement by two, the CNT_STAT.IC bit is set. Hardware sets the CNT_STAT.IC bit in QUAD_ENC and BIN_ENC encoder modes only.	
		0	No Error
		1	Error Occurred

26 Media Local Bus (MLB)

Media Local Bus (MediaLB®) is an on-PCB or inter-chip communication bus, which allows an application to access the MOST network data. Media Local Bus supports all the MOST network data transport methods including synchronous stream data, asynchronous packet data, control message data and isochronous data. The MLB topology supports communication among the MLB controller and MLB devices, where the MLB controller is the interface between the MLB devices and the MOST network.

The MLB module serves as an interface between the MediaLB and the processor, implementing the requirements of the physical layer and the link layer outlined in the MediaLB specification. It supports up to 64 logical channels with up to 128 bytes of data per MediaLB frame. Transmit and receive data can be transferred between MediaLB and on-chip memory with DMA block transfers.

The MLB supports the MOST25, MOST50 and MOST150 standards and this document assumes familiarity with these standards. For more information, refer to the Media Local Bus specification version 4.2.

- Copyright 1998-2016 Microchip Technology Inc. All rights reserved. Portions of this chapter are included with permission from Microchip Technology, Inc.

Features

The objective of MLB is to map all the MOST Network data types (transport methods) into a single low-cost, scalable, and standardized hardware interface between a MediaLB controller and at least one other MediaLB Device. The adoption of MediaLB simplifies the hardware interface, reduces the pin count, and facilitates the design of modular reusable hardware. From a software development perspective, the use of MediaLB relieves the system developer from the complexity of the MOST network, which simplifies software development and enables the design of reusable software for different applications. This simplified, standardized interface shortens time to market and makes software maintenance effortless.

- Compliant to Media Local Bus specification version 4.2
- Support MOST25, MOST50, MOST150 standards
- 3-pin mode support various data rates of 256 × FS, 512 × FS, 1024 × FS
- Support 64 logic channels
- Shared pins for 3-pin mode

- Dedicated internal RAM for data buffering and channel table
- MLB 3-pin clock available out on a shared pin for DAI

MLB Definitions

The following are standard MLB and MOST terms that are used in this chapter.

AGU

Address Generation Units. To access a particular HBI channel the HC must first configure one of two HBI address generation units. AGUs can be configured by writing the HBI command registers, HCMD0 and HCMD1.

CAT

Channel Allocation Table. The Channel Allocation Table (CAT) is comprised of 16 CTR entries. Each 16-bit CAT entry represents a logical connection to or from a transmit/receive device (for example MediaLB or HBI channel).

CDT

Channel Descriptor Table.

FCE

Flow Control Enable bit. The FCE bit is used by MediaLB isochronous Rx channels only.

HBI

Host Bus Interface. The HBI block provides 16-bit parallel completer access to all MOST channels and data types for the external Host Controller (HC). The HBI supports up to 64 independent channels.

HC

Host Controller (external).

HCMDx

HBI Command registers.

HSTSx

HBI status registers.

MFE

Multi-Frame per sub-buffer enable bit. The MFE bit is used by MediaLB synchronous channels only.

PML

Packet Message Length.

Clocking

The MLB controller provides an external clock pin—the media local bus clock. The MLB controller generates the clock. It is synchronized to the MOST network and provides the timing for the entire MLB interface at FS = 48 kHz.

Functional Description

The ***MediaLB Block Diagram*** figure shows the MLB high-level architecture. The MLB core serves as an interface between the MediaLB and the processor, implementing the requirements of the physical layer and the link layer outlined in the MediaLB specification. The MLB core has the following responsibilities.

- Transmit commands and data when functioning as the transmitting device associated with a *Channel Address*
- Receive data and transmit Rx status responses when functioning as the receiving device associated *Channel Address*
- MLB lock detection
- System channel command handling

The MediaLB interfaces are capable of exchanging data at speeds up to 1024 × Fs in 3-pin mode.

A set of physical channels for exchanging data over the MediaLB bus is supported. These physical channels (4 bytes in length, or a quadlet) can be grouped into logical channels, where each logical channel is referenced using a channel address and represents a uni-directional datapath between a specific MediaLB device transmitting the data and the MediaLB device(s) receiving the data. The MediaLB 3-pin interface provides support for up to 128 bytes of data per frame. The logical channels, configured by system software, can be any combination of channel types (synchronous, asynchronous, isochronous, or control) and direction (transmit or receive).

ADSP-2184x MLB Register List

The MediaLB Device Interface Macro 2 (MediaLB DIM 2), also referred to as OS62420, implements the required functionality of a Media Local Bus (MediaLB) device. This logic serves as an interface between the inter-chip MediaLB bus and a customer IC, implementing the physical- and link-layer requirements outlined in the MediaLB Specification.

Table 26-1: ADSP-2184x MLB Register List

Name	Description
MLB_ACMR0	Peripheral Channel Mask 0 Register

Table 26-1: ADSP-2184x MLB Register List (Continued)

Name	Description
MLB_ACMR1	Peripheral Channel Mask 1 Register
MLB_ACSR0	Peripheral Channel Status 0 Register
MLB_ACSR1	Peripheral Channel Status 1 Register
MLB_ACTL	Bus Control Register
MLB_CTL0	MediaLB Control 0 Register
MLB_CTL1	Control 1 Register
MLB_GCTL	MLB Global Control Register
MLB_HCBR0	HBI Channel Busy 0 Register
MLB_HCBR1	HBI Channel Busy 1 Register
MLB_HCER0	HBI Channel Error 0 Register
MLB_HCER1	HBI Channel Error 1 Register
MLB_HCMR0	HBI Channel Mask 0 Register
MLB_HCMR1	HBI Channel Mask 1 Register
MLB_HCTL	HBI Control Register
MLB_MADR	Memory Interface Address Register
MLB_MCTL	Memory Interface Control Register
MLB_MDAT0	Memory Interface Control Data 0 Register
MLB_MDAT1	Memory Interface Control Data 1 Register
MLB_MDAT2	Memory Interface Control Data 2 Register
MLB_MDAT3	Memory Interface Control Data 3 Register
MLB_MDWE0	Memory Interface Control Data Write Enable 0 Register
MLB_MDWE1	Memory Interface Control Data Write Enable 1 Register
MLB_MDWE2	Memory Interface Control Data Write Enable 2 Register
MLB_MDWE3	Memory Interface Control Data Write Enable 3 Register
MLB_MIEN	Interrupt Enable Register
MLB_MS0	Channel Status 0 Register
MLB_MS1	Channel Status 1 Register
MLB_MSD	System Data Register
MLB_MSS	System Status Register

ADSP-2184x MLB Interrupt List

Table 26-2: ADSP-2184x MLB Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
211	MLB0_INT0	MLB0 Interrupt 0		
212	MLB0_INT1	MLB0 Interrupt 1		
213	MLB0_STAT	MLB0 Status		

MediaLB Protocol

The MediaLB topology supports communication among all MediaLB devices, including the MediaLB controller. The bus interface consists of a uni-directional line for clock (MLBC), a bidirectional line for signal information (MLBS), and a bidirectional line for data transfer (MLBD). The MediaLB topology supports one controller connected to one or more devices, where the controller is the interface between the MediaLB devices and the MOST network.

The MediaLB controller includes MediaLB device functionality, and also generates the MediaLB clock (MLBC) that is synchronized to the MOST Network. This generated clock provides the timing for the entire MediaLB interface. The MLBS line is a multiplexed signal which carries channel addresses generated by the MediaLB controller, as well as command and RxStatus bytes from MediaLB devices. The MLBD line is driven by the transmitting MediaLB device and is received by all other MediaLB devices, including the MediaLB controller. The MLBD line carries the actual data (synchronous, asynchronous, control, or isochronous).

Once per MOST network frame, the MLB controller generates a unique frame sync pattern on the `MLB_SIG` line. The end of the frame sync pattern defines the byte boundary and the channel boundary for the `MLB_SIG` and `MLB_DAT` lines of all MLB devices.

The MLB controller manages the arbitration for all the channels on the MLB and grants bandwidth for all the MLB devices. An MLB *physical channel* is defined as four bytes wide, or a quadlet. Physical channels can be grouped into multiple quadlets (which do not have to be consecutive) to form an MLB *logical channel*, which is defined by a unique channel address.

As shown in ***MLB Data Structure***, the MLB controller initiates communication by sending out a channel address on the `MLB_SIG` line for each physical channel. The channel address indicates which MLB device is transmitting and which MLB devices are receiving in the following physical channel. Therefore, four bytes after the controller outputs the channel address on the `MLB_SIG` line, the transmitting device outputs a command byte command on the `MLB_SIG` line and outputs the respective data on the `MLB_DAT` line, concurrently. The MLB command byte contains the type of data currently being transmitted (for example synchronous, asynchronous or control).

The MLB device receiving the channel data outputs a status byte, RxStatus, on the `MLB_SIG` line immediately after the transmitting device outputs the command byte. The status response can indicate that the receiving device is busy and cannot receive the data at present, or the device is ready to receive the data. Since synchronous stream data is sent in a broadcast fashion, receiving devices cannot return a busy status and should not drive RxStatus onto the `MLB_SIG` line.

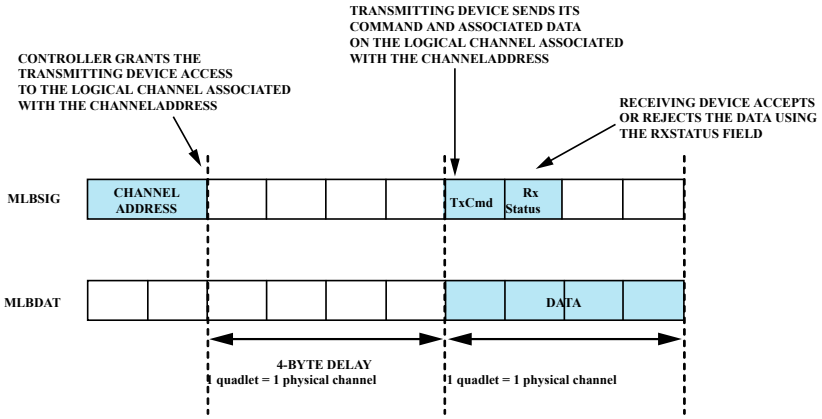


Figure 26-1: MLB Data Structure

MLB Architectural Concepts

The following sections provide information about the MLB architecture.

MediaLB Block Diagram

The ***MediaLB Block Diagram*** shows the various blocks within the interface its connections to the processor.

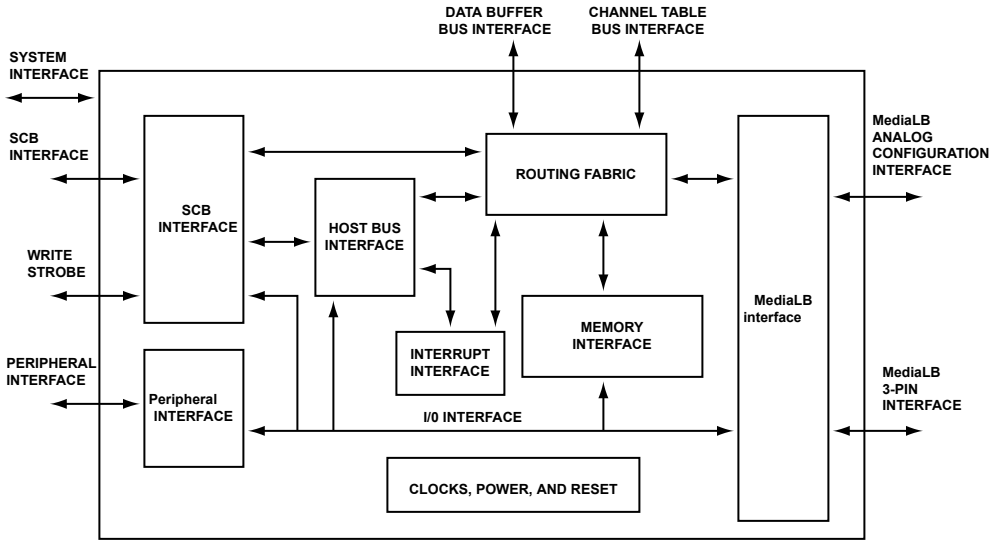


Figure 26-2: MediaLB Block Diagram

MediaLB Interface

The MediaLB interfaces provide real-time access to all network data types - synchronous, asynchronous, control, and isochronous data.

- MediaLB 3-pin interface–supports the MediaLB protocol for single-ended 3-pin mode, with a maximum data rate of 1024 FS (49.152 MHz at FS = 48 kHz).

MediaLB Channel Address to Logical Channel Mapping

The MediaLB channel addresses are mapped to the logical channels as shown in the ***MediaLB Channel Address to Logical Channel Mapping*** table.

Table 26-3: MediaLB Channel Address to Logical Channel Mapping

Channel Address	Logical Address
0x0002	1
0x0004	2
0x0006	3
...	...
0x007C	62
0x007E	63
0x01FE	0 (Logical channel 0 is the system channel and is reserved)

Routing Fabric

The Routing Fabric (RF) block manages the flow of data between the MediaLB port and the HBI port. It manages accessing the Channel Table RAM (CTR) and Data Buffer RAM (DBR), which are explained in the following subsections. The routing fabric uses channel descriptors (stored in the CTR) to manage access to dynamic buffers in the DBR.

Data Buffer RAM

The Data Buffer RAM (DBR) is an 8-bit x 16k entry buffer, single-port synchronous SRAM, and provides dynamic circular buffering between the transmit and receive devices. The size and location of each data buffer is defined by software in the Channel Descriptor Table (CDT), which is in the Channel Table RAM (CTR), described in following sections.

Channel Table RAM

The Channel Table RAM (CTR) is a 128-bit x 144-entry table that allows system software to dynamically configure channel routing and allocate data buffers in the DBR. The CTR is logically divided into three tables:

- Channel Descriptor Tables
- AHB Descriptor Table (ADT)
- Channel Allocation Table

Address Mapping

The ***CTR Address Mapping*** table shows the address mapping for the CTR.

Table 26-4: CTR Address Mapping

Label	Address	Bits [127:96]	Bits [95:64]	Bits [63:32]	Bits [31:0]
Channel Descriptor Table (CDT)					
CDT	0x00	CDT0[127:0], CL = 0			
	0x01	CDT1[127:0], CL = 1			
	0x02	CDT2[127:0], CL = 2			
	...				
	0x3D	CDT61[127:0], CL = 61			
	0x3E	CDT62[127:0], CL = 62			
	0x3F	CDT63[127:0], CL = 63			
AHB Descriptor Table (ADT)					
ADT	0x40	ADT0[127:0], CL = 0			
	0x41	ADT1[127:0], CL = 1			
	0x42	ADT2[127:0], CL = 2			
	...				
	0x7D	ADT61[127:0], CL = 61			
	0x7E	ADT62[127:0], CL = 62			
	0x7F	ADT63[127:0], CL = 63			

Channel Allocation Table (CAT)									
CAT for MediaLB	0x80	CAT7	CAT6	CAT5	CAT4	CAT3	CAT2	CAT1	CAT0
	...	...	...	...	...	...	...	...	...
	0x87	CAT63	CAT62	CAT61	CAT60	CAT59	CAT58	CAT57	CAT56
CAT for HBI*1	0x88	CAT71	CAT70	CAT69	CAT68	CAT67	CAT66	CAT65	CAT64
	...	...	...	...	...	...	...	...	...
	0x8F	CAT127	CAT126	CAT125	CAT124	CAT123	CAT122	CAT121	CAT120

*1 A fixed relationship exists between ADT entries and HBI CAT entries. When using HBI channel 0 (CAT64), program ADT0. When using HBI channel 1 (CAT65), program ADT1, and so on.

Channel Allocation Table

The ***Channel Allocation Table (CAT)*** table is comprised of 16 CTR entries (addresses 0x80 0x8F) as shown in the ***CTR Entry Map*** table. Each 16-bit CAT entry represents a logical connection to or from a transmit or receive device. (for example, MediaLB channel). All entries are indexed according to a fixed physical address assigned to every RX/TX channel as shown in the ***CAT Entry Formats*** table. The value stored in a CAT entry includes a 6-bit connection label, which provides a pointer to the CDT. To complete a logical channel and form a routing connection, system software must assign the same connection label to both the RX and TX channels.

Table 26-5: CAT Entry Map

Peripheral	TX Channels	RX Channels	CAT Start Index	CAT End Index	Entries
MediaLB	0 to 64	64 TX Channels	0	63	64
HBI	0 to 64	64 TX Channels	64	127	64

The format of a full CAT entry is shown in the ***CAT Entry Formats*** table, with field descriptions described in the ***CAT Field Definitions*** table. All reserved bits of a CAT entry field should be written as zero.

Table 26-6: CAT Entry Formats

Channel Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Isochronous	rsvd	FCE	rsvd	RNW	CE	CT[2:0] = 011			rsvd		CL[5:0]					
Asynchronous	rsvd		MT	RNW	CE	CT[2:0] = 010			rsvd		CL[5:0]					
Control	rsvd		MT	RNW	CE	CT[2:0] = 001			rsvd		CL[5:0]					
Synchronous	rsvd	MFE	MT	RNW	CE	CT[2:0] = 000			rsvd		CL[5:0]					

Table 26-7: CAT Field Definitions

Field	Description
CL[5:0]	Connection Label (offset into CDT)
CT[2:0]	Channel Type (others) 111 = Reserved 110 = Reserved 101 = Reserved 100 = Reserved 011 = Isochronous 010 = Asynchronous 001 = Control 000 = Synchronous
CE	Channel Enable. 0 = Disabled, 1 = Enabled
RNW	Read Not Write. 0 = Write, 1 = Read
MT	Mute Enable. 0 = Disabled When set for synchronous channels, the MT bit forces RX channels to write zeros into the channel data buffer, and TX channels to output zeros on the physical interface. When set for asynchronous and control channels, the MT bit causes DMA to halt at a packet boundary. Not valid for isochronous channels.
FCE	Flow Control Enable. 0 = Disabled, 1 = Enabled The FCE bit is used by MediaLB isochronous RX channels only.
MFE	Multi-Frame per sub-buffer enable. 0 = Disabled, 1 = Enabled The MFE bit is used by MediaLB synchronous channels only.

Table 26-7: CAT Field Definitions (Continued)

Field	Description
rsvd	Reserved. Software writes a 0 to all reserved bits when the entry is initialized. These bits are read-only after initialization.

Channel Set Up

Data direction is in reference to the DBR. The data direction of CAT entries corresponding to the same channel is reversed for the HBI CAT and the MediaLB CAT.

- For a Tx channel (from the HC to the MediaLB interface):
  - HBI CAT entry: RNW = 0 (write)
  - MediaLB CAT entry: RNW = 1 (read)
- Conversely, for an Rx channel (data from MediaLB to HC):
  - HBI CAT entry: RNW = 1 (read)
  - MediaLB CAT entry: RNW = 0 (write)

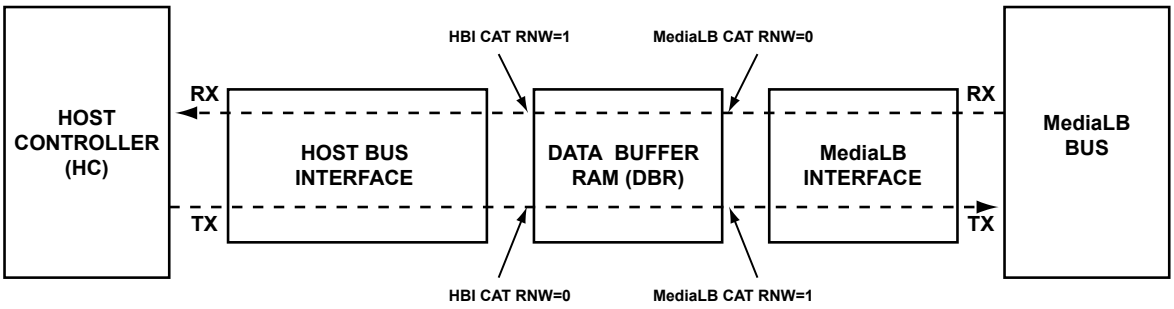


Figure 26-3: DBR Directional Relationship

Channel Descriptor Tables

The Channel Descriptor Table (CDT) is comprised of 64 CTR entries (addresses 0x00 - 0x3F), as shown in the Table 26-4 CTR Address Mapping table. Each 128-bit CDT entry (also referred to as a channel descriptor) is referenced by a connection label and contains information about a data buffer in the DBR (for example buffer size, address pointers). The format of each CDT entry is dependent on the channel type (synchronous, isochronous, asynchronous, or control).

NOTE: All reserved channel descriptor bits must be written to 0 by software when initialized.

Synchronous Channel Descriptors

The format and field definitions for a synchronous CDT entry are shown in the ***Synchronous CDT Entry Format*** and ***Synchronous CDT Entry Field Definitions*** tables.

Table 26-8: Synchronous CDT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WSBC[1:0]		Reserved													
16	RSBC[1:0]		Reserved													
32	Reserved															
48	Reserved															
64	WSTS[3:0]				WPTR[11:0]											
80	RSTS[3:0]				RPTR[11:0]											
96	Reserved				BD[11:0]											
112	Reserved		BA[13:0]													

Table 26-9: Synchronous CDT Entry Field Definitions

Field	Description	Details	Access
BA	Buffer Base Address	Can start at any byte in the 16k DBR	RW
BD	Buffer Depth	BD = size of buffer in bytes 1. Buffer end address = BA + BD. BD = 4 m bpf 1 where: m = frames per sub-buffer (for MFE = 0, m = 1) bpf = bytes per frame.	RW
RPTR	Read Pointer	Software initializes to 0, hardware updates. Counts the read address offset within a buffer. DMA read address = BA + RPTR.	RWU
WPTR	Write Pointer	Software initializes to 0, hardware updates. Counts the write address offset within a buffer. DMA read address = BA + WPTR.	RWU
RSBC	Read Sub-buffer Counter	Software initializes to 0, hardware updates. Counts the read sub-buffer offset. DMA uses for pointer management.	RWU
WSBC	Write Sub-buffer Counter	Software initializes to 0, hardware updates. Counts the write sub-buffer offset. DMA uses for pointer management.	RWU
RSTS	Read Status	Software initializes to 0, hardware updates. RSTS States: xxx0 = normal operation (no mute) xxx1 = normal operation (mute) xx0x = idle	RWU

Table 26-9: Synchronous CDT Entry Field Definitions (Continued)

Field	Description	Details	Access
WSTS	Write Status	Software initializes to 0, hardware updates. WSTS States: xxx0 = normal operation (no mute) xxx1 = normal operation (mute) xx0x = idle 1xxx = command protocol error	RWU
Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.			RWU

Isochronous Channel Descriptors

The format and field definitions for a synchronous CDT entry are shown in the ***Isochronous Entry Formats*** and ***Isochronous CDT Entry Field Definitions*** tables.

Table 26-10: Isochronous Entry Formats

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Reserved															
16	Reserved															
32	Reserved							BS[8:0]								
48	Reserved															
64	WSTS[3:0]				WPTR[12:0]											
80	RSTS[3:0]				RPTR[12:0]											
96	Reserved				BD[12:0]											
112	BF	Rsvd		BA[13:0]												

Table 26-11: Isochronous CDT Entry Field Definitions

Field	Description	Details	Access
BA	Buffer Base Address	Can start at any byte in the 16k DBR.	RW
BD	Buffer Depth	BD = size of buffer in bytes 1. Buffer end address = BA + BD. Isochronous buffers must be large enough to hold at least 3 blocks (packets) of data. BD Must be an integer multiple of blocks.	RW

Table 26-11: Isochronous CDT Entry Field Definitions (Continued)

Field	Description	Details	Access
BF	Buffer Full	Software initializes to 0, hardware updates. DMA write hardware sets BF when the buffer is full. DMA read hardware clears BF when the buffer is empty. BF is valid only when buffer is full or empty, otherwise ignore.	RWU
BS	Block Size	BS defines when to begin the DMA to the data buffer. BS = buffer block size in bytes 1. For RX channels, the DMA writes start when the number of empty bytes in the data buffer the block size. For TX channels, the DMA reads start when the number of valid bytes in the data buffer the block size.	RWU
RPTR	Read Pointer	Software initializes to 0, hardware updates. Counts the read address offset within a buffer. DMA read address = BA + RPTR.	RWU
WPTR	Write Pointer	Software initializes to 0, hardware updates. Counts the write address offset within a buffer. DMA write address = BA + WPTR.	RWU
RSTS	Read Status	Software initializes to 0, hardware updates. RSTS States: xx1 = active xx0 = idle	RWU
WSTS	Write Status	Software initializes to 0, hardware updates. WSTS States: xxx0 =active xxx1 = idle xx0x = command protocol error 1xxx = buffer overflow (FCE = 0 only)	RWU
Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.			RWU

Asynchronous/Control Channel Descriptors

The format and field definitions for an Asynchronous/Control CDT entry are shown in the ***Asynchronous/Control CDT Entry Format*** and ***Asynchronous/Control CDT Entry Field Definitions*** tables.

Table 26-12: Asynchronous/Control CDT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WPC[4:0]					Reserved										
16	RPC[4:0]					Reserved										
32	Rsvd	WPC[7:5]				Reserved										
48	Rsvd	RPC[7:5]				Reserved										
64	WSTS[3:0]					WPTR[11:0]										
80	RSTS[3:0]					RPTR[11:0]										
96	RSTS[4]	WSTS[4]	Rsvd			BD[11:0]										
112	Rsvd		BA[13:0]													

Table 26-13: Asynchronous/Control CDT Entry Field Definitions

Field	Description	Details	Access
BA	Buffer Base Address	Can start at any byte in the 16k DBR.	RW
BD	Buffer Depth	BD = size of buffer in bytes 1. Buffer end address = BA + BD. BD = max packet length 1.	RW
RPC	Read Packet Count	Software initializes to 0, hardware updates. Used with RPC, RPTR and WPTR to determine if the buffer is empty or full.	RWU
WPC	Write Packet Count	Software initializes to 0, hardware updates. Used with RPC, RPTR and WPTR to determine if the buffer is empty or full.	RWU
RPTR	Read Pointer	Software initializes to 0, hardware updates. Counts the read address offset within a buffer. DMA read address = BA + RPTR.	RWU
WPTR	Write Pointer	Software initializes to 0, hardware updates. Counts the write address offset within a buffer. DMA read address = BA + WPTR.	RWU
RSTS	Read Status	Software initializes to 0, hardware updates. RSTS States: x0x00 = idle xx1xx = *ReceiverProtocolError* response received from RX device 1xxxx = *ReceiverBreak* command received from RX device	RWU

Table 26-13: Asynchronous/Control CDT Entry Field Definitions (Continued)

Field	Description	Details	Access
WSTS	Write Status	Software initializes to 0, hardware updates. Status States (only valid for DMA pointers associated with the MLB block, not HBI block pointers): xxx0 = idle xxx1 = command protocol error detected xx0x = *AsyncBreak*/*ControlBreak* command received from TX device	RWU
Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.			RWU

AHB Descriptor Table (ADT)

The AHB block manages data exchange between local channel data buffers within MLB module and the system memory buffer. To support system memory buffering, a ping-pong memory structure is implemented on a per channel basis using 128-bit descriptors for AHB Descriptor Table (ADT) entries. The Table 26-4 CTR Address Mapping table provides a complete address map of the CTR, including the location of the ADT.

Each logical channel is assigned a separate 128-bit descriptor, defining the data buffers in the system memory used by the DMA interface for that channel. The descriptors are stored at fixed addresses in the CTR as described in previous section. The ***ADT Field Definitions*** table provides an overview of field definitions for ADT entries.

Table 26-14: ADT Field Definitions

Field	No. of Bits	Description	Access
CE	1	Channel Enable. 0 = Disabled	RW, U
LE	1	Endianness Select. 0 = Big Endian, 1 = Little Endian	RW
PG	1	Page pointer. Software initializes to 0, hardware writes thereafter. 0 = Ping buffer, 1 = Pong buffer	RW, U
RDY1	1	Buffer Ready bit for ping buffer page. 0 = Not ready, 1 = Ready	RW
RDY2	1	Buffer Ready bit for pong buffer page. 0 = Not ready, 1 = Ready	RW
DNE1	1	Buffer Done bit for ping buffer page. 0 = Not done, 1 = Done	R, U, c0
DNE2	1	Buffer Done bit for pong buffer page. 0 = Not done, 1 = Done	R, U, c0
ERR1	1	Error Response detected for ping buffer page. 0 = No error, 1 = Error	R, U, c0
ERR2	1	Error Response detected for pong buffer page. 0 = No error, 1 = Error	R, U, c0
PS1	1	Packet Start bit for ping buffer page. 0 = No packet start, 1 = Packet start Reserved for synchronous and isochronous channels.	RW, U both TX and RX
PS2	1	Packet Start bit for pong buffer page. 0 = No packet start, 1 = Packet start Reserved for synchronous and isochronous channels.	RW, U both TX and RX

Table 26-14: ADT Field Definitions (Continued)

Field	No. of Bits	Description	Access
MEP1	1	Most Ethernet Packet indicator for ping buffer page. 0 = Not MEP, 1 = MEP. MEP1 only valid for the first page of a segmented buffer. Reserved for control synchronous and isochronous channels.	Rsvd for TX, R, U, c0 for RX
MEP2	1	Most Ethernet Packet indicator for pong buffer page. 0 = Not MEP, 1 = MEP. MEP2 only valid for the first page of a segmented buffer. Reserved for control synchronous and isochronous channels.	Rsvd for TX, R, U, c0 for RX
BD1	11 to 13	Buffer Depth for ping buffer page. 11 or 12 bits for asynchronous and control channels. 13 bits for synchronous and isochronous channels.	RW
BD2	11 to 13	Buffer Depth for pong buffer page. 11 or 12 bits for asynchronous and control channels. 13 bits for synchronous and isochronous channels.	RW
BA1	32	Buffer Base Address for ping buffer page	RW
BA2	32	Buffer Base Address for pong buffer page.	RW
Reserved	varies	Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.	RW, U

The ***Ping-Pong System Memory Structure*** figure shows that this system memory structure is similar for all channel types and shows the relationship between the BAn, BDn, and PG descriptor fields.

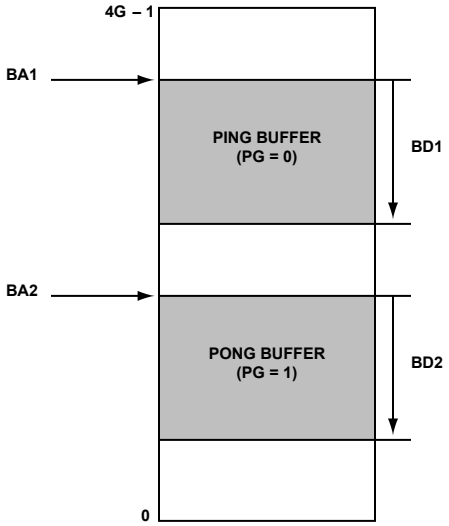


Figure 26-4: Ping-Pong System Memory Structure

Each ADT entry (also referred to as a *Channel Descriptor)* holds a 32-bit BAn field which defines the start of each ping or pong buffer within system memory. The BDn field is used to indicate the size for the respective ping or pong page. The maximum size is 2k entries for asynchronous and control channels and 8k entries for isochronous and synchronous channels.

Synchronous Channel Descriptors

The synchronous buffering scheme allows each ping or pong buffer to contain a single frame or a multiple number of frames. For this reason, the synchronous buffer depth (BDn) must be defined in terms of an integer number (n), frames per sub-buffer (m) and bytes per frame (bpf) of data (for example BDn = n m bpf 1). The ***Synchronous ADT Entry Format*** table shows the format for a synchronous ADT entry. The field definitions are defined in the ***ADT Field Definitions*** table. Each synchronous channel buffer can be up to 8k-bytes deep.

Table 26-15: Synchronous ADT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	BD1[12:0]												
48	RDY2	DNE2	ERR2	BD2[12:0]												
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Isochronous Channel Descriptors

The isochronous buffering scheme allows each ping or pong buffer to contain a single block or a multiple number of blocks. For this reason, the isochronous buffer depth (BDn) must be defined in terms of an integer number (n) and block size (BS) (for example BDn = n (BS + 1) 1).

The ***Isochronous ADT Entry Format*** table shows the format for an isochronous ADT entry. The field definitions are defined in the ***ADT Field Definitions*** table. Each isochronous channel buffer can be up to 8k-bytes deep.

Table 26-16: Isochronous ADT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	BD1[12:0]												
48	RDY2	DNE2	ERR2	BD2[12:0]												
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Asynchronous and Control Channel Descriptors

Every asynchronous and control packet adheres to the Port Message Protocol (PMP), which designates the first two bytes of each packet as the packet length (PML). Each packet must be no more than 2048-bytes (PML 2048).

Software must set the buffer ready bit (RDYn) for each buffer as it programs the DMA. As hardware processes each buffer, it sets the done bit (DNEn) and generates an interrupt to inform HC. When hardware finishes processing a buffer, it can begin processing another buffer if RDYn is set. The application is responsible for setting up and configuring the channel buffer descriptor prior to every DMA access on the channel.

Two packet buffering modes are supported by hardware for programming the DMA, single-packet mode (`MLB_ACTL.MPB` =0) and multiple-packet mode (`MLB_ACTL.MPB` =1). The MPB is written prior to enabling the channel DMA.

Single Packet Mode. The single-packet mode asynchronous and control buffering scheme supports a maximum of one packet per buffer (for example, ping or pong). Both non-segmented and segmented data packets are allowed while using single-packet mode. Non-segmented packets are exchanged when only one buffer (for example, ping or pong) is needed for packet transfer. Segmented packets are exchanged when a single packet is too long for one buffer and the packet must span multiple buffers.

The ***Single-Packet Asynchronous and Control Entry Format*** table shows the format for single-packet mode asynchronous and control ADT entries. The field definitions are defined in the ***ADT Field Definitions*** table.

Table 26-17: Single-Packet Asynchronous and Control Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	PS1	MEP1	BD1[10:0]										
48	RDY2	DNE2	ERR2	PS2	MEP2	BD2[10:0]										
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Multiple Packet Mode. The multiple-packet mode asynchronous and control buffering scheme supports more than one packet per system memory buffer, as shown in the ***Asynchronous/Control CDT Entry Format*** table. Multiple-packet mode reduces the interrupt rate for packet channels at the cost of increasing buffering and latency.

For TX packet channels in multiple-packet mode, software sets the packet start bit (PSn) for every buffer. Setting PSn informs hardware that the first two bytes of the buffer contains the port message length (PML) of the first packet. After the first packet, hardware keeps track of where packets start and end within the current buffer. Software should not write to PSn while the buffer is active (RDYn = 1 and DNEn = 0). For TX packet channels, the buffer is done (DNEn = 1) when the last byte of the last packet in the buffer is read from system memory. Software

should set the buffer depth to contain the exact number of complete packets for that buffer. Segmented buffers are not supported for TX packet channels in multiple-packet mode.

NOTE: The PS1 and PS2 bits are only valid for TX channels. Set PS1 and PS2 = 1 at the start of the buffer.

Table 26-18: Multiple-Packet Asynchronous and Control Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	PS1	BD1[11:0]											
48	RDY2	DNE2	ERR2	PS2	BD2[11:0]											
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Interrupt Interface Block

The interrupt interface raises an interrupt when specific changes to HBI channel descriptors occur, including:

- For asynchronous and control read/write channels:
  - a packet is available to read in the channel buffer, or
  - sufficient empty space is available in the channel buffer to accept a requested packet write
- For isochronous read/write channels:
  - the number of valid bytes in the channel buffer exceeds the block size, or
  - the number of empty bytes in the channel buffer exceeds the block size

Operating Modes

The following sections describe the operating modes of the MLB interface. The channel type selection enables the logical channels to operate in synchronous, asynchronous, isochronous, or control channels.

NOTE: The logical channels can be any combination of channel type (for example synchronous, asynchronous, or control) and direction (transmit or receive).

Isochronous Data Exchange

An isochronous HBI channel is initially opened and synchronized with HCMD0.CMD[2:0]= 010. For isochronous channels, no further synchronization is required from the HBI perspective; however, an optional resynchronization command is available for HC flexibility. Setting HCMD0.CMD[2:0]= 011 reinitializes the address pointer within

the data buffer, ensuring that subsequent data exchange with the channel is aligned at an isochronous packet boundary. When the HC must close an isochronous channel before it has read or written an entire data packet, setting HCMD0.CMD[2:0]= 000 reopens the channel without synchronizing the address pointer in the buffer. This action allows reading and writing to continue where the HC previously stopped.

For isochronous data transmission, the ***Exchanging Isochronous Data on an HBI Channel*** figure shows the flow control.

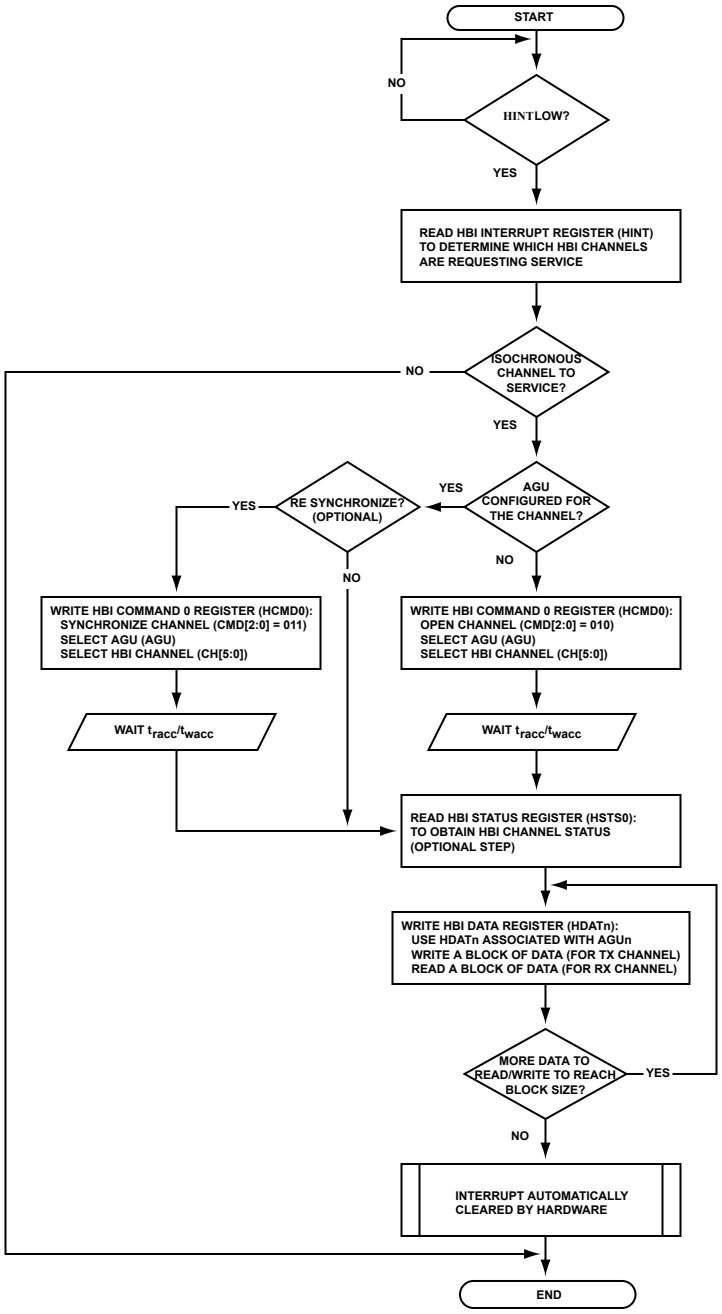


Figure 26-5: Exchanging Isochronous Data on an HBI Channel

Asynchronous and Control Data Exchange

An asynchronous or control HBI channel is initially opened using HCMD0.CMD[2:0]= 010. Occasionally, the HC may need to close a packet channel before it has completed writing or reading the current packet of data (for example, the HC needs to use this AGU to service another HBI channel). In this case, the HC can reopen the previous channel with HCMD0.CMD[2:0]= 000. This situation allows the HC to continue writing or reading a packet from the point it left off. In this situation, the PML is already known, and the packet length is not reread or rewritten by the HC.

During the reading of a packet, if the HC sets HCMD0.CMD[2:0]= 010 for the channel before the last byte of the packet is read, an internal hardware pointer is reset to the beginning of the packet buffer. This situation requires that the HC reread the PML (from HSTS1) and reread the packet data (from HDATn) from the beginning. In the same manner, if the HC resets HCMD0.CMD[2:0]= 010 for the channel before the last byte of a packet is written, an internal hardware pointer is reset to the beginning of the packet buffer. This situation requires that the HC rewrite the PML (to HCMD1) and rewrite the packet data (to HDATn) from the beginning. Any previous packet data in the buffer is overwritten.

Frame synchronization is not supported for asynchronous channels.

Synchronous Data Exchange

The MLB core provides two modes of operation; standard and multi-frame per sub-buffer which provide flexibility for implementing synchronous channels. Channels configured for standard mode require less buffer space but have higher interrupt rates and more stringent latency requirements. Channels setup for each multi-frame per sub-buffer mode requires more buffer space, but has lower interrupt rates and less stringent latency requirements.

To set up a channel in multi-frame per sub-buffer mode:

1. Program the `MLB_CTL0.FCNT` bit field to select the number of frames per sub-buffer.
2. Program the CAT to enable multi-frame sub-buffering (MFE= 1) for each channel.
3. Set the buffer depth in the CDT: BD= 4 × m × bpf 1 where: m = frames per sub-buffer, bpf = bytes per frame.
4. Repeat for additional synchronous channels

A sample synchronous data buffer is shown in the ***Synchronous Data Buffer Structure*** figure. Each data buffer contains four sub-buffers and each sub-buffer contains space for 1 to 64 frames of data, determined by the `MLB_CTL0.FCNT` bits.

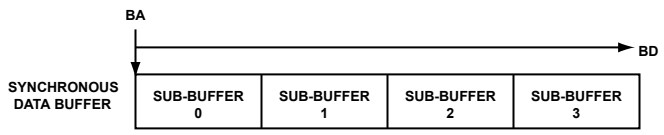


Figure 26-6: Synchronous Data Buffer Structure

Data Transfer

Two modes of operation are supported for transferring channel data between the MLB and internal memory. DMA allows the multi-channel DMA engine to manage data transfers without core intervention. Core driven mode (I/O mode) allows software to manage the transfer of data between MLB and internal memory.

NOTE: All hardware channels must use the same data transfer method. Mixed mode operation where hardware channels operate in both I/O mode and DMA mode is not supported.

DMA

The processor supports DMA mode which uses only INCR8, INCR4 and SINGLE beat bursts. Program the `MLB_ACTL.DMAMODE` = 1 to use DMA mode.

Programming Model

This section provides general guidelines for programming the MediaLB interface.

Channel Initialization

The software flow required to initialize a channel must be performed to ensure proper operation.

Configure the Hardware

1. Initialize the CTR and registers.
    a. Set all the CTR (CAT, CDT, and ADT) bits to 0.
    b. Set all bits of all registers to 0.
2. Configure the MediaLB interface.
    a. Select 3-pin operation by setting the `MLB_CTL0.PEN` =0.
    b. Select MediaLB clock speed via `MLB_CTL0.CLK`.
    c. Set MediaLB enable via `MLB_CTL0.EN`.
3. Configure the HBI interface.
    a. Set `MLB_HCMR0` and `MLB_HCMR1` = 0xFFFFFFFF to activate all channels.
    b. Set the HBI enable bit: `MLB_HCTL.EN` =1.

Program the CAT and the CDT

1. Initialize all bits of the CAT to 0.

2. Select a logical channel: N = 1 – 63.
3. Program the CDT for channel N.
    a. Set the 14-bit base address (BA)
    b. Set the 12-bit or 13-bit buffer depth (BD): BD = buffer depth in bytes 1
    - For synchronous channels: (BD + 1) = 4 frames per sub-buffer (m) bytes-per-frame (bpf)
    - For isochronous channels: (BD + 1) mod (BS + 1) = 0
    - For asynchronous channels: (BD + 1) max packet length (1024 for a MOST Data Packet (MDP)
    - 1536 for a MOST Ethernet Packet (MEP)
    - For control channels: (BD + 1) max packet length (64)
4. Program the CAT for the inbound DMA.
    a. For TX channels (to MediaLB) HBI is the inbound DMA
    b. For RX channels (from MediaLB) MediaLB is the inbound DMA
    c. Set the channel direction: RNW = 0
    d. Set the channel type: CT[2:0] = 010 (asynchronous), 001 (control), 011 (isochronous), or 000 (synchronous)
    e. Set the connection label: CL[5:0] = N
    f. If CT[2:0] = 000 (synchronous), set the mute bit (MT = 1)
    g. Set the channel enable: CE = 1
    h. Set all other bits of the CAT to 0
5. Program the CAT for the outbound DMA.
    a. For TX channels (to MediaLB) HBI is the outbound DMA
    b. For RX channels (from MediaLB) MediaLB is the outbound DMA
    c. Set the channel direction: RNW = 1
    d. Set the channel type: CT[2:0] = 010 (asynchronous), 001 (control), 011 (isochronous), or 000 (synchronous)
    e. Set the channel label: CL[5:0] = N
    f. If CT[2:0] = 000 (synchronous), set the mute bit (MT = 1)
    g. Set the channel enable: CE = 1
    h. Set all other bits of the CAT to 0
6. Repeat steps 2 through 5 to initialize all logical channels.

Program the ADT

1. Initialize all bits of the ADT to 0
2. Select a logical channel: N = 1 – 63
3. Program the AMBA AHB block ping page for channel N
    a. Set the 32-bit base address (BA1)
    b. Set the 11-bit buffer depth (BD1): BD1 = buffer depth in bytes – 1
        - For synchronous channels: (BD1 + 1) = n frames per sub-buffer (m) bytes-per-frame (bpf)
        - For isochronous channels: (BD1 + 1) mod (BS + 1) = 0
        - For asynchronous channels: 5 (BD1 + 1) 4096 (max packet length)
        - For control channels: 5 (BD1 + 1) 4096 (max packet length)
    c. For asynchronous and control Tx channels set the packet start bit (PS1) when the page contains the start of the packet
    d. Clear the page done bit (DNE1)
    e. Clear the error bit (ERR1)
    f. Set the page ready bit (RDY1)
4. Program the AMBA AHB block pong page for channel N
    a. Set the 32-bit base address (BA2)
    b. Set the 11-bit buffer depth (BD2): BD2 = buffer depth in bytes – 1
        - For synchronous channels: (BD2 + 1) = n frames per sub-buffer (m) bytes-per-frame (bpf)
        - For isochronous channels: (BD2 + 1) mod (BS + 1) = 0
        - For asynchronous channels: 5 (BD2 + 1) 4096 (max packet length)
        - For control channels: 5 (BD2 + 1) 4096 (max packet length)
    c. For asynchronous and control TX channels set the packet start bit (PS2) when the page contains the start of the packet
    d. Clear the page done bit (DNE2)
    e. Clear the error bit (ERR2)
    f. Set the page ready bit (RDY2)
5. Select Big Endian (LE = 0) or Little Endian (LE = 1)
6. Select the active page: PG = 0 (ping), PG = 1 (pong)

7. Set the channel enable (CE) bit for all active logical channels
8. Repeat steps 2 through 7 for all active logical channels.

Service

After initialization, each channel will require periodic servicing. Use the procedures in the following sections to service DMA and MLB interrupts and to poll for MLB system commands.

Servicing the DMA Channel Interrupts

1. Program the MLB_ACMR0/MLB_ACMR1 registers to enable interrupts from all active DMA channels.
2. Select the status clear method. MLB_ACTL.SCE =0 (hardware clears on read), MLB_ACTL.SCE = 1 (software writes a 1 to clear).
3. Select 1 or 2 interrupt signals. Configure the MLB_ACTL.SMX bit =0 (one interrupt for channels 0 through 31 and another interrupt for channels 32 through 63 on). Configure the MLB_ACTL.SMX bit =1 (single interrupt for all channels).
4. Wait for an interrupt.
5. Read the MLB_ACSR0/MLB_ACSR1 registers to determine which channel or channels are causing the interrupt.
6. If the MLB_ACTL.SCE bit =1, write the results of step 5 back to the MLB_ACSR0 and MLB_ACSR1 registers to clear the interrupt.
7. Select a logical channel (N = 0 – 63) with an interrupt to service.
8. Read the ADT entry for channel N to:
    a. Determine the active page (ping or pong) via the PG bit
    b. Determine which page(s) are done via the DNEn bits
    c. Determine which channels encountered an AHB error via the ERRn bit
    d. Determine which asynchronous and control Rx channel pages contain a packet start via the PSn bit (extract the PML)
9. Repeat steps 6 through 8 for all channels with pending interrupts
10. Repeat steps 4 through 9 while there are active channels.

Servicing the MediaLB Status Interrupts

1. Select the MediaLB channel status register (MLB_MS0/MLB_MS1) to be cleared by software, writing a 0 to the appropriate bits.

2. Program the MLB_MIEN register to enable protocol error interrupts for all active MediaLB channels. (MLB_MIEN.CTXPE =1, MLB_MIEN.CRXPE =1, MLB_MIEN.ATXPE =1, MLB_MIEN.ARXPE =1, MLB_MIEN.SYNCPE =1, and MLB_MIEN.ISOCPE =1).
3. Wait for an interrupt on the MLB_INT0/MLB_INT1 signal.
4. Read the MLB_MS0/MLB_MS1 registers to determine which channel(s) are causing the interrupt.
5. Read the RSTS/WSTS of the appropriate CDT(s) to determine the interrupt type.
6. Clear the RSTS/WSTS errors to ensure that the status of channel operations is reflected in the register:

Option	Description
For synchronous RX channels	WSTS[3] = 0
For synchronous TX channels	RSTS[3] = 0
For isochronous RX channels	WSTS[2:1] = 00
For isochronous TX channels	RSTS[2:1] = 00
For asynchronous and control RX channels	WSTS[4] = 0 and WSTS[2] = 0
For asynchronous and control TX channels	RSTS[4] = 0 and RSTS[2] = 0

Polling for MediaLB System Commands

The MediaLB System status (MLB_MSS) register is used to detect a system command received from the MediaLB controller. The processor's peripheral automatically sends the appropriate system response to the MediaLB controller. The procedure for the application is:

1. The application periodically polls the MLB_MSS register.
2. Clear by writing a 0 to the appropriate bit in the MLB_MSS register after the application finishes the service.
3. If MLB_MSS.SWSYSCMD = 1, read the MLB_MSD register to receive the system data sent from MediaLB controller.

ADSP-2184x MLB Register Descriptions

MediaLB Device Interface Macro 2 (MLB) contains the following registers.

Table 26-19: ADSP-2184x MLB Register List

Name	Description
MLB_ACMR0	Peripheral Channel Mask 0 Register
MLB_ACMR1	Peripheral Channel Mask 1 Register
MLB_ACSR0	Peripheral Channel Status 0 Register
MLB_ACSR1	Peripheral Channel Status 1 Register

Table 26-19: ADSP-2184x MLB Register List (Continued)

Name	Description
MLB_ACTL	Bus Control Register
MLB_CTL0	MediaLB Control 0 Register
MLB_CTL1	Control 1 Register
MLB_GCTL	MLB Global Control Register
MLB_HCBR0	HBI Channel Busy 0 Register
MLB_HCBR1	HBI Channel Busy 1 Register
MLB_HCER0	HBI Channel Error 0 Register
MLB_HCER1	HBI Channel Error 1 Register
MLB_HCMR0	HBI Channel Mask 0 Register
MLB_HCMR1	HBI Channel Mask 1 Register
MLB_HCTL	HBI Control Register
MLB_MADR	Memory Interface Address Register
MLB_MCTL	Memory Interface Control Register
MLB_MDAT0	Memory Interface Control Data 0 Register
MLB_MDAT1	Memory Interface Control Data 1 Register
MLB_MDAT2	Memory Interface Control Data 2 Register
MLB_MDAT3	Memory Interface Control Data 3 Register
MLB_MDWE0	Memory Interface Control Data Write Enable 0 Register
MLB_MDWE1	Memory Interface Control Data Write Enable 1 Register
MLB_MDWE2	Memory Interface Control Data Write Enable 2 Register
MLB_MDWE3	Memory Interface Control Data Write Enable 3 Register
MLB_MIEN	Interrupt Enable Register
MLB_MS0	Channel Status 0 Register
MLB_MS1	Channel Status 1 Register
MLB_MSD	System Data Register
MLB_MSS	System Status Register

Peripheral Channel Mask 0 Register

The MLB_ACMR0 register allows control over which channel(s) generate interrupts on MLB_INT[1:0]. All of the bits in this register default to 0 (masked) so the HC must initially write to the MLB_ACMR0 register to enable interrupts. Each bit of this register is read/write accessible.

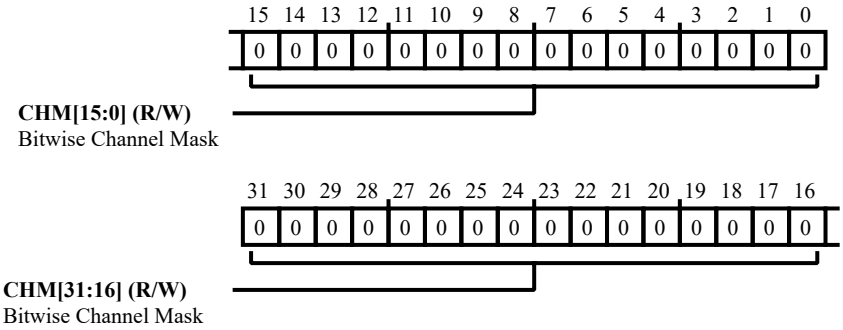


Figure 26-7: MLB_ACMR0 Register Diagram

Table 26-20: MLB_ACMR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	CHM	Bitwise Channel Mask. The MLB_ACMR0.CHM bit field masks or unmasks channels 31-0.	
		0	Mask interrupt for Channel 31:0 (bitwise; all channels shown)
		4294967295	Unmask interrupt for Channel 31:0 (bitwise; all channels shown)

Peripheral Channel Mask 1 Register

The MLB_ACMR1 register allows control over which channel(s) generate interrupts on MLB_INT[1:0]. All of the bits in this register default to 0 (masked) so the HC must initially write to the MLB_ACMR1 register to enable interrupts. Each bit of this register is read/write accessible.

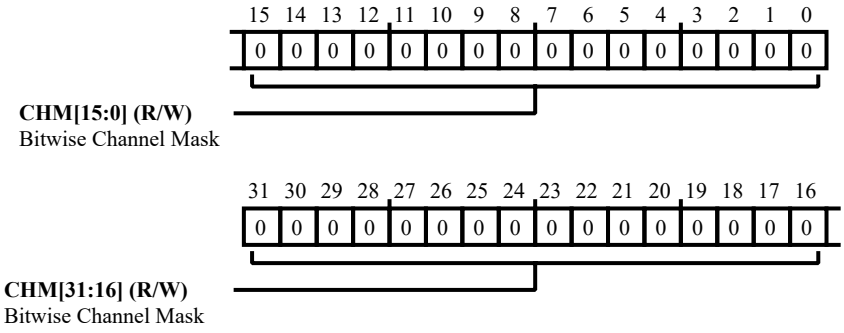


Figure 26-8: MLB_ACMR1 Register Diagram

Table 26-21: MLB_ACMR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	CHM	Bitwise Channel Mask. The MLB_ACMR1.CHM bit field masks or unmasks channels 63-32.	
		0	Mask interrupt for Channel 63:32 (bitwise; all channels shown)
		4294967295	Unmask interrupt for Channel 63:32 (bitwise; all channels shown)

Peripheral Channel Status 0 Register

The MLB_ACSR0 register contains interrupt bits for each of the 64 physical channels. When a bit in this register is set, it indicates that the corresponding physical channel has an interrupt pending.

A peripheral interrupt is triggered when either DNEn or ERRn is set within the Bus Channel Descriptor. The HC is notified of the channel interrupt via MLB_INT[1:0]. When an interrupt occurs in ACCUSER (for channels 31 to 0) MLB_INT[0] is set. When an interrupt occurs in MLB_ACSR1 (for channels 63 to 32) MLB_INT[1] is set.

Interrupts in the MLB_ACSR0 and MLB_ACSR1 registers can be optionally multiplexed onto a single interrupt signal, MLB_INT[0], if the MLB_ACTL.SMX bit = 1.

If the MLB_ACTL.SCE bit =0, hardware automatically clears the interrupt bit(s) after the HC reads the peripheral channel status registers. Alternatively, if the MLB_ACTL.SCE bit =1, software must write a 1 to the appropriate bit(s) of the peripheral channel status registers to clear the interrupt(s).

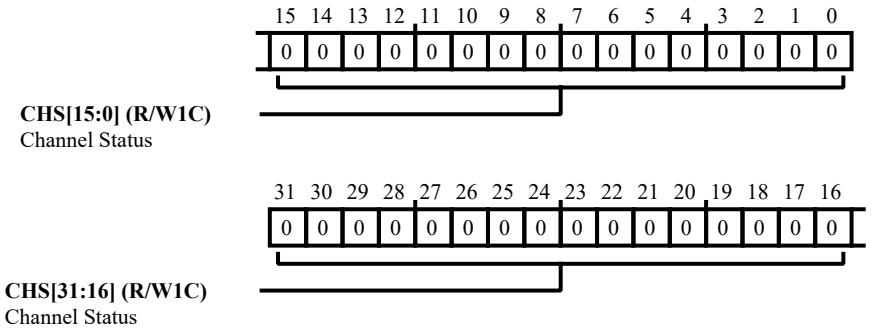


Figure 26-9: MLB_ACSR0 Register Diagram

Table 26-22: MLB_ACSR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W1C)	CHS	Channel Status. The MLB_ACSR0.CHS bit field indicates channel status for channels 31-0.	
		0	No interrupt on Channel 31:0 (bitwise; all channels shown)
		4294967295	Interrupt on Channel 31:0 (bitwise; all channels shown)

Peripheral Channel Status 1 Register

The MLB_ACSR1 register contains interrupt bits for each of the 64 physical channels. When a bit in this register is set, it indicates that the corresponding physical channel has an interrupt pending.

A peripheral interrupt is triggered when either DNEn or ERRn is set within the Bus Channel Descriptor. The HC is notified of the channel interrupt via MLB_INT[1:0]. When an interrupt occurs in ACCUSER (for channels 31 to 0) MLB_INT[0] is set. When an interrupt occurs in MLB_ACSR1 (for channels 63 to 32) MLB_INT[1] is set.

Interrupts in the MLB_ACSR1 and MLB_ACSR0 registers can be optionally multiplexed onto a single interrupt signal, MLB_INT[0], if the MLB_ACTL.SMX bit = 1.

If the MLB_ACTL.SCE bit =0, hardware automatically clears the interrupt bit(s) after the HC reads the peripheral channel status registers. Alternatively, if the MLB_ACTL.SCE bit =1, software must write a 1 to the appropriate bit(s) of peripheral channel status registers to clear the interrupt(s).

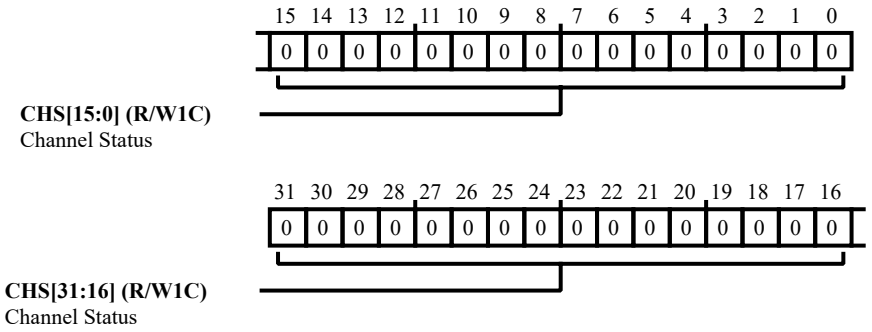


Figure 26-10: MLB_ACSR1 Register Diagram

Table 26-23: MLB_ACSR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W1C)	CHS	Channel Status. The MLB_ACSR1.CHS bit field indicates channel status for channels 63-32.	
		0	No interrupt on Channel 63:32 (bitwise; all channels shown)
		4294967295	Interrupt on Channel 63:32 (bitwise; all channels shown)

Bus Control Register

The MLB_ACTL register is written by the HC to configure the block for channel interrupts. This register contains bits to select the packet buffering mode and the DMA mode. This register also contains bits that are used to multiplex channel interrupts onto a single interrupt signal and to select the method of clearing channel interrupts.

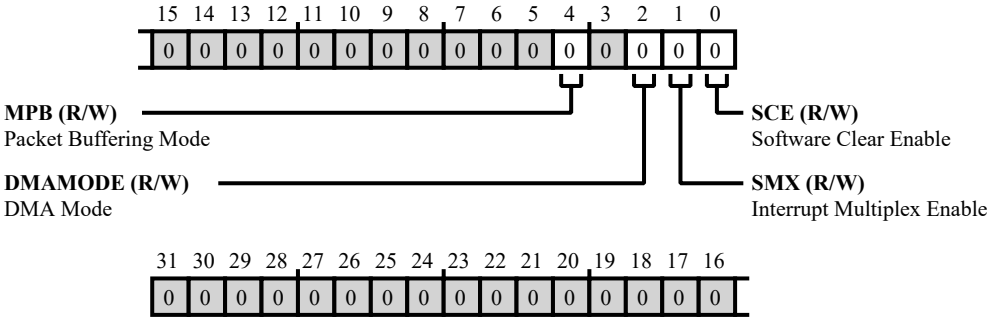


Figure 26-11: MLB_ACTL Register Diagram

Table 26-24: MLB_ACTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	MPB	Packet Buffering Mode. The MLB_ACTL.MPB bit selects whether the buffering mode is single-packet or multiple-packet.	
		0	single-packet mode
		1	multiple-packet mode
2 (R/W)	DMAMODE	DMA Mode. The MLB_ACTL.DMAMODE bit selects between DMA mode 1 and 0. DMA Mode 0 uses incrementing bursts of an unspecified length. This allows the block to perform single beat transfers as well as an incrementing (INCR) burst of unspecified length up to the maximum specified burst length (8 beats). DMA Mode 1 uses only INCR8, INCR4, and SINGLE beat bursts. The hburst[2:0] signal encodes the burst transfer used for a DMA operation.	
		0	DMA Mode 0
		1	DMA Mode 1

Table 26-24: MLB_ACTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SMX	Interrupt Multiplex Enable. The `MLB_ACTL.SMX` bit selects whether ACSR0 generates an interrupt on MLB_INT[0] and ACSR1 generates an interrupt on MLB_INT[1] or ACSR0 and ACSR1 generate an interrupt on MLB_INT[0] only.	
		0	ACSR0 generates an interrupt on MLB_INT[0]; ACSR1 generates an interrupt on MLB_INT[1]
		1	ACSR0 and ACSR1 generate an interrupts on MLB_INT[0] only
0 (R/W)	SCE	Software Clear Enable. The `MLB_ACTL.SCE` bit selects whether hardware clears the interrupt after a ACSRn register read or software clears the interrupt.	
		0	Hardware clears interrupt after a ACSRn register read
		1	Software clears interrupt

MediaLB Control 0 Register

The MLB_CTL0 register contains bit that enable the module and provide module status. Note that the maximum speed for ML3-pin mode is 1024 x FS

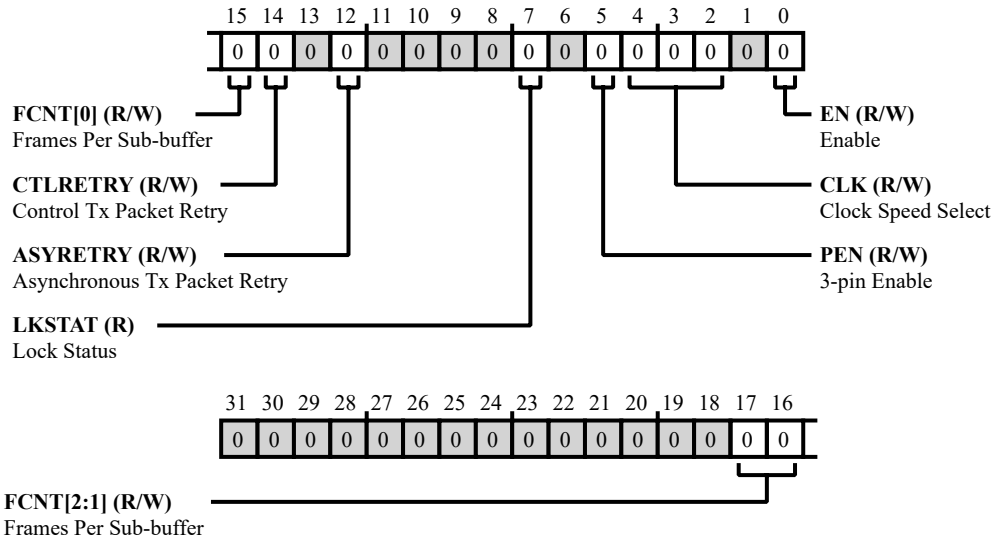


Figure 26-12: MLB_CTL0 Register Diagram

Table 26-25: MLB_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	FCNT	Frames Per Sub-buffer. The MLB_CTL0.FCNT bit field configures the frames per sub-buffer on synchronous channels.	
		0	1 Frame per sub-buffer (Operation is the same as Standard mode)
		1	2 frames per sub-buffer
		2	4 frames per sub-buffer
		3	8 frames per sub-buffer
		4	16 frames per sub-buffer
		5	32 frames per sub-buffer
		6	64 frames per sub-buffer
14 (R/W)	CTLRETRY	Control Tx Packet Retry. When the MLB_CTL0.CTLRETRY bit is set, a control packet that is flagged with a Break or Protocol error by the receiver is retransmitted. When cleared, a control packet that is flagged with a Break or Protocol error by the receiver is skipped.	

Table 26-25: MLB_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	ASYRETRY	Asynchronous Tx Packet Retry. When the MLB_CTL0.ASYRETRY bit is set, an asynchronous packet that is flagged with a Break or Protocol error by the receiver is retransmitted. When cleared, an asynchronous packet that is flagged with a Break or ProtocolError by the receiver is skipped.	
7 (R/NW)	LKSTAT	Lock Status. When the MLB_CTL0.LKSTAT bit is set (=0), the MediaLB block is synchronized to the incoming MediaLB frame with the following conditions. • If MLB_CTL1.LOCK =0 (unlocked), MLB_CTL1.LOCK is set after a FRAMESYNC is detected at the same position for three consecutive frames. • If MLB_CTL1.LOCK =1 (locked), MLB_CTL1.LOCK is cleared after not receiving a FRAMESYNC at the expected time for two consecutive frames. In this case FRAMESYNC patterns occurring at locations other than the expected one are ignored.	
5 (R/W)	PEN	3-pin Enable. The MLB_CTL0.PEN bit configures the 3-pin mode.	
		0	3-pin interface enabled
		1	invalid condition
4:2 (R/W)	CLK	Clock Speed Select. The MLB_CTL0.CLK bit field sets the clock speed.	
		0	256 x Fs (MLB_CTL.PEN = 0)
		1	512 x Fs (MLB_CTL.PEN = 0)
		2	1024 x Fs (MLB_CTL.PEN = 0)
0 (R/W)	EN	Enable. When the MLB_CTL0.EN bit is set (=1), MediaLB clock, signal, and data are received and transmitted on the appropriate MediaLB pins.	

Control 1 Register

The MLB_CTL1 register contains bits that provide lock status and control system commands.

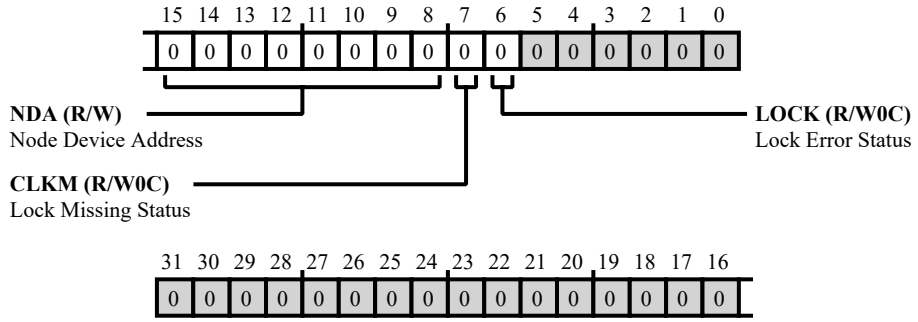


Figure 26-13: MLB_CTL1 Register Diagram

Table 26-26: MLB_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	NDA	Node Device Address. The MLB_CTL1.NDA bit field is used for system commands directed to individual MediaLB nodes.
7 (R/W0C)	CLKM	Lock Missing Status. The MLB_CTL1.CLKM bit is set when the MediaLB clock is not toggling at the pin. This bit is cleared by software.
6 (R/W0C)	LOCK	Lock Error Status. The MLB_CTL1.LOCK bit is set when the MediaLB is unlocked. This bit is cleared by software.

MLB Global Control Register

The MLB_GCTL register contains bits that manage the MLB clock.

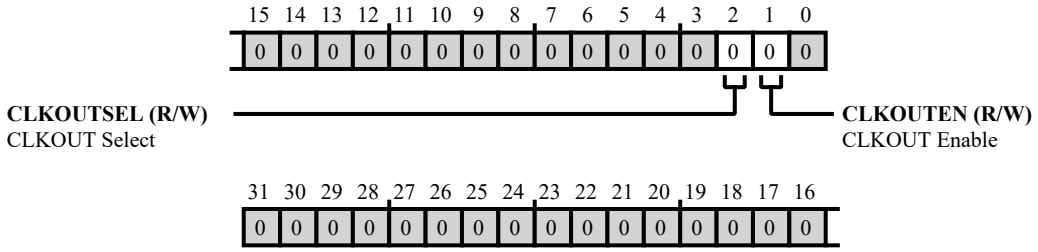


Figure 26-14: MLB_GCTL Register Diagram

Table 26-27: MLB_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	CLKOUTSEL	CLKOUT Select. The MLB_GCTL.CLKOUTSEL bit selects MLB 3 pin clock	
		0	MLB 3 pin clock is selected for MLBCLKOUT
1 (R/W)	CLKOUTEN	CLKOUT Enable.	

HBI Channel Busy 0 Register

The HC can determine which channel(s) are busy by reading the MLB_HCBR0 register. An HBI channel is busy if:

- it is currently loaded into one of the two AGUs
- the channel is enabled, CE = 1 from the Channel Allocation Table
- the DMA is active

When an HBI channel is busy, hardware may write back its local copy of the channel descriptor at any time. System software should not write a CDT descriptor for a channel that is busy. Only two HBI channels can be busy at any given time. Each bit of this register is read-only.

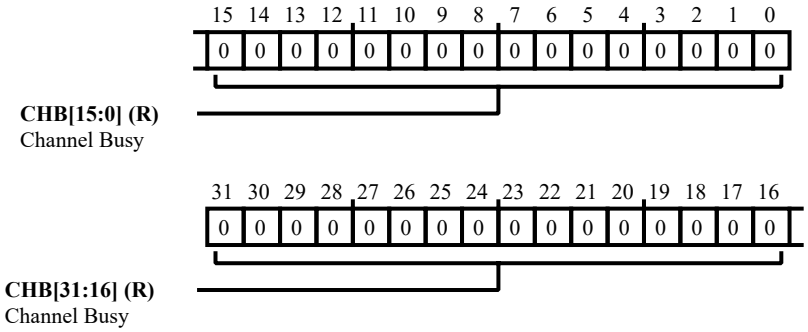


Figure 26-15: MLB_HCBR0 Register Diagram

Table 26-28: MLB_HCBR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CHB	Channel Busy. The MLB_HCBR0.CHB bit field contains the bitwise channel busy bit for channels 31:0. When a bit is cleared (=0) the channel is idle. When a bit is set (=1) the channel is busy.

HBI Channel Busy 1 Register

The HC can determine which channel(s) are busy by reading the MLB_HCBR1 register. An HBI channel is busy if:

- it is currently loaded into one of the two AGUs
- the channel is enabled, CE = 1 from the Channel Allocation Table
- the DMA is active

When an HBI channel is busy, hardware may write back its local copy of the channel descriptor at any time. System software should not write a CDT descriptor for a channel that is busy. Only two HBI channels can be busy at any given time. Each bit of this register is read-only.

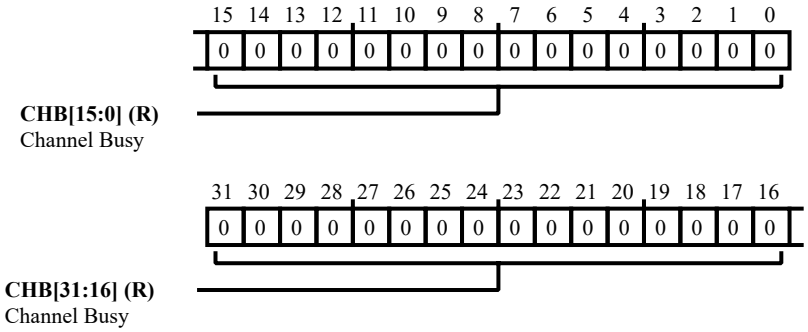


Figure 26-16: MLB_HCBR1 Register Diagram

Table 26-29: MLB_HCBR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CHB	Channel Busy. The MLB_HCBR1.CHB bit field contains the bitwise channel busy bit for channels 63:32. When a bit is cleared (=0) the channel is idle. When a bit is set (=1) the channel is busy.

HBI Channel Error 0 Register

The MLB_HCER0 register indicates which channels (channels 31:0) have encountered fatal errors.

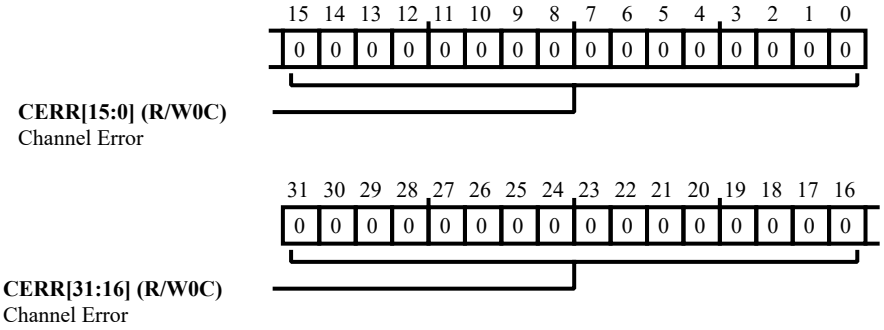


Figure 26-17: MLB_HCER0 Register Diagram

Table 26-30: MLB_HCER0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	CERR	Channel Error. The MLB_HCER0.CERR bit field reports bitwise errors for channels 31:0.

HBI Channel Error 1 Register

The MLB_HCER1 register indicates which channel(s) have encountered fatal errors.

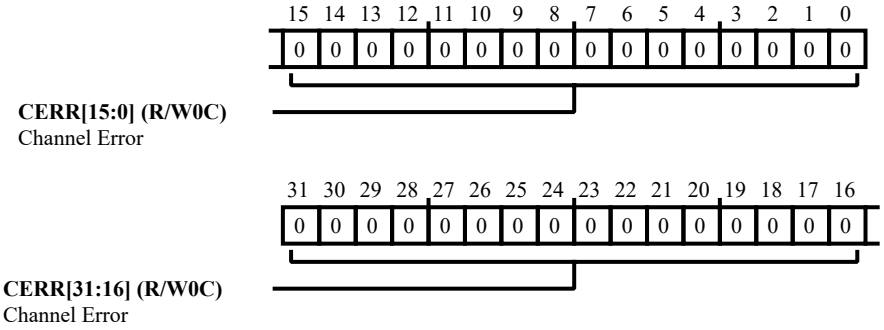


Figure 26-18: MLB_HCER1 Register Diagram

Table 26-31: MLB_HCER1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	CERR	Channel Error. The MLB_HCER1.CERR bite field reports bitwise errors for channels 63:32.

HBI Channel Mask 0 Register

The MLB_HCMR0 register controls which channels (for channels 31:0) are able to generate an HBI interrupt.

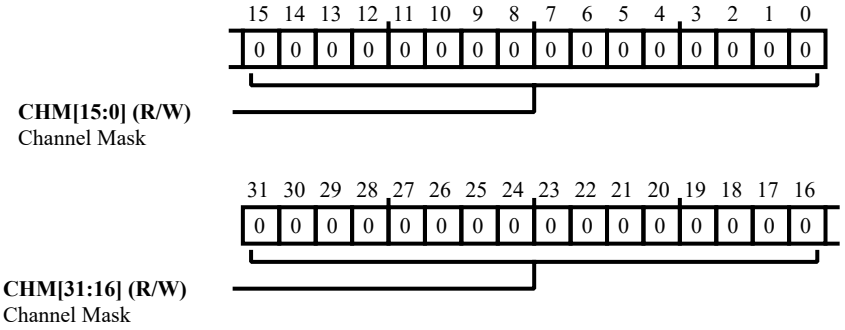


Figure 26-19: MLB_HCMR0 Register Diagram

Table 26-32: MLB_HCMR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CHM	Channel Mask. The MLB_HCMR0.CHM bit field contains the bitwise channel mask bit for channels 31:0. When a bit is cleared (=0) the channel is masked. When set (=1) the channel is unmasked.

HBI Channel Mask 1 Register

The MLB_HCMR1 register controls which channels (for channels 63:32) are able to generate an HBI interrupt.

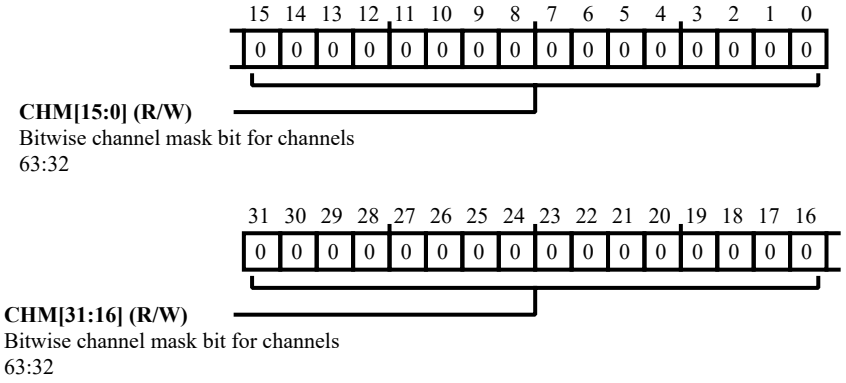


Figure 26-20: MLB_HCMR1 Register Diagram

Table 26-33: MLB_HCMR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CHM	Bitwise channel mask bit for channels 63:32. The MLB_HCMR1.CHM bit field contains the bitwise channel mask bit for channels 63:32. When a bit is cleared (=0) the channel is masked. When set (=1) the channel is unmasked.

HBI Control Register

The MLB_HCTL register controls and monitors general operation of the HBI block through the Address Generation Units) by reading and writing the register through the I/O interface. Each bit of this register is read/write.

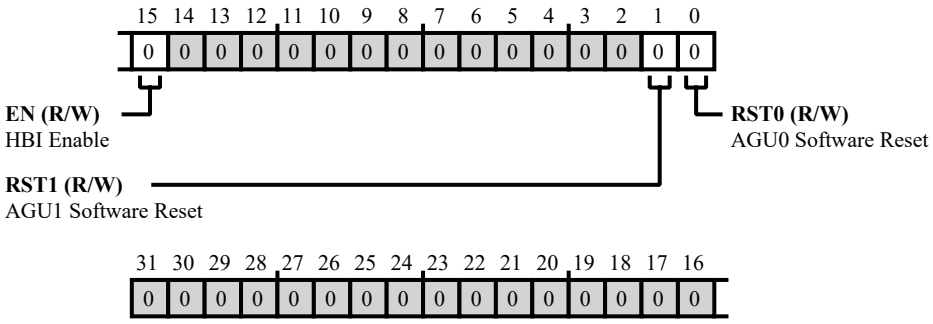


Figure 26-21: MLB_HCTL Register Diagram

Table 26-34: MLB_HCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	EN	HBI Enable. Setting the MLB_HCTL.EN bit enables the HBI.
1 (R/W)	RST1	AGU1 Software Reset. Setting the MLB_HCTL.RST1 bit resets AGU1.
0 (R/W)	RST0	AGU0 Software Reset. Setting the MLB_HCTL.RST0 bit resets AGU0.

Memory Interface Address Register

The MLB_MADR register contains bit fields that contain the Channel Table RAM (CTR) or Data Buffer RAM (DBR) addresses. It also contains bits that set target location and read-not-write parameters.

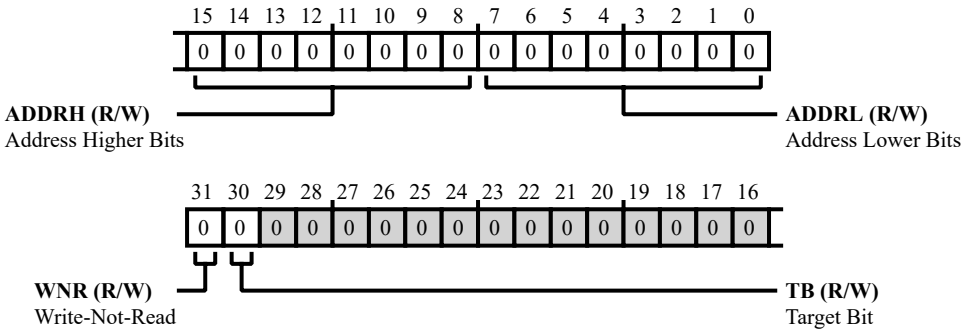


Figure 26-22: MLB_MADR Register Diagram

Table 26-35: MLB_MADR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	WNR	Write-Not-Read. The MLB_MADR.WNR bit selects Write-Not-Read.	
		0	Read
		1	Write
30 (R/W)	TB	Target Bit. The MLB_MADR.TB bit sets the target location.	
		0	Channel Table RAM (CTR)
		1	Data Buffer RAM (DBR)
15:8 (R/W)	ADDRH	Address Higher Bits. The MLB_MADR.ADDRH bit field contains the DBR address of the 8-bit entry (bits [13:8]).	
7:0 (R/W)	ADDRL	Address Lower Bits. The MLB_MADR.ADDRL bit field contains the CTR address of the 128-bit entry or the DBR address of the 8-bit entry (bits [7:0]).	

Memory Interface Control Register

The MLB_MCTL register contains a bit that indicates that the data transfer is complete.

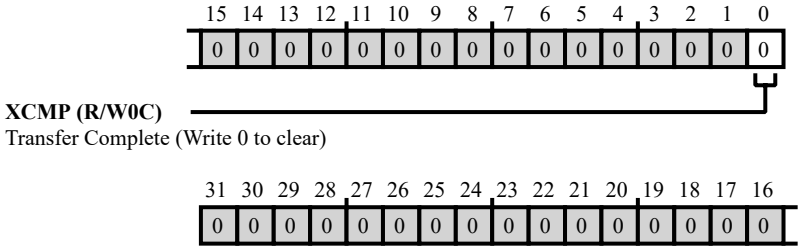


Figure 26-23: MLB_MCTL Register Diagram

Table 26-36: MLB_MCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W0C)	XCMP	Transfer Complete (Write 0 to clear). The MLB_MCTL.XCMP bit indicates that the data transfer is complete.

Memory Interface Control Data 0 Register

The MLB_MDAT0 register contains Channel Table RAM (CTR) data (bits [31:0] of 128-bit entry) or Data Buffer RAM (DBR) data (bits [7:0] of 8-bit entry).

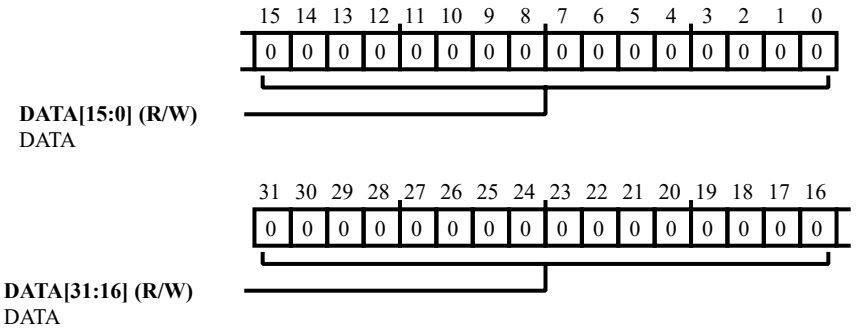


Figure 26-24: MLB_MDAT0 Register Diagram

Table 26-37: MLB_MDAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT0.DATA bit field contains CTR data or DBR data.

Memory Interface Control Data 1 Register

The MLB_MDAT1 register contains Channel Table RAM (CTR) data (bits [63:32] of 128-bit entry).

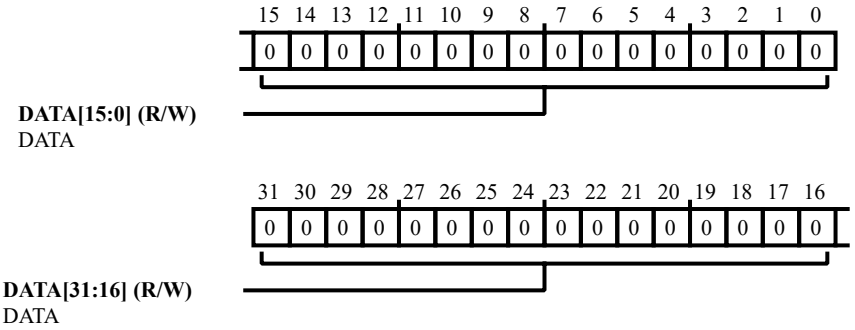


Figure 26-25: MLB_MDAT1 Register Diagram

Table 26-38: MLB_MDAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT1.DATA bit field contains CTR data.

Memory Interface Control Data 2 Register

The MLB_MDAT2 register contains Channel Table RAM (CTR) data (bits [95:64] of 128-bit entry).

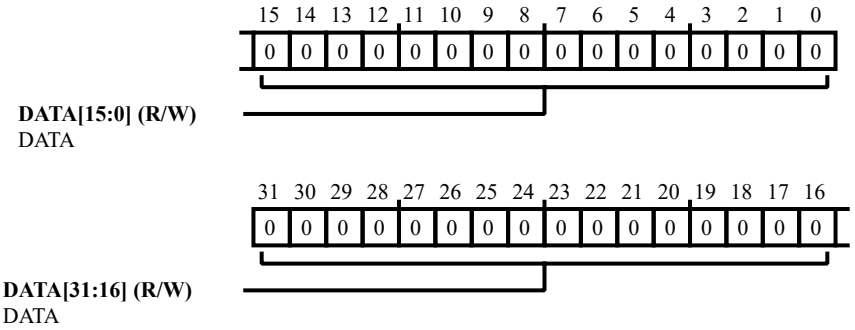


Figure 26-26: MLB_MDAT2 Register Diagram

Table 26-39: MLB_MDAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT2.DATA bit field contains CTR data.

Memory Interface Control Data 3 Register

The MLB_MDAT3 register contains Channel Table RAM (CTR) data (bits [127:96] of 128-bit entry).

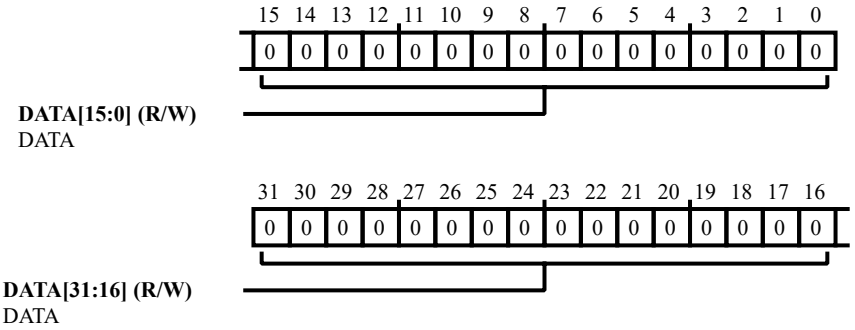


Figure 26-27: MLB_MDAT3 Register Diagram

Table 26-40: MLB_MDAT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT3.DATA bit field contains CTR data.

Memory Interface Control Data Write Enable 0 Register

The MLB_MDWE0 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [31:0]. When cleared (=0) the bit is disabled, when set (=1) the bit is enabled.

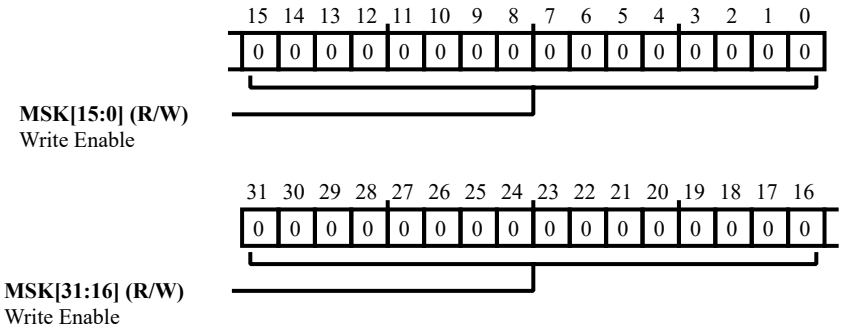


Figure 26-28: MLB_MDWE0 Register Diagram

Table 26-41: MLB_MDWE0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Write Enable. The MLB_MDWE0.MSK bit field contains the bitwise write enable for CTR data.

Memory Interface Control Data Write Enable 1 Register

The MLB_MDWE1 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [63:32]. When cleared (=0), the bit is disabled. When set (=1), the bit is enabled.

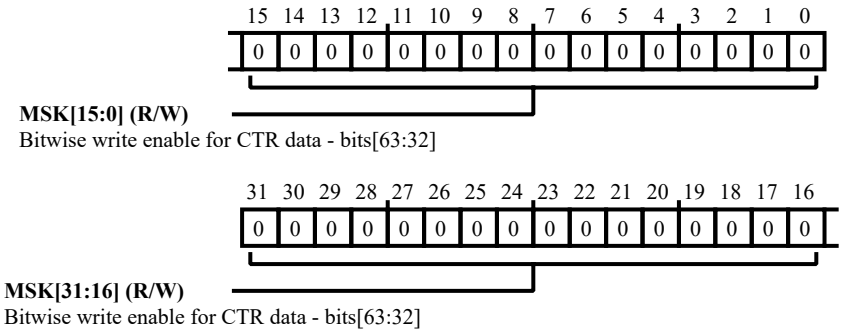


Figure 26-29: MLB_MDWE1 Register Diagram

Table 26-42: MLB_MDWE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Bitwise write enable for CTR data - bits[63:32]. The MLB_MDWE1.MSK bit field contains the bitwise write enable for CTR data.

Memory Interface Control Data Write Enable 2 Register

The MLB_MDWE2 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [95:64]. When cleared (=0), the bit is disabled. When set (=1), the bit is enabled.

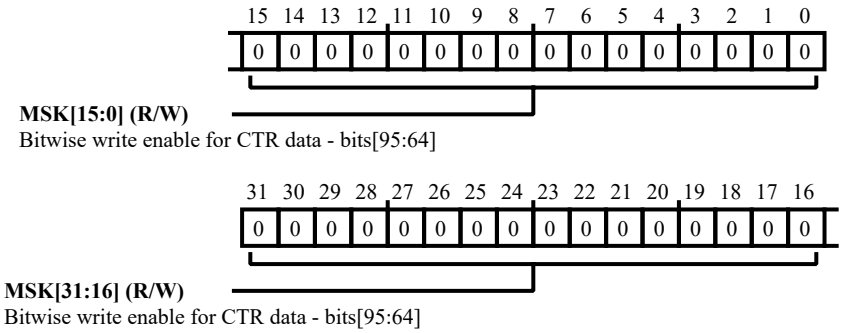


Figure 26-30: MLB_MDWE2 Register Diagram

Table 26-43: MLB_MDWE2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Bitwise write enable for CTR data - bits[95:64]. The MLB_MDWE2.MSK bit field contains the bitwise write enable for CTR data.

Memory Interface Control Data Write Enable 3 Register

The MLB_MDWE3 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [127:96]. When cleared (=0), the bit is disabled. When set (=1), the bit is enabled.

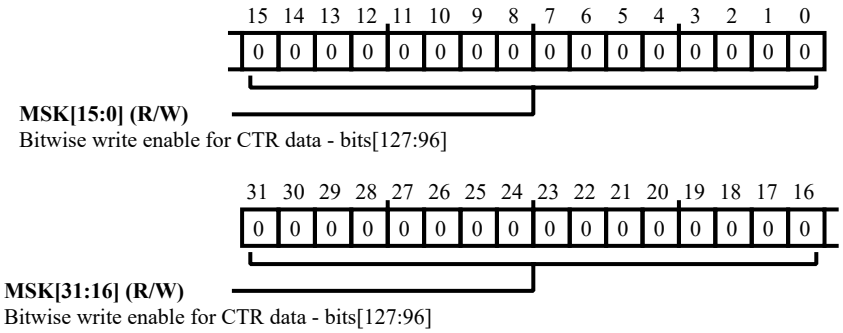


Figure 26-31: MLB_MDWE3 Register Diagram

Table 26-44: MLB_MDWE3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Bitwise write enable for CTR data - bits[127:96]. The MLB_MDWE3.MSK bit field contains the bitwise write enable for CTR data.

Interrupt Enable Register

The MLB_MIEN register is used to enable various interrupt conditions.

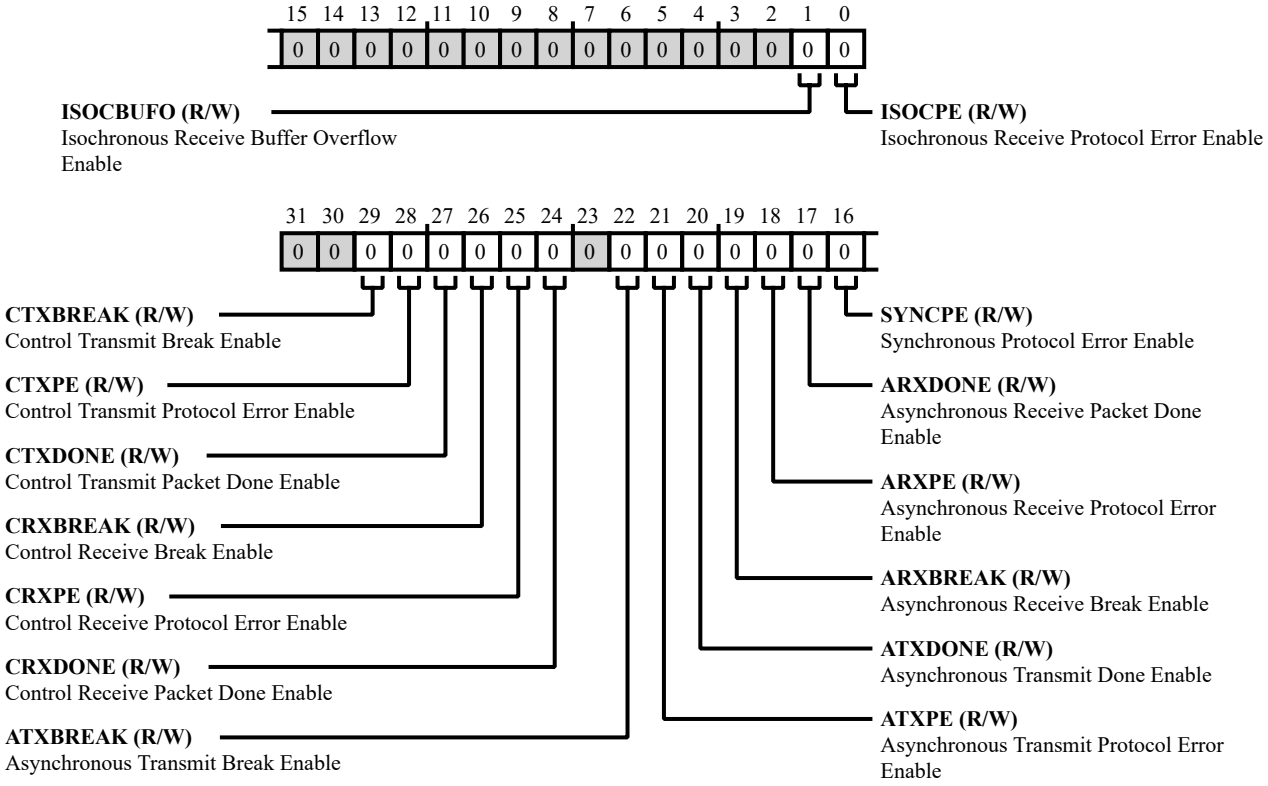


Figure 26-32: MLB_MIEN Register Diagram

Table 26-45: MLB_MIEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	CTXBREAK	Control Transmit Break Enable. When the MLB_MIEN.CTXBREAK bit is set, a ReceiverBreak response received from the receiver on a control Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
28 (R/W)	CTXPE	Control Transmit Protocol Error Enable. When the MLB_MIEN.CTXPE bit is set, a ProtocolError generated by the receiver on a control Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
27 (R/W)	CTXDONE	Control Transmit Packet Done Enable. When the MLB_MIEN.CTXDONE bit is set, a packet transmitted with no errors on a control Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.

Table 26-45: MLB_MIEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	CRXBREAK	Control Receive Break Enable. When the MLB_MIEN.CRXBREAK bit is set, a ControlBreak command received from the transmitter on a control Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
25 (R/W)	CRXPE	Control Receive Protocol Error Enable. When the MLB_MIEN.CRXPE bit is set, a ProtocolError detected on a control Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
24 (R/W)	CRXDONE	Control Receive Packet Done Enable. When the MLB_MIEN.CRXDONE bit is set, a packet received with no errors on a control Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
22 (R/W)	ATXBREAK	Asynchronous Transmit Break Enable. When the MLB_MIEN.ATXBREAK bit is set, a ReceiverBreak response received from the receiver on an asynchronous Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
21 (R/W)	ATXPE	Asynchronous Transmit Protocol Error Enable. When the MLB_MIEN.ATXPE bit is set, a ProtocolError generated by the receiver on an asynchronous Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
20 (R/W)	ATXDONE	Asynchronous Transmit Done Enable. When the MLB_MIEN.ATXDONE bit is set, a packet transmitted with no errors on an asynchronous Tx channel channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
19 (R/W)	ARXBREAK	Asynchronous Receive Break Enable. When the MLB_MIEN.ARXBREAK bit is set, a AsyncBreak command received from the transmitter on an asynchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
18 (R/W)	ARXPE	Asynchronous Receive Protocol Error Enable. When the MLB_MIEN.ARXPE bit is set, a protocol error detected on an asynchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
17 (R/W)	ARXDONE	Asynchronous Receive Packet Done Enable. When the MLB_MIEN.ARXDONE bit is set, a packet received with no errors on an asynchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.

Table 26-45: MLB_MIEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	SYNCPE	Synchronous Protocol Error Enable. When the MLB_MIEN.SYNCPE bit is set, a protocol error detected on a synchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set. This occurs only when isochronous flow control is disabled.
1 (R/W)	ISOCBUFO	Isochronous Receive Buffer Overflow Enable. When the MLB_MIEN.ISOCBUFO bit is set, a buffer overflow on an isochronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set. This occurs only when isochronous flow control is disabled.
0 (R/W)	ISOCPE	Isochronous Receive Protocol Error Enable. When the MLB_MIEN.ISOCPE bit is set, a protocol error detected on an isochronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.

Channel Status 0 Register

The MLB_MS0 register indicates the channel status for MediaLB channels 31 to 0. Channel status bits are set by hardware and cleared by software. Status is only set if the appropriate bits in the MLB_MIEN register are set.

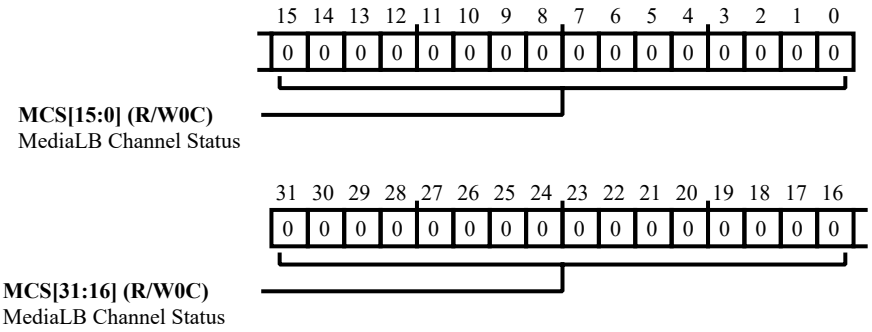


Figure 26-33: MLB_MS0 Register Diagram

Table 26-46: MLB_MS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	MCS	MediaLB Channel Status. The MLB_MS0.MCS bit field indicates the MediaLB channel status for channels 31 to 0.

Channel Status 1 Register

The MLB_MS1 register indicates the channel status for MediaLB channels 32 to 63. Channel status bits are set by hardware and cleared by software. Status is only set if the appropriate bits in the MLB_MIEN register are set.

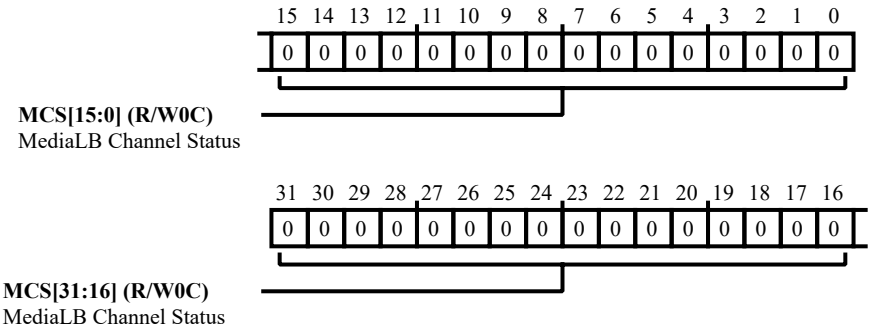


Figure 26-34: MLB_MS1 Register Diagram

Table 26-47: MLB_MS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	MCS	MediaLB Channel Status. The MLB_MS1.MCS bit field indicates the MediaLB channel status for channels 63 to 32.

System Data Register

The MLB_MSD register allows system software to receive control information from the MLB controller. The MLB_MSD register is updated once per frame by the hardware during the MLB system channel.

The MLB_MSD register is loaded with the data from the MLBDAT_IN signal during the system channel quadlet. System software must read this register before the start of the next MLB frame to prevent the current data from being lost.

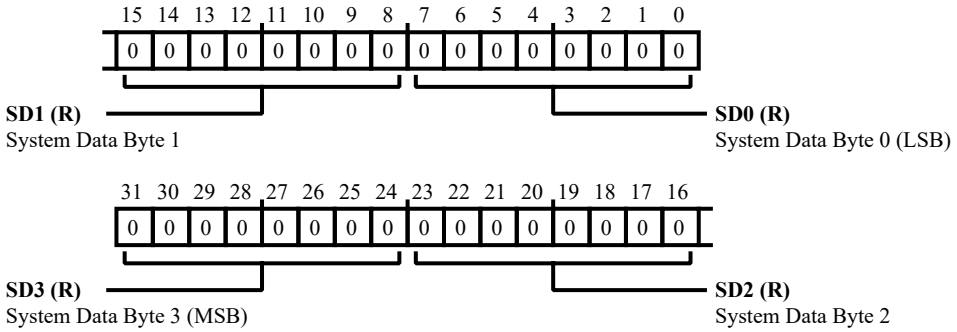


Figure 26-35: MLB_MSD Register Diagram

Table 26-48: MLB_MSD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	SD3	System Data Byte 3 (MSB). The MLB_MSD.SD3 bits are updated with MediaLB Data [31:24] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD3 is not updated.
23:16 (R/NW)	SD2	System Data Byte 2. The MLB_MSD.SD2 bits are updated with MediaLB Data [23:16] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD2 is not updated.
15:8 (R/NW)	SD1	System Data Byte 1. The MLB_MSD.SD1 bits are updated with MediaLB Data [15:8] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD1 is not updated.
7:0 (R/NW)	SD0	System Data Byte 0 (LSB). The MLB_MSD.SD0 bits are updated with MediaLB Data [7:0] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD0 is not updated.

System Status Register

The MLB_MSS register allows system software to monitor and control the status of the MLB network. The register is updated once per frame by hardware during the MLB system channel. The bits of this register are not valid until the processor is locked to the MLB interface (except for the bits associated with MLB lock and unlock, SDMU and SDML). System software must service events before the start of the next MLB frame to prevent the current frame status from being lost.

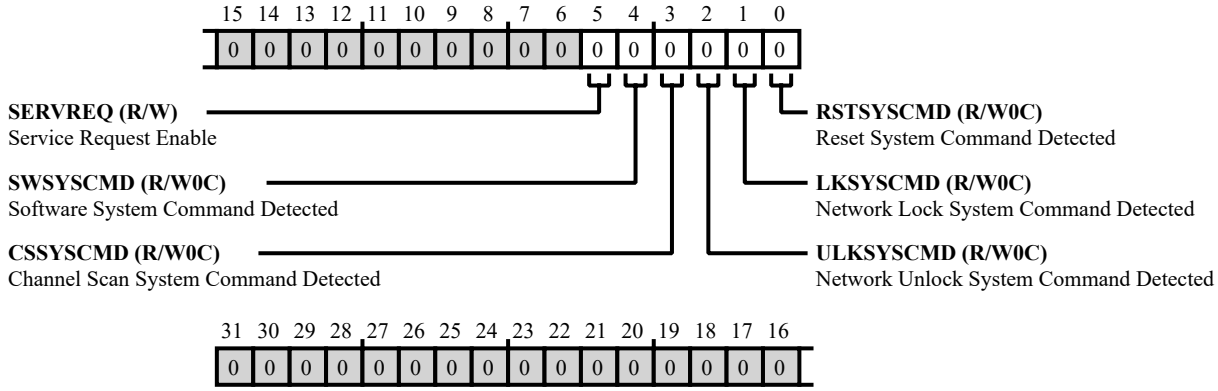


Figure 26-36: MLB_MSS Register Diagram

Table 26-49: MLB_MSS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	SERVREQ	Service Request Enable. When the MLB_MSS.SERVREQ bit set, the MediaLB block responds with a device present, request service system response if a matching channel scan system command is detected. When cleared, the MediaLB block responds with a device present system response.
4 (R/W0C)	SWSYSCMD	Software System Command Detected. The MLB_MSS.SWSYSCMD bit indicates that a software system command was detected (in the system quadlet). The MLB_MSS.SWSYSCMD bit is set by hardware, cleared by software. Data is stored in the MLB_MSD register for this command.
3 (R/W0C)	CSSYSCMD	Channel Scan System Command Detected. The MLB_MSS.CSSYSCMD bit indicates that a channel scan system command was detected (in the system quadlet). The MLB_MSS.CSSYSCMD bit is set by hardware, cleared by software. If the node address specified in the data quadlet matches the value in the MLB_CTL1.NDA bit field, the device responds either device present or device present, request service system response in the next system quadlet.
2 (R/W0C)	ULKSYSCMD	Network Unlock System Command Detected. The MLB_MSS.ULKSYSCMD bit indicates a network unlock system command was detected (in the system quadlet). The MLB_MSS.ULKSYSCMD bit is by hardware, cleared by software.

Table 26-49: MLB_MSS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W0C)	LKSYSCMD	Network Lock System Command Detected. The `MLB_MSS.LKSYSCMD` bit indicates a network lock system command was detected (in the system quadlet). The `MLB_MSS.LKSYSCMD` bit is set by hardware and cleared by software.
0 (R/W0C)	RSTSYSCMD	Reset System Command Detected. The `MLB_MSS.RSTSYSCMD` bit indicates a reset system command was detected (in the system quadlet). The `MLB_MSS.RSTSYSCMD` bit is set by hardware, cleared by software.

27 Two-Wire Interface (TWI)

The processor has a two-wire interface (TWI), that provides a simple exchange method of control data between multiple devices. The TWI module is compatible with the widely used I^2C bus standard. Additionally, the TWI module is fully compatible with serial camera control bus (SCCB) functionality for easier control of various CMOS camera sensor devices.

The TWI module offers the capabilities of simultaneous controller and target operation and support for both 7-bit addressing and multimedia data arbitration. The TWI interface uses two pins for transferring clock (`TWI_SCL`) and data (`TWI_SDA`) and supports the protocol at speeds up to 400K bits/s. The TWI interface pins are compatible with 3.3V logic levels.

To preserve processor bandwidth, the TWI module can be set up with transfer-initiated interrupts to only service FIFO buffer data reads and writes. Protocol-related interrupts are optional. The TWI externally moves 8-bit data while maintaining compliance with the I^2C bus protocol.

NOTE: All TWI instances operate in the SCLK0 domain.

TWI Features

The TWI is fully compatible with the widely used I^2C bus standard.

The TWI controller includes the following features.

- Simultaneous controller and target operation on multiple device systems
- Support for multi-controller bus arbitration
- 7-bit addressing
- 100K bits/second and 400K bits/second data rates
- General call address support
- Controller clock synchronization and support for clock low extension
- Separate multiple-byte receive and transmit FIFOs
- Low interrupt rate

- Individual override control of data and clock lines in the event of bus lock-up
- Input filter for spike suppression
- Serial camera control bus support as specified in the *OmniVision Serial Camera Control Bus (SCCB) Functional Specification*
- Programmable drive/tolerance of TWI pins. Refer to the PADS_PCFG0 register description in General-Purpose Ports (PORT) chapter for details.

TWI Functional Description

The TWI interface is a shift register that serially transmits and receives data bits. It moves data 1 bit at a time at the SCL rate, to and from other TWI devices. The SCL signal synchronizes the shifting and sampling of the data on the serial data pin.

ADSP-2184x TWI Register List

The Two-Wire Interface controller TWI allows a device to interface to an inter-IC bus as specified by the Philips I^2C Bus Specification version 2.1, dated January 2000. A set of registers governs TWI operations. For more information on TWI functionality, see the TWI register descriptions.

Table 27-1: ADSP-2184x TWI Register List

Name	Description
TWI_CLKDIV	SCL Clock Divider Register
TWI_CTL	Control Register
TWI_FIFOCTL	FIFO Control Register
TWI_FIFOSTAT	FIFO Status Register
TWI_IMSK	Interrupt Mask Register
TWI_ISTAT	Interrupt Status Register
TWI_CTLRADDR	Controller Mode Address Register
TWI_CTLRCTL	Controller Mode Control Registers
TWI_CTLRSTAT	Controller Mode Status Register
TWI_RXDATA16	Rx Data Double-Byte Register
TWI_RXDATA8	Rx Data Single-Byte Register
TWI_TGTADDR	Target Mode Address Register
TWI_TGTCTL	Target Mode Control Register
TWI_TGTSTAT	Target Mode Status Register
TWI_TXDATA16	Tx Data Double-Byte Register
TWI_TXDATA8	Tx Data Single-Byte Register

ADSP-2184x TWI Interrupt List

Table 27-2: ADSP-2184x TWI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
361	TWI0_DATA	TWI0 Data Interrupt	Level	
362	TWI1_DATA	TWI1 Data Interrupt	Level	
363	TWI2_DATA	TWI2 Data Interrupt	Level	
364	TWI3_DATA	TWI3 Data Interrupt	Level	
365	TWI4_DATA	TWI4 Data Interrupt	Level	
366	TWI5_DATA	TWI5 Data Interrupt	Level	

TWI Block Diagram

The *TWI Block Diagram* figure shows the basic blocks of the TWI interface.

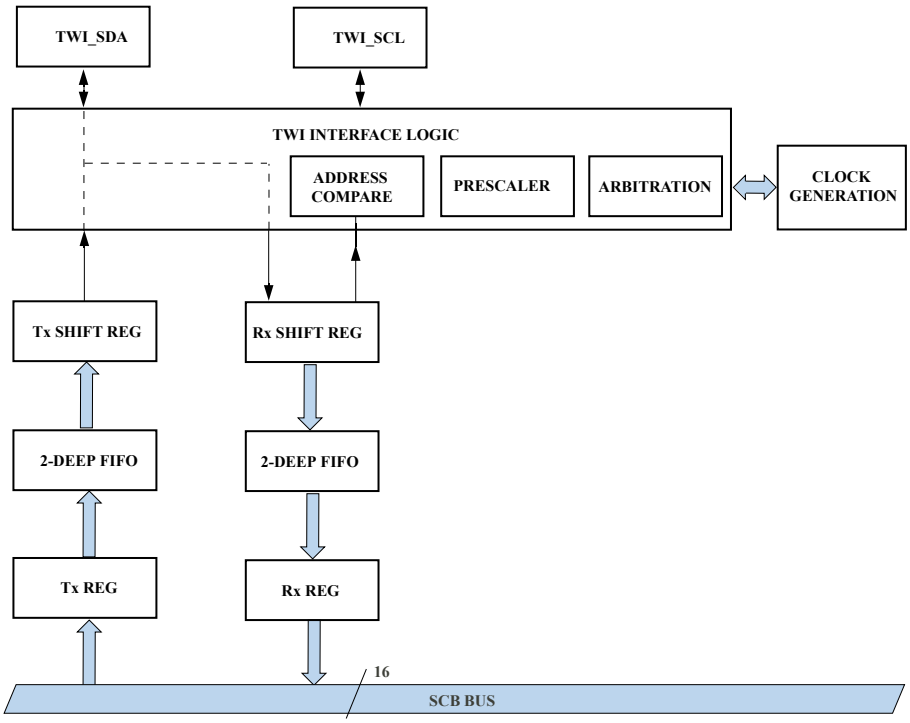


Figure 27-1: TWI Block Diagram

External Interface

The `TWI_SDA` (serial data) and `TWI_SCL` (serial clock) signals are open drain and require pull-up resistors. These bidirectional signals externally interface the TWI controller to the I^2C bus, and no other external connections or logic are necessary.

Serial Clock Signal (SCL)

The serial clock signal (TWI_SCL) is an input in controller mode. In controller mode, the TWI controller must set this signal to the desired frequency.

The TWI controller supports the standard mode of operation (up to 100 kHz) or fast mode (up to 400 kHz). The TWI control register (TWI_CTL) sets the TWI_CTL.PRESCALE value which sets the relationship between the system clock (SCLK) and the internally timed events of the TWI controller. The internal time reference is derived from SCLK using a prescaled value. The prescale value is the number of SCLK periods used in the generation of one internal time reference. Set the value of prescale to create an internal time reference with a period of 10 MHz. It is represented as a 7-bit binary value as follows.

PRESCALE = f_{SCLK}/10MHz

NOTE: It is not always possible to achieve 10-MHz accuracy. In such cases, it is safe to round up the PRESCALE value to the next highest integer. For example, if SCLK is 100 MHz, the PRESCALE value is calculated as 100 MHz/10 MHz = 10. A prescale value of 14 in this case ensures that all timing requirements are met.

During controller mode operation, the TWI module uses the TWI_CLKDIV register values to create the minimum TWI_CLKDIV.CLKHI and TWI_CLKDIV.CLKLO durations of the TWI_SCL signal. The TWI_CLKDIV.CLKHI field specifies the minimum number of 10-MHz time reference periods the TWI_SCL waits before a new clock low period begins, assuming a single controller. (The 10-MHz time reference periods are represented as an 8-bit binary value). The TWI uses the TWI_CLKDIV.CLKLO field to specify the minimum number of internal time reference periods (represented as an 8-bit binary value); the TWI_SCL signal is held low.

Serial clock frequencies can vary from 400 kHz to less than 20 kHz. The resolution of the clock generated is 1/10 MHz or 100 ns. The following equation describes the frequency.

CLKDIV = TWI_SCL period/10 MHz time reference.

For example, for an TWI_SCL of 400 kHz (period = 1/400 kHz = 2500 ns) and an internal time reference of 10 MHz (period = 100 ns), the following equation applies:

CLKDIV =2500 ns/100 ns = 25

Therefore, a TWI_SCL signal with a 30% duty cycle has TWI_CLKDIV.CLKLO=17 and TWI_CLKDIV.CLKHI=8. Adding TWI_CLKDIV.CLKLO and TWI_CLKDIV.CLKHI equals CLKDIV.

NOTE: The TWI_CLKDIV.CLKHI and TWI_CLKDIV.CLKLO fields are not intended to guarantee a certain frequency. Rather, they guarantee a certain minimum high and low duration for the TWI_SCL signal. Slew rate controls falling edges. The *RC* time constant governs the rising edges. The pull-up resistor and the TWI_SCL capacitance form the time constant. See the “Register Descriptions” section for more details.

Serial Data Signal (SDA)

The TWI transmits and receives serial data, depending on the direction of the transfer, on the bidirectional serial data signal (SDA).

Internal Interface

The peripheral bus interface supports the transfer of 16-bit wide data. The processor uses the interface in the support of register and FIFO buffer reads and writes. The TWI internal interface is comprised of the blocks described as follows.

Register block. Contains all control and status bits and reflects what can be written or read as outlined by the programming model. Each function block updates their corresponding status bits.

FIFO buffer. Configured as a 1-byte-wide, 2-deep transmit FIFO buffer and a 1-byte-wide, 2-deep receive FIFO buffer.

Transmit shift register. Serially shifts its data out externally off chip. The output can be controlled for generation of acknowledgments, or it can be manually overwritten.

Receive shift register. Receives its data serially from off chip. The receive shift register is 1 byte wide and data received can either be transferred to the FIFO buffer or used in an address comparison.

Address compare block. Supports address comparison in the event the TWI controller module is accessed as a target.

Prescaler block. Must be programmed to generate a 10-MHz time reference relative to the system clock. The block uses this time base for filtering of data and timing events specified by the electrical data sheet (See the Philips specification). The block uses the time base to generate the `TWI_SCL` clock as well.

Clock generation module. Generates an external `TWI_SCL` clock when in controller mode. It includes the logic necessary for synchronization in a multi-controller clock configuration and clock stretching when configured in target mode.

NOTE: The TWI does not support DMA based operation.

TWI Architectural Concepts

The TWI controller follows the transfer protocol of the Philips I^2C Bus specification version 2.1 dated January 2000.

NOTE: The TWI unit does not support DMA-based operation.

TWI Protocol

The ***Data Transfer*** figure shows a simple complete transfer.

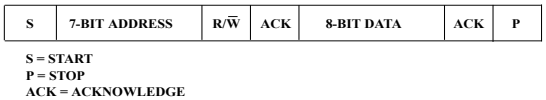


Figure 27-2: Data Transfer

The TWI controller register contents maps to a basic transfer. The ***Data Transfer with Bit Illustration*** figure details the same transfer from the ***Data Transfer*** figure noting the corresponding TWI controller bit names. In this

illustration, the TWI controller successfully transmits 1 byte of data. The completer has acknowledged both address and data.

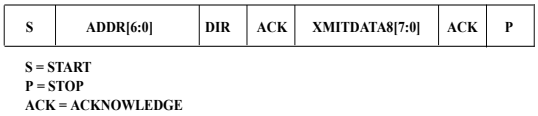


Figure 27-3: Data Transfer with Bit Illustration

Clock Generation and Synchronization

The TWI controller implementation only issues a clock during controller mode operation and only at the time a transfer initiates. If arbitration for the bus is lost, the serial clock output immediately three-states. If multiple clocks attempt to drive the serial clock line, the TWI controller synchronizes its clock with the other remaining clocks. The ***Clock Synchronization*** figure shows this functionality.

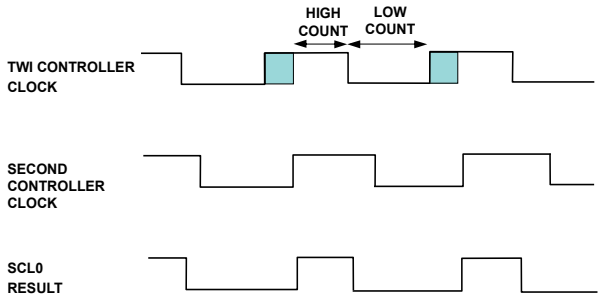


Figure 27-4: Clock Synchronization

The TWI controller serial clock (`TWI_SCL`) output follows these rules:

- Once the clock high (`TWI_CLKDIV.CLKHI`) count is complete, the serial clock output is driven low and the clock low (`TWI_CLKDIV.CLKLO`) count begins.
- Once the clock low count is complete, the serial clock line is three-stated. This state allows the external pull-up resistor to pull the `TWI_SCL` signal high. The clock synchronization logic enters a delay mode (shaded area) until the `TWI_SCL` signal is detected at logic 1 level. Now, the clock high count begins.

Bus Arbitration

The TWI controller initiates a controller mode transmission only when the bus is idle. When the bus is idle and two controllers initiate a transfer, arbitration for the bus begins. The ***Bus Arbitration*** figure shows the arbitration.

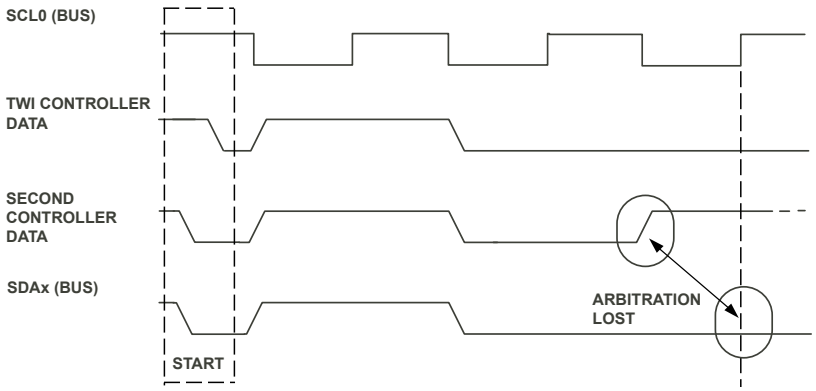


Figure 27-5: Bus Arbitration

The TWI controller monitors the serial data bus (SDA) while the `TWI_SCL` signal is high. If the `TWI_SDA` signal is determined to be an active logic 0 level while the data of the TWI controller is a logic 1 level, the TWI controller has lost arbitration. It stops generating the clock and data signals. Arbitration is not only performed at the serial clock edges, but also during the entire time the `TWI_SCL` signal is high.

Start and Stop Conditions

Start and stop conditions involve serial data transitions while the serial clock is a logic 1 level. The TWI controller generates and recognizes these transitions. Typically, start and stop conditions occur at the beginning and at the conclusion of a transmission, except repeated start combined transfers. The ***Start and Stop Conditions*** figure shows the transitions.

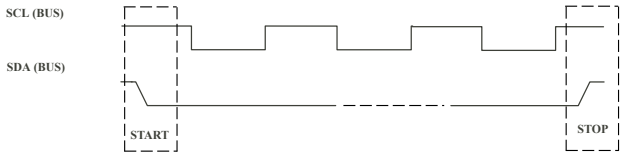


Figure 27-6: Start and Stop Conditions

The TWI special case start and stop conditions of the TWI controller include the following.

- Controller addressed as a target-receiver. If the controller asserts a stop condition during the data phase of a transfer, the TWI controller concludes the transfer (`TWI_ISTAT.TCOMP`).
- Controller addressed as a target-transmitter. If the controller asserts a stop condition during the data phase of a transfer, the TWI controller concludes the transfer (`TWI_ISTAT.TCOMP`) and indicates a target transfer error (`TWI_ISTAT.TERR`).
- Controller as a controller-transmitter or controller-receiver. If the stop bit (`TWI_CTLRCTL.STOP`) is set during an active controller transfer, the TWI controller issues a stop condition as soon as possible avoiding any error conditions. The TWI controller operates as if data transfer count had been reached.

General Call Support

The TWI controller always decodes and acknowledges a general call address if:

- The TWI controller is enabled as a target
- General call is enabled

The TWI_TGTCTL.GEN bit configures general call addressing (0x00) only when the TWI controller is a target-receiver.

If the data associated with the transfer is (NAK) not acknowledged, the TWI_TGTCTL.NAK bit can be set. If the TWI controller issues a general call as a controller-transmitter, set the appropriate address (TWI_CTLRADDR register) and transfer direction (TWI_CTLRCTL.DIR bit) and load the transmit FIFO data.

NOTE: The byte following the general call address usually defines the response of the target to the call. The interpretation of the command in the second byte is based on the value of its LSB. For a TWI target device, the bytes received after the general call address are considered data.

Fast Mode

Fast mode uses the same mechanics as the standard mode of operation. Fast mode affects electrical specifications and timing. When fast mode is enabled, (FAST) timing is modified to meet the following electrical requirements.

- Serial data rise times before arbitration evaluation (t_r)
- Stop condition set-up time from serial clock to serial data (t_{SUSTO})
- Bus free time between a stop and start condition (t_{BUF})

TWI Operating Modes

The TWI has two modes of operation: repeated start and clock stretching. The following sections describe the operating modes.

Repeated Start

A repeated start condition is the absence of a stop condition between two transfers. The two transfers can be of any direction type. Examples include a transmit followed by a receive, or a receive followed by a transmit. The following sections guide the programmer in developing a service routine.

Transmit Receive Repeated Start

The ***Repeated Start Followed by Data Receive*** figure shows a repeated start followed by a data receive sequence. The shading in the figure indicates that the target has control of the bus.

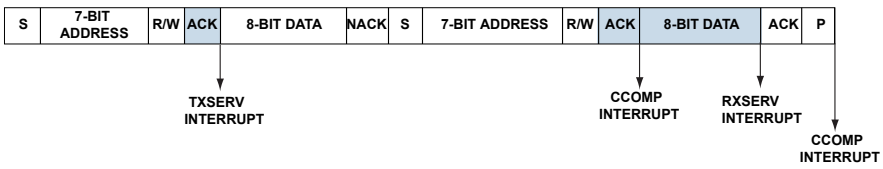


Figure 27-7: Repeated Start Followed by Data Receive

The tasks performed at each interrupt are:

- Transmit FIFO service (TWI_ISTAT.TXSERV) interrupt request. This interrupt is generated due to a FIFO access. Since this byte is the last of this transfer, the TWI uses the TWI_FIFOSTAT register to indicate that the transmit FIFO is empty. When read, TWI_CTLRCTL.DCNT bit field=0. Set the TWI_CTLRCTL.RSTART bit to indicate a repeated start and set the TWI_CTLRCTL.DIR bit if the following transfer is a data receive.
- Controller transfer complete (TWI_ISTAT.CCOMP) interrupt. This interrupt request is generated when all data transfers (TWI_CTLRCTL.DCNT bit field=0). If no errors occur, a start condition initiates. Clear the TWI_CTLRCTL.RSTART bit and program the TWI_CTLRCTL.DCNT bits with the desired number of bytes to receive.
- Receive FIFO service (TWI_ISTAT.RXSERV) interrupt. This interrupt request is generated due to the arrival of a byte in the receive FIFO. Simple data handling is the only requirement.
- Controller transfer complete (TWI_ISTAT.CCOMP) interrupt. This interrupt request is generated when the transfer completes.

Receive Transmit Repeated Start

The ***Repeated Start Data Receive Followed by Data Transmit*** figure illustrates a repeated start data receive followed by a data transmit sequence. The shading in the figure indicates that the target has control of the bus.

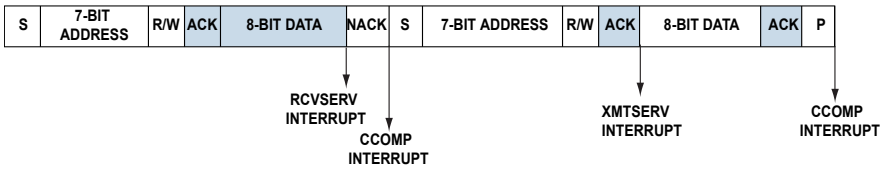


Figure 27-8: Repeated Start Data Receive Followed by Data Transmit

The tasks performed at each interrupt are:

- Receive FIFO service (TWI_ISTAT.RXSERV) interrupt. This interrupt request is generated due to the arrival of a data byte in the receive FIFO. Set the TWI_CTLRCTL.RSTART bit to indicate a repeated start and clear the TWI_CTLRCTL.DIR bit if the following transfer is a data transmit.
- Controller transfer complete (TWI_ISTAT.CCOMP) interrupt. This interrupt request has occurred due to the completion of the data receive transfer. If no errors occur, a start condition initiates. Clear the TWI_CTLRCTL.RSTART bit and program the TWI_CTLRCTL.DCNT bits with the desired number of bytes to transmit.
- Transmit FIFO service (TWI_ISTAT.TXSERV) interrupt. This interrupt request is generated due to a FIFO access. Simple data handling is the only requirement.
- Controller transfer complete (TWI_ISTAT.CCOMP) interrupt. This interrupt request is generated when the transfer completes.

NOTE: There is no timing constraint to meet the conditions—program the bits as required. Refer to Clock Stretching During Repeated Start section for more on how the controller stretches the clock during repeated start transfers.

Clock Stretching

Clock stretching is an added function of the TWI controller in controller mode operation. This behavior uses self-induced stretching of the I ^{2}C clock while waiting to service interrupts. Hardware initiates stretching automatically. No programming is necessary. The TWI controller as a controller supports three modes of clock stretching:

- Clock Stretching During FIFO Underflow
- Clock Stretching During FIFO Overflow
- Clock Stretching During Repeated Start

Clock Stretching During FIFO Underflow

During a controller mode transmit, an interrupt request occurs the instant the transmit FIFO becomes empty. The most recent byte begins transmission. If the `TWI_ISTAT.TXSERV` interrupt request is not serviced, the concluding acknowledge phase of the transfer stretches.

Stretching of the clock continues until new data bytes are written to the transmit FIFO (`TWI_TXDATA8` or `TWI_TXDATA16` registers). No other action is required to release the clock and continue the transmission. This behavior continues until the transmission completes (`TWI_CTLRCTL.DCNT`=0). The transmission concludes (`TWI_ISTAT.CCOMP`). The ***Clock Stretching during FIFO Underflow*** figure and table show the stretching.

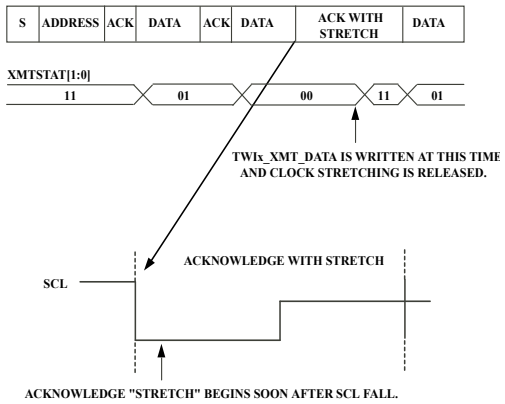


Figure 27-9: Clock Stretching during FIFO Underflow

TWI Controller	Processor
Interrupt: XMTSERV – Transmit FIFO buffer is empty	Acknowledge: Clear the interrupt request source bits. Write to the transmit FIFO buffer
...	...
Interrupt: CCOMP – Controller transmit complete (DCNT= 0x00)	Acknowledge: Clear the interrupt request source bits

Clock Stretching During FIFO Overflow

During a controller mode receive operation, an interrupt occurs at the instant the receive FIFO becomes full. It is during the acknowledge phase of this received byte that clock stretching begins. The TWI module makes no attempt to initiate the reception of another byte. Stretching of the clock continues until the data bytes previously received are read from the receive FIFO buffer (`TWI_RXDATA8` or `TWI_RXDATA16` registers). No other action is required to release the clock and continue the reception of data. This behavior continues until the reception is complete (`TWI_CTLRCTL.DCNT=0`). Reception concludes (`TWI_ISTAT.CCOMP`). The ***Clock Stretching During FIFO Overflow*** figure and table show the clock stretching.

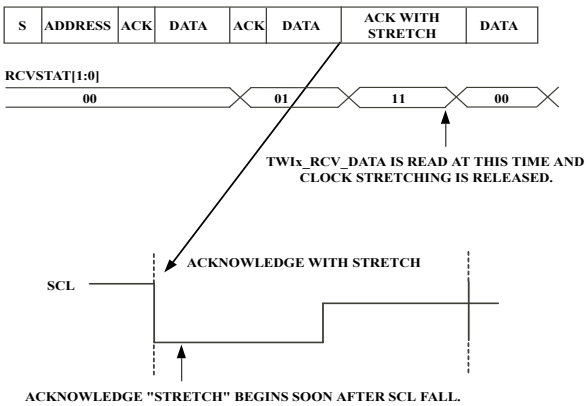


Figure 27-10: Clock Stretching During FIFO Overflow

TWI Controller	Processor
Interrupt: RCVSERV – Receive FIFO buffer is full	Acknowledge: Clear the interrupt request source bits. Read the receive FIFO buffer
...	...
Acknowledge: Clear the interrupt source bits	Interrupt: CCOMP – Controller receive complete

Clock Stretching During Repeated Start

The repeated start feature in I^2C protocol requires a transition between two subsequent transfers. With the use of clock stretching, the task of managing transitions becomes simpler and common to all transfer types.

Once an initial TWI controller transfer completes (transmit or receive), the clock initiates a stretch during the repeated start phase between transfers. Concurrent with this event, the initial transfer generates a `TWI_ISTAT.CCOMP` interrupt to signify the initial transfer has completed (`TWI_CTLRCTL.DCNT=0`). This initial transfer is handled without any special bit setting sequences or timing.

The clock stretching logic described applies here. With no system-related timing constraints, the subsequent transfer (receive or transmit) is set up and activated. This sequence can repeat as many times as required to string a series of repeated start transfers together. The ***Clock Stretching during Repeated Start Condition*** figure and table show the clock stretching.

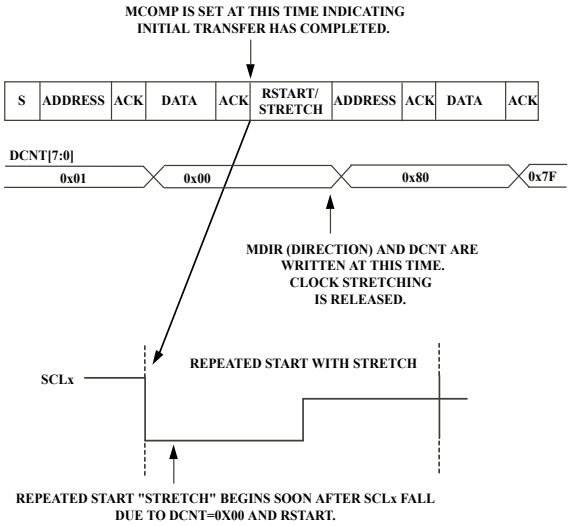


Figure 27-11: Clock Stretching during Repeated Start Condition

TWI Controller	Processor
Interrupt: CCOMP – Initial transmit has completed and DCNT = 0x00. Note: transfer in progress, RSTART previously set.	Acknowledge: Clear the interrupt request source bits. Write to `TWI_CTLRCTL`, setting DIR (receive), clearing RSTART, and setting new DCNT value (nonzero).
Interrupt: RCVSERV – Receive FIFO is full	Acknowledge: Clear the interrupt request source bits. Read the receive FIFO buffer
...	...
Interrupt: CCOMP – Controller receive complete	Acknowledge: Clear the interrupt request source bits

TWI Programming Model

The topics in this section provide information on the basic programming steps required to set up and run the two wire interface.

General Setup

General setup refers to register writes that are required for both target mode and controller mode operations.

Perform general setup before setting either the controller or target enable bits.

1. Program the `TWI_CTL.EN` bit to enable the TWI controller and set the prescale value (`TWI_CTL.PRESCALE` bit).
2. Program the prescale value to the binary representation of f_{SCLK0}/10 MHz. Round up all values to the next whole number.
3. Set the `TWI_CTL.EN` bit to enable the controller.

Once the TWI controller is enabled, a bus busy condition can be detected. This condition clears after t_{BUF} has expired, assuming no additional bus activity has been detected.

Target Mode

When enabled, target mode operation supports both receive and transmit data transfers.

It is not possible to enable only one data transfer direction and not acknowledge (NAK) the other. The following setup reflects this functionality.

1. Program the `TWI_TGTADDR` register. The TWI uses the appropriate 7 bits in determining a match during the address phase of the transfer.
2. Program the `TWI_TXDATA8.VALUE` or `TWI_TXDATA16` registers. These values are the initial data values for transmission when the target is addressed, and transmission is needed. This step is optional. If no data is written when the target is addressed and transmission is needed, the serial clock (`TWI_SCL`) stretches. An interrupt is generated until data is written to the transmit FIFO.
3. Program the `TWI_IMSK` register. There are enable-bits associated with the desired interrupt sources. For example, programming the value 0x000F results in an interrupt request output to the processor, when the TWI module detects a valid address match. An interrupt request also occurs when a valid target transfer completes or has an error, or a subsequent transfer has begun, and the previous transfer has not been serviced.
4. Program the `TWI_TGTCTL` register. This step prepares and enables target mode operation. For example, programming the value 0x0005 enables target mode operation and requires 7-bit addressing. It indicates that data in the transmit FIFO buffer is for target mode transmission.

The ***Target Mode Interaction*** table and ***TWI Target Mode Program Flow*** diagram represent the interaction between the TWI controller and the processor using this example.

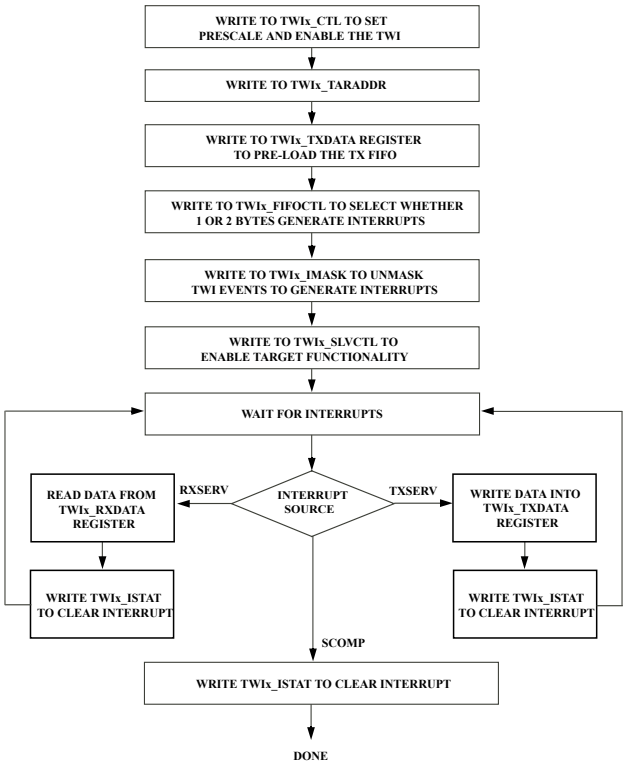


Figure 27-12: TWI Target Mode Program Flow

Table 27-3: Target Mode Interaction

TWI Controller	Processor
Interrupt: SINIT – Target transfer in progress.	Acknowledge: Clear the interrupt source bits.
Interrupt: RCVSERV – Receive buffer is full.	Acknowledge: Clear the interrupt source bits. Read TWIx_FIFO_STAT. Read the receive FIFO buffer.
...	...
Interrupt: SCOMP – Target transfer complete.	Acknowledge: Clear the interrupt source bits. Read the receive FIFO buffer.

Controller Mode Program Flow

The ***Controller Mode Program Flow*** figure shows the program for the TWI in controller mode.

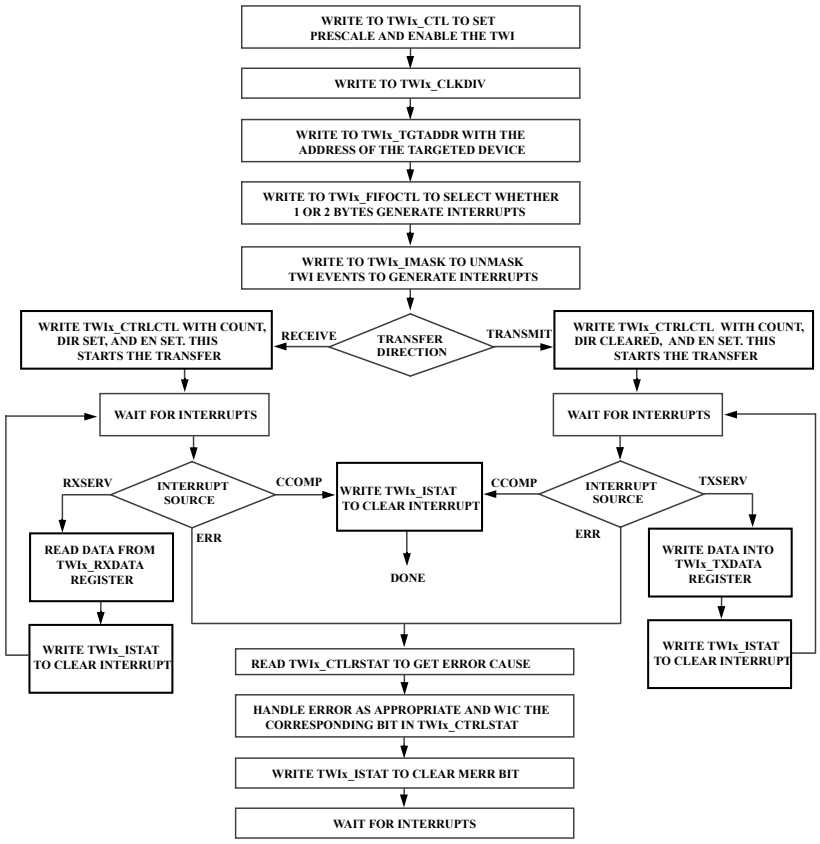


Figure 27-13: Controller Mode Program Flow

Controller Mode Clock Setup

Controller mode operation is set up and executed on a per-transfer basis.

An example of programming steps for a receive and for a transmit is given separately in following sections. The programming step for clock setup listed here is common to both transfer types.

1. Program the `TWI_CLKDIV` register to define the minimum high and minimum low duration for the clock.

The `TWI_CLKDIV.CLKHI` and `TWI_CLKDIV.CLKLO` fields do not guarantee a certain frequency. Rather, they guarantee a certain minimum high and low duration for `TWI_SCL`. The slew rate controls falling edges. The RC time constant formed by the pull-up resistor and the SCL capacitance govern rising edges. See the “Register Descriptions” section for more details.

Controller Mode Transmit

Follow these programming steps for a single controller mode transmission:

1. Program the `TWI_CTLRADDR` register. This step defines the address transmitted during the address phase of the transfer.

2. Program the TWI_TXDATA8 or TWI_TXDATA16 register. This step configures the initial data transmitted. It is an error to complete the address phase of the transfer and not have data available in the transmit FIFO buffer.
3. Program the TWI_FIFOCTL register. The programming indicates if the transmit FIFO buffer interrupt requests occur with each byte transmitted (8-bits) or with every 2 bytes transmitted (16-bits).
4. Program the TWI_IMSK register. This step enables the bits associated with the desired interrupt request sources. For example, programming the value 0x0030 results in an interrupt output to the processor when the controller transfer completes, and the controller transfer has an error.
5. Program the TWI_CTLRCTL register. This step prepares and enables controller mode operation. For example, programming the value 0x0201: enables controller mode operation, generates a 7-bit address, sets the direction to controller-transmit, uses standard mode timing, and transmits 8 data bytes before generating a stop condition.

The ***Controller Mode Transmit Setup Interaction*** table represents the interaction between the TWI controller and the processor using this example.

Table 27-4: Controller Mode Transmit Setup Interaction

TWI Controller	Processor
Interrupt: XMTSERV – Transmit buffer is empty	Acknowledge: Clear the interrupt request source bits and write to the transmit FIFO buffer
...	...
Interrupt: cCOMP – Controller transfer complete	Acknowledge: Clear the interrupt request source bits

Controller Mode Receive

Follow these programming steps for a single controller mode receive.

1. Program the TWI_CTLRADDR register. This step defines the address transmitted during the address phase of the transfer.
2. Program the TWI_FIFOCTL register. This step indicates if the receive FIFO buffer interrupt requests occur with each byte received (8-bits) or with every 2 bytes received (16-bits).
3. Program the TWI_IMSK register. This step configures the enable bits associated with the desired interrupt sources. For example, programming the value 0x0030 results in an interrupt request output to the processor when the controller transfer completes, and the controller transfer has an error.
4. Program the TWI_CTLRCTL register. This step prepares and enables controller mode operation. For example, programming the value 0x0205: enables controller mode operation, generates a 7-bit address, sets the direction to controller-receive, uses standard mode timing, and receives 8 data bytes before generating a stop condition.

The ***Controller Mode Receive Setup Interaction*** table shows the interaction between the TWI controller and the processor using this example.

Table 27-5: Controller Mode Receive Setup Interaction

TWI Controller	Processor
Interrupt: RCVSERV – receive buffer is full	Acknowledge: Clear the interrupt request source bits. Read the receive FIFO buffer.
...	...
Interrupt: cCOMP – Controller transfer complete	Acknowledge: Clear the interrupt request source bits and read the receive FIFO buffer

NOTE: After the `TWI_CTLRCTL.DCNT` bit decrements to zero, the TWI controller device sends a NAK to indicate to the target transmitter to release the bus. This operation allows the controller to send the stop signal to terminate the transfer.

ADSP-2184x TWI Register Descriptions

Two-Wire Interface (TWI) contains the following registers.

Table 27-6: ADSP-2184x TWI Register List

Name	Description
`TWI_CLKDIV`	SCL Clock Divider Register
`TWI_CTL`	Control Register
`TWI_FIFOCTL`	FIFO Control Register
`TWI_FIFOSTAT`	FIFO Status Register
`TWI_IMSK`	Interrupt Mask Register
`TWI_ISTAT`	Interrupt Status Register
`TWI_CTLRADDR`	Controller Mode Address Register
`TWI_CTLRCTL`	Controller Mode Control Registers
`TWI_CTLRSTAT`	Controller Mode Status Register
`TWI_RXDATA16`	Rx Data Double-Byte Register
`TWI_RXDATA8`	Rx Data Single-Byte Register
`TWI_TGTADDR`	Target Mode Address Register
`TWI_TGTCTL`	Target Mode Control Register
`TWI_TGTSTAT`	Target Mode Status Register
`TWI_TXDATA16`	Tx Data Double-Byte Register
`TWI_TXDATA8`	Tx Data Single-Byte Register

SCL Clock Divider Register

During controller mode operation, the TWI_CLKDIV holds values, which the TWI uses to create the high and low durations of the serial clock (SCL). The clock signal SCL is an output in controller mode and an input in target mode. The values in the TWI_CLKDIV.CLKLO and TWI_CLKDIV.CLKHI fields add up to the CLKDIV value the following equation.

CLKDIV = TWI SCL period / 10 MHz time reference

Serial clock frequencies can vary from 400 KHz to less than 20 KHz. The resolution of the clock generated is 1/10 MHz or 100 ns. For example, for an SCL of 400 KHz (period = 1/400 KHz = 2500 ns) and an internal time reference of 10 MHz (period = 100 ns):

CLKDIV = 2500 ns / 100 ns = 25

For an SCL with a 30% duty cycle, use TWI_CLKDIV.CLKLO = 17 and TWI_CLKDIV.CLKHI = 8.

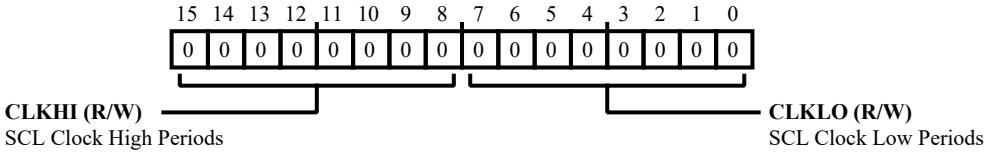


Figure 27-14: TWI_CLKDIV Register Diagram

Table 27-7: TWI_CLKDIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	CLKHI	SCL Clock High Periods. The TWI_CLKDIV.CLKHI specifies the number of 10 MHz time reference periods the serial clock (SCL) waits before a new clock low period begins, assuming a single requester.
7:0 (R/W)	CLKLO	SCL Clock Low Periods. The TWI_CLKDIV.CLKLO specifies the number of internal time reference periods the serial clock (SCL) is held low.

Control Register

The TWI_CTL enables the TWI, establishes a relationship between the system clock (eval{SCLK}) and the TWI controller internally timed events, and enables SCCB compatibility.

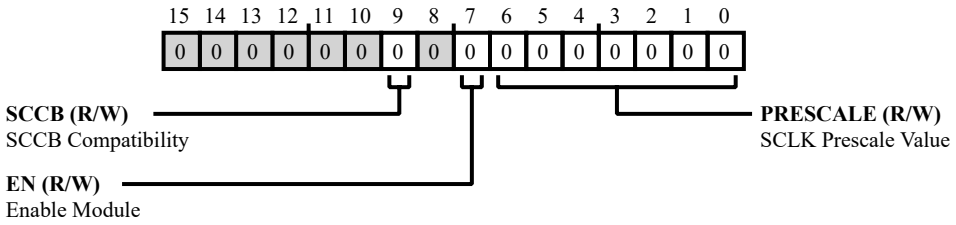


Figure 27-15: TWI_CTL Register Diagram

Table 27-8: TWI_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SCCB	SCCB Compatibility. The TWI_CTL.SCCB enables SCCB compatible operation for the TWI. SCCB compatibility is an optional feature and should not be used in an I^2C bus system. When this feature is enabled, all completer asserted acknowledgement bits are ignored by this requester. This feature is valid only during transfers where the TWI is requesting an SCCB bus. Completer mode transfers should be avoided when this feature is enabled because the TWI controller always generates an acknowledge in completer mode.	
		0	Disable SCCB Compatibility When disabled, requester transfers are not SCCB compatible.
		1	Enable SCCB Compatibility When enabled, requester transfers are SCCB compatible. All completer-asserted acknowledgment bits are ignored by this requester.
7 (R/W)	EN	Enable Module. The TWI_CTL.EN enables TWI controller operation for either requester and/or completer mode of operation. It is recommended that this bit be set at the time TWI_CTL.PRESCALE is initialized and remain set. This method guarantees accurate operation of bus busy detection logic.	
		0	Disable
		1	Enable

Table 27-8: TWI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	PRESCALE	SCLK Prescale Value. The `TWI_CTL.PRESCALE` holds the pre-scaled value for the TWI internal time reference. This reference is derived from SCLK according to the formula: `TWI_CTL.PRESCALE` = f_{SCLK}/10MHz The `TWI_CTL.PRESCALE` specifies the number of system clock (SCLK) periods used in the generation of one internal time reference. The value of `TWI_CTL.PRESCALE` must be set to create an internal time reference with a period of 10 MHz. It is represented as a 7-bit binary value.

FIFO Control Register

The TWI_FIFOCTL control bits affect only the FIFO and are not tied in any way with controller or target mode operation.

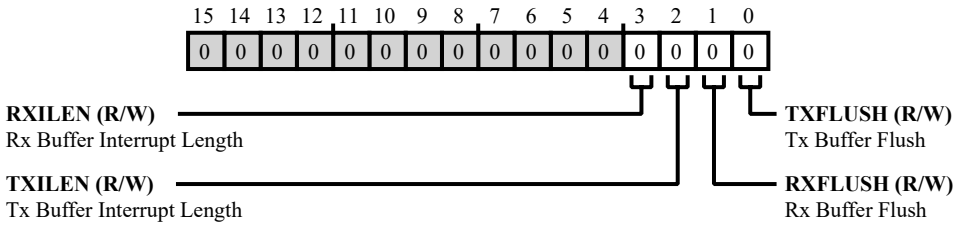


Figure 27-16: TWI_FIFOCTL Register Diagram

Table 27-9: TWI_FIFOCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXILEN	Rx Buffer Interrupt Length. The TWI_FIFOCTL.RXILEN determines the rate at which receive buffer interrupts are to be generated. Interrupts may be generated with each byte received or after two bytes are received. Interrupt status is available in TWI_FIFOSTAT.RXSTAT.	
		0	RXSERVI on 1 or 2 Bytes in FIFO
		1	RXSERVI on 2 Bytes in FIFO
2 (R/W)	TXILEN	Tx Buffer Interrupt Length. The TWI_FIFOCTL.TXILEN determines the rate at which transmit buffer interrupts are to be generated. Interrupts may be generated with each byte transmitted or after two bytes are transmitted. Interrupt status is available in TWI_FIFOSTAT.TXSTAT.	
		0	TXSERVI on 1 Byte of FIFO Empty
		1	TXSERVI on 2 Bytes of FIFO Empty
1 (R/W)	RXFLUSH	Rx Buffer Flush. The TWI_FIFOCTL.RXFLUSH directs the TWI to flush the contents of the receive buffer and update TWI_FIFOSTAT.RXSTAT to indicate the buffer is empty. This state is held until this bit is cleared. During an active receive, the receive buffer in this state responds to the receive logic as if it is full.	
		0	Normal Operation of Rx Buffer
		1	Flush Rx Buffer

Table 27-9: TWI_FIFOCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	TXFLUSH	Tx Buffer Flush. The `TWI_FIFOCTL.TXFLUSH` directs the TWI to flush the contents of the transmit buffer and update `TWI_FIFOSTAT.TXSTAT` to indicate the buffer is empty. This state is held until this bit is cleared. During an active transmit, the transmit buffer in this state responds to the transmit logic as if it is empty.	
		0	Normal Operation of Tx Buffer
		1	Flush Tx Buffer

FIFO Status Register

The TWI_FIFOSTAT fields indicate the state of the FIFO buffers' receive and transmit contents. The FIFO buffers do not discriminate between controller data and target data. By using the status and control bits provided, the FIFO can be managed to allow simultaneous controller and target operation.

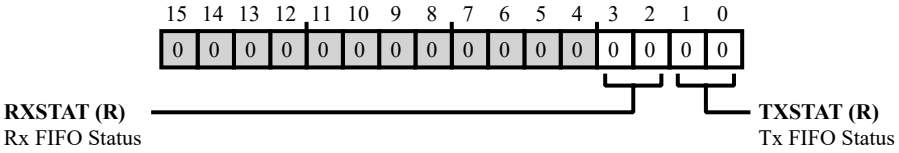


Figure 27-17: TWI_FIFOSTAT Register Diagram

Table 27-10: TWI_FIFOSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/NW)	RXSTAT	Rx FIFO Status. The read-only TWI_FIFOSTAT.RXSTAT indicates the number of valid data bytes in the receive FIFO buffer. The status is updated with each FIFO buffer read using the peripheral data bus or write access by the receive shift register. Simultaneous accesses are allowed.	
		0	Empty. The FIFO is empty.
		1	Contains 1 Byte. The FIFO contains one byte of data. A single byte peripheral read of the FIFO is allowed.
		2	Reserved
		3	Full. The FIFO is full and contains two bytes of data. Either a single or double byte peripheral read of the FIFO is allowed.
1:0 (R/NW)	TXSTAT	Tx FIFO Status. The read-only TWI_FIFOSTAT.TXSTAT field indicates the number of valid data bytes in the FIFO buffer. The status is updated with each FIFO buffer write using the peripheral data bus or read access by the transmit shift register. Simultaneous accesses are allowed.	
		0	Empty. The FIFO is empty. Either a single or double byte peripheral write of the FIFO is allowed.
		1	Contains 1 Byte. The FIFO contains one byte of data. A single byte peripheral write of the FIFO is allowed.
		2	Reserved
		3	Full. The FIFO is full and contains two bytes of data.

Interrupt Mask Register

The TWI_IMSK enables interrupt sources to assert the interrupt output. Each mask bit corresponds with one interrupt request source bit in TWI_ISTAT. Reading and writing TWI_IMSK does not affect the contents of the TWI_ISTAT.

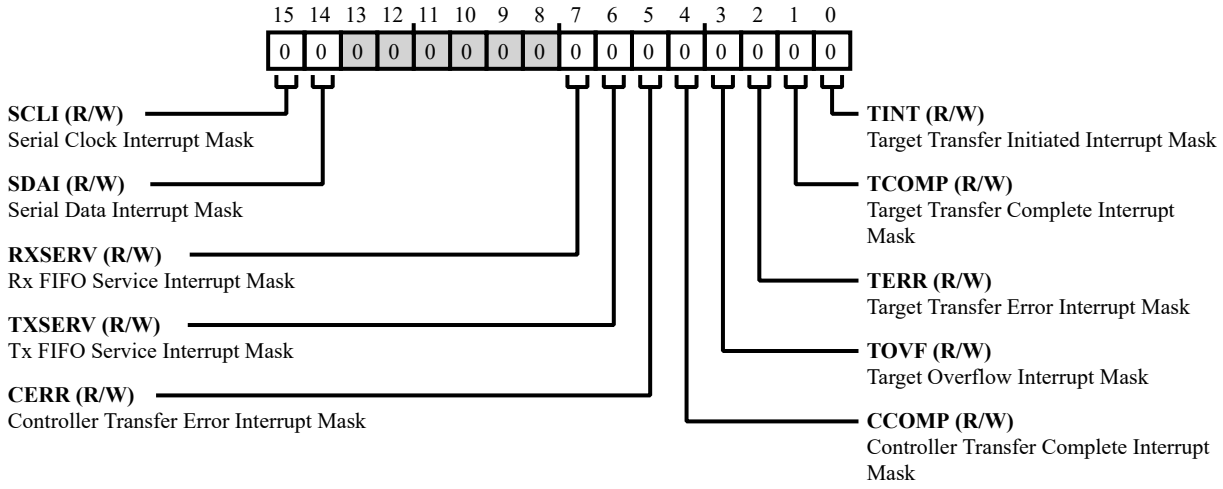


Figure 27-18: TWI_IMSK Register Diagram

Table 27-11: TWI_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	SCLI	Serial Clock Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
14 (R/W)	SDAI	Serial Data Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
7 (R/W)	RXSERV	Rx FIFO Service Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
6 (R/W)	TXSERV	Tx FIFO Service Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt

Table 27-11: TWI_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	CERR	Controller Transfer Error Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
4 (R/W)	CCOMP	Controller Transfer Complete Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
3 (R/W)	TOVF	Target Overflow Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
2 (R/W)	TERR	Target Transfer Error Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
1 (R/W)	TCOMP	Target Transfer Complete Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
0 (R/W)	TINT	Target Transfer Initiated Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt

Interrupt Status Register

The TWI_ISTAT contains information about functional areas requiring servicing. Many of the bits serve as an indicator to further read and service various status registers. After servicing the interrupt source associated with a bit, the user must clear that interrupt source bit by writing a 1 to it.

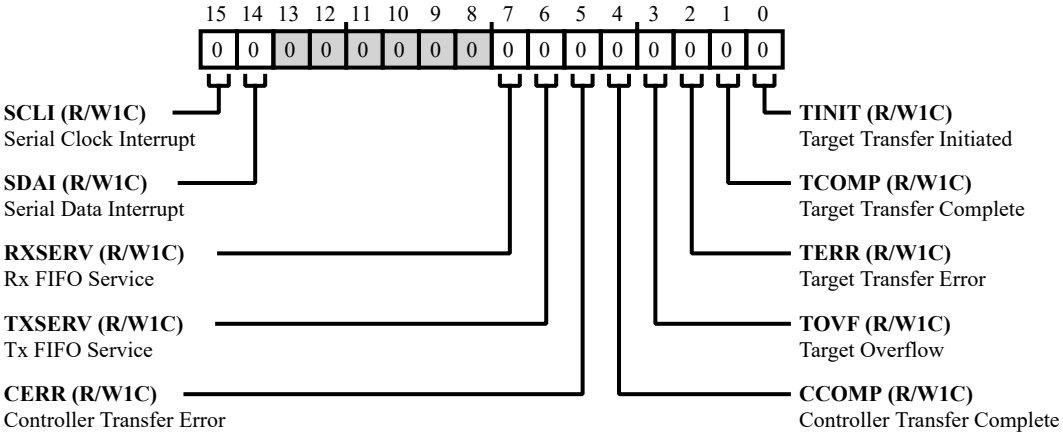


Figure 27-19: TWI_ISTAT Register Diagram

Table 27-12: TWI_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	SCLI	Serial Clock Interrupt. If the TWI is enabled (TWI_CTL.EN), SCLI is set on a high-to-low transition of the serial clock pin (SCLx). Normally, this bit is not required for I^2C bus transfers. It will be initially set on an I^2C transfer and does not require clearing.	
		0	No Interrupt. No transition was detected on the SCLx pin.
		1	Interrupt Detected. A high-to-low transition was detected on the SCLx pin. This bit is W1C.
14 (R/W1C)	SDAI	Serial Data Interrupt. If the TWI is enabled (TWI_CTL.EN), SDAI is set on a high-to-low transition of the serial data pin (SDAx). Normally, this bit is not required for I^2C bus transfers. It will be initially set on an I^2C transfer and does not require clearing.	
		0	No Interrupt. No transition was detected on the SDAx pin.
		1	Interrupt Detected. A high-to-low transition was detected on the SDAx pin. This bit is W1C.

Table 27-12: TWI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	RXSERV	Rx FIFO Service. If `TWI_FIFOCTL.RXILEN` =0, the `TWI_ISTAT.RXSERV` is set each time the `TWI_FIFOSTAT.RXSTAT` field is updated to either 01 or 11. If `TWI_FIFOCTL.RXILEN` =1, the `TWI_ISTAT.RXSERV` is set each time `TWI_FIFOSTAT.RXSTAT` is updated to 11.	
		0	No Interrupt. The FIFO does not require servicing, or the `TWI_FIFOSTAT.RXSTAT` field has not changed since this bit was last cleared.
		1	Interrupt Detected. The receive FIFO buffer has one or two 8-bit words of data available to be read.
6 (R/W1C)	TXSERV	Tx FIFO Service. If `TWI_FIFOCTL.TXILEN` =0, the `TWI_ISTAT.TXSERV` is set each time the `TWI_FIFOSTAT.TXSTAT` field is updated to either 01 or 00. If `TWI_FIFOCTL.TXILEN` =1, the `TWI_ISTAT.TXSERV` is set each time `TWI_FIFOSTAT.TXSTAT` is updated to 00.	
		0	No Interrupt. FIFO does not require servicing, or the `TWI_FIFOSTAT.TXSTAT` field has not changed since this bit was last cleared.
		1	Interrupt Detected. The transmit FIFO buffer has one or two 8-bit locations available to be written.
5 (R/W1C)	CERR	Controller Transfer Error. The `TWI_ISTAT.CERR` indicates that a controller error has occurred. The conditions surrounding the error are indicated by the controller status register (`TWI_CTLRSTAT`).	
		0	No Interrupt
		1	Interrupt Detected
4 (R/W1C)	CCOMP	Controller Transfer Complete. The `TWI_ISTAT.CCOMP` indicates that the initiated controller transfer has completed. In the absence of a repeat start, the bus has been released.	
		0	No Interrupt
		1	Interrupt Detected

Table 27-12: TWI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	TOVF	Target Overflow. The `TWI_ISTAT.TOVF` indicates that the `TWI_ISTAT.TCOMP` bit was set at the time a subsequent transfer has acknowledged an address phase. The transfer continues, however, it may be difficult to delineate data of one transfer from another.	
		0	No Interrupt
		1	Interrupt Detected
2 (R/W1C)	TERR	Target Transfer Error. The `TWI_ISTAT.TERR` indicates that a target error has occurred. A restart or stop condition has occurred during the data receive phase of a transfer.	
		0	No Interrupt
		1	Interrupt Detected
1 (R/W1C)	TCOMP	Target Transfer Complete. The `TWI_ISTAT.TCOMP` indicates that the transfer is complete and either a stop, or a restart was detected.	
		0	No Interrupt
		1	Interrupt Detected
0 (R/W1C)	TINIT	Target Transfer Initiated. The `TWI_ISTAT.TINIT` indicates whether or not a target transfer is in progress.	
		0	No Interrupt. A transfer is not in progress, or an address match has not occurred since the last time this bit was cleared.
		1	Interrupt Detected. The completer has detected an address match, and a transfer has been initiated.

Controller Mode Address Register

During the addressing phase of a transfer, the TWI controller, with its controller enabled, transmits the contents of TWI_CTLRADDR. When programming this register, omit the read/write bit. That is, only the upper 7 bits that make up the target address should be written to this register. For example, if the target address is b#1010000X, where X is the read/write bit, the TWI_CTLRADDR is programmed with b#1010000, which corresponds to 0x50. When sending out the address on the bus, the TWI controller appends the read/write bit as appropriate based on the state of the TWI_CTLRCTL.DIR bit.

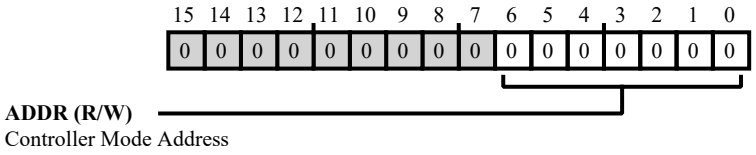


Figure 27-20: TWI_CTLRADDR Register Diagram

Table 27-13: TWI_CTLRADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	ADDR	Controller Mode Address.

Controller Mode Control Registers

The TWI_CTLRCTL controls the logic associated with controller mode operation. Bits in this register do not affect target mode operation and should not be modified to control target mode functionality.

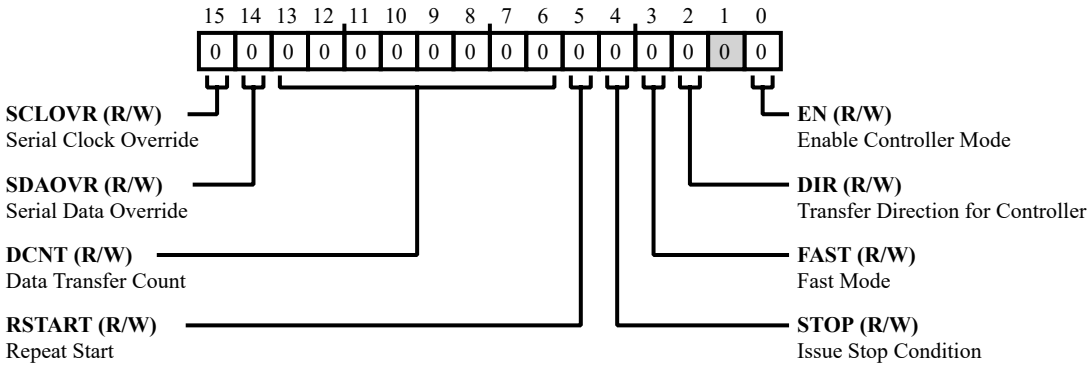


Figure 27-21: TWI_CTLRCTL Register Diagram

Table 27-14: TWI_CTLRCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	SCLOVR	Serial Clock Override. The TWI_CTLRCTL.SCLOVR provides direct control of the serial clock line when required. Normal controller and target mode operation should not require override operation. When TWI_CTLRCTL.SCLOVR is set, the TWI overrides normal serial clock output, driving it to an active 0 level and overriding all other logic. This state is held until this bit is cleared. When TWI_CTLRCTL.SCLOVR is cleared, the TWI permits normal serial clock operation under the control of controller mode clock generation and target mode clock stretching logic.	
		0	Permit Normal SCL Operation
		1	Override Normal SCL Operation
14 (R/W)	SDAOVR	Serial Data Override. The TWI_CTLRCTL.SDAOVR provides direct control of the serial data line when required. Normal controller and target mode operation should not require override operation. When TWI_CTLRCTL.SDAOVR is set, the TWI overrides normal serial data operation under the control of the transmit shift register and acknowledge logic, driving serial data output to an active 0 level and overriding all other logic. This state is held until this bit is cleared. When TWI_CTLRCTL.SDAOVR is cleared, the TWI permits normal serial data operation.	
		0	Permit Normal SDA Operation
		1	Override Normal SDA Operation

Table 27-14: TWI_CTLRCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:6 (R/W)	DCNT	Data Transfer Count. The TWI_CTLRCTL.DCNT indicates the number of data bytes to transfer. As each data word is transferred, the TWI decrements this counter. When TWI_CTLRCTL.DCNT decrements to 0, a stop condition is generated. Setting TWI_CTLRCTL.DCNT to 0xFF disables the counter. In this transfer mode, data continues to be transferred until it is concluded by setting the TWI_CTLRCTL.STOP bit. In the event a controller transmit is aborted due to a completer data NAK, the value of TWI_CTLRCTL.DCNT equals the number of bytes not sent. The byte which was NAK'ed by the completer is counted as a sent byte.	
5 (R/W)	RSTART	Repeat Start. The TWI_CTLRCTL.RSTART enables the TWI to issue a repeat start condition at the conclusion of the current transfer (TWI_CTLRCTL.DCNT =0) and begin the next transfer. The current transfer concludes with updates to the appropriate status and interrupt bits. If errors occurred during the previous transfer, a repeat start does not occur. In the absence of any errors, controller enable (TWI_CTLRCTL.EN) does not self clear on a repeat start.	
		0	Disable Repeat Start
		1	Enable Repeat Start
4 (R/W)	STOP	Issue Stop Condition. The TWI_CTLRCTL.STOP directs the TWI to issue a stop condition. The transfer concludes as soon as possible avoiding any error conditions (as if data transfer count had been reached). At that time, the TWI_IMSK is updated along with any associated status bits.	
		0	Permit Normal Operation
		1	Issue Stop
3 (R/W)	FAST	Fast Mode. The TWI_CTLRCTL.FAST selects whether the TWI operates in fast mode or standard mode. In fast mode, the TWI uses timing specifications for transfers at up to 400K bits/s. In standard mode, the TWI uses timing specifications for transfers at up to 100K bits/s.	
		0	Select Standard Mode
		1	Select Fast Mode
2 (R/W)	DIR	Transfer Direction for Controller. The TWI_CTLRCTL.DIR selects the transfer direction for the TWI as controller initiated receive or transmit.	
		0	Requester Transmit
		1	Requester Receive

Table 27-14: TWI_CTLRCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Enable Controller Mode. The `TWI_CTLRCTL.EN` enables controller mode functionality. A start condition is generated if the bus is idle. This bit self clears at the completion of a transfer (after `TWI_CTLRCTL.DCNT` decrements to zero), including transfers terminated due to errors. If disabled (=0) during operation, the transfer is aborted, and all logic associated with controller mode transfers are reset. Serial data and serial clock (SDA, SCL) are no longer driven. Write-1-to-clear status bits are not affected.	
		0	Disable
		1	Enable

Controller Mode Status Register

The TWI_CTLRSTAT holds information during controller mode transfers and at their conclusion. Generally, controller mode status bits are not directly associated with the generation of interrupt requests, but these bits offer information on the current transfer. Target mode operation does not affect controller mode status bits.

Note that while TWI_CTLRSTAT.SCLSEN is set (this condition could be due to having no pull-up resistor on TWI_SCL or another agent is driving TWI_SCL low), the acknowledge bits (TWI_CTLRSTAT.ANAK and TWI_CTLRSTAT.DNAK) do not update. This result occurs because the acknowledge conditions are sampled during the high phase of TWI_SCL.

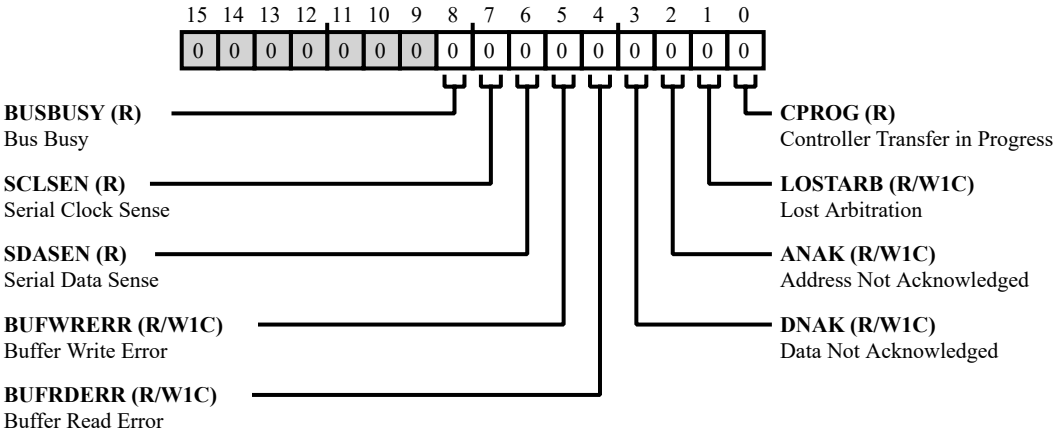


Figure 27-22: TWI_CTLRSTAT Register Diagram

Table 27-15: TWI_CTLRSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	BUSBUSY	Bus Busy. The TWI_CTLRSTAT.BUSBUSY indicates whether the bus is currently busy or free. This indication is not limited to only this device but is for all devices. On a start condition, the setting of the register value is delayed due to the input filtering. On a stop condition the clearing of the register value occurs after t_{BUF}.	
		0	Bus Free. The bus is free. The clock and data bus signals have been inactive for the appropriate bus free time.
		1	Bus Busy. The bus is busy. Clock or data activity has been detected.

Table 27-15: TWI_CTLRSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SCLSEN	Serial Clock Sense. The `TWI_CTLRSTAT.SCLSEN` indicates the active or inactive state of the serial clock. Use this status bit when direct sensing of the serial clock line is required. The register value is delayed due to the input filter (nominally 50 ns). Normal controller and target mode operation should not require this feature.	
		0	SCL Inactive "One". An inactive "one" is being sensed on the serial clock.
		1	SCL Active "Zero". An active "zero" is being sensed on the serial clock. The source of the active driver is not known and can be internal or external.
6 (R/NW)	SDASEN	Serial Data Sense. The `TWI_CTLRSTAT.SDASEN` indicates the active or inactive status of the serial data. Use this status bit when direct sensing of the serial data line is required. The register value is delayed due to the input filter (nominally 50 ns). Normal controller and target mode operation should not require this feature.	
		0	SDA Inactive "One". An inactive "one" is currently being sensed on the serial data line.
		1	SDA Active "Zero". An active "zero" is currently being sensed on the serial data line. The source of the active driver is not known and can be internal or external.
5 (R/W1C)	BUFWRERR	Buffer Write Error. The `TWI_CTLRSTAT.BUFWRERR` indicates whether the current controller transfer was aborted due to a receive buffer write error. The receive buffer and receive shift register were both full at the same time. This bit is W1C.	
		0	No Status
		1	Buffer Write Error
4 (R/W1C)	BUFRDERR	Buffer Read Error. The `TWI_CTLRSTAT.BUFRDERR` indicates whether the current controller transfer was aborted due to a transmit buffer read error. The error occurs if the buffer is empty when data is required by the transmit shift register. This bit is W1C.	
		0	No Status
		1	Buffer Read Error

Table 27-15: TWI_CTLRSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	DNAK	Data Not Acknowledged. The `TWI_CTLRSTAT.DNAK` indicates whether the current controller transfer was aborted due to the detection of a NAK during data transmission. This bit is W1C.	
		0	No Status
		1	Data NAK
2 (R/W1C)	ANAK	Address Not Acknowledged. The `TWI_CTLRSTAT.ANAK` indicates whether the current controller transfer was aborted due to the detection of a NAK during the address phase of the transfer. This bit is W1C.	
		0	No Status
		1	Address NAK
1 (R/W1C)	LOSTARB	Lost Arbitration. The `TWI_CTLRSTAT.LOSTARB` indicates whether the current transfer was aborted due to the loss of arbitration with another controller. This bit is W1C.	
		0	No Status
		1	Lost Arbitration
0 (R/NW)	CPROG	Controller Transfer in Progress. The `TWI_CTLRSTAT.CPROG` indicates whether or not a controller transfer is in progress. If clear (`TWI_CTLRSTAT.CPROG` =0), currently no transfer is taking place. This can occur after a transfer is complete or while an enabled controller is waiting for an idle bus.	
		0	No Status
		1	Requester Transfer in Progress

Rx Data Double-Byte Register

The TWI_RXDATA16 holds a 16-bit data value read from the FIFO buffer. To reduce interrupt output rates and peripheral bus access times, a double byte receive data access can be performed. Two data bytes can be read, effectively emptying the receive FIFO buffer with a single access.

The data is read in little endian byte order, where byte 0 is the first byte received and byte 1 is the second byte received. With each access, the receive status (TWI_FIFOSTAT.RXSTAT) field is updated to indicate it is empty. If an access is performed while the FIFO buffer is not full, the read data is unknown and the existing FIFO buffer data and its status remains unchanged.

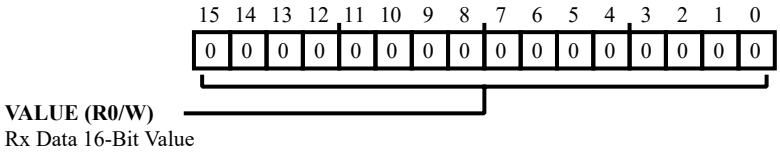


Figure 27-23: TWI_RXDATA16 Register Diagram

Table 27-16: TWI_RXDATA16 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W)	VALUE	Rx Data 16-Bit Value.

Rx Data Single-Byte Register

The TWI_RXDATA8 holds an 8-bit data value read from the FIFO buffer. Receive data is read from the corresponding receive buffer in a first-in first-out order. Although peripheral bus reads are 16 bits, a read access to TWI_RXDATA8 accesses only one transmit data byte from the FIFO buffer. With each access, the receive status (TWI_FIFOSTAT.RXSTAT) field is updated. If an access is performed while the FIFO buffer is empty, the data is unknown and the FIFO buffer status remains indicating it is empty.

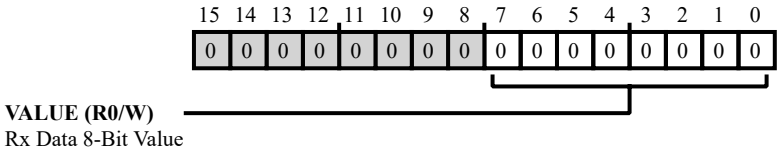


Figure 27-24: TWI_RXDATA8 Register Diagram

Table 27-17: TWI_RXDATA8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	VALUE	Rx Data 8-Bit Value.

Target Mode Address Register

The TWI_TGTADDR holds the target mode address, which is the valid address to which the target-enabled TWI controller responds. The TWI controller compares this value with the received address during the addressing phase of a transfer.

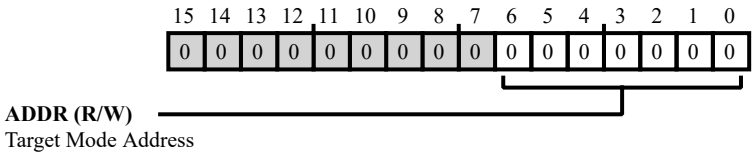


Figure 27-25: TWI_TGTADDR Register Diagram

Table 27-18: TWI_TGTADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	ADDR	Target Mode Address.

Target Mode Control Register

The TWI_TGTCTL controls the logic associated with target mode operation. Settings in this register do not affect controller mode operation and should not be modified to manage controller mode functionality.

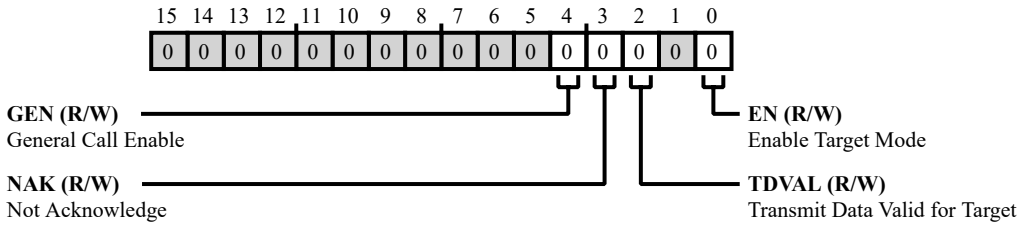


Figure 27-26: TWI_TGTCTL Register Diagram

Table 27-19: TWI_TGTCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	GEN	General Call Enable. The TWI_TGTCTL.GEN enables general call address matching. When enabled, a general call target receive transfer is accepted. All status and interrupt source bits associated with transfers are updated. Note that general call address detection is available only when target mode is enabled.	
		0	Disable General Call Matching
		1	Enable General Call Matching
3 (R/W)	NAK	Not Acknowledge. The TWI_TGTCTL.NAK directs the TWI to generate a NAK (if set) or an ACK (if cleared) at the conclusion of data transfer for target receive. For NAK, the target is still considered to be addressed at the conclusion of transfer.	
		0	Generate ACK
		1	Generate NAK
2 (R/W)	TDVAL	Transmit Data Valid for Target. The TWI_TGTCTL.TDVAL selects whether the data in the transmit FIFO is available (valid) for target transmission (TWI_TGTCTL.TDVAL set). If the FIFO data is not available (invalid) for target transmission (TWI_TGTCTL.TDVAL cleared), the data in the transmit FIFO is for controller mode transmits, and the data is not allowed to be used during a target transmit; the transmit FIFO is treated as if it is empty.	
		0	Data Invalid for Target Tx
		1	Data Valid for Target Tx

Table 27-19: TWI_TGTCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Enable Target Mode. The `TWI_TGTCTL.EN` enables target operation. Enabling target and controller modes of operation concurrently is allowed. If disabled, no attempt is made to identify a valid address. If `TWI_TGTCTL.EN` is cleared during a valid transfer, clock stretching ceases, the serial data line is released, and the current byte is not acknowledged.	
		0	Disable
		1	Enable

Target Mode Status Register

During and at the conclusion of register target mode transfers, the TWI_TGTSTAT holds information on the current transfer. Generally target mode status bits are not associated with the generation of interrupt requests. Controller mode operation does not affect target mode status bits.

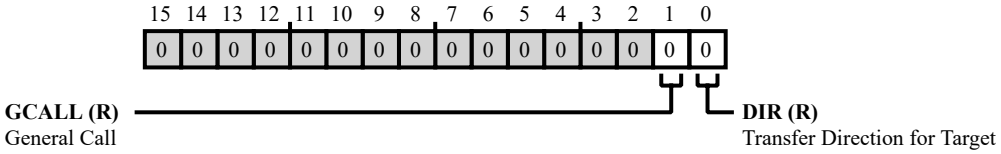


Figure 27-27: TWI_TGTSTAT Register Diagram

Table 27-20: TWI_TGTSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	GCALL	General Call. The TWI_TGTSTAT.GCALL indicates whether or not--at the time of addressing--the address was determined to be a general call. This bit self clears if target mode is disabled (TWI_TGTCTL.EN =0).	
		0	Not a General Call Address
		1	General Call Address
0 (R/NW)	DIR	Transfer Direction for Target. The TWI_TGTSTAT.DIR indicates whether--at the time of addressing--the transfer direction was determined to be target transmit or receive. This bit self clears if target mode is disabled (TWI_TGTCTL.EN =0).	
		0	Target Receive
		1	Target Transmit

Tx Data Double-Byte Register

The TWI_TXDATA16 register holds a 16-bit data value written into the FIFO buffer. To reduce interrupt latency output rates and peripheral bus access times, a double byte transfer data access can be done. Two data bytes can be written, effectively filling the transmit FIFO buffer with a single access.

The data is written in little endian byte order, where byte 0 is the first byte to be transferred and byte 1 is the second byte to be transferred. With each access, the transmit status (TWI_FIFOSTAT.TXSTAT) field is updated. If an access is performed while the FIFO buffer is not empty, the write is ignored and the existing FIFO buffer data and its status remains unchanged. This register when read back returns zero.

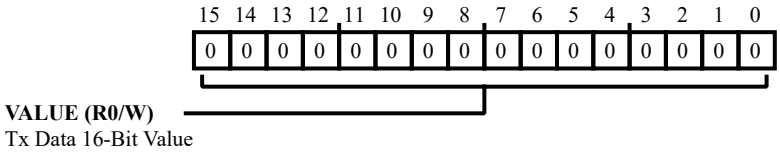


Figure 27-28: TWI_TXDATA16 Register Diagram

Table 27-21: TWI_TXDATA16 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W)	VALUE	Tx Data 16-Bit Value.

Tx Data Single-Byte Register

The TWI_TXDATA8 register holds an 8-bit data value written into the FIFO buffer. Transmit data is entered into the corresponding transmit buffer in a first-in first-out order. For 16-bit peripheral bus writes, a write access to this register adds only one transmit data byte to the FIFO buffer. With each access, the transmit status (TWI_FIFOSTAT.TXSTAT) field is updated. If an access is performed while the FIFO buffer is full, the write is ignored and the existing FIFO buffer data and its status remains unchanged. This register returns zero when read back.

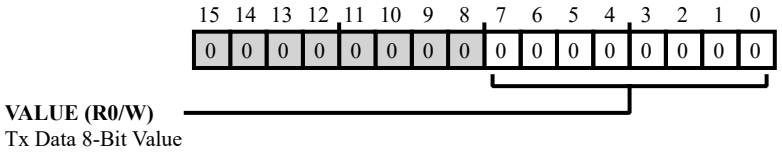


Figure 27-29: TWI_TXDATA8 Register Diagram

Table 27-22: TWI_TXDATA8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	VALUE	Tx Data 8-Bit Value.

28 Ethernet Media Access Controller (EMAC)

The EMAC peripheral in the processor enables network connectivity to applications through an Ethernet interface.

The module is fully compliant to the following standards:

- IEEE 802.3-2015 for Physical Layer Specifications.
- IEEE 1588-2008 for Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Forwarding and Queuing Enhancements for Time-Sensitive Streams or Audio Video (AV) traffic.
- IEEE 802.1AS-Rev/D4.0 for Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE).
- IEEE 802.1Qbv-2015, 802.1Qbu-2016, and 802.1AS-Rev D5.0 for Time-Sensitive Networking (TSN).
- IEEE 802.1Qaz-2011 and 802.1Qbb-2011 for Data Center Bridging (DCB).
- Reduced Gigabit Media Independent Interface (RGMII), Revision 2.6, HP/Marvell.
- Reduced Media Independent Interface Specification, Revision 1.2, RMII Consortium.

NOTE: Copyright © 2020 Synopsys, Inc.; portions of this chapter are included with permission from Synopsys, Inc.

NOTE: References to MII in this document are applicable to both MII and RMII; references to GMII are applicable to RGMII.

EMAC Features

The EMAC features include the following:

MAC Tx and Rx Features

- 10/100/1000 Mbps data transfer rates with external PHY interfaced through RGMII/RMII/MII

- Support for both full-duplex and half-duplex operations
- CSMA/CD protocol for half-duplex operation
- IEEE 802.3x flow control for full-duplex and using backpressure support for half-duplex

MAC Transmit Features

- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets with up to 16 KB of size
- Programmable Inter-Frame Gap (IFG)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted frames with per-frame control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in full-duplex mode
- Insert, replace, or delete queue/channel based VLAN tags
- Frame preemption for MAC Tx

MAC Receive Features

- Automatic pad and CRC stripping options
- Preamble and SFD deletion
- Programmable watchdog timeout limit
- Flexible address filtering modes
- Additional packet filtering (VLAN tag based, layer 3, layer 4)
- IEEE 802.1Q VLAN tag detection
- Frame preemption for MAC Rx

MAC Transaction Layer (MTL) Features

- Eight queues each on the transmit and receive path
- Programmable threshold capability for each queue
- Programmable burst length

MTL Transmit Features

- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC

- Programmable queue size in configurations with multiple queues. Each queue size can be programmed in terms of 256 bytes
- Automatic retransmission of collision packets in half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and underrun conditions with appropriate status
- Option to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum
- Programmable interrupt options for different operational conditions
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Statistics related to bandwidth consumption by each queue of up to 16 blocks over a 125 us period
- Optional packet-level control for:
    - VLAN tag insertion or replacement
    - Ethernet source address insertion
    - Layer 3/layer 4 checksum insertion control
    - One-step time stamp
    - Time stamp control
    - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues:
    - Weighted Round Robin (WRR)
    - (When Data Center Bridging is enabled) Deficit Weighted Round Robin (DWRR)
    - (When Data Center Bridging is enabled) Weighted Fair Queuing (WFQ)
    - Strict Priority (SP)
    - (When Audio-Video Bridging is enabled) Credit-based Shaper (CBS)
    - (When TSN is enabled), Enhancement to Scheduled Traffic (EST)
    - (When TSN is enabled), Time Based Scheduling (TBS)Option to support dropping of Tx status to improve the transmit throughput

MTL Receive Features

- Programmable Rx queue threshold in threshold (or cut through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets

- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of pause packet control or backpressure signal to the MAC based on the Rx queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted Round Robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict Priority (SP)
- Option to replicate received multicast packets for transfer by multiple Rx DMA channel
- Option to have a programmable lookup table based flexible parser for filtering and steering the Rx packets
- Address Recognition Protocol (ARP) offload for IPv4 packets

DMA Block Features

- Multichannel transmit and receive engines (eight transmit and eight receive channels)
- Separate DMA channel in the transmit path for each queue in MFL
- Single or multiple DMA channels for any number of queues in MFL receive path
- Dual-buffer (ring) descriptor support
- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst length for Tx DMA and Rx DMA engines
- Programmable interrupt options for different operational conditions
- Per-packet transmit or receive complete interrupt control
- Round-robin or fixed-priority arbitration between the receive and transmit engines
- Fixed-priority, weighted-strict-priority or weighted-round-robin arbitration for data read requests from multiple transmit DMA engines
- Start and stop modes
- Support for TCP Segmentation Offload (TSO) and UDP Fragmentation Offload (UFO) with selectable number of Tx DMA channels
- Routing of received packets to the DMA channels based on the DA or VLAN priority in multichannel DMA configurations
- Option to split the packet header (layer 3 and layer 4) and payload in different buffers
- Time-sensitive conditional packet fetching from system memory by comparing the slot time or IEEE 1588 time information provided in the descriptor (useful for AV applications)

- Programmable control for transmit descriptor posted writes to improve the throughput
- Separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes
- Up to eight queues on the receive paths for AV traffic and seven queues on the transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for transmit channels
- Single Tx FIFO and Rx FIFO (MFL) for all selected queues
- Programmable slot interval with range from 1µs to 4096 µs and granularity of 1µs

Data Center Bridging Features

- Separate channels or queues for DCB data transfer in 100 Mbps and 1000 Mbps modes
- Up to eight queues on the transmit and receive paths for DCB traffic
- The following Tx queues scheduling mechanisms meet the compliance specification in IEEE 802.1-Qaz Enhanced Transmission Selection (ETS) algorithm:
  - Weighted Round Robin (WRR)
  - Deficit Weighted Round Robin (DWRR)
  - Weighted Fair Queuing (WFQ)
  - Strict Priority (SP)
- Common memory for all selected Tx or Rx queues
- Priority-based Flow Control (PFC) on individual queues configured for DCB
- Programmable control to route received VLAN tagged packets to channels or queues

Time Sensitive Networking Features

- IEEE 802.1Qbv-2015, Enhancements to Scheduling Traffic
- IEEE802.1Qbu/802.3br, Frame preemption and Interspersing Express Traffic

Precision Time Protocol (PTP) Features

- Ethernet frame time stamping as described in IEEE 1588–2002 and IEEE 1588–2008. The transmit or receive status of each frame include 64-bit time stamps
- Hardware assisted time stamping capable of up to 8 ns resolution
- Automatic detection of PTP messages through Ethernet, IPv4, and IPv6 packets
- Four programmable PPS outputs that physically represent PTP system time
- Auxiliary snapshot to time stamp external events - four auxiliary input pins available

- PTP time stamp offload support
- Support for one-step time stamp

Table 28-1: EMAC0 Features

Feature	EMAC0
Speed	10/100/1000 Mbps
Line Interface	MII, RMII, RGMII
Double VLAN Processing	Yes
Queue/Channel-based VLAN Tag Insertion on Tx	Yes
Additional MAC Address Registers	31
Address Filter Hash Table Size	256 bits
Extended RX VLAN Tag-based Filters	4
Layer 3 and Layer 4 Filters	8
Multicast Packet Duplication Support	Yes
Number of Tx/Rx Queues and DMA Channels	8
Data Center Bridging (DCB)	Yes
Audio Video Bridging (AVB)	Yes
Additional AVB DMA channels	7
Precision Time Protocol (PTP)	Yes
PTP - Number of Auxiliary snapshots	4 with 16 deep FIFO
PTP - Timestamp Offloading (Auto Send PTP frames)	Yes
PTP - One Step Timestamp Feature	Yes
TSN-EST (Enhancements to Scheduling Traffic) (802.1Qbv)	Yes
Frame Preemption Support (802.3br, 802.1Qbu)	Yes
Time Based Scheduling (launch time)	Yes
TCP Segmentation Offloading for TCP/IP Packets	Yes
Split Header Feature	Yes
IPv4 ARP Offload	Yes
Energy Efficient Ethernet (EEE)	Yes

EMAC Functional Description

This section provides information on the function of Ethernet MAC peripheral.

Hardware for the Media Access Control protocol

This function allows applications to support TCP/IP based network communication. At the system end, the module supports direct connection with the system crossbar bus for memory or MMR transactions. It supports RMII (Reduced Media Independent Interface), RGMII (Reduced Gigabit Media Independent Interface), MII (Media Independent Interface), and SMI (Station Management Interface) for interfacing with the external PHY chip.

Dedicated DMA Controller with independent read/write channels

Performs both data and status transfers between the application and the media independent interfaces. Internal transmit and receive FIFOs are used to buffer and regulate the frames. Dedicated interrupt lines connect the EMAC interrupt sources to the System Event Controller (SEC) for the SHARC core or Generic Interrupt Controller (GIC) for ARM core.

MAC Management Counters (MMC) block

An extended set of registers that collect various statistics compliant with IEEE 802.3 definitions regarding the operation of the interface. The registers are updated for each new transmitted or received frame when the condition to update the counter is met. The EMAC provides a set of such counters, along with extended usage control.

PTP (Precision Time Protocol) engine

Provides hardware assistance for the implementation of the IEEE 1588 version 1 and version 2 standards, which allows time synchronization between systems.

Audio Video (AV) functionality

Enables transmission of time-sensitive traffic over bridged local area networks (LANs). The EMAC provides hardware support for IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm. In addition, slot number function allows scheduling of fetching of data by DMA from system memory.

Data Center Bridging (DCB)

Data center bridging (DCB) is a set of enhancements to the Ethernet local area network communication protocol for use in data center environments, for use with clustering and storage area networks. DCB aims, for selected traffic, to eliminate loss due to queue overflow (sometimes called lossless Ethernet) and to be able to allocate bandwidth on links. DCB enables, to some extent, the treatment of different priorities as if they were different pipes.

Time Sensitive Networking

Time sensitive network (TSN) is a set of standards developed by the IEEE 802.1 work group. The standards define mechanisms for the time-sensitive transmission of data over Ethernet.

ADSP-2184x EMAC Register List

RGMII, RMII, MII Ethernet Controller (EMAC)

Table 28-2: ADSP-2184x EMAC Register List

Name	Description
EMAC_DMA[n]_CTL	DMA Channel n Control Register
EMAC_DMA[n]_RXBUF_CUR	DMA Channel n Current Application Receive Buffer Address Register
EMAC_DMA[n]_RXDSC_CUR	DMA Channel n Current Application Receive Descriptor Register
EMAC_DMA[n]_TXBUF_CUR	DMA Channel n Current Application Transmit Buffer Address Register
EMAC_DMA[n]_TXDSC_CUR	DMA Channel n Current Application Transmit Descriptor Register
EMAC_DMA[n]_IEN	DMA Channel n Interrupt Enable Register
EMAC_DMA[n]_MISSFRM_CNT	DMA Channel n Missed Frame Counter Register
EMAC_DMA[n]_RXDSC_ADDR	DMA Channel n Receive Descriptor List Address Register
EMAC_DMA[n]_RXDSC_TLPTR	DMA Channel n Receive Descriptor Tail Pointer Register
EMAC_DMA[n]_RXCTL	DMA Channel n Receive Control Register
EMAC_DMA[n]_RXCTL2	DMA Channel n Extended Receive Control Register
EMAC_DMA[n]_RXINTWDTMR	DMA Channel n Receive Interrupt Watchdog Timer Register
EMAC_DMA[n]_SFN_CTLSTAT	DMA Channel n Slot Function Control and Status register
EMAC_DMA[n]_STAT	DMA Channel n Status Register
EMAC_DMA[n]_TXDSC_ADDR	DMA Channel n Transmit Descriptor List Address Register
EMAC_DMA[n]_TXDSC_RLEN	DMA Channel n Transmit Descriptor Ring Length Register
EMAC_DMA[n]_TXDSC_TLPTR	DMA Channel n Transmit Descriptor Tail Pointer Register
EMAC_DMA[n]_TXCTL	DMA Channel n Transmit Control Register
EMAC_DMA_DBG_STAT0	DMA Debug Status 0 Register
EMAC_DMA_DBG_STAT1	DMA Debug Status 1 Register
EMAC_DMA_DBG_STAT2	DMA Debug Status 2 Register
EMAC_DMA_ISTAT	DMA Interrupt Status Register
EMAC_DMA_MODE	DMA Mode Register
EMAC_DMA_SYSBMODE	DMA System Bus Mode Register
EMAC_DMA_TBS_CTL0	DMA TBS Control 0 Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_DMA_TBS_CTL1	DMA TBS Control 1 Register
EMAC_DMA_TBS_CTL2	DMA TBS Control 2 Register
EMAC_DMA_TBS_CTL3	DMA TBS Control 3 Register
EMAC_ONEUS_TIC_CNT	LPI One Microsecond Tic Counter Register
EMAC_ARP_ADDR	Address Resolution Protocol (ARP) Register
EMAC_ADDR0_HI	MAC Address 0 High Register
EMAC_ADDR0_LO	MAC Address 0 Low Register
EMAC_ADDR[i]_HI	MAC Address n High Register
EMAC_ADDR[i]_LO	MAC Address n Low Register
EMAC_TM_AUX_CTL	Auxiliary Time Stamp Control Register
EMAC_TM_AUXTS_NSEC	Auxiliary Time Stamp Nanoseconds Register
EMAC_TM_AUXTS_SEC	Auxiliary Time Stamp Seconds Register
EMAC_CSR_SW_CTL	CSR Software Control Register
EMAC_MAC_CFG	MAC Configuration Register
EMAC_MAC_DBG	Debug Register
EMAC_SPLM_OFST_CFG	MAC Split Mode and Offset Configuration Register
EMAC_MAC_EXT_CFG	MAC Extended Configuration Register
EMAC_FPE_CTLSTAT	Frame Preemption Control and Status Register
EMAC_HW_FTR_PSNT0	Hardware Feature Presence Register 0
EMAC_HW_FTR_PSNT1	Hardware Feature Presence Register 1
EMAC_HW_FTR_PSNT2	Hardware Feature Presence Register 2
EMAC_HW_FTR_PSNT3	Hardware Feature Presence Register 3
EMAC_MAC_HW_FEATURE4	MAC Hardware Feature Presence Register 4
EMAC_HASHTBL_REG0	Hash Table Register 0
EMAC_HASHTBL_REG1	Hash Table Register 1
EMAC_HASHTBL_REG2	Hash Table Register 2
EMAC_HASHTBL_REG3	Hash Table Register 3
EMAC_HASHTBL_REG4	Hash Table Register 4
EMAC_HASHTBL_REG5	Hash Table Register 5
EMAC_HASHTBL_REG6	Hash Table Register 6
EMAC_HASHTBL_REG7	Hash Table Register 7

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_INDR_ACC_CTL	Indirect Access Control Register
EMAC_INDR_ACC_DAT	Indirect Access Data Register
EMAC_INVLAN_INCL	Inner VLAN Tag Inclusion or Replacement Register
EMAC_MAC_IEN	MAC Interrupt Enable Register
EMAC_MAC_ISTAT	MAC Interrupt Status Register
EMAC_L3L4_CTL7	Layer 3 and Layer 4 Control Register
EMAC_L3L4_CTL[i]	Layer 3 and Layer 4 Control Register
EMAC_LPI_CTLSTAT	LPI Control and Status Register
EMAC_LPI_ENTR_TMR	Transmit LPI Entry Timer Register
EMAC_LPI_TMRSCTL	LPI Timers Control Register
EMAC_L3_ADDR0_REG7	Layer 3 Address 0 Register 7
EMAC_L3_ADDR0_REG[i]	Layer 3 Address 0 Register [i]
EMAC_L3_ADDR1_REG7	Layer 3 Address 1 Register 7
EMAC_L3_ADDR1_REG[i]	Layer 3 Address 1 Register [i]
EMAC_L3_ADDR2_REG7	Layer 3 Address 2 Register 7
EMAC_L3_ADDR2_REG[i]	Layer 3 Address 2 Register [i]
EMAC_L3_ADDR3_REG7	Layer 3 Address 3 Register 7
EMAC_L3_ADDR3_REG[i]	Layer 3 Address 3 Register [i]
EMAC_L4_ADDR7	Layer 4 Address Register [i]
EMAC_L4_ADDR[i]	Layer 4 Address Register [i]
EMAC_LOG_MSG_INTVL	PTP Log Message Interval Register
EMAC_MDIO_ADDR	MDIO Address Register
EMAC_MDIO_DATA	MDIO Data Register
EMAC_PHYIF_CTLSTAT	PHY Interface Control and Status Register
EMAC_PPS[n]_INTVL	PPS n Interval Register
EMAC_PPS[n]_TGTM_NSEC	PPS n Target Time Nanoseconds Register
EMAC_PPS[n]_TGTM_SEC	PPS n Target Time Seconds Register
EMAC_PPS[n]_WID	PPS n Width Register
EMAC_PPS_CTL	PPS Control Register
EMAC_PTO_CTL	PTP Time Stamp Offload Control Register
EMAC_MACPKT_FILT	MAC Packet Filter Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_PRSNTM_UPDT	Presentation Time Update Register
EMAC_PRSNTM_NSEC	Presentation Time Nanosecond Register
EMAC_Q0_TXFLOW_CTL	Queue 0 Transmit Flow Control Register
EMAC_Q[i]_TXFLOW_CTL	Queue [i] Transmit Flow Control Register
EMAC_RXQ_CTL0	Receive Queue Control 0 Register
EMAC_RXQ_CTL1	Receive Queue Control 1 Register
EMAC_RXQ_CTL2	Receive Queue Control 2 Register
EMAC_RXQ_CTL3	Receive Queue Control 3 Register
EMAC_RXQ_CTL4	Receive Queue Control 4 Register
EMAC_MAC_RX_DOMAIN_TIME_INCR	MAC Receive Time Domain Increment Register
EMAC_RXFLOW_CTL	Receive Flow Control Register
EMAC_RXTX_STAT	Receive Transmit Status Register
EMAC_SRCPRT_IDNTY0	Source Port Identity 0 Register
EMAC_SRCPRT_IDNTY1	Source Port Identity 1 Register
EMAC_SRCPRT_IDNTY2	Source Port Identity 2 Register
EMAC_TM_SUBSEC	Subsecond Increment Register
EMAC_TM_HISEC	System Time Higher Word Seconds Register
EMAC_TM_NSEC	System Time Nanoseconds Register
EMAC_TM_NSECUPDT	System Time Nanoseconds Update Register
EMAC_TM_SEC	System Time Seconds Register
EMAC_TM_SECUPDT	System Time Seconds Update Register
EMAC_TM_ADDEND	Time Stamp Addend Register
EMAC_TM_CTL	Time Stamp Control Register
EMAC_TM_EGASYM_CORR	Time Stamp Egress Asymmetry Correction Register
EMAC_TM_EGCORR_NSEC	Time Stamp Egress Correction Nanosecond Register
EMAC_TM_EGCORR_SNSEC	Time Stamp Egress Correction Sub Nanosecond Register
EMAC_TM_EG_LTNCY	Time Stamp Egress Latency Register
EMAC_TM_INGASYM_CORR	Time Stamp Ingress Asymmetry Correction Register
EMAC_TM_INGCORR_NSEC	Time Stamp Ingress Correction Nanosecond Register
EMAC_TM_INGCORR_SNSEC	Time Stamp Ingress Correction Sub Nanosecond Register
EMAC_TM_ING_LTNCY	Time Stamp Ingress Latency Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_TM_STAT	Time Stamp Status Register
EMAC_TXQ_PRTY_MAP0	Transmit Queue Priority Mapping 0 Register
EMAC_TXQ_PRTY_MAP1	Transmit Queue Priority Mapping 1 Register
EMAC_MAC_TX_DOMAIN_TIME_INCR	MAC Transmit Time Domain Increment Register
EMAC_TXTMSTAT_NSEC	Transmit Time Stamp Status Nanoseconds Register
EMAC_TXTMSTAT_SEC	Transmit Time Stamp Status Seconds Register
EMAC_VLAN_HASHTBL	VLAN Hash Table Register
EMAC_VLAN_INCL	VLAN Tag Inclusion or Replacement Register
EMAC_VLANTAG_CTL	VLAN Tag Control Register
EMAC_VLANTAG_DAT	VLAN Tag Indirect Access Data Register
EMAC_WDT_JB_TMOUT	Watchdog and Jabber Timeout Register
EMAC_MMC_CTRL	MMC Control Register
EMAC_MMC_FPE_RXINT	MMC Receive FPE Interrupt Status Register
EMAC_MMC_FPE_RXIMSK	MMC Receive FPE Interrupt Mask Register
EMAC_MMC_FPE_TXINT	MMC Transmit FPE Interrupt Status Register
EMAC_MMC_FPE_TXIMSK	MMC Transmit FPE Interrupt Mask Register
EMAC_MMC_IPC_RXINT	MMC Receive IPC Interrupt Status Register
EMAC_MMC_IPC_RXIMSK	MMC Receive IPC Interrupt Mask Register
EMAC_RXFPE_FRGMNT_CNT	MMC Receive FPE Fragment Counter Register
EMAC_MMC_RXINT	MMC Receive Interrupt Status Register
EMAC_MMC_RXIMSK	MMC Receive Interrupt Mask Register
EMAC_RXPKT_ASERR_CNT	MMC Receive FPE Packet Assembly Error Counter Register
EMAC_RXPKT_ASOK_CNT	MMC Receive FPE Packet Assembly Okay Counter Register
EMAC_RXPKT_SMDERR_CNT	MMC Receive FPE Packet SMD Error Counter Register
EMAC_TXFPE_FRGMNT_CNT	MMC Transmit FPE Packet Fragment Counter Register
EMAC_TXHLDRQ_CNT	MMC Transmit FPE Hold Request Counter Register
EMAC_MMC_TXINT	MMC Transmit Interrupt Status Register
EMAC_MMC_TXIMSK	MMC Transmit Interrupt Mask Register
EMAC_EST_CTL	EST Control Register
EMAC_EST_EXT_CTL	EST Extended Control Register
EMAC_EST_FRMSZ_CAP	EST Frame Size Capture Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_EST_FRMSZ_ERR	EST Frame Size Error Register
EMAC_EST_GCL_CTL	EST Gate Control List Control Register
EMAC_EST_GCL_DAT	EST Gate Control List Data Register
EMAC_EST_IEN	EST Interrupt Enable Register
EMAC_EST_SCH_ERR	EST Scheduling Error Register
EMAC_EST_STAT	EST Status Register
EMAC_FPE_ADV	Frame Preemption Advance Register
EMAC_FPE_CTRLSTS	MTL Frame Preemption Control and Status Register
EMAC_MTL_ISTAT	MTL Interrupt Status Register
EMAC_MTL_OPMODE	MTL Operation Mode Register
EMAC_Q0_INT_CTLSTAT	Queue 0 Interrupt Control and Status Register
EMAC_Q[n]_INT_CTLSTAT	Queue n Interrupt Control and Status Register
EMAC_RQ0_CTL	Receive Queue Control Register
EMAC_RQ0_DBG	Receive Queue 0 Debug Register
EMAC_RQ0_MSPKTOF_CNT	Receive Queue 0 Missed Packet and Overflow Counter Register
EMAC_RQ0_OPMODE	Receive Queue 0 Operation Mode Register
EMAC_RQ[n]_CTL	Receive Queue n Control Register
EMAC_RQ[n]_DBG	Receive Queue n Debug Register
EMAC_RQ[n]_MSPKTOF_CNT	Receive Queue n Missed Packet and Overflow Counter Register
EMAC_RQ[n]_OPMODE	Receive Queue n Operation Mode Register
EMAC_RXQ_DMA_MAP0	Receive Queue to DMA Channel Mapping Register 0
EMAC_RXQ_DMA_MAP1	Receive Queue to DMA Channel Mapping Register 1
EMAC_TBS_CTRL	MTL Time Based Scheduling Control Register
EMAC_TQ0_DBG	Transmit Queue 0 Debug Register
EMAC_TQ0_ETS_STAT	Transmit Queue 0 ETS Status Register
EMAC_TQ0_OPMODE	Transmit Queue 0 Operation Mode Register
EMAC_TQ0_QTMWGT	Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ0_URFL	Transmit Queue 0 Underflow Packet Counter Register
EMAC_TQ[n]_DBG	Transmit Queue n Debug Register
EMAC_TQ[n]_ETS_CTL	Transmit Queue n ETS Control Register
EMAC_TQ[n]_ETS_STAT	Transmit Queue n ETS Status Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_TQ[n]_HICRDT	Transmit Queue n CBS HiCredit Register
EMAC_TQ[n]_LOCRDT	Transmit Queue n CBS LoCredit Register
EMAC_TQ[n]_OPMODE	Transmit Queue n Operation Mode Register
EMAC_TQ[n]_QTMWGT	Transmit Queue n IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ[n]_SSCRDT	Transmit Queue n SendSlopeCredit Register
EMAC_TQ[n]_URFL	Transmit Queue n Underflow Packet Counter Register
EMAC_RXICMP_ERR_OCNT	Receive ICMP Error Octets Counter Register
EMAC_RXICMP_ERR_PCNT	Receive ICMP Error Packets Counter Register
EMAC_RXICMP_OCNT_G	Receive ICMP Good Octets Counter Register
EMAC_RXICMP_PCNT_G	Receive ICMP Good Packets Counter Register
EMAC_RXIPV4_FRGMT_OCNT	Receive IPv4 Fragmented Octets Counter Register
EMAC_RXIPV4_FRGMT_PCNT	Receive IPv4 Fragmented Packets Counter Register
EMAC_RXIPV4_OCNT_G	Receive IPv4 Good Octets Counter Register
EMAC_RXIPV4_PCNT_G	Receive IPv4 Good Packets Counter Register
EMAC_RXIPV4_HDRERR_OCNT	Receive IPv4 Header Error Octets Counter Register
EMAC_RXIPV4_HDRERR_PCNT	Receive IPv4 Header Error Packets Counter Register
EMAC_RXIPV4_PYLD_OCNT	Receive IPv4 Payload Octets Counter Register
EMAC_RXIPV4_PYLD_PCNT	Receive IPv4 Payload Packets Counter Register
EMAC_RXIPV4_UDPCSD_OCNT	Receive IPv4 UDP Checksum Disable Octets Counter Register
EMAC_RXIPV4_UDPCSD_PCNT	Receive IPv4 UDP Checksum Disable Packets Counter Register
EMAC_RXIPV6_OCNT_G	Receive IPv6 Good Octets Counter Register
EMAC_RXIPV6_PCNT_G	Receive IPv6 Good Packets Counter Register
EMAC_RXIPV6_HDRERR_OCNT	Receive IPv6 Header Error Octets Counter Register
EMAC_RXIPV6_HDRERR_PCNT	Receive IPv6 Header Error Packets Counter Register
EMAC_RXIPV6_PYLD_OCNT	Receive IPv6 Payload Octets Counter Register
EMAC_RXIPV6_PYLD_PCNT	Receive IPv6 Payload Packets Counter Register
EMAC_RXTCP_ERR_OCNT	Receive TCP Error Octets Counter Register
EMAC_RXTCP_ERR_PCNT	Receive TCP Error Packets Counter Register
EMAC_RXTCP_OCNT_G	Receive TCP Good Octets Counter Register
EMAC_RXTCP_PCNT_G	Receive TCP Good Packets Counter Register
EMAC_RXUDP_ERR_OCNT	Receive UDP Error Octets Counter Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_RXUDP_ERR_PCNT	Receive UDP Error Packets Counter Register
EMAC_RXUDP_OCNT_G	Receive UDP Good Octets Counter Register
EMAC_RXUDP_PCNT_G	Receive UDP Good Packets Counter Register
EMAC_RX1024TOMAX_GB	Receive 1024 - Maximum Octets Packets Good Bad Counter Register
EMAC_RX128TO255_GB	Receive 128 - 255 Octets Packets Good Bad Counter Register
EMAC_RX256TO511_GB	Receive 256 - 511 Octets Packets Good Bad Counter Register
EMAC_RX512TO1023_GB	Receive 512 - 1023 Octets Packets Good Bad Counter Register
EMAC_RX64_GB	Receive 64 Octets Packets Good Bad Counter Register
EMAC_RX65TO127_GB	Receive 65 - 127 Octets Packets Good Bad Counter Register
EMAC_RX_ALGNERR_PCNT	Receive Alignment Error Packets Counter Register
EMAC_RX_BCERR_PCNT_G	Receive Broadcast Packets Good Counter Register
EMAC_RX_CRCERR_PCNT	Receive CRC Error Packets Counter Register
EMAC_RX_CTL_PCNT_G	Receive Control Packets Good Counter Register
EMAC_RX_FIFOOVFL_PCNT	Receive FIFO Overflow Packets Counter Register
EMAC_RX_JBERR_PCNT	Receive Jabber Error Packets Counter Register
EMAC_RX_LPITRAN_CNT	Receive LPI Transition Counter Register
EMAC_RX_LPIUSEC_CNT	Receive LPI Microseconds Counter Register
EMAC_RX_LENERR_PCNT	Receive Length Error Packets Counter Register
EMAC_RX_MCST_PCNT_G	Receive Multicast Packets Good Counter Register
EMAC_RX_OCNT_G	Receive Octet Good Counter Register
EMAC_RX_OCNT_GB	Receive Octet Good Bad Counter Register
EMAC_RX_ORTYPE_PCNT	Receive Out of Range Type Packets Counter Register
EMAC_RX_OVRSZ_PCNT	Receive Oversize Packets Good Counter Register
EMAC_RX_PCNT_GB	Receive Packets Good Bad Counter Register
EMAC_RX_PAUSE_PCNT	Receive Pause Packets Counter Register
EMAC_RX_ERR_PCNT	Receive Error Packets Counter Register
EMAC_RX_RNTERR_PCNT	Receive Runt Error Packets Counter Register
EMAC_RX_UNDRSZ_PCNT_G	Receive Undersize Packets Good Counter Register
EMAC_RX_UCST_PCNT_G	Receive Unicast Packets Good Counter Register
EMAC_RX_VLAN_PCNT_GB	Receive VLAN Packets Good Bad Counter Register
EMAC_RX_WDTERR_PCNT	Receive Watchdog Error Packets Counter Register

Table 28-2: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_TX1024TOMAX_GB	Transmit 1024-Max Octets Packets Good Bad Counter Register
EMAC_TX128TO255_GB	Transmit 128-255 Octets Packets Good Bad Counter Register
EMAC_TX256TO511_GB	Transmit 256-511 Octets Packets Good Bad Counter Register
EMAC_TX512TO1023_GB	Transmit 512-1023 Octets Packets Good Bad Counter Register
EMAC_TX64_GB	Transmit 64 Octets Packets Good Bad Counter Register
EMAC_TX65TO127_GB	Transmit 65-127 Octets Packets Good Bad Counter Register
EMAC_TX_BCST_PCNT_G	Transmit Broadcast Packets Good Counter Register
EMAC_TX_BCST_PCNT_GB	Transmit Broadcast Packets Good Bad Counter Register
EMAC_TX_CRERR_PCNT	Transmit Carrier Error Packets Counter Register
EMAC_TX_DFRD_PCNT	Transmit Deferred Packets Counter Register
EMAC_TX_EXCOL_PCNT	Transmit Excessive Collision Packets Counter Register
EMAC_TX_EXDFRL_PCNT	Transmit Excessive Deferral Error Counter Register
EMAC_TX_LPITRAN_CNT	Transmit LPI Transition Counter Register
EMAC_TX_LPIUSEC_CNT	Transmit LPI Microseconds Counter Register
EMAC_TX_LTCOL_PCNT	Transmit Late Collision Packets Counter Register
EMAC_TX_MCST_PCNT_G	Transmit Multicast Packets Good Counter Register
EMAC_TX_MCST_PCNT_GB	Transmit Multicast Packets Good Bad Counter Register
EMAC_TX_MCOL_PCNT_G	Transmit Multiple Collision Packets Good Counter Register
EMAC_TX_OSIZE_PCNT_G	Transmit Osize Packets Good Counter Register
EMAC_TX_OCNT_G	Transmit Octet Good Counter Register
EMAC_TX_OCNT_GB	Transmit Octet Good Bad Counter Register
EMAC_TX_PCNT_G	Transmit Packet Good Counter Register
EMAC_TX_PCNT_GB	Transmit Packet Good Bad Counter Register
EMAC_TX_PAUSE_PCNT	Transmit Pause Packets Counter Register
EMAC_TX_SNGLCOL_PCNT_G	Transmit Single Collision Packets Good Counter Register
EMAC_TX_URFL_PCNT	Transmit Underflow Error Packets Counter Register
EMAC_TX_UCST_PCNT_GB	Transmit Unicast Packets Good Bad Counter Register
EMAC_TX_VLAN_PCNT_G	Transmit VLAN Packets Good Counter Register

ADSP-2184x EMAC Interrupt List

Table 28-3: ADSP-2184x EMAC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
68	EMAC0_DMA2_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
70	EMAC0_DMA3_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
71	EMAC0_DMA4_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
72	EMAC0_DMA5_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
73	EMAC0_DMA6_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
74	EMAC0_DMA7_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
75	EMAC0_DMA0_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
82	EMAC0_DMA1_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
84	EMAC0_DMA3_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
86	EMAC0_DMA4_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
87	EMAC0_DMA5_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
88	EMAC0_DMA6_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
89	EMAC0_DMA7_RX	EMAC0 DMAn Rx Channel Interrupt	Edge	
94	EMAC0_STAT	EMAC0 Status	Edge	
95	EMAC0_PWR	EMAC0 Power Interrupt	Edge	
96	EMAC0_DMA0_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
97	EMAC0_DMA1_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
98	EMAC0_DMA2_TX	EMAC0 DMAn Tx Channel Interrupt	Edge	
99	EMAC0_MAC	EMAC0 MAC Interrupt	Edge	

ADSP-2184x EMAC Trigger List

Table 28-4: ADSP-2184x EMAC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
27	`EMAC0_MCGR_DMA_REQ0`	EMAC0 MCGR DMA request	Edge
28	`EMAC0_MCGR_DMA_REQ1`	EMAC0 MCGR DMA request	Edge
29	`EMAC0_MCGR_DMA_REQ2`	EMAC0 MCGR DMA request	Edge
30	`EMAC0_MCGR_DMA_REQ3`	EMAC0 MCGR DMA request	Edge
31	`EMAC0_DMA0_RX`	EMAC0 DMAn Rx Channel Interrupt	None
32	`EMAC0_DMA1_RX`	EMAC0 DMAn Rx Channel Interrupt	None
33	`EMAC0_DMA2_RX`	EMAC0 DMAn Rx Channel Interrupt	None

Table 28-4: ADSP-2184x EMAC Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
34	EMAC0_DMA3_RX	EMAC0 DMAn Rx Channel Interrupt	None
35	EMAC0_DMA4_RX	EMAC0 DMAn Rx Channel Interrupt	None
36	EMAC0_DMA5_RX	EMAC0 DMAn Rx Channel Interrupt	None
37	EMAC0_DMA6_RX	EMAC0 DMAn Rx Channel Interrupt	None
38	EMAC0_DMA7_RX	EMAC0 DMAn Rx Channel Interrupt	None

Table 28-5: ADSP-2184x EMAC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
26	EMAC0_DMA0_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
27	EMAC0_DMA1_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
28	EMAC0_DMA2_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
29	EMAC0_DMA3_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
30	EMAC0_DMA4_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
31	EMAC0_DMA5_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
32	EMAC0_DMA6_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
33	EMAC0_DMA7_RX_Start	EMAC0 DMAn Rx Channel Start Trigger	Pulse
34	EMAC0_DMA0_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
35	EMAC0_DMA1_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
36	EMAC0_DMA2_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
37	EMAC0_DMA3_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
38	EMAC0_DMA4_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
39	EMAC0_DMA5_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
40	EMAC0_DMA6_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
41	EMAC0_DMA7_TX_Start	EMAC0 DMAn Tx Channel Start Trigger	Pulse
42	EMAC0_DMA0_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
43	EMAC0_DMA1_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
44	EMAC0_DMA2_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
45	EMAC0_DMA3_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
46	EMAC0_DMA4_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
47	EMAC0_DMA5_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
48	EMAC0_DMA6_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse

Table 28-5: ADSP-2184x EMAC Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
49	EMAC0_DMA7_RX_STOP	EMAC0 DMAn Rx Channel Stop Trigger	Pulse
50	EMAC0_DMA0_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
51	EMAC0_DMA1_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
52	EMAC0_DMA2_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
53	EMAC0_DMA3_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
54	EMAC0_DMA4_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
55	EMAC0_DMA5_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
56	EMAC0_DMA6_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
57	EMAC0_DMA7_TX_STOP	EMAC0 DMAn Tx Channel Stop Trigger	Pulse
58	EMAC0_MCGR_DMA_ACK0	EMAC0 MCGR DMA acknowledge	Pulse
59	EMAC0_MCGR_DMA_ACK1	EMAC0 MCGR DMA acknowledge	Pulse
60	EMAC0_MCGR_DMA_ACK2	EMAC0 MCGR DMA acknowledge	Pulse
61	EMAC0_MCGR_DMA_ACK3	EMAC0 MCGR DMA acknowledge	Pulse

EMAC Definitions

The following definitions are helpful for using the EMAC.

ARI

Application Receive Interface

ATI

Application Transmit Interface

CPT

Current Presentation Time

DA

Destination Address

DCB

Data Center Bridging

EMAC CORE

CORE layer inside EMAC which performs the actual Ethernet operations, including interface with PHY through the reduced media interface(s).

EMAC DMA

DMA Controller of EMAC

EMAC MTL

MAC Transaction Layer inside EMAC

EMAC SCB

System Crossbar Interface of EMAC

EOP

End of Packet

EST

Enhancement to Scheduled Traffic

ETS

Enhanced Transmission Selection

FPE

Frame Preemption

ICMP

Internet Message Control Protocol

MAC

Media Access Control

MII

Media Independent Interface

MMC

MAC Management Counter

OST

One Step Time Stamp

PTO

PTP Time Stamp Offload

PTP

Precision Time Protocol

LPI

Low Power Interface

MDIO

Management Data Input/Output

RGMII

Reduced Gigabit Media Independent Interface

RMII

Reduced Media Independent Interface

SA

Source Address

SMI

Station Management Interface that controls PHY through MDIO and MDC signals.

SOP

Start Of Packet

TBS

Time Based Scheduling

TPT

Target Presentation Time

TSN

Time Sensitive Networking

VLAN

Virtual LAN

EMAC Block Diagram and Interfaces

The ***EMAC Simplified Block Diagram*** illustrates the overall functional architecture of the Ethernet MAC peripheral. The EMAC module is comprised of four major layers: EMAC SCB, EMAC DMA, EMAC MTL, and EMAC CORE. Each of these layers (sub blocks) is explained in depth in their respective sections in this chapter.

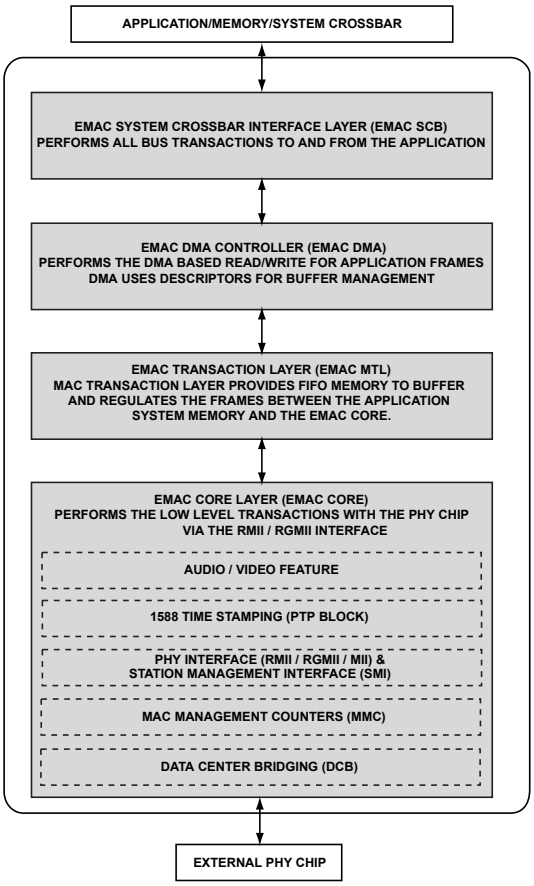


Figure 28-1: EMAC Simplified Block Diagram

A more comprehensive block diagram is shown in the ***EMAC Complete Block Diagram***. It includes most of the important blocks inside the EMAC. The EMAC is connected to processor memory and the system crossbar through the System Crossbar Bus Interface (SCB) and System Peripheral Bus Interface (SPB). These connections are which are part of the SCB layer. The SPB interface is connected to all modules that require MMR programming.

The DMA controller performs application data transfer frame by frame, through well-defined descriptor structures. A FIFO layer acts as a buffer between the DMA controller and EMAC CORE.

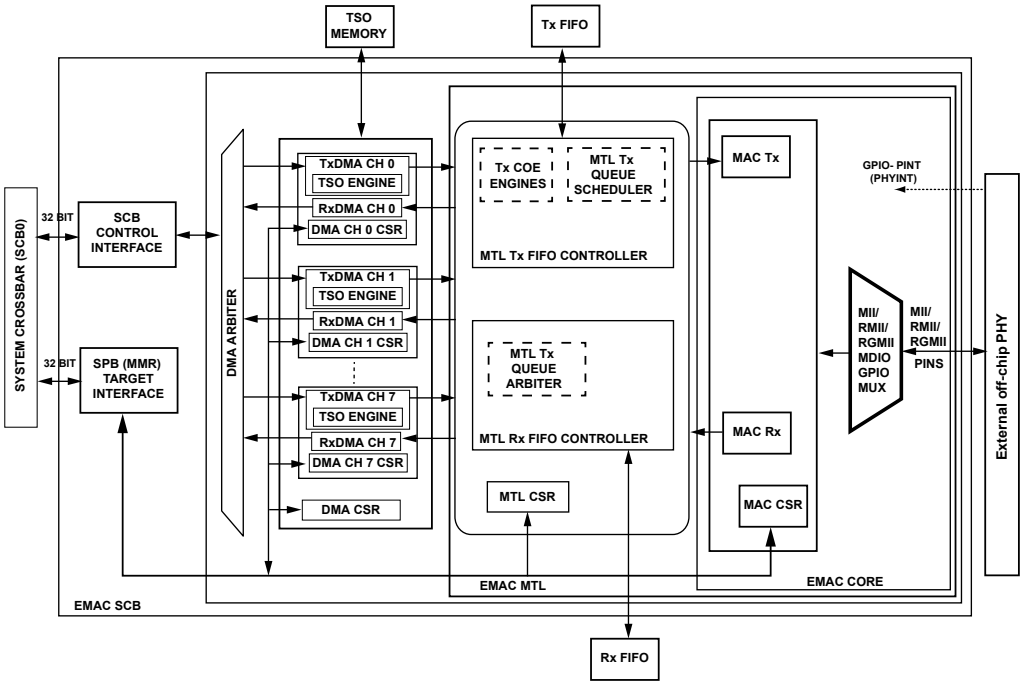


Figure 28-2: EMAC Complete Block Diagram

The *EMAC System Level Block Diagram* shows how the EMAC module interacts with rest of the system inside the processor.

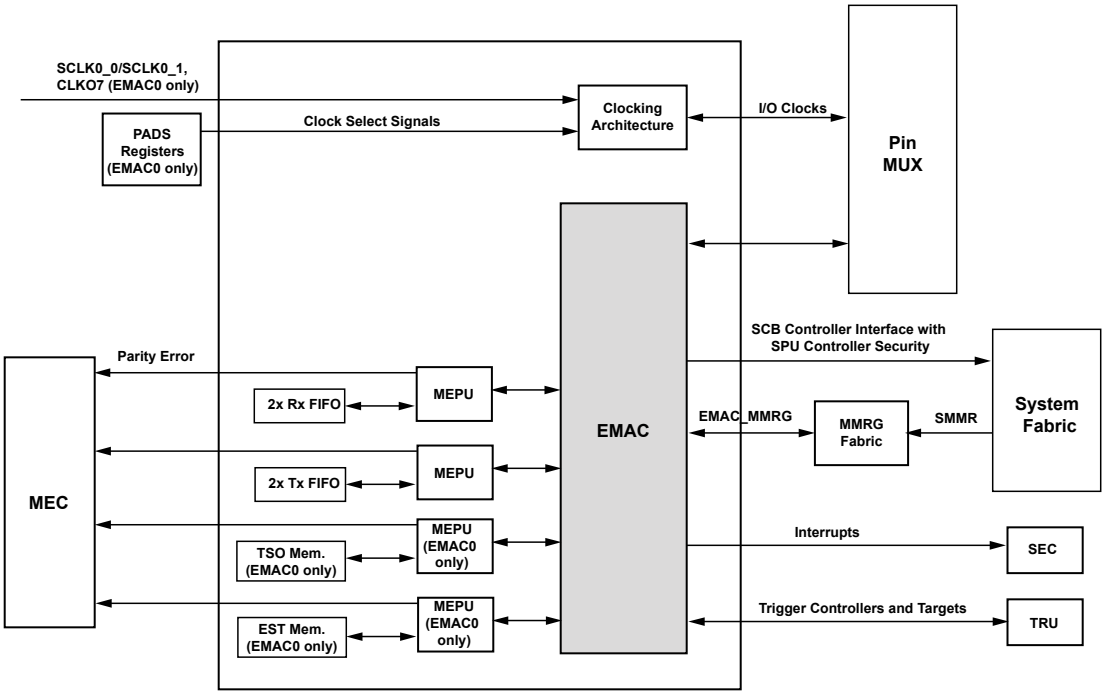


Figure 28-3: EMAC System Level Block Diagram

The *EMAC FIFO/Memory Details* table shows details about various FIFO and memory instances used inside the EMAC module. Each SPRAM/DPRAM stores data along with parity bits generated by MEPU. The parity errors are handled by the Memory Error Controller (MEC) unit.

Table 28-6: EMAC FIFO/Memory Details

EMAC Instance	FIFO/Memory Type	Single (SPRAM)/ Dual Ported (DPRAM)	Runs on	No. of Instances	Size	No. of data bits	No. of parity bits
EMAC0	Rx/Tx FIFOs (Odd/Even)	SPRAM	SCLK0_0/ SCLK0_1	4	2048x40	35	5
	TSO Memory		SCLK0_0/ SCLK0_1	1	512x36	32	4
	EST Memory		PTP Clock	1			

EMAC CORE Sub Blocks

The ***Core Transmit Engine Sub Blocks*** table summarizes the core transmit engine sub blocks and their functions. Refer to the EMAC CORE section for further explanation of each of these sub blocks.

Table 28-7: CORE Transmit Engine Sub Blocks

CORE Transmit Engine Sub Block	Function
Transmit Bus Interface	Interface to the FIFO.
Transmit Packet Controller	Appends Zero-PAD data, if required, for short frames. Appends CRC for frame checksum from the CRC generator.
Transmit Protocol Controller	Generates preamble and SFD, as per 802.3 protocol. Generates jam pattern in half-duplex mode, for collisions. Jabber timeout, for excessively large frames. Flow control for half-duplex mode (back pressure). Generates transmit frame status.
Transmit Scheduler	Maintains the inter-frame gap between two transmitted frames. Follows the truncated binary exponential back-off algorithm for half-duplex mode.
Transmit CRC Generator	Generate CRC for the frame checksum field of the Ethernet frame.
Transmit Flow Control	Receives the pause frame, appends the calculated CRC, and sends the frame to the protocol engine module.
Transmit Checksum Offload Engine	Supports checksum calculation and insertion in the transmit path, for IPV4/TCP/UDP/ICMP packets.
TCP/IP Segmentation Offload (TSO) Engine	Useful in offloading the TCP segmentation functions to the hardware.

Table 28-7: CORE Transmit Engine Sub Blocks (Continued)

CORE Transmit Engine Sub Block	Function
UDP/IPv4 Fragment Offload (UFO) Engine	Supports breaking a large UDP packet into smaller multiple packets for transmission.

The ***Core Receive Engine Sub Blocks*** table summarizes the core receive engine sub blocks and their function. Refer to the EMAC CORE section for more information on each of these sub blocks.

Table 28-8: Core Receive Engine Sub Blocks

CORE Receive Engine Sub block	Functionality Overview
Receive Protocol Engine	Strips the incoming preamble and SFD. Checks for correct length or type field. Performs internal loop back, if necessary. Generates receive status. Supports watchdog of received frames. Supports jumbo frames.
Receive CRC Module	Checks for CRC error, by comparing with FCS.
Receive Packet Controller Module	Packs incoming 8-bit input stream to 32-bit data internally. Performs frame filtering, for uni-cast, multi-cast, and broadcast frames. Attaches the calculated IP checksum input from checksum offload engine. Updates the receive status to bus interface.
Receive Flow Control Module	Detects the receiving pause frame and pauses the frame transmission for the delay specified within the received pause frame. Works in full duplex mode.
Receive IP Checksum Offload Engine	Calculates IPv4 header checksums and verify against the received IPv4 header checksums. Identifies a TCP, UDP, or ICMP payload in the received IP datagrams.
Receive Bus Interface Unit Module	Interface to the FIFO.
Address Filtering Module	Filters destination and source address based on uni-cast, multi-cast, and broadcast frames. Provides CRC hash filtering.
IPv4 ARP Offload Engine	This feature allows the processing of the IPv4 ARP request packet in the receive path and generating corresponding ARP response packet in the transmit path

EMAC PHY Interface

The EMAC can interface to the PHY through the RMII interface standard. The ***RMII Pins*** table shows the RMII pins available in the EMAC, in terms of their generic names. Refer to the data sheet for exact pin names.

Table 28-9: RMII Pins

Signal No.	Generic Signal Name (IEEE Standards)	RMII Pin Functionality
1.	TXD0	RMII transmit data pin D0 (di-bit lower)
2.	TXD1	RMII transmit data pin D1 (di-bit higher)
3.	RXD0	RMII receive data pin D0 (di-bit lower)
4.	RXD1	RMII receive data pin D1 (di-bit higher)
5.	RMII CLK	RMII common clock (for TX and RX), also called reference clock
6.	TXEN	RMII transmit enable pin (TX valid)
7.	CRS	RMII carrier sense / receive data valid
8.	MDC	Serial management clock driven by EMAC
9.	MDIO	Serial management bidirectional data

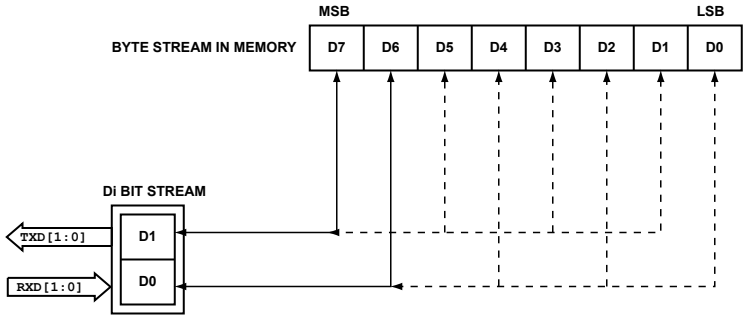


Figure 28-4: RMII Di-bit Data Transfer

RGMII Board Design Recommendations

Use the following guidelines during when performing board design when using the RGMII interface.

MAC to PHY (Transmit)

The Ethernet MAC transmits data to the Ethernet PHY. The Ethernet MAC sends data with tskewT (the timing of TXC at the MAC) that meets the RGMII specification (tskewT = –500 ps to +500 ps skew window for transmitter to drive data). The RGMII specification requires that at the PHY end, tskewR is sampled at 1.0 to 2.6 ns. According to the RGMII standard, clocks must be routed such that an additional trace delay of greater than 1.5 ns and less than 2 ns is added to the associated clock signal.

To meet this standard without adding trace delays, most of the PHYs in the industry already include delay logic that can compensate for this on-board delay. These PHY types can manage a tskewR of ±500 ps (the skew for TXC data sampling seen inside the PHY).

The PHY or the on-board delay must delay the clock signal by 1.5 to 2.0 ns so that tskewR is sampled at 1.0 to 2.6 ns. The *MAC to PHY Delay Diagram (Transmitting Data)* figure shows where the delays must occur.

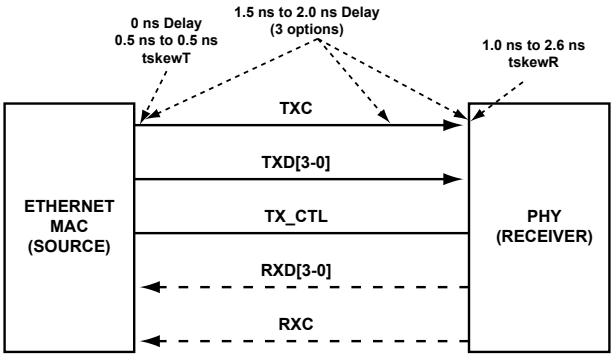


Figure 28-5: MAC to PHY Delay Diagram (Transmitting Data)

PHY to MAC (Receive)

The Ethernet MAC receives data from the Ethernet PHY. Just as in the transmit case, a trace delay of greater than 1.5 ns and less than 2 ns must be added to the associated clock signal, as required by the RGMII specification. Also similar to the transmit case, most of the PHYs in the industry already include delay logic that can compensate for the RXC clock as well.

As shown in the *MAC to PHY Delay Diagram (Receiving Data)* figure, a board trace or the PHY can be used to generate the required 1.5–2.0 ns delay to RXC and the 1.0–2.6 ns tskewR.

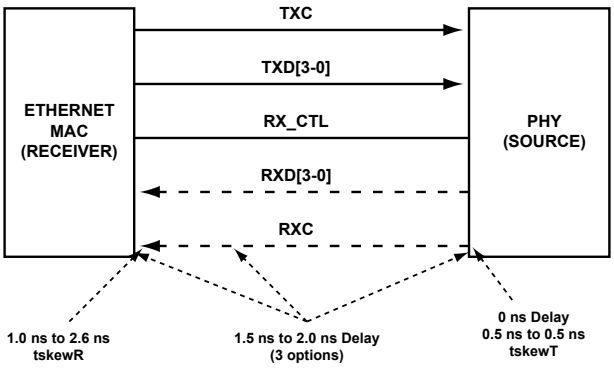


Figure 28-6: MAC to PHY Delay Diagram (Receiving Data)

There following are two options for board design.

- The on-board delay must delay the clock signal by 1.5 to 2.0 ns. In this case, no additional delay must be introduced by the PHY.
- Most of the PHYs in the industry already include a mode that can introduce a delay logic. When this PHY mode is used, then the trace lengths on the board must be matched exactly.

The second option is recommended since it is easier to implement the delay by using the mode in the PHY rather than during the board design. This is the approach followed in the ADI EZ-Kits.

NOTE: Refer to the product specific data sheet for exact processor timing.

For the exact requirements as recommended by the RGMII protocol, see the RGMII specification.

Clock Sources

The Ethernet MAC is clocked internally from CLKO7. Check the processor data sheet for the valid frequency range of the appropriate CLKO7 signal for Ethernet operation.

EMAC0 Clock Sources

EMAC0 supports RGMII, RMII and MII interfaces. The external PHY sources a 2.5 MHz or 25 MHz clock (for 10/100 or gigabit Ethernet respectively) to operate the EMAC RXCLK in RGMII mode. The RGMII TXCLK is driven from CLK07 of the CDU (Clock Distribution Unit) and needs to be configured to 125 MHz regardless of the EMAC0 speeds (10/100/1000 Mbit/s). The `EMAC_MAC_CFG.PS` and `EMAC_MAC_CFG.FES` bits are used to divide the clocks down.

The following tables show the clock sources for the RGMII, MII and RMII interfaces.

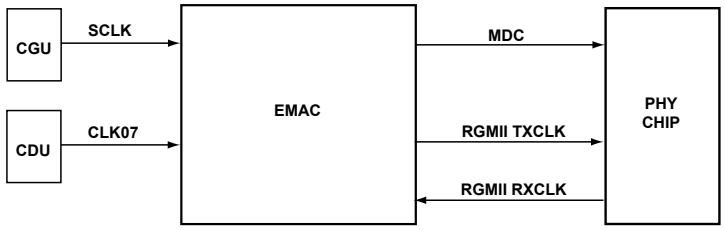


Figure 28-7: EMAC Clock Sources - RGMII PHY Interface

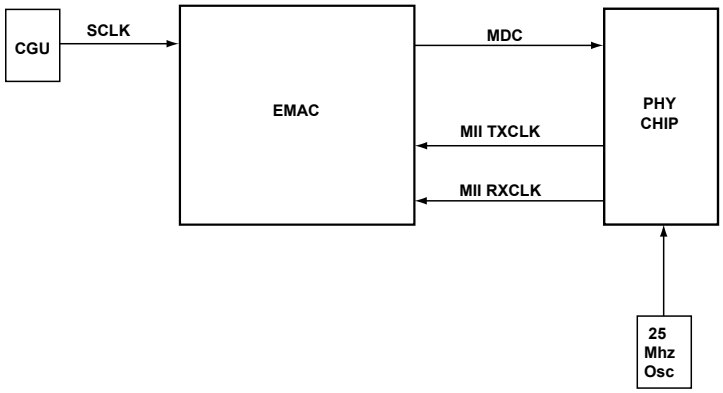


Figure 28-8: EMAC Clock Sources - MII PHY Interface

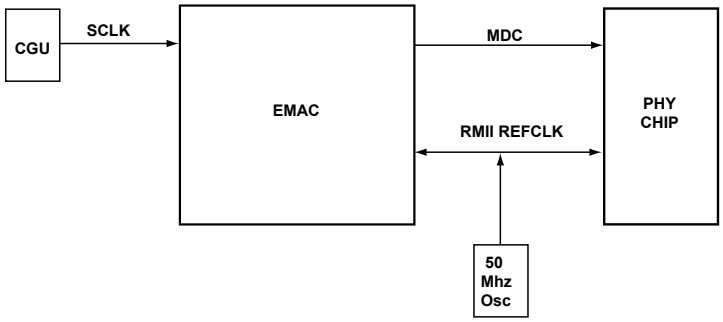


Figure 28-9: EMAC Clock Sources - RMII PHY Interface

EMAC Architectural Concepts

This section explains different architectural concepts relevant to EMAC peripheral, such as EMAC SCB, EMAC DMA, EMAC MTL, EMAC CORE, and others.

EMAC System Crossbar Interface (EMAC SCB)

The EMAC SCB bus interface provides the bus connectivity to support highly effective throughput of data traffic. System bus use is maximized by allowing simultaneous read and write transfers initiated from different DMA channels. The EMAC controller connects directly to the SCB0 crossbar. The following interfaces are available with the design.

- A 32-bit SCB controller interface for reading and writing to and from the application memory.
- A 32-bit SPB target interface for register programming.

Refer to the “System Crossbars (SCB)” chapter for more information on how the crossbar operates. This chapter details only the EMAC-specific information.

Table 28-10: EMAC-SCB Interface Data Transfer Specifications with Crossbar

Specification Term	Comments
1 beat in SCB	SINGLE burst
BLEN4 bursts	4 beats in SCB
BLEN8 bursts	8 beats in SCB
BLEN16 bursts	16 beats in SCB
Bus size	32-bit fixed bus size; equals 1 beat
INCR bursts	Incrementing Bursts
INCR ALIGNED bursts	Incrementing aligned bursts
UNDEF bursts	Undefined burst length
PBL	Programmable Burst Length for DMA

The *EMAC DMA Read/Write channels with System Crossbar* figure shows DMA write channel and read channel data paths and their connection to the system crossbar.

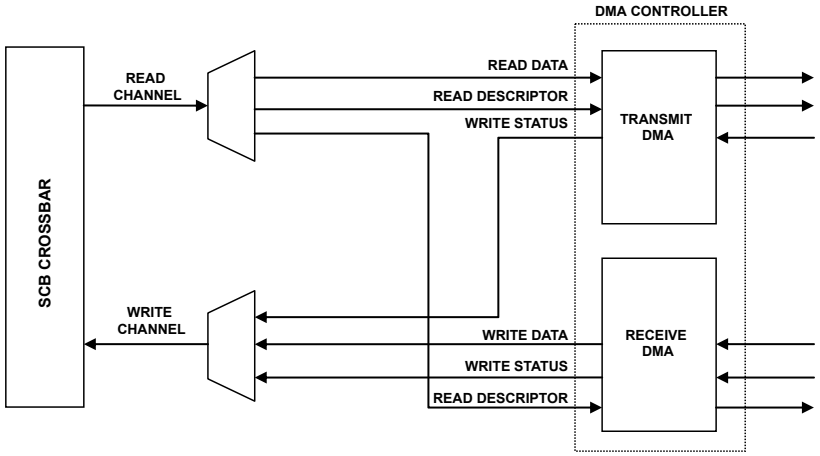


Figure 28-10: EMAC DMA Read/Write channels with System Crossbar

NOTE: Transmit descriptor read and receive descriptor write-back (status update) operations can occur simultaneously. However, transmit descriptor read and write-back operations cannot occur simultaneously. Transmit DMA (or receive DMA) does not initiate the next transfer unless the previous one is complete.

Priority of SCB Requests

The descriptor transfers have higher priority than the data transfers. For example, if there are two bus requests, such as a receive descriptor read and a transmit data read, the receive descriptor read has a higher priority. The next receive data write (after the receive descriptor read) does not depend on the completion of the transmit data-read transfer.

If there are requests for descriptor reads from both DMA channels, they are serviced based on a first-come first-serve. Receive DMA has higher priority if the descriptor-read requests are generated from both the DMA channels in the same clock cycle. Similarly, in the write channel, descriptor writes from DMA have higher priority than the data-write transfers for the receive DMA.

When programs enable the AV or DCB feature, the DMA may contain multiple channels. If the requests are generated from different channels simultaneously in the same clock cycles, the request priority among the channels is: Channel 7 (if present), Channel 6, Channel 5, Channel 4, Channel 3, Channel 2, Channel 1, and Channel 0.

SCB Interface Programming Options

The SCB bus interface supports the following programmable options for the EMAC module. These options are available using the `EMAC_DMA_SYSBMODE` and `EMAC_DMA[n]_CTL` registers.

- *Outstanding transactions.* The EMAC-SCB supports up to 16 (EMAC0) outstanding read/write requests on the SCB bus. Software can control these requests by programming the `EMAC_DMA_SYSBMODE.WR_OSR_LMT` and `EMAC_DMA_SYSBMODE.RD_OSR_LMT` bits. Maximum outstanding requests = `EMAC_DMA_SYSBMODE.WR_OSR_LMT` + 1 or `EMAC_DMA_SYSBMODE.RD_OSR_LMT` + 1.

- ***Allowed burst sizes.*** The allowed burst sizes are 4 (EMAC_DMA_SYSBMODE.BLEN4), 8 (EMAC_DMA_SYSBMODE.BLEN8), 16 (EMAC_DMA_SYSBMODE.BLEN16) and the SINGLE burst. The EMAC-SCB uses only those burst sizes configured by the program (through the EMAC_DMA_SYSBMODE register) for data transfer through the SCB bus. Data transfers are restricted to the maximum burst size from this list of programmed burst sizes.
- ***Burst splitting and burst selection.*** The EMAC-SCB splits the DMA requests into multiple bursts on the SCB system bus. Splitting is based on DMA count and software controllable burst enable bits (EMAC_DMA_SYSBMODE.FB, BLEN16, BLEN8, BLEN4) as well as burst types (INCR and INCR_ALIGNED). Burst types are also controllable through the software. Burst length select priority is in the sequence: FB, BLEN16, BLEN8, BLEN4
- ***INCR burst type***
  - If the EMAC_DMA_SYSBMODE.FB bit is disabled, the EMAC-SCB always chooses the maximum allowed burst length based on the BLEN16, BLEN8 and BLEN4, bits. When the DMA requests are not multiples of the maximum allowed burst length, the SCB can choose a burst-length of any value less than the maximum enabled. (All lesser burst-length enables are redundant). For example, when length bits are enabled and the DMA requests a burst of 42 beats, the SCB splits it into three bursts of 16, 16 and 10 beats respectively.
  - If EMAC_DMA_SYSBMODE.FB is enabled, then the burst length is based on the priority of the enabled bits in the following order: BLEN16, BLEN8 and BLEN4. When the DMA requests a burst transfer, the SCB interface splits the requested bursts into multiple transfers using only the enabled burst lengths. This splitting can occur when the requested burst is not a multiple of the maximum enabled burst. If it cannot choose any of the enabled burst lengths, then it selects the burst length as 1.

  For example, the DMA requests a burst transfer of 42 beats, the SCB interface splits it into multiple bursts of size 16, 16, 8, 1 and 1 beats respectively. (In this case, the allowed burst sizes are enabled and the sequence is in decreasing burst sizes).
- ***INCR_ALIGNED burst type.*** When the address-aligned burst-type is enabled (EMAC_DMA_SYSBMODE.AAL), the SCB interface splits the DMA requested bursts. The "INCR Burst Type" section explains burst splitting conditions further. Each burst-size aligns to the least significant bits of the start address. The SCB interface initially generates smaller bursts so that the remaining transfers move with the maximum (enabled) fixed burst lengths.

  For example, in the same setting as explained earlier for the EMAC_DMA_SYSBMODE.FB bit disabled, the DMA requests a burst size of 42 beats at the start address of 0x000003A4 (EMAC_DMA_SYSBMODE.BLEN4, EMAC_DMA_SYSBMODE.BLEN8 and (EMAC_DMA_SYSBMODE.BLEN16 are enabled). The SCB starts the first transfer with size 3 such that the address of the next burst is aligned (0x000003B0) for a burst of 16. Therefore, the sequence of bursts is 3, 16, 16, and 7, respectively.

  When the EMAC_DMA_SYSBMODE.FB bit is enabled, then (having a start address of 0x000003A4 with 42 beats), the sequence of burst transfers is 1, 1, 1, 16, 16, 4, and 3 respectively. The sequence of smaller bursts at the beginning is used to align the address to the next higher enabled burst-lengths programmed in the register.

Burst Reordering and Data Interleaving

The SCB protocol allows reordering of data transfers with different SCB IDs with respect to the sequence of requests. It also allows data interleaving between transfers of different SCB IDs for read channels. In the EMAC, the requests from each internal requester (receive DMA and transmit DMA) are generated with different SCB IDs. Therefore, reordering and data interleaving can be performed as two DMAs operate independently and are allocated different address space on the target memory.

The SCB requester supports reordering and interleaving from the SCB completer on the read channel. Each DMA internally ensures that a read request and a write request is never generated at the same time to the same address (it can occur for only descriptor accesses). The transmit DMA ensures that transmit descriptor reads, transmit data reads, and transmit descriptor write-back requests are generated only after the previous requests are complete. The receive DMA generates only descriptor reads on the read channel with a different ID. Therefore, data reordering and interleaving on the read channel does not cause any problems in the MAC.

Posted Writes

In posted writes, the write channel of SCB requester transfers an OKAY response to the DMA as soon as the last cycle or beat of data is accepted by the SCB interconnect. The SCB requester sends this response without waiting for the response from the target SCB completer on the write response channel.

In non-posted writes, the SCB controller interface transfers the response received from the SCB target channel to the DMA. To transfer the Ethernet data to system memory, the receive DMA always issues posted write requests to the SCB controller interface. This enables pipelining of the data requests without delays.

The SCB controller does not support out-of-order write transfers. However, the receive DMA ensures that the sequence is maintained for descriptor writes, descriptor reads, and data transfers. For writing descriptors (status or time stamp), the DMA always issues the non-posted write requests. This is required because the transfer complete interrupt is generated based on the descriptor writes for which the DMA needs completion response from the memory target. This ensures no race condition between the hardware and the software because the interrupt is generated only after the data and descriptor are written to the target memory.

SCB Bus Transaction Status

The SCB uses the `EMAC_DMA_DBG_STAT0.AXRHSTS` and `EMAC_DMA_DBG_STAT0.AXWHSTS` bits to indicate whether the channel is active or not.

DMA Controller (EMAC DMA)

The DMA has independent transmit and receive engines, and a CSR (control and status register) space. The transmit engine transfers data from the system memory to the FIFO (MTL), whereas the receive engine transfers data from the FIFO (MTL) to the system memory. The DMA engine uses descriptors to efficiently move data from source to destination with minimal processor core intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The DMA controller can be programmed to interrupt the application for situations such as packet transmit and receive transfer completion, and other normal or error conditions.

The DMA and the application communicate through two internal data structures:

1. Control and status registers (CSR).

2. Descriptor lists and data buffers

The EMAC supports the ring structure for DMA descriptor as shown in the following figure.

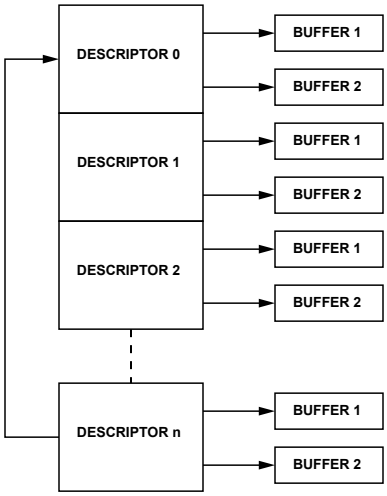


Figure 28-11: Descriptor Ring Structure

In the ring structure, descriptors are separated by the number programmed in the DSL field (32-bit unit) of the EMAC_DMA[n]_CTL registers. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- Transmit Descriptor Ring Length register (EMAC_DMA[n]_TXDSC_RLEN)
- Receive Descriptor Ring Length register (EMAC_DMA[n]_RXCTL2.RDRL)

The EMAC_DMA[n]_TXDSC_TLPTR or EMAC_DMA[n]_RXDSC_TLPTR register contains the pointer to the descriptor address (N). The base address registers (EMAC_DMA[n]_TXDSC_ADDR and EMAC_DMA[n]_RXDSC_ADDR) and the current descriptor pointer registers (EMAC_DMA[n]_TXDSC_CUR or EMAC_DMA[n]_RXDSC_CUR) determine the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA goes into the Suspend mode when this condition occurs. The application must perform a write to the Descriptor Tail pointer register and update the tail pointer so that the following condition is true:

```
Current Descriptor Pointer < Descriptor Tail Pointer;
```

The DMA automatically wraps around the base address when the end of ring is reached, as shown in the *DMA Descriptor Ring* figure.

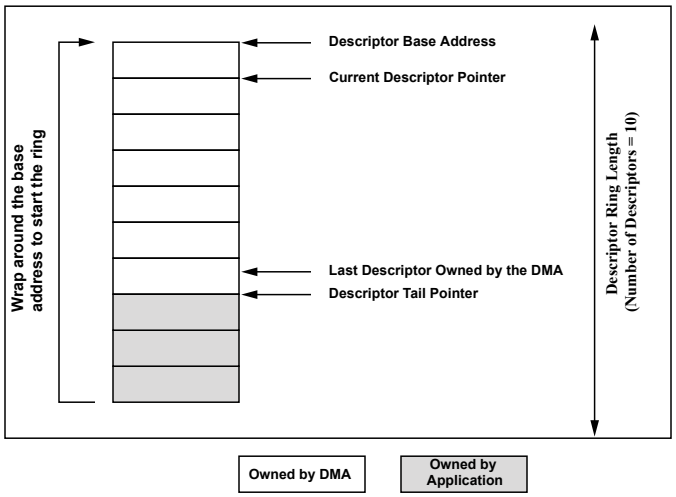


Figure 28-12: DMA Descriptor Ring

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

The DMA supports up to 8 transmit and 8 receive descriptor lists (or DMA channels). The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers in the system memory. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet but cannot exceed a single packet. Buffers contain only data. Buffer status is maintained in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP (End of Packet) is detected.

For more information on the descriptor, see DMA Descriptors section that describes the descriptor structure and how the DMA accesses the descriptors.

DMA Bursts Using the SCB Interface

The DMA engines attempt to transfer data in maximum size bursts that are programmed using the `EMAC_DMA[n]_TXCTL` and `EMAC_DMA[n]_RXCTL` bits. The receive and transmit descriptors are always accessed at the maximum possible burst length to read 16 bytes (limited by PBL or 32). The burst transfers initiated by the DMA can be split into multiple burst transfers as per the SCB protocol requirements and the settings of the `EMAC_DMA_SYSBMODE` register. The transmit DMA initiates a data transfer only when sufficient space is available in the MTL transmit queue to accommodate either of the following:

- Bytes corresponding to the configured burst (PBL x 4)
- Remaining bytes in the transmit Buffer without EOP
- Number of bytes until EOP

The receive DMA initiates a data transfer in the following conditions:

- Sufficient data is available in MTL receive queue to accommodate the configured burst
- EOP (when it is less than the configured burst length) is detected in the receive queue

The DMA indicates the start address and the number of transfers required to the SCB interface. When the SCB interface is configured for a fixed-length burst, it transfers the data by using the best combination of INCR4, INCR8, or INCR16 and SINGLE transactions. If EOP is reached before the fixed-burst ends on the SCB interface, dummy transfers are performed in-order to complete the fixed-burst. Otherwise, the DMA transfers the data using INCR (undefined length) and SINGLE transactions. When the SCB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer initiated by the SCB is less than or equal to the size of the configured PBL. Therefore, all subsequent beats start at an address that is aligned to the configured PBL.

Application Data Buffer Alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. The start address for the buffers aligns to any of the 4 bytes. However, the DMA always initiates write transfers with address aligned to the bus width and dummy data (old data) in the invalid byte lanes. This typically happens during the transfer of the beginning or end of an Ethernet packet. The software driver should discard the dummy bytes based on the start address of the buffer and size of the packet.

Example for Buffer Read

If the transmit buffer address is 0x0002 and 15 bytes must transfer, the DMA reads 5 full words (5 x 32-bit data) from address 0x0000. However, when transferring data to the EMAC transmit FIFO, the extra bytes (the first 2 bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always transfers a full 32-bit data to the transmit FIFO unless it is the end-of-frame.

Example for Buffer Write

If the receive buffer address is 0x0002 and 15 bytes of a received frame must transfer, the DMA writes 5 full words (5 x 32-bit data) to address 0x0000. However, the first 2 bytes of first transfer and the last 3 bytes of the third transfer have dummy data.

Buffer Size Calculations

The DMA engines do not update the size fields in the transmit and receive descriptors alone. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver must perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES2) towards the MAC. If a descriptor is marked as first (FD bit of TDES3 is set), the DMA marks the first transfer from the buffer as SOP. If a descriptor is marked as last (LD bit of TDES3), the DMA marks the last transfer from that data buffer as EOP to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or the end of packet is received from the MTL. When the FD bit of a descriptor is set, the amount of valid data in a buffer is accurately indicated by the buffer size field (programmed in the `EMAC_DMA[n]_RXCTL` registers) minus the data buffer pointer offset. The offset

is zero when the data buffer pointer is aligned to the data bus width (32 bits). If a descriptor is marked as last, the buffer may not be full (as indicated by the buffer size in Bits [14:1] of the EMAC_DMA[n]_RXCTL registers). To compute the amount of valid data in this final buffer, the driver must:

- Read the packet length (PL bits of RDES3 [14:0]), and
- Subtract the sum of the buffer sizes of the preceding buffers in this packet

The receive DMA always transfers the start of next packet with a new descriptor.

DMA Transmit Operation

The following sections describe how the transmission process works for direct memory access on the EMAC controller.

Default (Non-OSF) Mode

The following sequence describes the default process for DMA transmit works.

1. The application sets up the transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application advances the descriptor tail pointer offset value of the transmit channel.
3. While in the run state, the DMA runs an arbitration cycle to select the next transmit DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended, the EMAC_DMA[n]_STAT.TBU and EMAC_DMA[n]_STAT.TEB bits are set, and the transmit engine proceeds to step 11:
   - The descriptor is flagged as owned by the application (TDES3 [31] = 1'b0)
   - The descriptor tail pointer is equal to the current descriptor pointer in ring descriptor list mode
   - An error condition occurs
6. If the acquired descriptor is flagged as owned by DMA (TDES0 [31] = 1#b1), the DMA decodes the transmit data buffer address from the acquired descriptor.
7. The DMA fetches the transmit data from the application memory and transfers the data to the MTL for transmission.
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end of- Ethernet-packet data is transferred to the MTL.
9. When packet transmission is complete, if the IEEE 1588 time stamp feature was enabled for the packet (as indicated in the transmit status), the time stamp value obtained from MTL is written to the transmit descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this

transmit descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step.

If time stamp feature is not enabled for this packet, the DMA does not alter the contents of TDES0 and TDES1.

NOTE: By default, the transmit DMA places subsequent request after the current descriptor write request is completed to the descriptor memory. If posted descriptor write is enabled by setting the EMAC_DMA_MODE.DSPW bit, the transmit DMA places subsequent request after the current descriptor write request is placed without waiting for its completion, thereby improving the throughput.

10. The transmit interrupt (EMAC_DMA[n]_STAT.TI) bits are set after completing transmission of a packet that has interrupt on completion (TDES2[31]) set in its last descriptor. The DMA engine returns to step 3.
11. In the suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the EMAC_DMA[n]_TXDSC_TLPTR registers when it receives a transmit poll demand and the underflow interrupt status bit is cleared. If the application stopped the DMA by clearing transmit interrupt bit, the DMA enters the stop state.

NOTE: In non-OSF mode, it is not required for the driver/application to extract the packet status and release the descriptor for the packet, even though FD is set to 1. The driver must track the descriptor that has OWN = 0 and LD = 1, as 1; such a descriptor contains the transmission status for the packet. This indicates that descriptors up to this descriptor can be released for reuse by the driver/application for subsequent packets.

OSF Mode

While in the run state, the transmit process can simultaneously acquire two packets without closing the status descriptor of the first packet (if the EMAC_DMA[n]_TXCTL.OSF bit is set). As the transmit process finishes transferring the first frame, it immediately polls the transmit descriptor list for the second packet. If the second packet is valid, the transmit process transfers this frame before writing the status information of the first packet.

In OSF mode, the run state transmit DMA operates in the following sequence.

1. The DMA operates as described in steps 1–7 of default (non-OSF) mode
2. Without closing the previous last descriptor of the frame, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into suspend mode and skips to Step 7.
4. The DMA fetches the transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and time stamp of previous packet. When the status is available, the DMA writes the time stamp to TDES0 and TDES1 if such time stamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES1, thus closing the descriptor.

If the time stamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES0 and TDES1.

6. If enabled, the transmit interrupt is set; the DMA fetches the next descriptor, and then proceeds to Step 3 (when status is normal). If the previous transmission status shows an underflow error, the DMA goes into suspend mode (Step 7).
7. In suspend mode, if a pending status and time stamp are received from the MTL, the DMA:
   - Writes the time stamp (if enabled for the current packet) to TDES0 and TDES1
   - Writes the status to the corresponding TDES1
   - Sets relevant interrupts and returns to Suspend mode

   If no status is pending and the application stopped the DMA by clearing the `EMAC_DMA[n]_TXCTL.ST` bits, the DMA enters the Stop state.
8. The DMA can exit suspend mode and enter the run state (goes to step 1 or step 2 depending on pending status) only after receiving a transmit poll demand in transmit descriptor tail pointer register of the corresponding channel.

NOTE: The DMA fetches the next descriptor before closing the current descriptor. Therefore, the descriptor ring length must be more than 2. A minimum descriptor length of 4 is recommended.

In OSF mode, all except the last descriptor is closed immediately. The last descriptor is closed after the packet is transmitted on the line. Therefore, to minimize the complexity, the transmission status and only the required control bits are updated in the pending last descriptor of the previous packet.

Transmit Packet Processing

The transmit DMA engine expects that the data buffers contain complete Ethernet packets, excluding the preamble, pad bytes, and FCS fields. The DA, SA, and type/length fields contain valid data. When the transmit descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet packets (excluding preamble), including the CRC bytes.

Packets can be data-chained and can span several buffers. Packets must be delimited by the first descriptor (TDES3[29]) and the last descriptor (TDES3[28]). As transmission starts, the first descriptor must have TDES3[29] set. When this occurs, the packet data is transferred from the application buffer to the MTL transmit queue. Concurrently, iwhen the current packet has the last descriptor (TDES3[28]) clear, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES3[29] clear. When TDES3[28] is clear, it indicates an intermediary buffer. When TDES3[28] is set, it indicates the last buffer of the packet.

After the last buffer of the packet has been transmitted, the DMA writes back the final status information to the transmit descriptor 3 (TDES3) word of the descriptor that has the last descriptor bit set in transmit descriptor 3 (TDES3[28]). At this time, if interrupt on completion (TDES2[31]) is set, the `EMAC_DMA[n]_STAT.TI` bit is set, the next descriptor is fetched, and the process repeats. The actual packet transmission begins after either of the following:

- The MTL transmit queue has reached a programmable transmit threshold (programmed in the EMAC_TQ0_OPMODE.TTC bit field)
- A full packet is contained in the FIFO

Programs can also use the store-and-forward mode (EMAC_TQ0_OPMODE.TSF bit is set). In this mode, descriptors are released (OWN bit in TDES0[31] clears) when the DMA finishes transferring the packet.

NOTE: To ensure proper transmission of a packet and the next packet, specify a non-zero buffer size for the transmit descriptor that has the last descriptor (TDES3[28]) set.

Transmit Polling Suspended

Either of the following conditions suspends transmit polling:

- The DMA detects a descriptor owned by the application (TDES3[31] = 0). To resume, the driver must give descriptor ownership to the DMA and then issue a poll demand command by writing the EMAC_DMA[n]_TXDSC_TLPTR registers. If the DMA goes into the SUSPEND state because of this condition, the EMAC_DMA[n]_STAT.NIS and EMAC_DMA[n]_STAT.TBU bits are set.
- A packet transmission is aborted when a transmit error is detected because of underflow.

  The appropriate transmit descriptor 3 (TDES3) bit is set. When this condition occurs, the following bits are set and the information is written to transmit descriptor 0, causing the suspension:

  - EMAC_DMA[n]_STAT.AIS
  - Transmit Underflow bit of corresponding queue (EMAC_Q0_INT_CTLSTAT.TXUNFIS bits)

In all conditions, the position in the transmit list is retained. The retained position is that of the descriptor following the last descriptor closed by the DMA. The driver must explicitly issue a transmit poll demand command after rectifying the suspension cause.

DMA Transmit Channel Arbitration

An arbiter provides access to multiple DMAs trying to access the Bus Interface Unit (BIU).

When there is a request in the transmit DMA, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests. Therefore, when there is a descriptor fetch request, the DMA arbiter acknowledges the DMA channel that is requesting for a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (channel 1 having priority over channel 0, channel 2 having priority over channel 1 and so on). For packet buffer fetches, the DMA arbiter uses the programmed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms, which can be selected by programming the EMAC_DMA_MODE.TAA bit field.

- Weighted Strict Priority (WSP). In WSP arbitration mode, the arbiter first processes channel 7 (or the last selected channel) followed by channel 6, channel 5, and so on. If a channel does not have a frame to transmit,

the weight of the channel is reassigned to channel 7 (or the last selected channel). If channel 7 has no frames to transmit, the remaining weight is assigned to channel 6 and so on.

- Weighted Round Robin (WRR). In WRR arbitration mode, the arbiter first selects the channel with the highest weight programmed, and then the channel with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel is equally distributed to all channels that have frames to transmit. The weight of the channel is programmed in the EMAC_DMA[n]_TXCTL.TCW (transmit channel weight) bit fields.
- Fixed priority (FP). In fixed priority mode, channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programmed in the TCW bits of a channel is ignored. In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

DMA Receive Operation

In the receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel. The ARI data at the start of the frame indicates the channel number to which the current frame must be written. If only one DMA channel is selected, this information is not provided.

The following list shows the reception sequence for the receive DMA engine:

1. The application sets up the receive descriptors (RDES0-RDES3) and the OWN bit (RDES3[31]). The application must set the correct value in the EMAC_DMA[n]_RXDSC_TLPTR register.
2. When the EMAC_DMA[n]_RXCTL.SR bit is set, the DMA enters the run state. The DMA looks for free descriptors based on the EMAC_DMA[n]_RXDSC_ADDR and EMAC_DMA[n]_RXDSC_TLPTR register values. If there are no free descriptors, the DMA channel enters the suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from the acquired descriptors.
4. If IEEE 1588 time stamping is enabled and the time stamp is available for the previous packet, the DMA writes the time stamp (if available) to the RDES0 and RDES1 registers of the current descriptor and sets the CTXT bit field (RDES3[30]).
5. The DMA processes the incoming packets and places these in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA takes the status of the receive frame from the MTL and writes the status word to current descriptor with the OWN bit cleared and the last descriptor bit set.

8. The DMA writes the frame length to RDES3 and VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. If the IEEE 1588 time stamp feature is enabled, the DMA stores the time stamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the receive DMA descriptor ring, go to step 3; otherwise, go to the suspend state (step 11).
11. The receive DMA exits the suspend state when a receive poll demand is given, and the application advances the receive tail pointer register of a channel.

The engine proceeds to step 2 and refetches the next descriptor.

Receive Descriptor Acquisition

The receive engine always attempts to acquire an extra descriptor in anticipation of an incoming packet. descriptor acquisition is attempted if any of the following conditions is satisfied:

- The `EMAC_DMA[n]_RXCTL.SR` bit is set immediately after being placed in the run state.
- The `EMAC_DMA[n]_RXDSC_TLPTR` register value is ahead of the current descriptor acquired by the receive DMA.
- The controller has completed packet reception, but the current receive descriptor is not yet closed.
- A receive poll demand is issued (update of the tail pointer register).

Receive Packet Processing

The sequence for processing a receive packet is as follows:

1. The MAC transfers the received packets to the MTL memory only if the packet passes the address filter. If the packet fails the address filtering, it is dropped in the MAC block (unless the `EMAC_MACPKT_FILT.RA` bit is set).
2. If the packet size is greater than or equal to the configurable threshold bytes set for the receive queue of the MTL, or when the complete packet is written to the queue in the store-and-forward mode, the MTL block requests the DMA block to begin transferring the packet data to the receive buffer pointed to by the current descriptor.

   Because of collision or premature termination, packets smaller than 64 bytes are removed from the MTL receive queue.
3. When the DMA application interface (SCB or MDC) is ready, it transfers the data and sets the following:
   - If the packet fits in a single descriptor, the DMA sets both last descriptor (RDES3[28]) and first descriptor (RDES3[29]).
   - If the packets fits into more than one descriptor, the DMA sets the first descriptor (RDES3[29]) to delimit the packet.

4. The DMA releases the descriptors by resetting the OWN (RDES3[31]) bit to 1'b0, either because the receive is full or the last segment of the packet is transferred to the receive buffer. The received packet's status is updated in the last descriptor.
5. If the interrupt enabled on completion (RDES3[30]) bit is set in any of the descriptors between the first and last descriptor of the packet and the `EMAC_DMA[n]_IEN.RIE` bit is set, the DMA sets the `EMAC_DMA[n]_STAT.RI` bit.

The same process repeats unless the DMA encounters a descriptor flagged as being owned by the application or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if the `EMAC_DMA[n]_IEN.RBUE` bits are set, the receive process sets the `EMAC_DMA[n]_STAT.RBU` bits and then enters the suspend state. The position in the receive list is retained.

DMA Error Response

For any data transfer initiated by a DMA channel, if the completer replies with an error response, the DMA stops all operations and updates the error bits and the `EMAC_DMA[n]_STAT.FBE` bits. The application can either perform a reset to EMAC or re-initialize the DMA descriptor list and start again. The rest of the DMA channels are not affected by such errors.

DMA Descriptors

The DMA in the EMAC transfers data based on a linked list of descriptors. The descriptor addresses must be aligned to the 32-bit bus width. The application creates the descriptors in the system memory. The EMAC supports the following two types of descriptors.

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted. Each normal descriptor contains two buffers and two address pointers. These buffers enable the EMAC to be compatible with various types of memory management schemes.

NOTE: There is no limit for the number of descriptors that can be used for a single packet.

For basic details that describe the descriptor ring structure, see DMA Controller (EMAC DMA).

Transmit Descriptor

The DMA in EMAC requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The transmit normal descriptor has two formats: read format and write-back format.

Transmit Normal Descriptor (Read Format)

The *Transmit Normal Descriptor (Read Format)* figure shows the read format for a transmit normal descriptor. The following tables describe the read format for the transmit normal descriptors: TDES0, TDES1, TDES2, and TDES3.

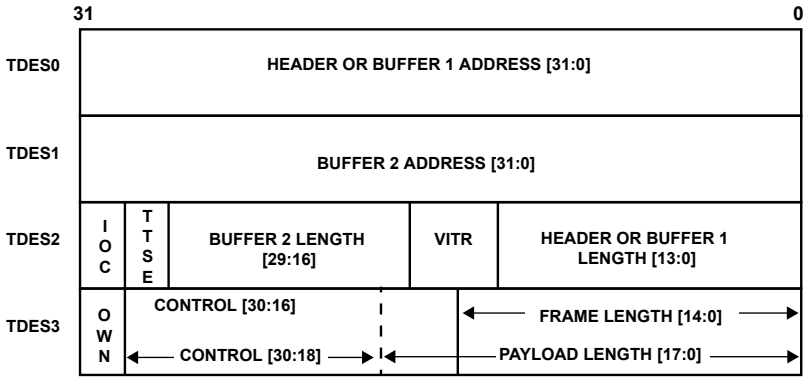


Figure 28-13: Transmit Normal Descriptor (Read Format)

Table 28-11: TDES0 Normal Descriptor (Read Format)

Bit	Name	Description
31–0	BUF1AP	Buffer 1 Address Pointer or TSO Header Address Pointer. These bits indicate the physical address of buffer 1. These bits also indicate the TSO header address pointer when the following bits are set: • TSE bit of TDES3 • FD bit of TDES3

Table 28-12: TDES1 Normal Descriptor (Read Format)

Bit	Name	Description
31–0	BUF2AP	Buffer 2 or Buffer 1 Address Pointer. These bits indicate the physical address of buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment.

Table 28-13: TDES2 Normal Descriptor (Read Format)

Bit	Name	Description
31	IOC	Interrupt on Completion. This bit controls the setting of TI and ETI status bits in the EMAC_DMA[n]_STAT register. When ETIC = 1 and TDES2[LD] = 0, this bit sets the ETI bit. When TDES3[LD] = 1, this bit sets the TI status bit.
30	TTSE/TMWD	Transmit Time stamp Enable or External TSO Memory Write Enable. This bit enables the IEEE1588 time stamping for the transmit packet referenced by the descriptor, if the TSE bit is not set. If the TSE bit is set and external TSO memory is enabled, setting this bit disables external TSO memory writing for this packet.
29:16	B2L	Buffer 2 Length. The driver sets this field. When set, this field indicates buffer 2 length.

Table 28-13: TDES2 Normal Descriptor (Read Format) (Continued)

Bit	Name	Description
15:14	VTIR	VLAN Tag Insertion or Replacement. These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC pad control bits appropriately when VLAN tag insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits: • 00 = Do not add a VLAN tag • 01 = Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. • 10 = Insert a VLAN tag with the tag value programmed in the MAC_VLAN_Incl register or context descriptor • 11 = Replace the VLAN tag in packets with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets. These bits are valid when the enable SA and VLAN insertion on transmit option is selected while configuring the core.
13:0	HL or B1L	Header Length or Buffer 1 Length. For header length only bits [9:0] are taken. The size 13:0 is applicable only when interpreting buffer 1 length. If the TCP segmentation offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length in bytes from the Ethernet source address until the end of the TCP header. The maximum header length supported for the TSO feature is 1023 bytes. If the TCP segmentation offload feature is not enabled, this field is equal to buffer 1 length

Table 28-14: TDES3 Normal Descriptor (Read Format)

Bit	Name	Description
31	OWN	Ownership. When set, this bit indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit either when it completes the transfer of data given in the associated buffers.
30	CTXT	Context Type. This bit should be set to 1'b0 for normal descriptor.
29	FD	First Descriptor. When set, this bit indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor. When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.

Table 28-14: TDES3 Normal Descriptor (Read Format) (Continued)

Bit	Name	Description
27:26	CPC	Disable CRC. When this bit is set, the EMAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. • 00 = CRC and Pad Insertion. The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. • 01 = CRC Insertion (Disable Pad Insertion). The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the transmit buffer, that is, the packet being transferred from the transmit buffer is of length greater than or equal to 60 bytes. • 10 = Disable CRC Insertion. The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. • 11 = CRC Replacement. The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. This field is valid only for the first descriptor. Note: When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	SA Insertion Control. When set, these bits request the MAC to add or replace the source address field in the Ethernet packet with the value given in the MAC address 0 register. The application must set the CRC pad control bits appropriately when SA insertion control is enabled for the packet. Bit 25 specifies the MAC address register (1 or 0) value that is used for source address insertion or replacement. Bits [24:23] encodings are as follows • 00 = Do not include the source address • 01 = Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. • 10 = Replace the source address. For reliable transmission, the application must provide frames without source addresses. • 11 = Reserved These bits are valid when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core and when the First descriptor (FD) bit (TDES3 [29]) is set.
22:19	SLOTNUM or THL	SLOTNUM: Slot Number Control bits in AV mode. These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field in the `EMAC_DMA[n]_SFN_CTLSTAT` register. Fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels.

Table 28-14: TDES3 Normal Descriptor (Read Format) (Continued)

Bit	Name	Description
		THL: TCP/UDP Header Length. If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. The value must be equal to 2 for UDP header. This field is valid only for the first descriptor
18	TSE	TCP Segmentation Enable. When this bit is set, the DMA performs the TCP/UDP segmentation or UDP fragmentation for a packet depending on the TSE_MODE[1:0] bit of the `EMAC_DMA[n]_TXCTL` register. This bit is valid only if the FD bit is set.
17:16	CIC/TPL	Checksum Insertion Control or TCP Payload Length. These bits control the checksum calculation and insertion. Bit encodings are as follows: • 00 = Checksum insertion disabled. • 01 = Only IP header checksum calculation and insertion are enabled • 10 = IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware • 11 = IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware When the TSE bit is set, this field contains the upper bits [17:16] of the TCP payload (or IP payload for UDP fragmentation). This allows the TCP/UDP packet length field to be spanned across TDES3[17:0] to provide 256 KB packet length support. This field is valid only for the first descriptor
15	TPL	Reserved or TCP Payload Length. When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0]. This field is valid only when the enable TCP segmentation offloading for TCP/IP packets option is selected while configuring the core
14:0	FL/TPL	Frame Length or TCP Payload Length. This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: `Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length` When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length in case of segmentation and IP payload in case of UDP fragmentation. In case of segmentation, this length does not include Ethernet header or TCP/UDP/IP header length. In case of fragmentation, this length does not include Ethernet header and IP header. When DWRR/WFQ algorithm is NOT enabled, value written into this field is not used when TSE = 0.

Transmit Normal Descriptor (Write-Back Format)

The write-back format of the transmit descriptor includes time stamp low, time stamp high, OWN, and status bits.

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and time stamp information for the corresponding transmit packet.

The *Transmit Normal Descriptor (Write-Back Format)* figure illustrates the write-back format of the transmit descriptor.

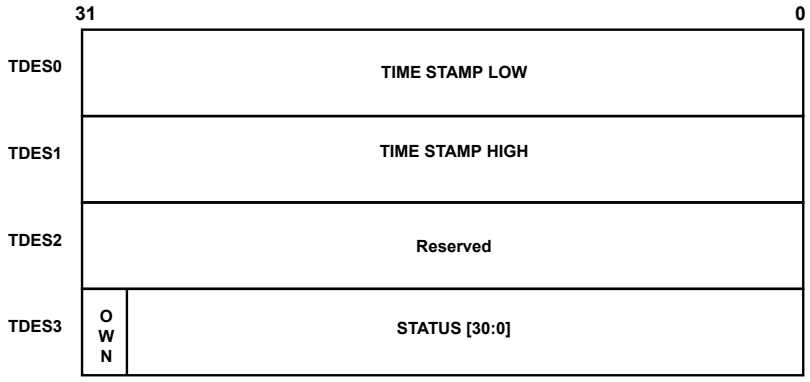


Figure 28-14: Transmit Normal Descriptor (Write-Back Format)

TDES0 Normal Descriptor (Write-Back Format)

As described in the following table, this format is only applicable to the last descriptor of a packet.

Table 28-15: TDES0 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–0	TTSL	Transmit Packet Time Stamp Low. The DMA updates this field with least significant 32 bits of the time stamp captured for the corresponding transmit packet. The DMA writes the time stamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the time stamp only if the last segment bit (LS) in the descriptor is set and the time stamp status (TTSS) bit is set.

TDES1 Normal Descriptor (Write-Back Format)

As described in the following table, this format is only applicable to the last descriptor of a packet.

Table 28-16: TDES1 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–0	TTSH	Transmit Packet Time Stamp High. The DMA updates this field with the most significant 32 bits of the time stamp captured for corresponding transmit packet. The DMA writes the time stamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the time stamp only if the last segment bit (LS) in the descriptor is set and time stamp status (TTSS) bit is set.

TDES2 Normal Descriptor (Write-Back Format)

This format is only applicable to the last descriptor of a packet.

Table 28-17: TDES2 Normal Descriptor (Write-Back Format)

Bit	Name
31–0	Reserved

TDES3 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

Table 28-18: TDES3 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31	OWN	Ownership. When this bit is set, it indicates that the EMAC DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	Context Type. This bit should be cleared (=0) for normal descriptor.
29	FD	First Descriptor. When set, this bit indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor. This bit is set (=1) for the last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27–24	Reserved	
23	DE	Descriptor Error. When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor errors include: • Incorrect sequence from the context descriptor. For example, a location after the first descriptor for a packet. • All 1s • CTXT is set to 1 and the LD or FD bits are set to 1. **NOTE:** When a descriptor error occurs due to all 1s or CTXT, LD, and FD bits set to 1, the transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When the IOC bit in the TDES2 register of the corresponding first descriptor is set to 1, the transmit DMA sets the TI bit in the `EMAC_DMA[n]_STAT` register. **NOTE:** Based on the CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the first descriptor (even if the FD bit is not set) and partial packet is sent.
22–18	Reserved	
17	TTSS	Transmit Time Stamp Status. This status bit indicates that a time stamp has been captured for the corresponding transmit packet. When this bit is set, the TDES0 and TDES1 registers have time stamp values that were captured for the transmit packet. This field is valid only when the last segment control bit (TDES3 [28]) in a descriptor is set. This bit is valid only when the IEEE1588 time stamping feature is enabled; otherwise, it is reserved.
16	EUE	ECC Uncorrectable Error Status. Indicates an ECC uncorrectable error in the TSO memory.

Table 28-18: TDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
		NOTE: An uncorrectable error in transmit FIFO memory is reported when the packet flushed bit (bit 13, FF) = 1. This is because all such packets are flushed by the EMAC module.
15	ES	Error Summary. Indicates the logical OR of the following bits: TDES3[0]: IP Header Error DES3[14]: Jabber Timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error This bit is also set when the EUE (bit 16) is set.
14	JT	Jabber Timeout. This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the EMAC_MAC_CFG register is not set.
13	FF	Packet Flushed. This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	Payload Checksum Error. This bit indicates that the checksum offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure is caused by: • insufficient bytes in the payload, as indicated by the payload length field of the IP header. • the MTL starts to forward the packet to the MAC transmitter in store-and-forward mode without the checksum being calculated. This error condition only occurs when the transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock. In this case the MTL starts forwarding the packet when the FIFO is full, even in store-and-forward mode. • a bus error is detected during packet transfer. When the full checksum offload engine is not enabled, this bit is reserved.
11	LoC	Loss of Carrier. This bit indicates that a loss of carrier occurred during packet transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
10	NC	No Carrier. This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	Late Collision. This bit indicates that packet transmission was terminated because a collision occurred after the collision window (64 byte times including preamble in MII mode and 512

Table 28-18: TDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
		byte times including preamble and carrier extension in GMII mode). This bit is not valid if underflow error is set.
8	EC	Excessive Collision. This bit indicates that the transmission was terminated after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after first collision and the transmission of the packet is terminated.
7–4	CC	Collision Count. This 4-bit counter value indicates the number of collisions that occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	Excessive Deferral. This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbps mode or jumbo packet enabled mode) if the DC bit is set in the MAC_Configuration register. When TBS is enabled in full duplex mode and this bit is set, it indicates that the frame has been dropped after the expiry time has reached.
2	UF	Underflow Error. This bit indicates that the MAC terminated the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: • The DMA encountered an empty transmit buffer while transmitting the packet • The application filled the MTL transmit FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the EMAC_MTL_ISTAT register.
1	DB	Deferred Bit. This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
0	IHE	IP Header Error. When this bit is set, it indicates that the checksum offload engine detected an IP header error. This bit is valid only when transmit checksum offload is enabled. Otherwise, it is reserved. If the COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet type field indicates an IPv4 payload. In full duplex mode, when EST/Qbv is enabled and this bit is set, it indicates the frame drop status due to frame size error or schedule error.

Transmit Context Descriptor

The context descriptor is used to provide the time stamps for one-step time stamp correction, VLAN tag ID for VLAN insertion feature.

The transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the time stamps for one-step time stamp correction and VLAN Tag ID for VLAN insertion feature. Write back is done on a context descriptor only to reset the OWN bit.

NOTE: The VLAN Tag IDs and MSS values, provided by the application in a context descriptor with their corresponding valid bits set, are stored internally by the DMA. When the outer or inner VLAN tag is provided with the valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The

application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The inner VLAN tag control input is used only for the next packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

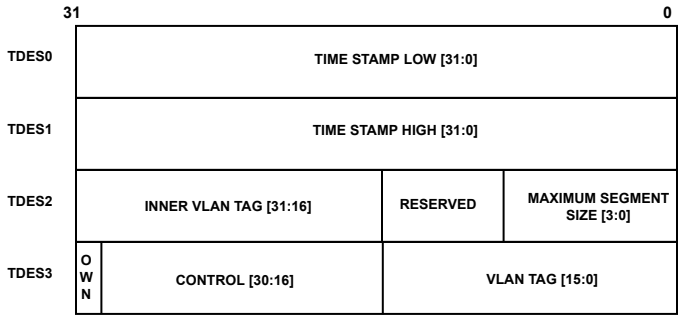


Figure 28-15: Transmit Context Descriptor Format

Table 28-19: TDES0 Context Descriptor

Bit	Name	Description
31–0	TTSL	Transmit Packet Time Stamp Low. For one-step correction, the driver can provide the lower 32 bits of time stamp in this descriptor word. The DMA uses this value as the low word for doing one-step time stamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Table 28-20: TDES1 Context Descriptor

Bit	Name	Description
31–0	TTSH	Transmit Packet Time Stamp High. For one-step correction, the driver can provide the upper 32 bits of time stamp in this descriptor. The DMA uses this value as the high word for doing one-step time stamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Table 28-21: TDES2 Context Descriptor

Bit	Name	Description
31–16	IVT	Inner VLAN Tag. When the IVLTV bit of the TDES3 context descriptor is set and the TCMSSV and OSTC bits of the TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN tag that is inserted in the subsequent transmit packets.
15–14	Reserved	
13–0	MSS	Maximum Segment Size. When the enable TCP segmentation offloading for TCP/IP packets option is selected, the driver provides maximum segment size in this field. This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of the TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

Table 28-22: TDES3 Context Descriptor

Bit	Name	Description
31	OWN	Own Bit. When this bit is set, it indicates that the EMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit immediately after the read.
30	CTXT	Context Type. This bit should be cleared (=0) for normal descriptor.
29–28	Reserved	
27	OSTC	One-Step Time stamp Correction Enable. When this bit is set, the DMA performs a one-step time stamp correction with reference to the time stamp values provided in TDES0 and TDES1.
26	TCMSSV	One-Step Time stamp Correction Input or MSS Valid. When this bit and the OSTC bit are set, it indicates that the time stamp correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25–24	Reserved	
23	CDE	Context Descriptor Error. When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the context descriptor. Context descriptor errors can be: • Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. • All 1s • CD, LD, and FD bits set to 1. **NOTE:** When the context descriptor error occurs due to all 1s or the CTXT, LD, and FD bits set to 1, the transmit DMA closes the transmit descriptor with the DE and LD bits set to 1. When the IOC bit in TDES2 of the corresponding first descriptor is set to 1, transmit DMA sets the TI bit in the DMA_CH#_Status register. **NOTE:** Based on the CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the first descriptor (even if the FD bit is not set) and a partial packet is sent. **NOTE:** This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported).
22–20	Reserved	

Table 28-22: TDES3 Context Descriptor (Continued)

Bit	Name	Description
19–18	IVTIR	Inner VLAN Tag Insert or Replace. When these bits are set, the MAC is requested to perform inner VLAN tagging or un-tagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: • 00 = Do not add the inner VLAN tag. • 01 = Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. • 10 = Insert an inner VLAN tag with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. • 11 = Replace the inner VLAN tag in packets with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. This option should be used only with the VLAN frames. These bits are valid when the Enable SA and VLAN Insertion on transmit and Enable Double VLAN Processing options are selected.
17	IVLTV	Inner VLAN Tag Valid. When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid. When this bit is set, it indicates that the VT field of TDES3 is valid.
15–0	VT	VLAN Tag. This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the MAC_VLAN_Incl register is reset.

Receive Descriptor

The DMA in the EMAC module attempts to read a descriptor only if the tail pointer is different from the base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the receive FIFO in the MTL becomes full and starts dropping packets.

The following receive descriptors are present:

- Normal descriptors
- Context descriptors

All receive descriptors are prepared by the software and given to the DMA as “Normal” descriptors with the content as shown in Receive Normal Descriptor (Read Format). The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the receive DMA closes the descriptor with the corresponding packet status. The format of this status is given in Receive Normal Descriptor (Write-Back Format).

For some packets, the normal descriptor bits are not enough to write the complete status. For such packets, the receive DMA writes the extended status to the next descriptor (without processing or using the buffers/pointers embedded in that descriptor). The format and content of the descriptor write back is described in Receive Context Descriptor.

Receive Normal Descriptor (Read Format)

The read format for a receive normal descriptor is made up of a header or buffer 1 address, reserved field, payload or buffer 2 or next descriptor address, a 30-bit reserved filed, OWN bit, and an interrupt bit.

The *Receive Normal Descriptor Read Format* figure shows the read format for a receive normal descriptor.

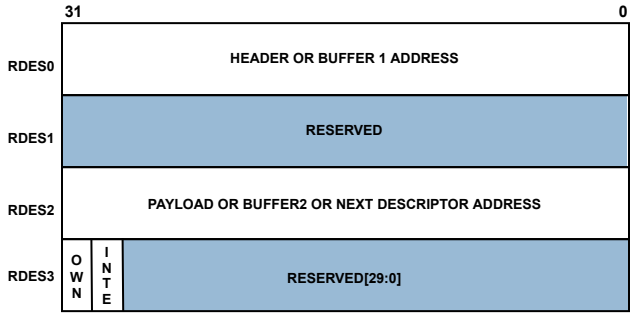


Figure 28-16: Receive Normal Descriptor Read Format

RDES0 Normal Descriptor (Read Format)

The following table explains the read format of the RDES0 normal descriptor.

Table 28-23: Receive Descriptor Fields 0 (RDES0)

Bit	Name	Description
31–0	BUFF1AP	Header or Buffer 1 Address Pointer. These bits indicate the physical address of buffer 1. There are no limit when the SPH bit of the control register of a channel is reset, these bits indicate the physical address of buffer 1. When the SPH bit is set, these bits indicate the physical address of header buffer, where the receive DMA writes the L2/L3/L4 header bytes of the received packet. The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES0[1:0] as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data width.

RDES1 Normal Descriptor (Read Format)

The following table describes the read format of the RDES1 normal descriptor.

Table 28-24: Receive Descriptor Fields 1 (RDES1)

Bit	Name	Description
31–0	Reserved	Reserved

RDES2 Normal Descriptor (Read Format)

The following table describes the read format of the RDES2 normal descriptor.

Table 28-25: Receive Descriptor Fields 2 (RDES2)

Bit	Name	Description
31–0	Reserved or BUF1AP	Buffer 2 Address Pointer. These bits indicate the physical address of buffer 2. When the `EMAC_DMA[n]_CTL.SPH` bit is set, the buffer address pointer must be bus width-aligned, that is, RDES2[3:0, 2:0, or 1:0] = 0 corresponding to 128, 64, or 32 bus width. LSBs are ignored internally. When the `EMAC_DMA[n]_CTL.SPH` bit is reset, there is no limitations on the RDES2 value. However, the receive DMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP and writes to the complete location.

RDES3 Normal Descriptor (Read Format)

The following table describes the read format of the RDES3 normal descriptor.

Table 28-26: Receive Descriptor Fields 3 (RDES3)

Bit	Name	Description
31	OWN	Own Bit. When this bit is set, it indicates that the EMAC module owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	IOC	Interrupt Enabled on Completion. When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29–26	Reserved	
25	BUF2V	Buffer 2 Address Valid. When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address, to which the buffer 2 address in RDES2 is pointing, to write received packet data.
24	BUF1V	Buffer 1 Address Valid. When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by Buffer 1 address in RDES1 can be used by the DMA to write received packet data.
23–16	Reserved	
15–0	Reserved	Reserved

Receive Normal Descriptor (Write-Back Format)

The *Receive Normal Descriptor Write-Back Format* figure illustrates the write-back format for a receive normal descriptor.

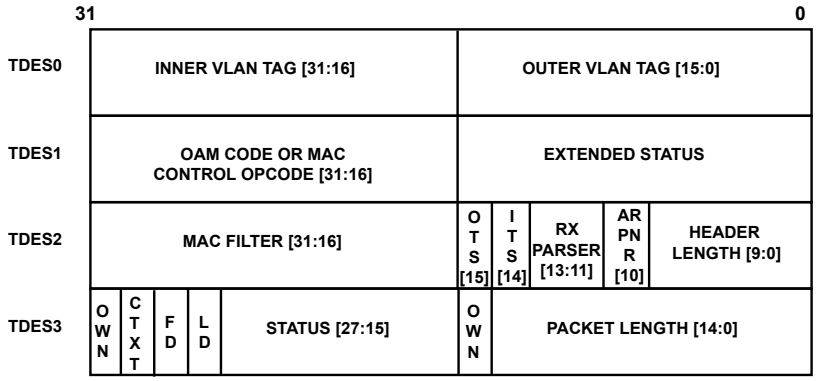


Figure 28-17: Receive Normal Descriptor Write-Back Format

RDES0 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES0 normal descriptor.

Table 28-27: RDES0 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–16	IVT	Inner VLAN Tag. This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when double VLAN tag processing and VLAN tag stripping are enabled.
15–0	OVT	Outer VLAN Tag. This field contains the outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

RDES1 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES1 normal descriptor.

Table 28-28: RDES1 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–16	OPC	OAM Subtype Code, or MAC Control Packet opcode. OAM subtype code: If bits [18:16] of RDES3 are set to 3'b111, this field contains the OAM subtype and code fields. MAC Control Packet opcode: The bits [15:8] of RDES3 contains the subtype and bits [7:0] contains the code.
15	TD	Time Stamp Dropped. This bit indicates that the time stamp was captured for this packet but was dropped in the MTL receive FIFO because of overflow. This bit is available only when the time stamp feature is selected. Otherwise, this bit is reserved.
14	TSA	Time Stamp Available. When time stamp is present, this bit indicates that the time stamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the last descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.

Table 28-28: RDES1 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
13	PV	PTP Version. This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format. This bit is available only when time stamp feature is selected. Otherwise, this bit is reserved.
12	PFT	PTP Packet Type. This bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when time stamp feature is selected. Otherwise, this bit is reserved.
11–8	PMT	PTP Message Type. These bits are encoded to give the type of the message received: • 0000 = No PTP message received • 0001 = SYNC (all clock types) • 0010 = Follow_Up (all clock types) • 0011 = Delay_Req (all clock types) • 0100 = Delay_Resp (all clock types) • 0101 = Pdelay_Req (in peer-to-peer transparent clock) • 0110 = Pdelay_Resp (in peer-to-peer transparent clock) • 0111 = Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000 = Announce • 1001 = Management • 1010 = Signaling • 1011–1110 = Reserved • 1111 = PTP packet with Reserved message type These bits are available only when the time stamp feature is selected.
7	IPCE	IP Payload Error. When this bit is set, it indicates either of the following: • The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. • The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. • The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.
6	IPCB	IP Checksum Bypassed. This bit indicates that the checksum offload engine is bypassed. This bit is available when the enable receive TCP/IP checksum check feature is selected.
5	IPV6	IPv6 Header Present. This bit indicates that an IPV6 header is detected. When the enable split header feature option is selected and the SPH bit of control register of a channel is set, the IPV6 header is available in the header buffer area to which RDES0 is pointing.

Table 28-28: RDES1 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
4	IPV4	IPV4 Header Present. This bit indicates that an IPV4 header is detected. When the SPH bit of RDES3 is set, the IPV4 header is available in the header buffer area to which RDES0 is pointing.
3	IPHE	IP Header Error. When this bit is set, it indicates either of the following: • The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. • The IP datagram version is not consistent with the Ethernet type value. • Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set. This bit is available when the enable receive TCP/IP checksum check feature is selected.
2–0	PT	Payload Type. These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): • 000: Unknown type or IP/AV payload not processed • 001 = UDP • 010 = TCP • 011 = ICMP • 110 = AV Tagged Data Packet • 111 = AV Tagged Control Packet • 101 = AV Untagged Control Packet • 100 = IGMP if IPV4 Header Present bit is set else DCB (LLDP) Control Packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 000.

RDES2 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES2 normal descriptor.

Table 28-29: RDES2 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–29	L3L4FM	Layer 3 and Layer 4 Filter Number Matched. These bits indicate the number of the layer 3 and layer 4 filter that matched the received packet: • 000 = Filter 0 • 001 = Filter 1 • 010 = Filter 2 • 011 = Filter 3 • 100 = Filter 4 • 101 = Filter 5 • 110 = Filter 6 • 111 = Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter. Note: This status is not available when flexible RX parser is enabled.
28	L4FM	Layer 4 Filter Match. When this bit is set, it indicates that the received packet matches one of the enabled layer 4 port number fields. This status is given only when one of the following conditions is true: • Layer 3 fields are not enabled and all enabled layer 4 fields match • All enabled layer 3 and layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31–29]. Note: This status is not available when the flexible receive parser is enabled
27	L3FM	Layer 3 Filter Match. When this bit is set, it indicates that the received packet matches one of the enabled layer 4 port number fields. This status is given only when one of the following conditions is true: • All enabled layer 3 fields match and all enabled layer 4 fields are bypassed • All enabled filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31–29]. Note: This status is not available when the flexible receive parser is enabled
26–19	MADRM	MAC Address Match or Hash Value. When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the hash value computed by the MAC. A packet passes the hash filter when the bit corresponding to the hash value is set in the hash filter register. Note: This status is not available when the flexible receive parser is enabled
18	HF	Hash Filter Status. When this bit is set, it indicates that the packet passed the MAC address hash filter. Bits[26:19] indicate the hash value. Note: This status is not available when the flexible receive parser is enabled.

Table 28-29: RDES2 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
17	DAF/RXPI	Destination Address Filter Fail. When flexible RX parser is disabled, and this bit is set, it indicates that the packet failed the DA filter in the MAC. When flexible RX parser is enabled, this bit is set to indicate that the packet parsing is incomplete (RXPI) due to ECC error. Note: When this bit is set, ES bit of RDES3 is also set.
16	SAF/RXPD	SA Address Filter Fail. When the flexible receive parser is disabled, and this bit is set, it indicates that the packet failed the SA filter in the MAC. When the flexible receive parser is enabled, this bit is set to indicate that the packet is dropped (RXPD) by the parser. Note: When this bit is set, ES bit of RDES3 is also set.
15	OTS	Outer VLAN Tag Filter Status. This bit is valid for both single and double VLAN tagged frames.
14	ITS	Inner VLAN Tag Filter Status (ITS). This bit is valid only for double VLAN tagged frames, when double VLAN processing is enabled. For more information, see the Filter Status topic.
13–11	RxParser	Rx Parser • 000 = Rx Parser filter passed • 001 = Packet rejected in Rx Parser • 010 = Rx Parser overflow errors • 011 = Incomplete Rx parsing of the packet • 100 = AF = 1, RF = 1, Bypass • 101 = RA = 1 and L2 filter fails (Rx parser bypass) • 110 = Reserved • 111 = Incomplete parsing due to ECC error
10	ARPNR	ARP Reply Not Generated. When this bit is set, it indicates that the MAC did not generate the ARP reply for received ARP request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time). This bit is reserved when the Enable IPv4 ARP Offload option is not selected.
9–0	HL	L3/L4 Header Length. This field contains the length of the header of the packet split by the MAC at L3 or L4 header boundary as identified by the MAC receiver. This field is valid only when the first descriptor bit is set (FD = 1). The header data is written to the buffer 1 address of corresponding descriptor. If header length is zero, this field is not valid. It implies that the MAC did not identify and split the header. This field is valid when the enable split header feature option is selected.

RDES3 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES3 normal descriptor.

Table 28-30: RDES3 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31	OWN	Ownership. When this bit is set, it indicates that the EMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	CTXT	Receive Context Descriptor. When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} • 00 = Intermediate Descriptor • 01 = First Descriptor • 10 = Reserved • 11 = Descriptor Error (due to all 1s) **NOTE:** When a descriptor error occurs, the receive DMA closes the receive descriptor indicating descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. Receive DMA sets the CDE field of the `EMAC_DMA[n]_STAT` register but does not the RI field even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	FD	First Descriptor. When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. See the CTXT bit description for details of using the CTXT bit and FD bit together.
28	LD	Last Descriptor. When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	Receive Status RDES2 Valid. When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	Receive Status RDES1 Valid. When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	Receive Status RDES0 Valid. When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
24	CE	CRC Error. When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	Giant Packet. When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Note: Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	Receive Watchdog Timeout When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.

Table 28-30: RDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
21	OE	Overflow Error. When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. Note: This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	Receive Error. When this bit is set, it indicates that the gmii_rxer_i signal is asserted while the gmii_rxdv_i signal is asserted during packet reception. This error also includes carrier extension error in the GMII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	Dribble Bit Error. When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18–16	LT	Length/Type Field. This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: • 000 = The packet is a length packet • 001 = The packet is a type packet • 011 = The packet is a ARP Request packet type • 100 = The packet is a type packet with VLAN Tag • 101 = The packet is a type packet with Double VLAN Tag • 110 = The packet is a MAC Control packet type • 111 = The packet is a OAM packet type • 010 = Reserved
15	ES	Error Summary. When this bit is set, it indicates the logical OR of the following bits: • RDES3[24]: CRC Error • RDES3[19]: Dribble Error • RDES3[20]: Receive Error • RDES3[22]: Watchdog Timeout • RDES3[21]: Overflow Error • RDES3[23]: Giant Packet • RDES2[17]: Destination Address Filter Fail, when flexible RX parser is enabled • RDES2[16]: SA Address Filter Fail, when flexible RX parser is enabled This field is valid only when the LD bit of RDES3 is set.
14–0	PL	Packet Length. These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the last descriptor and error summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

Receive Context Descriptor

This descriptor is read-only for the application. Only the DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. The bit 30 of RDES3 indicates the context type descriptor.

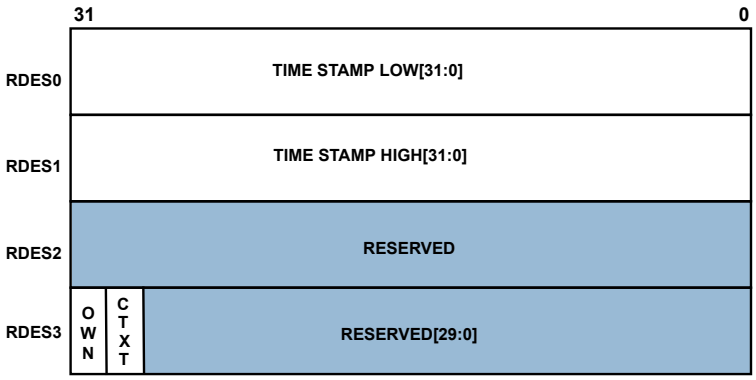


Figure 28-18: Receive Context Descriptor

RDES0 Context Descriptor

The following table describes the format for the RDES0 context descriptor.

Table 28-31: RDES0 Context Descriptor

Bit	Name	Description
31–0	RTSL	Receive Packet Time Stamp Low. The DMA updates this field with least significant 32 bits of the time stamp captured for corresponding receive packet. When this field and the RTSH field of RDES1 show all-one value, the time stamp must be considered as corrupt.

RDES1 Context Descriptor

The following table describes the format for the RDES1 context descriptor.

Table 28-32: RDES1 Context Descriptor

Bit	Name	Description
31–0	RTSH	Receive Packet Time Stamp High. The DMA updates this field with most significant 32 bits of the time stamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the time stamp must be considered as corrupt.

RDES2 Context Descriptor

The following table describes the format for the RDES2 context descriptor.

Table 28-33: RDES2 Context Descriptor

Bit	Name	Description
31–0	Reserved	

RDES3 Context Descriptor

The following table describes the format for the RDES3 context descriptor.

Table 28-34: RDES3 Context Descriptor

Bit	Name	Description
31	OWN	Ownership. When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	CTXT	Receive Context Descriptor. When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 10 to this bit for context descriptor. DMA writes 11 to indicate a descriptor error due to all 1s. When CTXT and DE bits are used together, {CTXT, DE} • 00 = Reserved • 01 = Reserved • 10 = Context Descriptor • 11 = Descriptor Error **NOTE:** When a descriptor error occurs, the receive DMA closes the receive descriptor indicating a descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the `EMAC_DMA[n]_STAT.CDE` bit but does not set the `EMAC_DMA[n]_STAT.RI` bit field even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	DE	Descriptor Error. See the CTXT bit description for details of using the DE bit along with the CTXT bit.
28–0	Reserved	

Descriptor Structure for Split Header Support

EMAC supports splitting the incoming receive packet header such that the header and the payload are stored in different buffers (buffer-1 and buffer-2) in the system memory. For more details about splitting header on receive packets, see Header-Payload Split.

The *Descriptors without Split Header Feature* figure shows the descriptor structure without the split header feature.

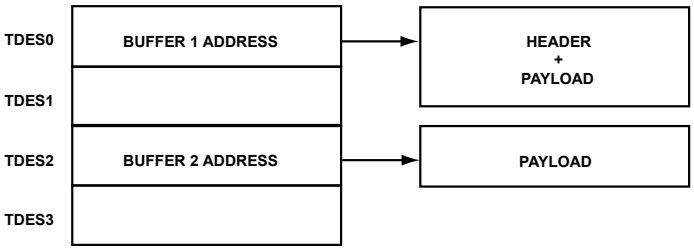


Figure 28-19: Descriptors without Split Header Feature

The *Descriptors with Split Header Feature* figure shows the descriptor structure with the split header feature.

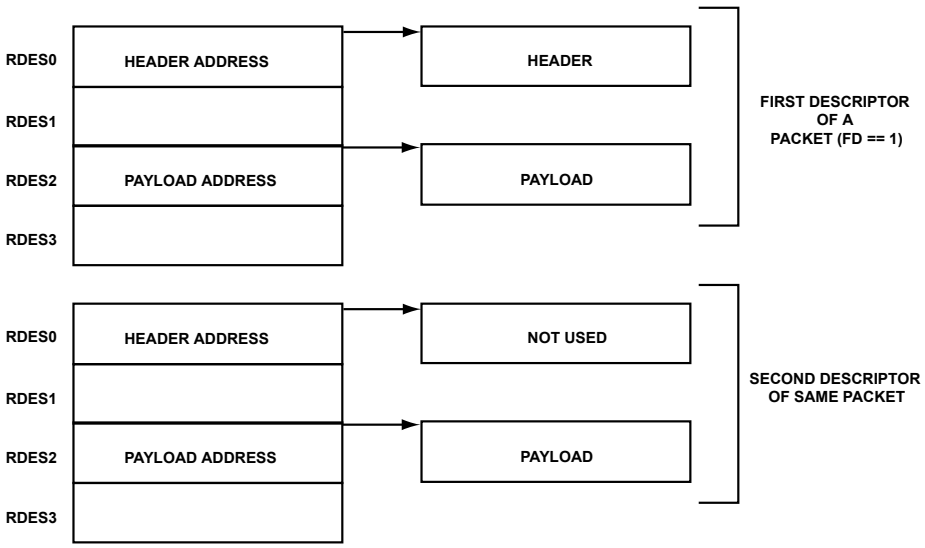


Figure 28-20: Descriptors with Split Header Feature

The DMA writes the header of the received packet by using the header address to which the RDES0 in the first descriptor is pointing (FD bit of RDES3 is set). The DMA writes the payload of the received packet into the buffer address to which the RDES2 is pointing. For subsequent descriptors (FD is set to 0), the address to which RDES0 (Header address) is pointing is not used. The payload is written only to buffers to which the RDES2 (payload address) is pointing.

The `EMAC_MAC_EXT_CFG.HDSMS` bit field indicates the maximum header size allowed for splitting the header data in the received packet based on the value programmed. When the `EMAC_DMA[n]_RXCTL2.ARBS` bit field is programmed to a non-zero value, it indicates the receive buffer size for buffer-1 and the `EMAC_DMA[n]_RXCTL.RBSZ_X_0` and `EMAC_DMA[n]_RXCTL.RBSZ_13_Y` indicates the receive buffer size for buffer-2. The `EMAC_DMA[n]_RXCTL2.ARBS` field can be set to 0, for backward compatibility.

The value in the `EMAC_MAC_EXT_CFG.HDSMS` field should be greater than or equal to the Buffer-1 size indicated by the `EMAC_DMA[n]_RXCTL2.ARBS` field for all the channels.

The following table shows how DMA processes a packet based on the packet type.

Table 28-35: Split Header Support Depending on the Packet Type

Packet Type	Name	Description
TCP or UDP	00 (L3L4 split	The DMA writes the Ethernet header + IP header + TCP or UDP header into the header buffer.
IP (not TCP/UDP)		The DMA writes the Ethernet header + IP header into the header buffer.
Non-IP		The DMA does not split the header and payload
Any packet	01 (L2 split)	The DMA writes the Ethernet header based on split offset (SPLOFST) into the header buffer.
IP	10 (Combination of L2 or L3/L4 split)	L3/L4 split
Non-IP		L2 split
NA	11	Reserved

The IP header includes IPv4 options in case of IPv4 packet and IPv6 extension headers in case of IPv6 frames. The points at which the header is split are shown in the *L3/L4 Header Split Points* figure.

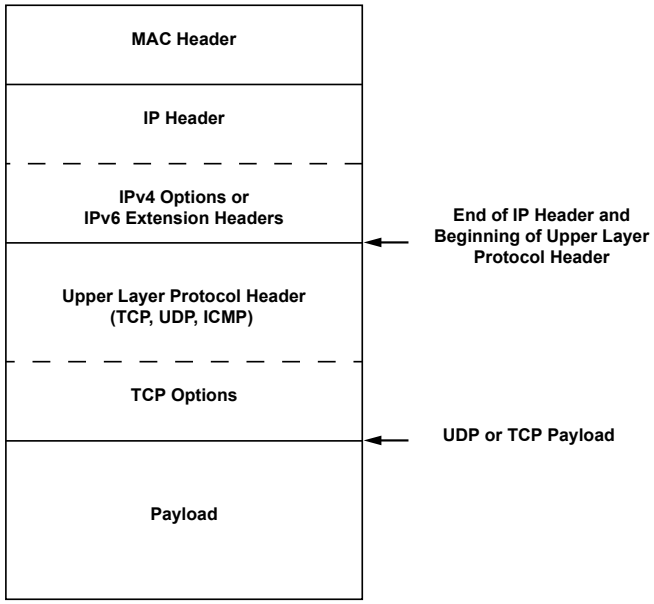


Figure 28-21: L3/L4 Header Split Points

NOTE:
- The VLAN tag stripping must be set for the split function. For instance, the DMA separates the header and payload of an untagged packet only. So, when a tagged packet is received, then program EMAC such that the VLAN tags are deleted/stripped from the received packets.
- L3/L4 split is applicable for IP packets that are either untagged or VLAN stripped. If VLAN tag is retained in the packet forwarded to the DMA, L3-L4 split is not performed. However, if the `EMAC_SPLM_OFST_CFG.SPLM` bit field is set to 2, L2 split is performed for VLAN tagged IP packets.

- For AV packets, to specify an alternative L2 split value, use the EMAC_SPLM_OFST_CFG.SAVE and EMAC_SPLM_OFST_CFG.SAVO bit fields.
- L2 Split is not supported for packets with variable preamble.

The following table shows the header length availability in RDES2.

Table 28-36: Header Length Availability in RDES2

First Descriptor (FD) Value	Last Descriptor (LD) Value	Header Length Availability in RDES2
0	1	Header length not available
1	0	HL[9:0] for all packets
1	1	For IP packets, HL[9:0] For non-IP packets, HL[9:2]

The DMA writes the header length in RDES2 of the first receive descriptor (RDES3[29] FD bit is set) for the packet. The packet length is written in RDES3 of the last receive descriptor (RDES3[28] LD bit set).

The buffer length for the payload is set by the driver through the EMAC_DMA[n]_RXCTL.RBSZ_X_0 and EMAC_DMA[n]_RXCTL.RBSZ_13_Y fields.

The DMA fills receive buffers fully in all except the last descriptor. The header length is taken to be the value based on the bits programmed in the EMAC_MAC_EXT_CFG.HDSMS field.

Enhanced Descriptor for Time-Based Scheduling

The time-based scheduling feature needs Enhanced Descriptors (that are 32 Bytes) to be enabled on all the DMA channels that intend to use the feature (by setting the EMAC_DMA[n]_TXCTL.EDSE bit).

The structure of the 32-byte descriptor for the context and the normal descriptor in read and write formats are described in the following sections.

Enhanced Normal Descriptor - Read

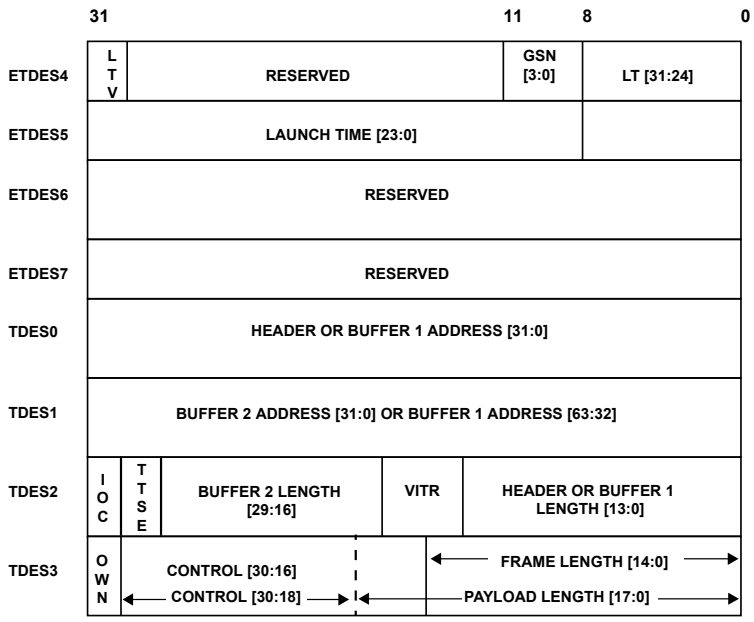


Figure 28-22: Enhanced Normal Descriptor - Read (32-bit Mode)

Table 28-37: Enhanced Normal Descriptor - Read

Bit	Name	Description
31–12	LTV	Launch Time Valid indicates the Launch Time (LT) and GSN fields present in the descriptor are valid. The LTV must be set only if the FD bit of the descriptor is not set.
11–8	GSN	GCL slot number associated with the packet
7–0	LT	Launch Time associated with the packet.

For details of the other fields, see Transmit Normal Descriptor (Read Format).

Enhanced Normal Descriptor - Write

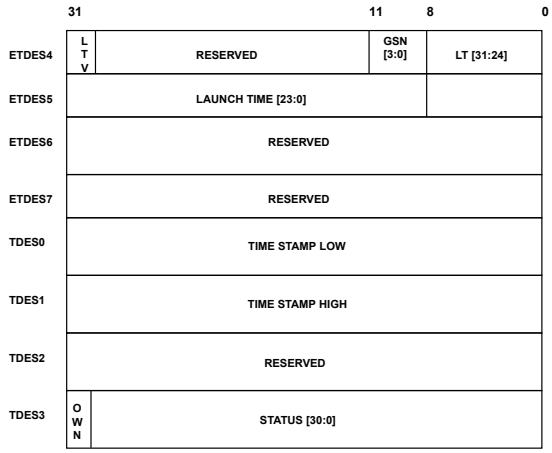


Figure 28-23: Enhanced Normal Descriptor - Write (32-Bit Mode)

For details of LTV, GSN, LT fields, see Enhanced Descriptor for Time-Based Scheduling. For details of the other fields, see Transmit Normal Descriptor (Write-Back Format).

NOTE: In both the write back formats, no modifications can be done to the extended 16 bytes. The rest of the 16 bytes (TDESC0 to TDESC3) are written back as per the previous 16 bytes Descriptor format.

NOTE: Fetch time when enabled overrides the AV slot function

NOTE: When an unaligned new GCL list (with a different CTR) is installed it is recommended not to have traffic during the installation of the new list. Any traffic during the switching of the lists might have unpredictable behavior regarding Fetch, Launch, and Launch expiry as the CTR and BTR values get updated while the frame is being processed.

Enhanced Context Descriptor (Read)

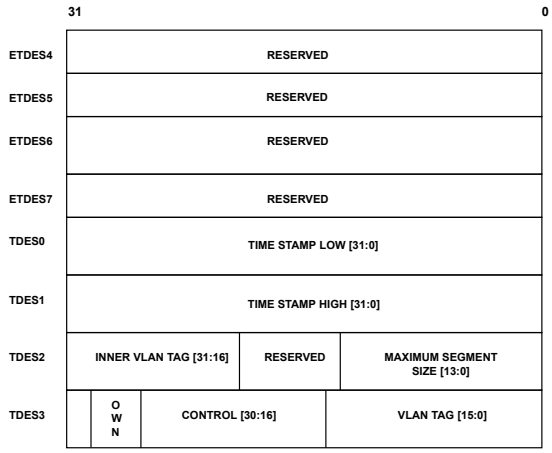


Figure 28-24: Enhanced Context Descriptor - Read

For details of the various fields, see Transmit Context Descriptor.

Enhanced Context Descriptor (Write)

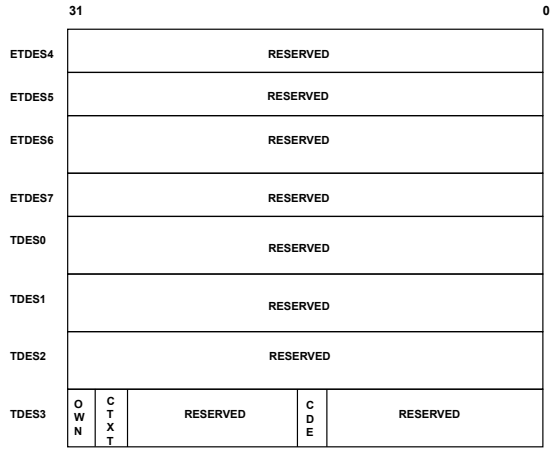


Figure 28-25: Enhanced Context Descriptor (Write)

For details of the various fields, see Receive Context Descriptor.

MAC Transaction Layer (MTL)

The MAC Transaction Layer (MTL) provides the FIFO memory interface to the buffer and regulates the packets between the application system memory and the MAC. It also enables the data to transfer between the application and MAC clock domains. The MTL layer has transmit and receive data paths. The data path for both directions is 32 bits wide and each transmit/receive DMA channel has a dedicated FIFO.

The transmit FIFO size for EMAC0 is 2048 bytes. The receive FIFO size for EMAC0 is 2048 bytes.

Transmit Path and Operation

The DMA engine controls all transactions for the transmit path with the application. The DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or the complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the DMA.

The queue fill level is indicated to DMA (using PBL and watermark) so that it can initiate a data fetch in required bursts from the application or system memory. The DMA indicates the SOP and EOP as packet delimiters through the Application Transmit Interface (ATI). In configurations with multiple queues, the DMA should also indicate the queue number for which the transaction is being addressed.

The following modes of operation trigger reading the data towards the MAC.

1. Threshold mode. When the number of bytes in the FIFO queue crosses the configured threshold level, (or when the end of packet is written before the threshold is crossed), the data is ready to be popped out and

forwarded to the MAC. The threshold level is configured using the EMAC_TQ0_OPMODE.TTC bit field that corresponds to an MTL queue.

2. Store-and-forward mode. In this mode, the MTL pops the packet towards the MAC only when one or more of the following conditions are true:

   - A complete packet is stored in the queue
   - The transmit FIFO becomes almost full
   - The ATI watermark becomes low

   The watermark becomes low when the requested queue does not have space to accommodate the requested burst length on the ATI. Therefore, the MTL, when operating in the store-and-forward mode, allows the packet transmission even if the packet length is bigger than the transmit queue size. The application can flush the complete content of the transmit queue using the EMAC_TQ0_OPMODE.FTQ bit that corresponds to a MTL queue. This bit is self-clearing and initializes the queue pointers to the default state.

   If the EMAC_TQ0_OPMODE.FTQ bit is set during a packet transfer from the MTL to the MAC, the MTL stops further transfer because the queue is empty. Therefore, an underflow event occurs at the MAC transmitter. For information about the initialization and transmit operations for the MTL layer, see the following sections.

Initialization Flow

At reset, the MTL is ready to manage the flow of data to and from the application or DMA, and MAC. For configurations with a single transmit queue, there are no initialization requirements for enabling the MTL. For configurations with multiple transmit queues, the program must initialize the queue size for each of the queues using the EMAC_TQ0_OPMODE.TQS bits that correspond to a transmit queue. The program must also initialize the MAC block. The DMA controllers must be individually enabled through their respective registers.

Transmit DMA Channels and Queues

The number of transmit DMA channels is equal to the transmit queues. This is because the DMA is designed to arbitrate among multiple channels (to fetch the descriptor and packet data from the system memory) in terms of PBLs (Programmable Burst Lengths). However, the transmit queues are packet-based storage memory. Therefore, one to one mapping is required between the DMA channels and transmit queues to maintain the packet-level coherency.

Transmit Operation with Multiple Packets in the Buffer

The transmit FIFO can be configured to accept up to two packets at a time.

Retransmission During Collision

While a packet is being transferred from the MTL to the MAC, a collision event can occur on the MAC line interface in half-duplex mode. The MAC indicates a retry attempt to the MTL by giving the status even before the EOP is transferred from the MTL. The MTL then enables the retransmission by popping out the packet again from the queue.

After more than 96 bytes (or 548 bytes in 1000-Mbps mode) are read out towards the MAC, the queue controller frees up that space and makes it available to the application or the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the MAC indicates a late-collision event.

Transmit Queue Flush Operation

The MTL allows a transmit Queue to be flushed at any moment using the EMAC_TQ0_OPMODE.FTQ bit. The application can read back the same bit (cleared after flush is complete) to check the status of the flush operation.

Receive Path and Operation

The MTL receive module receives packets from the MAC and pushes them into the receive queue. The status (fill level) of the queue is indicated to the DMA when it crosses the configured receive threshold (EMAC_RQ0_OPMODE.RTC bits of the corresponding MTL queue), or the complete packet is received. The MTL also indicates the fill level of the queue so that the DMA can initiate preconfigured burst transfers towards the SCB interface.

The receive operation occurs as follows:

1. When the MAC receives a packet, it indicates the availability of receive data.
2. The MAC indicates the SOP and EOP delimiters.
3. The MTL accepts the data and pushes it into corresponding receive queue.
4. After the EOP is transferred, the MAC drives the status word which is also pushed into the corresponding receive queue by the MTL.

   NOTE: In Threshold (cut-through) mode, the status words are stored after the packet EOP. In store-and-forward mode, the location for the maximum status words are reserved before writing the SOP and the status is written to reserved locations after writing the EOP.

5. If IEEE 1588 time stamp feature is enabled and the 64-bit time stamp is available along with the packet status, it is pushed into the receive queue as a part of the status word. Therefore, two additional locations are taken per packet to store the time stamp in the receive queue.
6. The MTL takes the data out of the FIFO and sends it to the DMA.
   - Threshold mode (default). In this mode, the MTL reads the data and indicates its availability to the application or DMA when one of the following occurs:
     - Data bytes equal to the threshold amount are written to the receive queue (EMAC_RQ0_OPMODE.RTC bit field value)
     - A full packet of data is received into the queue
   - Receive FIFO Store-and-forward mode. This mode is configured using EMAC_RQ0_OPMODE.RSF bit. The initial receive queue locations are reserved for the status words before writing the SOP. A packet is read out only after it is completely written into the receive queue. In this mode, all error packets are dropped (if configured through the EMAC_RQ0_OPMODE.FEP of a queue) such that only valid packets are read and forwarded to the application.

Multi-Packet Receive Operation

In Threshold mode, the packet status is available immediately after the packet data. In store-and-forward mode, the packet data is available after the packet status. The MTL can store any number of packets into the queue when it is not full. When the MAC receives a packet and the corresponding receive queue is full, the MTL ignores that packet and an overflow event occurs. Additionally, the MTL increments the overflow counter (EMAC_RQ0_MSPKTOF_CNT.OVFPKTCNT) of the corresponding queue.

Receive Error Handling

If the MTL receive queue is full before it receives the EOP data from the MAC, the following happens:

- An overflow is declared
- The whole packet (including the status word) is dropped
- The overflow counter in the DMA (EMAC_RQ0_MSPKTOF_CNT.OVFPKTCNT bits of corresponding MTL queue) is incremented.

This is true even if the EMAC_RQ0_OPMODE.FEP bit of the corresponding MTL queue is set.

If the start address of such a packet has already been transferred to the read controller, the rest of the packet is dropped and a dummy EOP is written to the queue along with the status word with overflow status. The status indicates a partial packet because of overflow. In such packets, the packet length field is invalid. If the MTL receive queue is configured to operate in the store-and-forward mode and the length of the received packet is more than the queue size, overflow occurs, and all such packets are dropped.

The MTL receive control logic can filter error and undersized packets using the EMAC_RQ0_OPMODE.FEP and EMAC_RQ0_OPMODE.FUP bits of the corresponding MTL queue. If the start address of such a packet has already been transferred to the receive queue read controller, that packet is not filtered. The start address of the packet is transferred to the Read Controller after the packet crosses the receive threshold set by the EMAC_RQ0_OPMODE.RTC of the corresponding MTL queue.

EMAC CORE

The EMAC CORE is the lowest block in the EMAC peripheral and it performs all operations with the external world (PHY chip). It has independent transmit and receive modules. The modules interact with the EMAC Transaction Layer (MTL) at one end and interacts with the PHY chip through the RGMII/RMII/MII interface at the other end. Both modules have several sub blocks which are discussed in subsequent sections.

Transmission is initiated when the MTL pushes in data with the start of packet (SOP). The CORE then transmits to the RGMII/RMII/MII. After the end of packet transfers out, the CORE gives the status of the transmission back to the MTL. The MTL forwards the transmission to the application through DMA.

A receive operation initiates when the EMAC detects an SFD on the RGMII/RMII/MII. The CORE strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The frame drops in the core when it fails the address filter.

NOTE: The term *CORE* (written in capitals) refers to the internal block of Ethernet peripheral. Do not confuse the term with the *processor core*.

EMAC CORE Transmission Engine

The following modules constitute the transmission function (transmission engine components) of the EMAC:

- Transmit Bus Interface Module (TBU)
- Transmit Packet Controller Module (TPC)
- Transmit Protocol Engine Module (TPE)
- Transmit Scheduler Module (STX)
- Transmit CRC Generator Module (CTX)
- Transmit Flow Control Module (FTX)

Transmit Bus Interface Module (TBU)

This module interfaces the transmit path of the EMAC CORE with the MAC Layer FIFO interface. This module outputs the transmit status to the application at the end of normal transmission or collision.

Transmit Packet Controller Module (TPC)

The Transmit Packet Controller (TPC) module consists of eight registers to hold the data and the last data control received from the TBU.

The register provides a buffer between the application and the Transmit Protocol Engine (TPE) to regulate data flow.

When the number of bytes received from the application are less than 60 (DA+SA+LT+DATA), the state machine that interfaces with TBU automatically appends zeros to the packet being transmitted. This is done to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE 802.3. The EMAC module can also be programmed to not append any padding.

The cyclic redundancy check (CRC) for the Frame Check Sequence (FCS) field is calculated before transmission to the TPE module. This value is computed by the CTX module. The TPC module receives the computed CRC and appends it to the data being transmitted to the TPE module. When the MAC is programmed to not append the CRC value to the end of Ethernet packets, the TPC module ignores the computed CRC and transmits only the data received from the TBU module to the TPE module. An exception to this rule is that when the MAC is programmed to append pads for packets (DA+SA+LT+DATA) less than 60 bytes sent by the TBU module, the TPC module always appends the CRC at the end of padded packet.

Transmit Protocol Engine Module (TPE)

The Transmit Protocol Engine (TPE) module consists of a Transmit State Machine that controls the operation of Ethernet packet transmission. The Transmit State Machine of this module contains the following features to meet the IEEE 802.3/802.3z specification:

- Generates preamble and SFD
- Generates jam pattern in half-duplex mode after normal collision

- Generates carrier extension in the half-duplex (only in the RGMII) mode when packet is smaller than 512 bytes
- Supports packet bursting in the half-duplex (only in the RGMII) mode
- Jabber timeout
- Flow control for half-duplex mode (back pressure)
- Generates transmit packet status
- Contains time stamp snapshot logic for IEEE 1588 support

When the TPC module requests the TPE module for a new packet transmission, the Transmit State Machine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of 10101010 pattern and the SFD is defined as 1 byte of 10101011 pattern.

The collision window is defined as 1 slot time (512-bit times for 10/100 Mbps Ethernet and 4096-bit times for 1,000 Mbps Ethernet). The jam pattern generation is applicable only to half-duplex mode, not to full duplex mode.

In MII/RMII mode, if a collision occurs any time from the beginning of the packet to the end of the CRC field, the Transmit State Machine sends a 32-bit jam pattern of 0x55555555 on MII/RMII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the Transmit state machine completes the transmission of preamble and SFD, and then sends the jam pattern.

In RGMII mode, if a collision occurs any time between the beginning of the packet and the end of the extension field, the Transmit State Machine sends a 32-bit jam pattern of 0x55555555 on RGMII to inform all other stations of the collision. If the collision is seen during the preamble transmission phase, the Transmit State Machine completes the transmission of preamble and SFD, and then sends the jam pattern. If a collision occurs during the extension field, the Transmit State Machine sends a 32-bit jam pattern of 0x1F1F1F1F.

If the collision occurs after the collision window and before the end of the FCS field (or the end of burst if the packet burst mode is enabled), the Transmit State Machine sends a 32-bit jam pattern and sets the late collision bit in the transmit packet status.

In RGMII half-duplex mode (1000 Mbps), the Transmit State Machine ensures that all valid carrier events exceed a slot time of 4,096-bit times. To accomplish this, any transmit packet shorter than 512 bytes from the TFC module is extended using a carrier extension.

When the packet burst mode is enabled, only the first packet of the burst is carrier extended if it is shorter than 512 bytes. The carrier extension is not applicable for MII/RMII half-duplex and RGMII or MII/RMII full-duplex modes. When the packet burst mode is enabled, the Transmit State Machine transmits a burst of packets (when packets are available from the TFC module) without releasing the carrier of the PHY. To accomplish this, the state machine inserts the carrier extension for a minimum IPG period (96-bit times) between the packets. The Transmit State Machine continues to burst packets when additional packets are available from the TPC module and a burst limit of 8192-byte times is not exceeded. When an additional packet is not available at the end of the IPG period in the middle of the burst, the Transmit State Machine releases the carrier RGMII.

Packet bursting is applicable only for the RGMII half-duplex mode. It is not applicable in the MII/RMII and RGMII full-duplex modes. In the RGMII half-duplex mode, the size of the first packet in a packet burst should at least be equal to the slot time. If the first packet (including carrier extension) is less than the slot time, the MAC considers all data bytes, received for the first packet and subsequent packets, which end immediately after the slot time is reached as first packet in the burst. This may result in CRC error. However, if the packet burst ends (both RXDV and RXER go low) before the slot time, the subsequent packet is considered as a new packet. This behavior is according the IEEE 802.3.

The TPE module maintains a jabber timer to stop the transmission of Ethernet packets if the TFC module transfers more than 2,048 (default) bytes. The timeout is changed to 10,240 bytes when the jumbo packet is enabled.

The Transmit State Machine uses the deferral mechanism for flow control (backpressure) in the half-duplex mode. When the application requests to stop receiving packets, the Transmit State Machine sends a JAM pattern of (8'h55) 32 bytes whenever it senses a reception of a packet, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. The application requests the flow control through by setting the `EMAC_Q[i]_TXFLOW_CTL.FCB_BPA` bit of the corresponding MTL queue. If the application requests a packet to transmit, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort their transmissions because of excessive collisions.

If IEEE 1588 time stamp is enabled for the transmit packet, this block takes a snapshot of the system time when the SFD is put onto the transmit RGMII or MII/RMII bus. The system time source is either an external input or it is internally generated according to the configuration selected.

Transmit Scheduler Module (STX)

The transmit scheduler is responsible for scheduling the frame transmission on the RGMII or MII/RMII. The two major functions of this module are:

- Maintains the inter-packet gap between two transmitted packets. The STX module maintains an idle period of the configured inter-packet gap (`EMAC_MAC_CFG.IPG`) between any two transmitted packets. If packets from the TPC arrive at the TPE module sooner than the configured IPG time, the TPE module waits for the enable signal from the STX module before starting the transmission on GMII or MII. The STX module starts its IPG counter as soon as the carrier signal of RGMII or MII/RMII goes inactive. At the end of the programmed IPG value, the module issues an enable signal to the TPE module in the full-duplex mode.

  In the half-duplex mode and when IPG is configured for 96-bit times, the STX module follows the rule of deference specified in the IEEE 802.3, Section 4.2.3.2.1. The module resets its IPG counter if a carrier is detected during the first two-thirds (64-bit times for all IPG values) of the IPG interval. If the carrier is detected during the final one-third of the IPG interval, the STX module continues the IPG count and enables the transmitter after the IPG interval.

- Implements the Truncated Binary Exponential Back-off algorithm in half-duplex mode.

Transmit CRC Generator Module (CTX)

The MAC Transmit CRC Generator (CTX) module interfaces with the TFC module to generate the CRC for the FCS field of the Ethernet packet.

The TPC module sends the packet data and any necessary padding to the CTX module through an 8-bit interface.

The CTX module calculates the 32-bit CRC for the FCS field of the Ethernet packet. The encoding is defined by the following generating polynomial:

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

The CTX module gets the byte data of Ethernet packet from the TPC module (DA + SA + LT + DATA + PAD) qualified with a data valid signal. The TPC also indicates to the CTX when to reset the previously calculated CRC and to start the new CRC calculation for the coming packet. The TPC module issues the start command before sending the new packet data for calculation. The calculated CRC is valid on the next clock after the data is received.

In RGMII mode, the data valid signal is valid for every clock, from the first data byte through the last data byte. In MII/RMII mode, this signal is valid every alternate clock.

Transmit Flow Control Module (FTX)

The Transmit Flow Control (FTX) module generates and transmits the pause packets to the TFC module based on the flow control triggers in full-duplex mode. The TFC module receives the pause packet from the FTX module, appends the calculated CRC, and sends the packet to the TPE module.

EMAC Core Reception

The following are the functional blocks (reception engine components) in the receive path of the EMAC core.

- Receive Protocol Engine Module (RPE)
- Receive CRC Module (CRX)
- Receive Packet Controller Module (RPC)
- Receive Flow Control Module (FRX)
- Receive Bus Interface Unit Module (RBU)
- Address Filtering Module (AFM)

Receive Protocol Engine Module (RPE)

The Receive Protocol Engine (RPE) consists of the Receive State Machine which strips the preamble, SFD, and carrier extension of the received Ethernet packet (in half-duplex 1000-Mbps mode). The sequence is as follows:

1. When the RX_DV signal of GMII or MII becomes active, the Receive State Machine of RPE starts looking for the SFD field (byte 0x5D in GMII mode; 0xD nibble in MII mode). The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine begins sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).

3. If the IEEE 1588 time stamp feature is enabled, the RPE takes a snapshot of the system time at which SFD of any packet is detected on GMII or MII. If this packet is not dropped during MAC filtering, the time stamp is passed to the application. In MII mode, the RPE converts the received nibble data into bytes and forwards the valid packet data to the RFC module.
4. The Receive State Machine of the RPE module decodes the length/type field of the receiving Ethernet packet.

   If the length/type field is less than 1,536 and if the MAC is programmed for the Auto CRC/pad stripping (`EMAC_MAC_CFG.ACS` bit), the state machine sends the data of the packet up to the count specified in the length/type field and starts dropping bytes (including the FCS field). The state machine of the RPE module decodes the length/type field and checks for the length interpretation.

   NOTE: In Audio Video (AV) or Data Center Bridging (DCB) mode when additional receive queues are selected, the packets that are less than or equal to 16 bytes in length after pad stripping, always get dropped inside the MAC receiver. This happens even if the packets have passed the address filter and have no CRC error.

   If the length/type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if the CRC stripping for type packet has not been enabled with the `EMAC_MAC_CFG.CST` bit. However, if CRC stripping for type packets is enabled and the Receive Checksum Offload Engine is not enabled, the MAC strips and drops the last 4 bytes of all packets of Ether type before forwarding the packets to the application.
5. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets larger than 2,048 (10,240 if Jumbo Packet is enabled) bytes, (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module.

At the end of every received packet, the RPE module generates received packet status and sends it to the RPC module. Control, missed packet, and filter fail status are added to the receive status in the RPC module.

Receive CRC Module (CRX)

The receive CRC module checks for any CRC errors in the receiving frame.

This module calculates the 32-bit CRC for the received frame that includes the destination address field through the FCS field (DA+SA+LT+DATA+PAD+FCS). The following generating polynomial defines the encoding.

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

Irrespective of the auto pad or CRC strip, the CRC module receives the entire frame to compute the CRC check for received frame.

Receive Packet Controller Module (RPC)

The MAC Receive Packet Controller (RPC) receives the Ethernet packet data and status from the RPE module.

The RPC module consists of a FIFO of parameterized depth (default set to 4 deep and 33-bits wide) and two state machines for writing and reading the FIFO. The FIFO holds the received Ethernet packet data and byte enables, along with a control bit to indicate the last data. The state machines manage the FIFO and provide a packet

buffering for the Ethernet packet being received from the RPE module. The following are main functions of the RPC module:

- Converting Data path — converts 8-bit data to 32-bit data to the RBU module
- Packet filtering
- Attaching the calculated IP checksum input form the IPC
- Updating the receive status and forwarding it to RBU

If the EMAC_MACPKT_FILT.RA bit is set, the RPC module initiates the data transfer to the RBU module when 4 bytes of Ethernet data are received from the RPE module. At the end of the data transfer, the RPC module sends out the received packet status that includes the packet filter bits (SA Filterfail and DA Filterfail) and status from the RPC module. These bits are generated based on the filter-fail signals from the AFM module. This status bit indicates to the application whether the received packet has passed the filter controls (both address filter and Packet Filter controls from CSR). The RPC module does not drop any packet on its own in this mode.

If the EMAC_MACPKT_FILT.RA bit is reset, the RPC module performs packet filtering based on the destination or source address. If the application does not want to receive any bad packets such as runt, CRC error packets, the application still needs to perform another level of filtering. The RPC module waits to receive the first 14 bytes of received data (type field) from the RPE module. Until then, the module does not initiate any transfers to the RBU module. After receiving the destination or source address bytes, the RPC checks the filter-fail signal from the AFM module for an address match. On detecting filter-fail from AFB, the packet is dropped at the RPC module and not transferred to the application.

On a delayed filter response from the AFM (this can only occur if the AFM logic is changed), the RPC module waits until the FIFO is full, and then proceeds with the packet transfer to the RBU module. However, the RPC module still takes the delayed response from the AFM module and if it is a (DA or SA) filter failure, it drops the rest of the packet and sends the receive status word (with zero packet-length, CRC error and runt error bits set) immediately indicating the filter fail. If there is no response from the AFM until EOP is transmitted, the filter fail status in the receive status word is updated accordingly.

Receive Flow Control Module (FRX)

The Receive Flow Controller (FRX) detects the pause packet being received and pauses the packet transmission for the delay specified within the received pause packet. The FRX module is enabled only in the full-duplex mode.

Receive Bus Interface Unit Module (RBU)

The Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RPC module into a 32-bit protocol on the Application side.

Address Filtering Module (AFM)

The Address Filtering Module (AFM) performs the destination and source address checking function on all received packets and reports the address filtering status to the RPC module.

The address checking is done based on different parameters (EMAC_MACPKT_FILT register) chosen by the application. These parameters are inputs to the AFM module as control signals, and the AFM module reports the

status of the address filtering based on the combination of these inputs. The AFM module does not filter the receive packets but reports the status of the address filtering (whether to drop the packet or not) to the RFC module. The AFM module also reports address filter status and whether the received packet is a multicast packet or a broadcast packet.

The AFM module probes the 8-bit receive data path between the RPE module and the RFC module and checks the destination and source address field of each incoming packet. In GMII mode, the module takes 8/14 clocks (from the start of packet) to compare the destination/source address of the packet being received. Similarly, in MII mode, the module takes 14/26 clocks (from the start of packet) to compare the destination or source address of the receiving packet. The AFM module gets the physical (MAC) address of the station and the Multicast Hash table from CSR module for address checking. The CSR module provides the packet filter register parameters to AFM.

Source Address, VLAN, CRC Insertion/Replacement

The following sections provide information on Source Address and VLAN Insertion, Replace, or Delete, Queue/Channel Based VLAN Tag Insertion on Transmission, and CRC Insertion and Replacement.

Double VLAN Processing

The EMAC module supports two VLAN tags, namely inner and outer, for processing double VLANs.

This feature is referred called double VLAN tagging where the MAC can process two VLAN tags. With this feature, the EMAC supports:

- Insertion, replacement, or deletion of up to two VLAN tags in the transmit path.
- Packet filtering and stripping based on any one of the two VLAN tags in the receive path. Stripping and providing up to two VLAN tags in the receive path as a part of the receive status.

Transmit Path

The following table describes the double VLAN features supported by the MAC on the transmit side.

Table 28-38: Double VLAN Processing Features in Transmit Path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the EMAC_VLAN_INCL.CSVL bit and the EMAC_INVLAN_INCL.CSVL bit. The EMAC supports processing of any sequence of outer and inner VLAN tags. **NOTE:** The EMAC does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN tag types or whether the insertion or replacement operation results in an invalid sequence of VLAN tag type. Therefore, the application must provide a correct sequence of VLAN tag types and program the MAC so that the VLAN tag types in the transmitted packet contains the correct sequence. The application must ensure the following: • The inner tag should not be S-VLAN when the outer C-VLAN tag insertion is enabled. • The outer tag should not be C-VLAN when the inner S-VLAN tag insertion is enabled. • The inner tag cannot be S-VLAN when the outer tag is replaced with C-VLAN. • The outer tag cannot be C-VLAN when the inner tag is replaced with S-VLAN.
VLAN Tag deletion	Programs can enable the VLAN tag deletion for outer or inner tag using the EMAC_VLAN_INCL.VLC and EMAC_INVLAN_INCL.VLC bit fields, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	Programs can enable the VLAN tag insertion or replacement for outer or inner tag using the EMAC_VLAN_INCL.VLC EMAC and EMAC_INVLAN_INCL.VLC bit fields, respectively. When VLAN tag insertion or replacement is enabled, the EMAC_VLAN_INCL.VLTI bit and the EMAC_INVLAN_INCL.VLTI bit are used to determine whether the VLAN tag is taken from the register or the control word.

Receive Path

The following table describes the features supported by the MAC on the receive side and the corresponding bits in the EMAC_VLANTAG_CTL register.

Table 28-39: Double VLAN Processing in Receive Path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the EMAC_VLANTAG_CTL.ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the EMAC_VLANTAG_CTL.ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EMAC_VLANTAG_CTL.EVLS and EMAC_VLANTAG_CTL.EIVLS bits.
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN tag and type in the Rx status based on the EMAC_VLANTAG_CTL.EVLRXS and EMAC_VLANTAG_CTL.EIVLRXS bits, respectively.

Table 28-39: Double VLAN Processing in Receive Path (Continued)

Feature	Description
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN tag type to match C-VLAN or S-VLAN based on the EMAC_VLANTAG_CTL.DOVLTC bit.

Source Address and VLAN Insert, Replace, or Delete

The EMAC supports inserting or replacing the source address based on the information in the MAC address registers. The MAC can also insert, replace, or delete the VLAN fields (VLAN type and VLAN tag) based on the setting of the EMAC_VLAN_INCL.VLTI bit. Programs can enable the SA insert or replace feature for all transmit packets or selective packets. Similarly, programs can enable the VLAN insert, replace, or delete feature for all transmit packets or selective packets.

Programming Source Address Insert or Replace

The software can use the SA insert or replace feature to instruct the MAC to do the following for transmit packets:

- Insert the content of the MAC address registers in the SA field
- Replace the content of the SA field with the content of the MAC address registers

When SA insert is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the transmit packet and it inserts the content of MAC address registers in the SA field. Similarly, when SA replace is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the destination address field in the transmit packet with the content of the MAC address registers.

The program can enable the SA insert or replace feature for all transmit packets or selective packets:

- Enable SA insert or replace for all packets: program the EMAC_MAC_CFG.SARC bit field
- Enable SA insert or replace for selective packets: program the SA insertion control field (bits [25:23] of TDES3) in the first transmit descriptor of the packet. When bit 25 of TDES3 is set, the SA insertion control field indicates insertion or replacement by the MAC address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by the MAC address0 registers.

If the MAC address1 registers are not enabled, the MAC address0 registers are used to insert or replace irrespective of the value of the most-significant bit of the SA insertion control field.

Programming VLAN Insert, Replace, or Delete

The software can use the VLAN insert, replace, or delete feature to instruct the MAC to do the following for transmit packets:

- Delete the VLAN type and VLAN tag fields
- Insert or replace the VLAN type and VLAN tag fields using the EMAC_VLAN_INCL.VLTI bit as described in the following table.

Table 28-40: Double VLAN Processing in Receive Path

Condition	Description
EMAC_VLAN_INCL.VLTI bit is set	The MAC inserts or replaces the following: • VLAN type field (C-VLAN or S-VLAN as indicated by the EMAC_VLAN_INCL.CSVL bit • VLAN Tag field content of the VT field of transmit context descriptor of the packet
EMAC_VLAN_INCL.VLTI bit is reset	The MAC inserts or replaces the following: • VLAN Type field (C-VLAN or S-VLAN as indicated by the EMAC_VLAN_INCL.CSVL bit • VLAN tag field with the EMAC_VLAN_INCL.VLT bit field

When VLAN replace or delete is enabled, the MAC checks if the VLAN type field (0x8100 or 0x88a8) is present after the DA and SA fields in the transmit packet. The replace or delete operation does not occur if the VLAN type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check for the presence of the VLAN type field in the transmit packet and just inserts the VLAN type and VLAN tag fields.

Programs can enable the VLAN insert, replace, or delete feature for all transmit packets or selective packets:

- Enable VLAN insert, replace, or delete for all packets: program the EMAC_VLAN_INCL.VLC and EMAC_VLAN_INCL.VLP bit fields
- Enable VLAN insert, replace, or delete for selective packets: program the VTIR field of the TDES2 normal descriptor

Queue/Channel Based VLAN Tag Insertion on Transmit

The EMAC supports channel/queue based VLAN tag insertion on all transmitted packets. Queue/channel specific VLAN tag registers are accessed using indirect addressing through the EMAC_VLAN_INCL register. VLAN type and tag values can be independently programmed for each queue/channel. See Programming Queue/Channel Based VLAN Inclusion Registers for more details.

CRC Insert and Replace

The software can use the CRC replace feature to instruct the MAC to replace the FCS field in the transmit frame with the CRC computed by the MAC. This feature works on per-frame basis. To enable the CRC replacement feature, program the CRC replacement control field (bit 26 of TDES3) in the first transmit descriptor of the frame.

Packet Filtering

the *Packet Filtering Sequence* figure shows the filtering sequence for receive packets. The sequence shown in the figure is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the layer filters are not enabled, that filter is bypassed, and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control. For example, when the EMAC_MACPKT_FILT.RA bit is set to 1, all discarded packets are forwarded to the host but with its packet status indicating the specific filter-failure. If RA=0, the EMAC_MACPKT_FILT.VTFE and

EMAC_MACPKT_FILT.IPFE bits of control whether the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

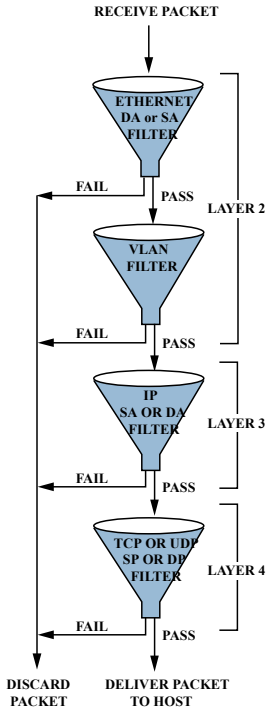


Figure 28-26: Packet Filtering Sequence

Source Address or Destination Address Filtering

The Address Filtering Module of the MAC checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast Destination Address Filtering

The MAC supports 32 MAC addresses for unicast perfect filtering. If perfect filtering is selected (the EMAC_MACPKT_FILT.HUC bit is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default address0 register (EMAC_ADDR0_HI, EMAC_ADDR0_LO) is always enabled.

The address1 through 31 register addresses (both HIGH and LOW) are selected with an individual enable bit. For these addresses, programs can mask each byte during comparison with corresponding received DA byte by setting the corresponding mask byte control bit in the register. This enables group address filtering for the DA.

In hash filtering mode (when the EMAC_MACPKT_FILT.HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a hash table (256 bits for EMAC0). For hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the hash table. A value of 00000 selects bit 0 of the selected register, and a value of 11111 selects bit 63 of the hash table register (registers 7-0). When the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the hash filter; otherwise, the packet is considered to have failed the hash filter.

Multicast Destination Address Filtering

To program the MAC to pass all multicast packets, set the EMAC_MACPKT_FILT.PM bit. If the EMAC_MACPKT_FILT.PM bit is reset, the MAC performs the filtering for multicast addresses based on the EMAC_MACPKT_FILT.HMC bit.

The multicast address is compared with the programmed MAC destination address registers (1–31). Group address filtering is also supported. In hash filtering mode, the MAC performs imperfect filtering using a 64-bit hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the hash table. A value of 000000 selects bit 0 of the selected register and a value of 111111 selects bit 63 of the hash table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the hash filter. Otherwise, the packet is considered to have failed the hash filter.

Hash or Perfect Address Filtering

To configure the DA filter to pass a packet when its DA matches either the hash filter or the perfect filter, set the EMAC_MACPKT_FILT.HPF bit and the corresponding EMAC_MACPKT_FILT.HUC or EMAC_MACPKT_FILT.HMC bits. This is applicable to both unicast and multicast packets. If the EMAC_MACPKT_FILT.HPF bit is reset, only one of the filters (hash or perfect) is applied to the received packet.

Broadcast Address Filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the EMAC_MACPKT_FILT.DBF bit.

Unicast Source Address Filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. Programs can configure the MAC address registers [1–31] to use SA instead of DA for comparison by setting bit 30 of corresponding register.

The MAC also supports group filtering with SA. Programs can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the EMAC_MACPKT_FILT.SAF bit is set. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word. When the EMAC_MACPKT_FILT.SAF bit is set, the SA filter and DA filter result is AND'ed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in order.

Inverse Filtering

For DA and SA filtering, programs can invert the filter-match result at the final output by setting the EMAC_MACPKT_FILT.DAIF and EMAC_MACPKT_FILT.SAIF bits. The EMAC_MACPKT_FILT.DAIF bit is applicable for both unicast and multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

The following tables summarize the DA and SA filtering based on the type of packets received.

NOTE: When the EMAC_MACPKT_FILT.RA bit is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 28-41: Destination Address Filtering

Packet Type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA Filter Operation
Broad-cast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multi-cast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x

Table 28-42: Source Address Filtering

Packet Type	PR	SAIF	SAF	DA Filter Operation
Unicast	1	X	X	Pass all packets
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN Filtering

The following sections provide information on EMAC VLAN filtering.

VLAN Tag Perfect Filtering

In VLAN tag perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. Programs can enable the inverse matching for VLAN packets by setting the `EMAC_VLANTAG_CTL.VTIM` bit. In addition, programs can enable processing of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the `EMAC_VLANTAG_CTL.ESVL` bit. The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

NOTE: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results. By default, the VLAN tag-based perfect filter is available in all configurations.

VLAN Tag Hash Filtering

The 16-bit VLAN hash table is used for group address filtering based on the VLAN tag. The MAC performs the VLAN hash matching using the `EMAC_VLANTAG_CTL.VTHM` (VLAN tag hash table match enable) bit. The MAC provides VLAN tag hash filtering with a 16-bit hash table.

If the `EMAC_VLANTAG_CTL.VTHM` bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the `EMAC_VLAN_HASHTBL` register. A value of 1 in the register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet is forwarded. A value of 0 indicates that VLAN-tagged packet is dropped.

NOTE:
- The 16 or 12 bits of the VLAN tag are considered for CRC-32 computation based on the `EMAC_VLANTAG_CTL.ETV` bit.
- When the bit is reset, most significant four bits of CRC-32 of VLAN tag are inverted and used to index the content of `EMAC_VLANTAG_CTL` register.
- When the bit is set, most significant four bits of CRC-32 of VLAN tag are directly used to index the content of the `EMAC_VLANTAG_CTL` register.

If the VLAN tag of a packet matches the perfect or hash filter, the packet is dropped. If the VLAN perfect and VLAN hash match are enabled, a packet is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse matching is set, a packet is forwarded only when both perfect and hash filters indicate mismatch.

The *VLAN Match Status* table shows the different possibilities for VLAN matching and the final VLAN match status. When the `EMAC_MACPKT_FILT.RA` bit is set, all packets are received, and the VLAN match status is indicated in the VF bit of RDES2 normal descriptor (write-back format). When the RA bit is not set and the

EMAC_MACPKT_FILT.VTFE bit is set, the packet is dropped if the final VLAN match status is fail. In the following table, X signifies that entry can have any value.

When VLAN VID is programmed to 0 in the EMAC_VLANTAG_DAT.VID bit field, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN hash match depends on the EMAC_VLANTAG_CTL.VTHM and EMAC_VLANTAG_CTL.VTIM bits.

Table 28-43: VLAN Match Status

VID	VLAN Perfect Filter Match Result	VTHM Bit	VLAN Hash Filter Match Result	VTIM Bit	Final VLAN Match Status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID! = 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

Extended Receive VLAN Filtering and Routing

The MAC receiver can classify the received packets based on a VLAN tag and steer them to a specific Rx DMA channel. The MAC compares the received frame's VLAN tag with all the enabled and relevant filters and provides a filtering result. If any of the perfect filters give a pass result and if the respective filter's DMA channel number is enabled, the frame is routed to that DMA channel.

In addition to filtering, routing can also be done. For more details about routing, see Extended VLAN Based DMA Selection in Dynamic Mapping.

Comparison Modes

For each VLAN tag filter, the application has the following comparison options:

- It can program the MAC to compare an outer VLAN tag or an inner VLAN tag with the programmed VID.
- It can choose if 12 or 16 bits of the VID field need to be compared.
- Type check can be disabled or enabled for each filter; if enabled, the application can choose if the VID comparison is for SVLAN or CVLAN type frames only. For example, if a filter is enabled for 16-bit comparison,

SVLAN type, and outer VLAN tag, any single or double VLAN tagged frames with outer SVLAN tags are compared with this filter, and a pass or fail result is obtained.

NOTE: The inner VLAN tag comparison is applicable only if double VLAN tag processing is enabled using the EMAC_VLANTAG_CTL.EDVLP bit.

Filtering

When extended receive VLAN filtering and routing is enabled, the application can enable both perfect and hash filtering. The overall VLAN filter result is based on the perfect filter result and the hash filter result (if enabled). The filter result is passed to the application as part of the status bits.

Perfect filtering is done using the EMAC_MAC_VLAN_TAG_FILTER[n] registers (these registers are accessed indirectly using the EMAC_VLANTAG_DAT and EMAC_VLANTAG_CTL registers). For each VLAN tag filter, the MAC compares the relevant VLAN tag ID and gives a result. If any one of the VLAN tag filters gives a match, the frame is considered to have passed the VLAN tag filters. If the frame mismatches all the filters, the frame is considered to have failed the VLAN filter. This behavior is applicable only when the inverse filtering is not enabled in the EMAC_VLANTAG_CTL register.

If inverse filtering is enabled and the frame has mismatched all the relevant filters, then it is considered to have passed the VLAN filter. If the frame matches any one of the relevant filters, then it is considered as a fail. If none of the enabled filters can perform a comparison or if none of the filters are enabled, then the frame is bypassed to the application.

The overall filter result and the values programmed in the EMAC_MACPKT_FILT.VTFE and EMAC_MACPKT_FILT.RA fields determine if the frame must be dropped or forwarded to the application.

- When RA = 1 or VTFE = 0, The MAC forwards the frame, irrespective of the filter result.
- When RA = 0 and VTFE = 1, The MAC forwards the frame only if the VLAN tag filter status is pass. After forwarding a frame to the application, the relevant filter result is indicated through the status bits.

See Programming Extended VLAN Filtering and Routing on Receive.

VLAN Filter Status

The extended receive VLAN filtering and routing feature provides two status bits to indicate the comparison result of the VLAN tags.

By default, the MAC indicates the VLAN filter status through one bit in the status – VF in RDES2. When extended receive VLAN filtering and routing is enabled, two status bits are used to indicate the comparison result of VLAN tags. The outer VLAN tag filter pass and inner VLAN tag filter pass bits are defined in the following positions in various configurations. The status indicated through these bits is updated in RDES2 as follows:

- Bit 15 – Outer VLAN tag filter status
- Bit 14 – Inner VLAN tag filter status

Outer VLAN Tag Filter Status (OTS)

The outer VLAN tag filter pass bit provides the following status:

- In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame's outer VLAN tag has matched one of the VLAN tag filters.
- When this bit is reset, it indicates that the frame's outer VLAN tag has either failed the relevant outer VLAN tag filters or bypassed them.
- When none of the filters are enabled for outer VLAN tag comparison, then this bit is reset.
- When Inverse filtering is enabled and this bit is set, then the frame's VLAN tag has passed all the relevant VLAN tag filters. If it is reset, then it has failed at least one of or bypassed all the filters programmed for outer VLAN tag comparison.
- This bit is valid for both single and double VLAN tagged frames.

Inner VLAN Tag Filter Status (ITS)

The inner VLAN tag filter pass bit provides the following status:

- In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame's inner VLAN tag has matched one of the VLAN tag filters.
- When this bit is reset, it indicates that the frame's inner VLAN tag has either failed the relevant inner VLAN tag filters or bypassed them.
- When none of the filters are enabled for inner VLAN tag comparison, then this bit is reset.
- When inverse filtering is enabled and this bit is set, then the frame's VLAN tag has passed all the relevant VLAN tag filters. When it is reset, then it has failed at least one of or bypassed all the filters programmed for inner VLAN tag comparison.
- This bit is valid for only double VLAN tagged frames, when double VLAN processing is enabled.

The application must look at the status bits and the programming to determine if the frame has passed or failed the VLAN filter.

The *OTS and ITS Bit Values with At Least 1 Perfect Filter Enabled* table and *OTS and ITS Bit Values with Only VLAN Hash Filter Enabled* table show the possible filter combinations and the corresponding filter results. These tables explain the scenarios when double VLAN processing and hash VLAN filter are enabled in the design.

In the following tables:

- `EMAC_VLANTAG_CTL.VTIM`: VLAN tag inverse match enable – bit 17.
- HFO: Hash filter enabled for outer VLAN tag comparison – bit 25 (VHTM) in the `EMAC_VLANTAG_CTL` register.
- HFI: hash filter enabled for inner VLAN tag comparison – bit 27 (ERIVLT) in the `EMAC_VLANTAG_CTL` register.

- PFO – Perfect filter comparison enabled for outer VLAN tag – Any of the EMAC_MAC_VLAN_TAG_FILTER registers is enabled (bit 16 is set) and programmed for outer VLAN tag comparison (bit 20 is set to 0).
- PFI – Perfect filter comparison enabled for inner VLAN tag – Any of the EMAC_MAC_VLAN_TAG_FILTER registers is enabled (bit 16 is set) and programmed for inner VLAN tag comparison (bit 20 is set to 1).
- OTS – Outer VLAN tag filter status
- ITS – Inner VLAN tag filter status

The following table shows the possible values of status bits (OTS and ITS) when at least one perfect filter is enabled.

Table 28-44: OTS and ITS Bit Values with At Least 1 Perfect Filter Enabled

VTIM	HFO	HFI	PFO	PFI	OTS	ITS
0	0	0	0	1	0	1/0
0	0	0	1	0	1/0	0
0	0	0	1	1	1/0	1/0
0	1	0	1	1	1/0	1/0
0	1	0	1	0	1/0	0
0	1	0	0	1	1/0	1/0
0	0	1	1	1	1/0	1/0
0	0	1	1	0	1/0	1/0
0	0	1	0	1	0	1/0
1	0	0	0	1	0	1/0
1	0	0	1	0	1/0	0
1	0	0	1	1	1/0	1/0
1	1	0	1	1	1/0	1/0
1	1	0	1	0	1/0	0
1	1	0	0	1	1/0	1/0
1	0	1	1	1	1/0	1/0
1	0	1	1	0	1/0	1/0
1	0	1	0	1	0	1/0

The following table shows the possible values of status bits (OTS and ITS) when none of the perfect filters are enabled and only the VLAN hash filter is enabled.

Table 28-45: OTS and ITS Bit Values with Only VLAN Hash Filter Enabled

VTIM	HFO	HFI	OTS	ITS
0	0	0	0	0

Table 28-45: OTS and ITS Bit Values with Only VLAN Hash Filter Enabled (Continued)

VTIM	HFO	HFI	OTS	ITS
0	1	0	1/0	0
0	0	1	1/0	1/0
1	0	0	1/0	0
1	1	0	1/0	0
1	0	1	1/0	1/0

- When no perfect filters are enabled, any VLAN packet is considered to have bypassed the perfect filter.
- When VLAN hash filter is enabled for one of the tags, the respective status bit depends on the result of the filter. Status bits are set to 0 when VLAN hash filter is not enabled.
- The value 1/0 for the ITS/OTS field indicates that the final result is dependent on the result of the enabled relevant filter.

Example 1: The second row of the above table indicates that at least one perfect filter is enabled for outer VLAN tag comparison and none of the filters are enabled for Inner VLAN tag comparison. Inverse VLAN filtering is not enabled. The bit OTS is given as 1/0. When the received frame passes at the least one of the enabled Outer VLAN Tag filters, then the bit is set to 1. When the frame does not pass any of the enabled Outer VLAN Tag filters, then the bit is set to 0.

Example 2: Last row of the above table indicates that inverse filtering is enabled, Hash filter and at least one perfect filter is enabled for inner VLAN tag comparison, then when the received frame's inner VLAN tag mismatches with both the hash filter and all the enabled perfect filters, then the frame has the ITS bit set to 1. Otherwise, the bit is set to 0. The OTS bit is set to 0 as the comparison is not performed.

VLAN Stripping

Each of the VLAN tags has individual control over stripping. The programming options of always strip, never strip, strip on pass and strip on fail are available. Inner or outer VLAN tag stripping is based on the pass or fail results of the individual tag. If a tag is bypassed by all the relevant filters, stripping is not applicable for the tag.

- If strip on pass is enabled for the outer VLAN tag, stripping is done only if the outer VLAN tag has passed the relevant filters. The outer VLAN tag filter result bit is set.
- If strip on fail is enabled for the outer VLAN tag, stripping is done only if the outer VLAN tag has failed relevant filters. The Outer VLAN tag filter result bit is reset.
- If the Outer VLAN tag of the received frame is bypassed by the entire filter (no comparison has been made), the tag is not stripped, though the status bit is still 0.
- As multiple filters are enabled, it is possible that the received VLAN frame matches more than one filters. The VLAN tag's value is not always deterministic from the filter status bits.

- If the application strips the VLAN tag based on the filter result, it might lose the VID. If stripping is enabled for any of the tags, the tag can be put in the status. The application must enable the respective EMAC_VLANTAG_CTL.EVLRXS or EMAC_VLANTAG_CTL.EIVLRXS bits.

VLAN Filter Fail Packets Queue

When VLAN filtering is enabled, the VLAN filter fail packets can be routed to a programmable queue (EMAC_RXQ_CTL4.VFFQ) when the EMAC_MACPKT_FILT.RA bit = 1 or EMAC_MACPKT_FILT.VTFE = 0 and the enable bit (EMAC_RXQ_CTL4.VFFQE) for the queue is set.

The packets that pass the VLAN filtering are routed based on the VLAN TAG priority field. The VLAN tag priorities can be assigned to receive queues by programming the PSRQ field in the corresponding EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The packets that fail the VLAN filter are discarded if EMAC_MACPKT_FILT.RA=0 or EMAC_MACPKT_FILT.VTFE =1. However, when RA=1 or VTFE=0, the VLAN filter fail packets are still forwarded to the application. In such a scenario, when the VLAN filter fail queue enable (EMAC_RXQ_CTL4.VFFQE) bit is set, the VLAN filter fail packets are forwarded to the receive queue number programmed in the EMAC_RXQ_CTL4.VFFQ bit. If VFFQE=0, the receive queue number is determined by the VLAN priority mapping as per the PSRQ fields.

The *Receive Queue Routing Table for Unicast Tagged Packets* shows the receive queue routing table for Unicast tagged packets, with the DA/SA filter enabled.

In the following table X = don't care; * = when UFFQE is enabled else PSRQ

Table 28-46: Receive Queue Routing Table for Unicast Tagged Packets

RA	VFTE	SA/DA Filter Result	VLAN Filter Result	VFFQE	Queue Routing
X	X	Pass	Pass	X	PSRQ
0	0	Pass	Fail	0	PSRQ
0	0	Pass	Fail	1	VFFQ
0	X	Fail	X	X	Dropped
0	1	Pass	Fail	X	Dropped
1	X	Fail	X	0	UFFQ*/PSRQ
1	X	Fail	X	1	UFFQ*/PSRQ
1	X	Pass	Fail	0	PSRQ
1	X	Pass	Fail	1	VFFQ

Layer 3 and Layer 4 Frame Filtering

The MAC supports layer 3 and layer 4 based frame filtering. The layer 3 filtering refers to the IP source or destination address filtering in the IPv4 or IPv6 frames whereas layer 4 filtering refers to the source or destination port number filtering in TCP or UDP.

When layer 3 and layer 4 filtering is enabled, the frames are filtered in the following way:

Matched Packets. The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the `EMAC_MAC_CFG.IPC` bit is set and one of the following conditions is true:

- All enabled layer 3 and layer 4 fields match
- At least one of the enabled fields match and other fields are bypassed or disabled

When multiple layer 3 and layer 4 filters are enabled, any filter match is considered as a match. IWhen more than one filter matches, the MAC provides the status of the lowest filter where filter 0 is the lowest filter and filter 3 is the highest filter. For example, when filter 0 and filter 1 match, the MAC gives the status corresponding to filter 0.

NOTE: The source or destination address and VLAN tag filters (if enabled) have precedence over layer 3 and layer 4 filter. This means that a packet which fails the source or destination address, or VLAN tag filter is dropped irrespective of the layer 3 and layer 4 filter results.

Unmatched Packets. The MAC drops the packets that do not match any of the enabled fields. Programs can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets. When a packet is dropped, the aborted or partial packets can be dropped in the MTL Rx FIFO. If the Rx FIFO operates in the threshold (cut-through) mode and the threshold is programmed to a small value, such that packet transfer to application starts before the failed layer 3 and layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

Non-TCP or UDP IP Packets. By default, all non-TCP or UDP IP packets are bypassed from the layer 3 and layer 4 filters. The program can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 Filtering

The EMAC supports perfect matching or inverse matching for IP source address and destination address. The matching compares all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, the program can enable the last four data registers of a register set to contain the 128-bit IP source address or IP destination address. The IP source address or destination address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP source address or destination address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, the program can enable the second and third data registers of a register set to contain the 32-bit IP source address and IP destination address. The remaining two data registers are reserved. The IP source address or destination address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP source address and destination address in the received packet in the higher byte of the respective register.

Layer 4 Filtering

The EMAC supports perfect matching or inverse matching for TCP or UDP source and destination port numbers. However, only one type (TCP or UDP) can be programmed at a time. The first data register contains the 16-bit source and destination port numbers of TCP or UDP, that is, the lower 16 bits for source port number and higher 16 bits for destination port number.

The TCP or UDP source and destination port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP source and destination port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 Filters Registers

The MAC implements a set of registers for layer 3 and layer 4 based frame filtering. In a register set, there is a control register, such as the `EMAC_L3L4_CTL[i]` register (layer 3 and layer 4 control register), to control the frame filtering. In addition, there are five address registers to program the layer 3 and layer 4 fields to be matched, which are:

- `EMAC_L4_ADDR[i]` (layer 4 Address Register)
- `EMAC_L3_ADDR0_REG[i]` (layer 3 address 0 register)
- `EMAC_L3_ADDR1_REG[i]` (layer 3 address 1 register)
- `EMAC_L3_ADDR2_REG[i]` (layer 3 address 2 register)
- `EMAC_L3_ADDR3_REG[i]` (layer 3 address 3 register)

EMAC Station Management Interface (SMI)

The IEEE 802.3 MII station management interface, also known as the MDIO management interface, allows the processor to monitor and control one or more external Ethernet physical-layer transceivers. (Physical-layer transceivers are commonly called PHYs). The management interface physically consists of a 2-wire serial connection composed of the MDC (management data clock) output signal and the MDIO (management data input/output) bidirectional data signal. The IEEE 802.3 MII station management interface applies to both MII/RMII.

The application can address only one register in the PHY in any given time and send control data or receive status information. All the transfers are initiated by the EMAC CORE, and the PHY chip only acts as a target device.

Standard PHY control and status registers typically provide

- Device capability status bits (for example: auto-negotiation, duplex modes, 10/100 speeds, and protocols)
- Device status bits (for example: auto-negotiation complete, link status, remote fault)
- Device control bits (for example: reset, speed selection, loop back, and auto-negotiation start)

Upon power-up, an MDIO read access (at default rates) of device capabilities in PHY status registers can determine the supported PHY features.

The MII management logical interface specifies:

- A set of 16-bit device control or status registers within the PHYs, including both required registers with standardized bit definitions as well as optional vendor-specified registers.
- A 5-bit device addressing scheme which allows the MAC to select one of up to 32 externally connected PHY devices.

- A 5-bit register addressing scheme for selecting the target register within the addressed device.
- A transfer frame protocol for 16-bit read and write accesses to PHY registers through the MDC and MDIO signals under control of the MAC.

Table 28-47: Station Management Interface pins

Station Management Interface Pins	Pin Description
MDIO – Management Data I/O	A periodic clock that runs at a maximum period of 400 ns. Always driven by the EMAC to PHY.
MDC – Management Data Clock	Data signal driven by EMAC or PHY, depending on write or read access based on EMAC; synchronous to MDC.

MDC Clock Frequency

The EMAC uses the `EMAC_MDIO_ADDR.CR` bit field to determine the frequency of MDC as shown in the *MDC Clock Frequency Selection* table. The clock range selection determines the frequency of the clock relative to the CLKO7 frequency. The table shows the suggested range of CLKO7 frequency applicable for each value of the `EMAC_MDIO_ADDR.CR` field. The programmability based on CLKO7 frequency range ensures that the MDC clock frequency range is within the IEEE specifications of 1.0 MHz to 2.5 MHz. However, the EMAC MDC can also support higher frequencies for PHY devices that support the frequencies.

Table 28-48: MDC Clock Frequency Selection

Selection	Programmed Frequency Range	Frequency of MDC	Min and Max MDC Freq (Per Specifications)
0000	60–100 MHz	CLKO7/42	MIN = 1.43 MHz and MAX = 2.39 MHz
0001	100–150 MHz	CLKO7/62	MIN = 1.61 MHz and MAX = 2.42 MHz
0010	20–35 MHz	CLKO7/16	MIN = 1.25 MHz and MAX = 2.19 MHz
0011	35–60 MHz	CLKO7/26	MIN = 1.35 MHz and MAX = 2.31 MHz

The ***MDIO Frame Parameters*** table provides MDIO data transfer parameters. The write and read sequences provided in the tables, ***MDIO Write Data Sequence*** and ***MDIO Read Data Sequence***, are based on these parameters.

Table 28-49: MDIO Frame Parameters

Parameter	Description
IDLE	The MDIO line is three-state (noted as Z in sequence); there is no clock on MDC.
PREAMBLE	32 continuous bits, each of value 1
START	Start of frame is 01
OPCODE	10 for read and 01 for write
PHY ADDR	5-bit address select for one of 32 PHYs (noted as AAAAA in sequence)
REG ADDR	Register address in the selected PHY (noted as RRRRR in sequence)

Table 28-49: MDIO Frame Parameters (Continued)

Parameter	Description
TA	Turnaround is Z0 for read and 10 for write (Z = high impedance)
DATA	Any 16-bit value. Driven by MAC or PHY based on direction (noted as DDD...DDD).

Table 28-50: MDIO Write Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	01	AAAAA	RRRRR	10	DDD... DDD	Z

Table 28-51: MDIO Read Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	10	AAAAA	RRRRR	Z0	DDD... DDD	Z

MDIO Frame Structure

Both Clause 22 and Clause 45 MDIO frame structures are supported by the EMAC. The `EMAC_MDIO_ADDR.C45E` bit can be programmed to enable Clause 22 or Clause 45 mode of operation. The *MDIO Clause 45 Frame Structure* table shows the Clause 45 frame format and the *MDIO Clause 22 Frame Structure* table shows the Clause 22 frame format. In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

Table 28-52: MDIO Clause 45 Frame Structure

Field	Description
IDLE	The MDIO line is in three-state; there is no clock on MDC
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2’b00
OPCODE	00/01/10/11
PHY ADDR	5-bit address select for one of 32 PHYs
DEV ADDR	5-bit address select for one of 32 devices
TA	Turnaround is Z0 for read and 10 for write (Z = high impedance)
DATA/ADDRESS	16-bit value. For an address cycle (OPCODE = 2'b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data that is written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. • In address and data write cycles, the EMAC drives the MDIO line during the transfer of these 16 bits.

Table 28-52: MDIO Clause 45 Frame Structure (Continued)

Field	Description
	• In read and post-read increment address cycles, the PHY drives the MDIO line during the transfer of these 16 bits.

Table 28-53: MDIO Clause 22 Frame Structure

Field	Description
IDLE	The MDIO line is in tri-state; there is no clock on MDC
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2'b01
OPCODE	10 for read and 01 for write
PHY ADDR	5-bit address select for one of 32 PHYs
DEV ADDR	5-bit address to select the register within each MMD
TA	Turnaround is Z0 for read and 10 for write (Z = high impedance)
DATA	Any 16-bit value. Driven by MAC or PHY based on direction (noted as DDD...DDD).

Table 28-54: MDIO Write Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

Table 28-55: MDIO Read Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	01	AAAAA	RRRRR	Z0	DDD...DDD	Z

SMI Write Operation

When programs set the EMAC_MDIO_ADDR.GOC_1/EMAC_MDIO_ADDR.GOC_0 bit field to 2'b01 and the EMAC_MDIO_ADDR.GB bit to 1, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (EMAC_MDIO_DATA register) to the SMI to initiate a write operation into the PHY registers. At this point, the SMI module starts a write operation on the GMII Management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002, Section 22.2.4.5).

When the SMA module starts a write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The EMAC_MDIO_ADDR.GB bit is set high until the write operation is complete. The CSR ignores the write operations performed to the EMAC_MDIO_ADDR register or the EMAC_MDIO_DATA register during this period (the EMAC_MDIO_ADDR.GB bit is high). When the write operation is complete, the SMI module indicates this to the CSR, and the CSR resets the EMAC_MDIO_ADDR.GB bit.

The *SMI Write Operation through MDIO/MDC Pins* figure shows this operation.

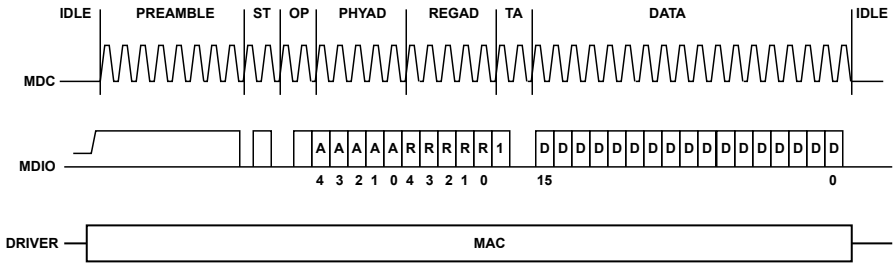


Figure 28-27: SMI Write Operation through MDIO/MDC Pins

SMI Read Operation

When programs set the EMAC_MDIO_ADDR.GOC_1/EMAC_MDIO_ADDR.GOC_0 bit field to 2'b11 and the EMAC_MDIO_ADDR.GB bit to 1, the MAC CSR module transfers the PHY address and the register address in PHY to the SMI to initiate a read operation in the PHY registers. The SMI starts a read operation on the GMII Management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002, Section 22.2.4.5).

When the SMI starts a read operation on the MDIO, the CSR ignores the write operations to the EMAC_MDIO_ADDR or EMAC_MDIO_DATA registers during this period (the EMAC_MDIO_ADDR.GB bit is high) and the transaction is completed without any error. When the read operation is complete, the SMI indicates this to the CSR. The CSR resets the GB bit and updates the EMAC_MDIO_DATA register with the data read from the PHY.

The *SMI Read Operation through MDIO/MDC Pins* figure shows this operation.

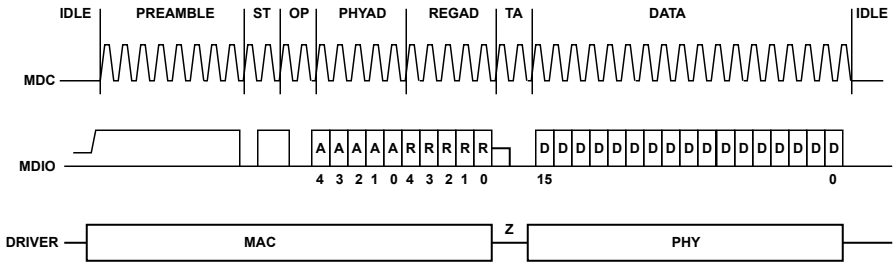


Figure 28-28: SMI Read Operation through MDIO/MDC Pins

Preamble Suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The EMAC provides controls to support preamble suppression. EMAC transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the EMAC_MDIO_ADDR.PSE bit.

Trailing Clocks and Back-to-Back Transactions

The EMAC drives MDC clock on the MDC port for duration of the MDIO frame. There is no clock driven during the idle period. The trailing clocks feature can be used if the PHY needs the MDC clock to be active for some cycles after the MDIO frame. The EMAC_MDIO_ADDR.NTC bit field allows programming of trailing clocks from 0 to 7.

The EMAC supports back to back transactions which allows start of next MDIO frame even before the trailing clocks are completed for previous MDIO frame. This feature can be enabled by setting the EMAC_MDIO_ADDR.BTB bit when trailing clocks feature is also enabled. When back to back transactions is enabled, the GMII busy is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by EMAC while trailing clocks are still on for the previous MDIO frame. When back to back transactions is not enabled, the GMII busy is cleared after the trailing clocks are completed for MDIO frame.

Interrupt for MDIO Transaction Completion

The EMAC can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy field of the EMAC_MDIO_ADDR register to know the completion of MDIO commands.

EMAC Management Counters (MMC)

The EMAC provides a comprehensive set of 32-bit MAC management counters. It uses these counters for gathering statistics on the received and transmitted frames. The MMC sub block also includes

- A control register (EMAC_MMC_CTRL) for managing the behavior of the counters
- Two 32-bit registers containing interrupts generated (EMAC_MMC_RXINT and EMAC_MMC_TXINT)
- Two 32-bit registers containing masks for the interrupt register (EMAC_MMC_RXIMSK and EMAC_MMC_TXIMSK)

The MMC receive counters are updated for frames passed by the address filtering sub block in the EMAC CORE. Statistics of frames dropped by the AFM module are not updated unless they are runt frames of less than 6 bytes. (Destination address bytes are not received fully.) The module is also capable of gathering statistics on encapsulated IPv4, IPv6, and TCP, UDP, or ICMP payloads in received Ethernet frames.

The MMC register naming conventions are as follows:

- Tx as a prefix or suffix indicates counters associated with transmission.
- Rx as a prefix or suffix indicates counters associated with reception.
- G as a suffix indicates registers that count good frames only.
- GB as a suffix indicates registers that count frames regardless of whether they are good or bad.

Transmitted frames are considered *good* when transmitted successfully. In other words, a transmitted frame is good if the frame transmission does not abort due to any of the following errors:

- Jabber timeout
- No carrier or loss of carrier
- Late collision
- Frame underflow

- Excessive deferral
- Excessive collision

Received frames are good when none of the following errors exists:

- CRC error
- Runt frame (shorter than 64 bytes)
- Alignment error
- Length error (non-type frames only)
- Out-of-range (non-type frames only, longer than maximum size)

The maximum frame size depends on the frame type, as follows:

- Untagged frame maximum size = 1518
- VLAN frame maximum size = 1522
- Jumbo frame maximum size = 9018
- Jumbo VLAN frame maximum size = 9022

The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Size of the Maximum Receive Packet* table.

Table 28-56: Size of the Maximum Receive Packet

JE	S2KP	GPSLCE	EDVLP	Untagged Frame maximum size in bytes	Single VLAN Frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

The EMAC_MMC_CTRL register also contains bits that control preset, freeze and roll-over of counters. The EMAC uses the EMAC_MMC_CTRL.RSTONRD bit to enable an auto-reset feature whenever the counters are read. The EMAC uses the EMAC_MMC_CTRL.CNTRST bit to reset all the counters.

The MMC can trigger an interrupt when the corresponding bits are enabled in the transmit, receive, and IPC mask registers, and when the counter reaches half or full. The status is also updated in the corresponding interrupt register.

MMC Receive Interrupt Register

The EMAC_MMC_RXINT register maintains the interrupts that are generated when the receive statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When EMAC_MMC_CTRL.CNTSTOPRO bit is set, then interrupts are set, but the counter remains at all ones. The EMAC_MMC_RXINT register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits 7:0) of the respective counter must be read to clear the interrupt bit.

MMC Transmit Interrupt Register

The EMAC_MMC_TXINT register maintains the interrupts generated when the transmit statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When EMAC_MMC_CTRL.CNTSTOPRO is set (= 1), then interrupts are set, but the counter remains at all ones. The EMAC_MMC_TXINT register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits 7:0) of the respective counter must be read to clear the interrupt bit.

MMC Receive Checksum Offload Interrupt Register

The EMAC_MMC_IPC_RXINT register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When the EMAC_MMC_CTL.NOROLL is set, then interrupts are set, but the counter remains at all ones. The EMAC_MMC_IPC_RXINT register is 32 bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The least-significant byte lane (bits 7:0) of the counter must be read to clear the interrupt bit.

EMAC Precision Time Protocol (PTP) Engine

The following sections describe the precision time protocol (PTP) engine.

IEEE1588 and the PTP Engine

The Ethernet MAC peripheral includes a PTP engine to assist applications requiring time synchronization. The PTP module is tightly integrated with the EMAC CORE to aid hardware time stamping defined in the IEEE1588 2002/2008 standards. Applications can use accurate hardware time stamps through TCP/IP stacks (if using network layer communication) to exchange time information across devices connected over network. Applications can also use accurate hardware time stamps through Ethernet device drivers (if using MAC layer communication) to exchange time information.

PTP Engine

For calculation of drift in time between two Ethernet devices, the device records its system time whenever a timing message is sent or received (IEEE 1588 protocol). Due to the indeterministic delay of a software system for a node, the software is unable to capture an accurate time when the message is sent or received. However, the hardware can monitor the signal on the communication media and get an accurate message of arrival and departure time.

The PTP (precision time protocol) module is closely integrated with the EMAC module. It provides hardware assistance to implement both the IEEE 1588–2002 and IEEE 1588–2008 standards on Ethernet (IEEE 802.3).

It takes one input clock signal as its PTP clock and maintains the timing information (called *system time*) at the nanosecond level.

The PTP module includes hardware for clock and system time adjustment. The pulse-per-second (PPS) signals physically represent the system time. PPS can be programmed to a fixed frequency or provide flexibility to the signal in terms of pulse width, interval, start, and stop time of the signals. The PTP module can be programmed to trigger an alarm interrupt when system time reaches specified time.

The PTP module can be programmed to detect different types of received frames, capture the system time, and time stamp those frames with the captured system time. Programs can configure any frame so that the PTP module capture the system time when it is transmitted. The PTP module can also capture the system time when an event is detected on the auxiliary snapshot trigger input pins (`EMAC_PTPAUX_MCG_IN[n]`).

IEEE 1588 Standard

Many systems require two independent devices to operate in a time synchronized fashion. If each system relied solely on its oscillator, differences between the characteristics and operating conditions of each oscillator would limit the ability of the clocks to operate synchronously. To serve applications requiring synchronized clocks, the system uses a periodic correction mechanism.

A simple way to synchronize multiple systems is to choose one system (with the best clock) as a requester. The system requester broadcasts the clock and timing information to other systems (completers); subsequently, the completers adjust their clocks and timing according to that of requester. However, this method has limitations. The requester cannot broadcast the time at infinitesimal intervals. Path delay (propagation delay) exists between a requester and a completer and the delay varies between each completer and requester.

IEEE 1588 is also known as precision time protocol or PTP. The standard specifies a protocol used to synchronize the time and clock of multiple devices, dispersed but interconnected by any communication, (for example, Ethernet IEEE 802.3). According to the protocol, timing messages are exchanged between two devices. Then, one of the devices calculates its drift from other device and corrects its system time. (Both devices must have the same representation of their system time). The protocol resolves path delay between devices. It also helps synchronize the clocks of multiple devices and all of the limitations mentioned are resolved.

IEEE 1588 was published in 2002 where four types of timing messages were defined: Sync, Follow_Up, Delay_Req, and Delay_Resp. Here the protocol synchronizes two or more devices where one is a requester and others are completers. The requester device sends Sync, Follow_Up, and Delay_Resp messages to the completer device in the system. The target sends the Delay_Req messages to the requester device. A following section provides more information on IEEE 1588–2002.

In 2008, a newer version of IEEE 1588 was introduced which provides further mechanisms to measure the peer-to-peer delay. Three more timing messages (PdelayReq, PdelayResp, and PdelayRespFollowup) were added to implement peer-to-peer synchronization. The following section provides more information on IEEE 1588–2008.

IEEE 1588–2002

The IEEE 1588–2002 standard defines the precision time protocol (PTP). The protocol allows precise synchronization of clocks in measurement and control systems that use network communication, local computing, and distributed objects. The protocol applies to systems that communicate by local area networks that support multicast

messaging, including (but not limited to) Ethernet. This protocol also allows heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range using minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into requester and completer nodes for distributing the timing or clock information. The ***IEEE 1588–2002 PTP Process*** figure shows the PTP process used for synchronizing a target node to a controller node by exchanging PTP messages.

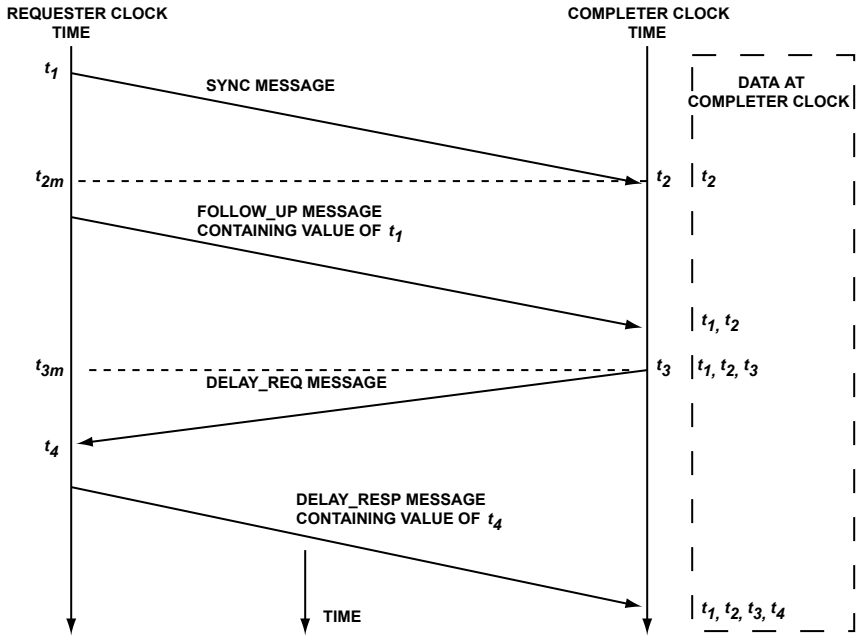


Figure 28-29: IEEE 1588–2002 PTP Process

As shown in the figure, the PTP uses the following process:

1. The requester broadcasts the PTP sync messages to all its nodes. The sync message contains the reference time information of the requester. The time at which this message leaves the system of the requester is t_1. The requester must capture this time for Ethernet ports, at RMII.
2. The completer receives the Sync message and also captures the exact time, t_2, using its timing reference.
3. The requester sends a Follow_up message to the completer, which contains t1 information for later use.
4. The completer sends a Delay_Req message to the requester, noting the exact time, t_3, at which this frame leaves the RMII.
5. The requester receives the message, capturing the exact time, t_4, at which it enters its system.
6. The requester sends the t_4 information to the completer in the Delay_Resp message.
7. The completer uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference to the timing reference of the requester.

Most of the PTP implementation occurs in the software above the UDP layer. However, the hardware support must capture the exact time when specific PTP packets enter or leave the Ethernet port at the RMII. Hardware must capture this timing information and return it to the software for the proper implementation of PTP with high accuracy.

IEEE 1588–2008 Advanced Time Stamps

In addition to the basic time stamp features mentioned in IEEE 1588–2002 time stamps, the EMAC supports the following advanced time stamp features defined in the IEEE 1588–2008 standard.

- Support for the IEEE 1588–2008 (Version 2) time stamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to select the node to be a requester or completer.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to run nanoseconds time in digital or binary format.

Peer-to-Peer (P2P) PTP Message Support

The IEEE 1588–2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. Refer to the ***Propagation Delay Calculation between Nodes Supporting P2P Path Correction*** figure. The figure shows the method to calculate the propagation delay between nodes supporting peer-to-peer path correction.

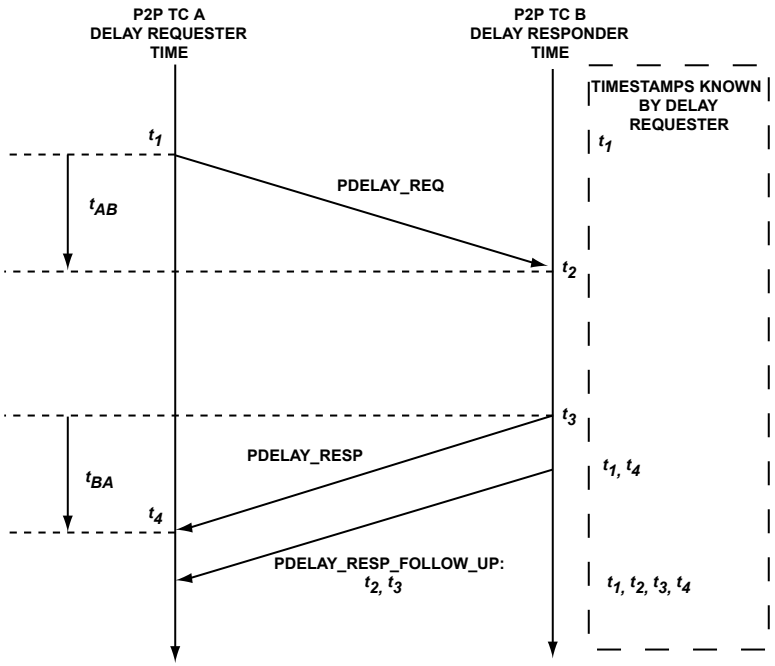


Figure 28-30: Propagation Delay Calculation between Nodes Supporting P2P Path Correction

As shown in the figure, the propagation delay is calculated in the following way:

1. Port 1 issues a Pdelay_Req message and generates a time stamp, t1, for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a time stamp, t2, for this message.
3. Port 2 returns a Pdelay_Resp message and generates a time stamp, t3, for this message. To minimize errors due to frequency offset between the two ports, port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The port 2 returns any one of the following:
   - The difference between the time stamps t2 and t3 in the Pdelay_Resp message.
   - The difference between the time stamps t2 and t3 in the Pdelay_Resp_Follow_Up message.
   - The time stamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.
4. Port 1 generates a time stamp, t4, on receiving the Pdelay_Resp message.

Port 1 uses all four time stamps to compute the mean link delay.

Block Diagram

The ***PTP Block Diagram*** figure shows the functional block diagram of PTP module.

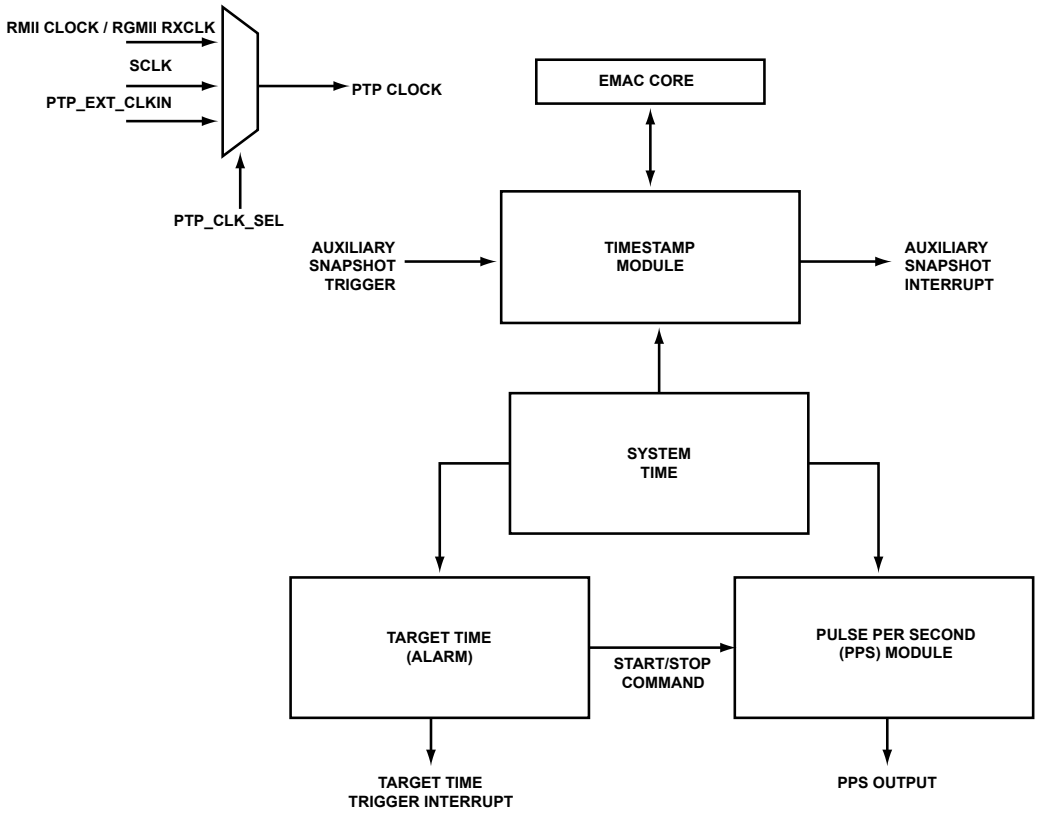


Figure 28-31: PTP Block Diagram

A system time module is present which keeps the time of PTP module. It consists of hardware which can be programmed for time initialization, time correction, and clock correction.

The time stamp module captures the time (provided by the system time module) at various conditions. For example, when the EMAC sends or receives a frame or during the rising edge of the auxiliary snapshot trigger EMAC_PTPAUX_MCG_IN[n] pins. When system time is captured after detection of a frame, the time stamp module automatically includes the time information in the frame descriptor. Time stamping on the detection of a frame can be programmed on a per frame basis.

The PTP clock drives the PTP module. This clock can be selected from three different clock sources.

The Pulse per Second (PPS) module generates a pulse or train of pulse on the PPS output pins, (EMAC_PTPPPS[n]). It is the physical representation of system time. PPS can be fixed (where only frequency varies) or flexible (where width, interval, start time, and stop time can be programmed).

The target time module acts as an alarm for the PTP module. Whenever system time reaches a value equal to programmed target time, the target time trigger interrupt is generated. By appropriate programming, The target time trigger can also start or stop flexible PPS output at specific time.

PTP Module Clock

The PTP module clock features include Clock Source Selection and Clock Frequency Range .

Clock Source Selection

The PTP module can take one of these clock sources as its input clock — CLKO7, RMII reference RGMII Rx clock, or PTP external clock.

As shown in the ***PTP Clock Source Selection*** table, the PADS_PCFG0 register [bits 1:0] selects the PTP clock source.

Table 28-57: PTP Clock Source Selection

PADS_PCFG0 [1:0]	PTP Clock Source	Clock Description
00 – RMII or MII	ETH0_RXCLK_REFCLK/ ETH1_REFCLK	MII/RMII reference clock
00 – RGMII	CLKO7	Processor Ethernet clock
10	PTP external clock	Clock available on EMAC_PTPCLKIN[n] pin
01 or 11	SCLK0	Processor system clock

Clock Frequency Range

The resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance. The timing constraints achievable for logic operating on the selected PTP clock source limit the maximum PTP clock frequency.

The minimum PTP clock frequency depends on the time required between two consecutive frames. The IEEE specification determines the RMII clock frequency. The minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC. See the ***Minimum PTP Clock Frequency*** table.

A minimum delay required between two consecutive time stamp captures is 8 clock cycles of RMII and 3 clock cycles of PTP clocks. If the delay between two time stamp captures is less than this delay, the EMAC does not take a time stamp snapshot for the second frame.

The ***Minimum PTP Clock Frequency*** table assumes:

- Minimum Ethernet packet size = 64 bytes
- Mininum IFG = 96 bit times or 12 bytes
- Preamble = 8 bytes
- 3 PTP clock + 4 GMII/MII clock minimum gap between two SFDs (start of frame delimiters)
- 3 PTP clock + 8 RMII clock minimum gap between two SFDs

Table 28-58: Minimum PTP Clock Frequency

Mode (Full Duplex in all cases)	Minimum Gap between two SFDs	PTP clock	Comments
10 Mbps RMII	64 bytes of data + 8 bytes of preamble + min IFG	3 PTP clock cycle + 8RMII clock cycle 3360 RMII clock cycles	In RMII @ 10 Mbps, 1 byte transmitted in 40 clock cycles
	(2560 + 320 + 480) RMII clock cycles	3 PTP clock cycle 3352 RMII clock cycles	RMII clock = 50 MHz
	3360 RMII clocks	PTP clock cycle 1117.33 20 ns = 22346 ns	1 RMII clock cycle = (1/50 MHz) = 20 ns
		PTP frequency min = 0.045 MHz	
100 Mbps RMII	64 bytes of data + 8 bytes of preamble + min IFG	3 PTP clock cycle + 8RMII clock cycle 336 RMII clock cycles	In RMII, 1 byte transmitted in 4 clock cycles
	(256 + 32 + 48) RMII clock cycles	3 PTP clock cycle 328 RMII clock cycles	RMII clock = 50 MHz
	336 RMII clocks	PTP clock cycle 109.33 20 ns = 2186 ns	1 RMII clock cycle = (1/50 MHz) = 20 ns
		PTP frequency min = 0.45 MHz	

The minimum PTP clock frequency also depends on the maximum value of the `EMAC_TM_SUBSEC` register. Even at the highest `EMAC_TM_SUBSEC.SSINC` value, the `EMAC_TM_SEC` register value can be incremented every second. Since the `EMAC_TM_SUBSEC.SSINC` is an 8-bit field, the minimum PTP clock frequency allowed is approximately 4 MHz.

Time Stamp Module

The time stamp module captures time in seconds and nanoseconds maintained as system time. The time stamp module also captures time when specific events occur. Events include detection of a frame transmitted or received

over the EMAC and a rising edge on the EMAC_PTPAUX_MCG_IN[n] pins. The time stamp module does not need to time stamp all of the transmitted or received frames over the EMAC. The PTP module can be programmed to detect specific kinds of frames for time stamping.

Frame Detection and Time Stamping

The PTP module automatically monitors all received and transmitted IEEE 1588 event messages on the Ethernet. If the module detects an event message, it takes a snapshot of the system time. The PTP module stores the value to the 64-bit fields in transmit or receive descriptor.

The time stamping occurs at the EMAC RMII/RGMII interface when the module sees the start of frame of an event message packet.

Transmit Path Time Stamping

The MAC captures a time stamp when the Start Packet Delimiter (SFD) of a packet is sent on the MII/RMII/RGMII interface. The packets for which the time stamps must be captured, can be controlled on per-packet basis. Each transmit packet can be marked to indicate whether a time stamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The program needs to specify the packets to capture time stamps.

The program can specify the packets by using the control bits in the transmit descriptor. The MAC returns the time stamp to the software inside the corresponding transmit descriptor, connecting the time stamp automatically to the specific PTP packet. The 64-bit time stamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the time stamp.

Receive Path Time Stamping

The MAC can be programmed to capture the time stamp of all packets received on the GMII or MII interface or to process packets to identify the valid PTP messages. Use the following options of the EMAC_TM_CTL register to control the snapshot of the time to be sent to the application:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 time stamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable time stamp snapshot for the received packet for IPv4 or IPv6
- Enable time stamp snapshot only for event messages (Sync, Delay_Req, Pdelay_Req, or Pdelay_Resp)
- Enable the node to be a main or target and select the snapshot type

This feature controls the type of messages for which snapshots are taken.

Refer to the ***PTP Frame Type Selections*** table.

Table 28-59: PTP Frame Type Selections

TSENALL (bit 8)	SNAPTYPSEL (bits [17:16])	TSMSTRENA (bit 15)	TSEVNTENA (bit 14)	Frames
1	X	X	X	All
0	00	X	0	Sync, Follow_Up, Delay_Req, Delay_Resp
0	00	0	1	Sync
0	00	1	1	Delay_Req
0	01	X	0	Sync, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_ Follow_Up
0	01	0	1	Sync, Pdelay_Req, Pdelay_Resp
0	01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
0	10	X	X	Sync, Delay_Req
0	11	X	X	Pdelay_Req, Pdelay_Resp

Time Stamp Correction

According to the IEEE 1588 specification, a time stamp must be captured when the PTP message time stamp point (leading edge of the first bit of the octet immediately following the SFD octet) crosses the boundary between the node and the network.

Because the MAC takes the time stamp at an internal point far from the actual boundary of the node and network, this captured time stamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (RGMII Tx, Rx clock). The clock can be different at the capture point when compared with the PTP clock that is used to generate the time. The resultant Clock Domain Crossing (CDC) circuits add error, depending on the clock period of the GMII and PTP clocks.

Ingress Correction

In the receive side, the time stamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet's SFD bit is received at the port's boundary. Therefore, the captured time stamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the `EMAC_TM_INGCORR_NSEC` and `EMAC_TM_INGCORR_SNSEC` registers.

The correction value consists of the following components:

1. External latency in the PHY layer between boundary point and the input of the core. If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has a register indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from its data sheet or timing characteristics.

2. Internal latency from the input of the core to the internal capture point. The internal ingress latency can be read from the EMAC_TM_ING_LTNCY register. This read-only register gives the latency in scaled nanoseconds format, as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface (RGMII, RMII, so on) and the operating speed. Therefore, the software must read this register after any speed change in the MAC to determine the current internal latency.
3. CDC Synchronization. The CDC synchronization error is approximately equal to two times the clock period of the PTP clock.

Software should add the values determined from these components and write them into the EMAC_TM_INGCORR_NSEC.TSIC and EMAC_TM_INGCORR_SNSEC.TSICSNS bit fields.

NOTE: The value written to the register must be negative (two's complement), because it must be subtracted from the captured time stamp. The MAC receiver adds the value in this register to the captured time stamp and then gives the resultant value as the time stamp of the received packet.

When the EMAC_TM_CTL.TSCTRLSSR bit is set, the nanoseconds field of the captured time stamp is in decimal format with a granularity of 1 ns. So, bit 31 of EMAC_TM_INGCORR_NSEC.TSIC must be set (= 1) (for negative value). Bits[30:0] must be written with "109 - total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is -5 ns, the value is 0xBB9A_C9FB.

When EMAC_TM_CTL.TSCTRLSSR bit is reset, the nanoseconds field of the captured time stamp is in binary format with a granularity of ~0.466 ns. Therefore, bits [30:0] must be written with "231 - total ingress_correction_value" represented in binary with bit[31] = 1.

Egress Correction

In the transmit side, the time stamp captured at the internal snapshot point is earlier (advanced in time) when compared with the time at which that packet's SFD bit is output at the port's boundary. Therefore, the captured time stamp must be compensated by the egress latency and the errors in CDC sampling. Software determines/calculates this correction value and writes it into the EMAC_TM_EGCORR_NSEC and EMAC_TM_EGCORR_SNSEC registers.

The correction value consists of the following components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network. If the PHY is compliant to the IEEE 802.3 Clause 45 MMD registers, it has a register indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Otherwise, (if the PHY does not support these registers), the egress latency must be determined from its data sheet or timing characteristics.
2. Internal latency from the internal capture point and the output of the core. This internal egress latency can be read from the EMAC_TM_EG_LTNCY register. This read-only register gives the latency in scaled nanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface (RGMII, RMII, so on) and the operating speed. Therefore, the software must read this register after any speed change in the MAC to determine the current internal latency.

3. CDC synchronization error. The CDC synchronization error value depends on the one-step time stamping mode. When one-step time stamping is enabled, the value = (1 x period of PTP clock + 4 x period of Tx clock).

The ***PHY Interface Latency*** table lists the egress and ingress latency for various PHY interfaces.

Table 28-60: PHY Interface Latency

PHY Interfaces		Egress Latency	Ingress Latency
RGMII	1.0 Gbps	12	12
RGMII	100 Mbps	40	40
RGMII	10 Mbps	400	400
RMII	100 Mbps	40	40
RMII	10 Mbps	400	400

PTP Processing and Control

When the EMAC module receives a frame, frame detection and time stamping are based on some of the PTP fields in the frame. The ***PTP Message Format (IEEE 1588–2008)*** table shows the common message header for the PTP messages. This format is derived from the IEEE standard 1588–2008. When the EMAC module sends a PTP frame, the frame follows this format.

When a frame is received, the PTP module compares these fields with standard values and finds out the type of PTP frame and other information (for example, PTP version, IP version, and others). The module then updates the related fields in RDES4. When a frame is transmitted, programs must ensure that all the fields are appropriate. The PTP module on the other end of a communication must correctly detect and decode the frame.

Table 28-61: PTP Message Format (IEEE 1588–2008)

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30

Table 28-61: PTP Message Format (IEEE 1588–2008) (Continued)

Bits	Octets	Offset
controlField (used in version 1. For version 2, messageType field is used for detecting different message types.)	1	32
logMessageInterva	1	33

There are some fields in the Ethernet payload that can be used to detect the PTP packet type and control the snapshot taken. These fields are different for the following PTP frames:

- PTP Frames Over IPv4
- PTP Frames Over IPv6
- PTP Frames Over Ethernet

For these PTP frames, EMAC does not consider the PTP version 1 messages as valid when the frame consists of peer delay multicast address as destination address (DA).

PTP Frame Over IPv4

The ***IPv4-UDP PTP Frame Fields Required for Control and Status*** table provides information about the fields that are matched to control snapshot for the PTP packets. The packets are sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. The positions are based on IEEE 1588–2008 standards and the message format. The format is defined in the ***PTP Message Format (IEEE 1588–2008)*** table in the PTP Processing and Control section.

Table 28-62: IPv4-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x0800	IPv4 datagram
IP Version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 Version 1)	30, 31, 32, 33	0xE0,0x00, 0x01,0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 Version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-Primary multicast address: 224.0.1.129 PTP-Peer delay multicast address: 224.0.0.107

Table 28-62: IPv4-UDP PTP Frame Fields Required for Control and Status (Continued)

Field Matched	Octet Position	Matched Value	Description
UDP Destination Port	36, 37	0x013F 0x0140	0x013F - PTP Event Messages. These are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only). 0x0140 - PTP general messages
PTP Control Field (IEEE version 1)	74	0x00/0x01/0x02/ 0x03/0x04	0x00 - SYNC 0x01 - Delay_Req 0x02 - Follow_Up 0x03 - Delay_Resp 0x04 - Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD	0x0 - SYNC 0x1 - Delay_Req 0x2 - Pdelay_Req 0x3 - Pdelay_Resp 0x8 - Follow_Up 0x9 - Delay_Resp 0xA - Pdelay_Resp_Follow_Up 0xB - Announce 0xC - Signaling 0xD - Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1 - Supports PTP version 1 0x2 - Supports PTP version 2

PTP Frame Over IPv6

The ***IPv6-UDP PTP Frame Fields Required for Control And Status*** table provides information about the fields that are matched to control the snapshots for the PTP packets. The packets are sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. The positions are based on IEEE 1588–2008 standards and the message format defined in PTP Message Format (IEEE 1588–2008).

Table 28-63: IPv6-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x86DD	IP datagram
IP Version	14 (bits [7:4])	0x06	IP version is IPv6
Layer 4 Protocol	20 (IPv6 extension header not defined for PTP packets)	0x11	UDP

Table 28-63: IPv6-UDP PTP Frame Fields Required for Control and Status (Continued)

Field Matched	Octet Position	Matched Value	Description
PTP Multicast Address	38–53	FF0x:0:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:0:6B (Hex)	PTP - primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP - Peer delay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57 (IPv6 extension header not defined for PTP packets)	0x013F, 0x0140	0x013F - PTP event messages 0x0140 - PTP general messages
PTP Control Field (IEEE 1588 version 1)	93 (IPv6 extension header not defined for PTP packets)	0x00/0x01/0x02/ 0x03/0x04	0x00 - SYNC 0x01 - Delay_Req 0x02 - Follow_Up 0x03 - Delay_Resp 0x04 - Management (version1)
PTP Message Type Field (IEEE version 2)	74 (nibble) (IPv6 extension header not defined for PTP packets)	0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD	0x0 - SYNC 0x1 - Delay_Req 0x2 - Pdelay_Req 0x3 - Pdelay_Resp 0x8 - Follow_Up 0x9 - Delay_Resp 0xA - Pdelay_Resp_Follow_Up 0xB - Announce 0xC - Signaling 0xD - Management
PTP Version	75 (nibble)	0x1 or 0x2	0x1 - Supports PTP version 1 0x2 - Supports PTP version 2

PTP Frame Over Ethernet

Refer to the ***Ethernet PTP Frame Fields Required for Control and Status*** table. The table provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. The positions are based on IEEE 1588–2008 standards and the message format defined in the table.

Table 28-64: Ethernet PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched value	Description
MAC Destination Multicast Address (The address match of destination address (DA) programmed in MAC address 0 is used when the EMAC_TM_CTL.TSENMACADDR bit is set)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses: 01-1B-19-00-00-00 01-80-C2-00-00-0E
MAC Frame Type	12, 13	0x88F7	PTP Ethernet frame
PTP control field (IEEE Version 1)	45	0x00/0x01/0x02/ 0x03/0x04	0x00 - SYNC 0x01 - Delay_Req 0x02 - Follow_Up 0x03 - Delay_Resp 0x04 - Management
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD	0x0 - SYNC 0x1 - Delay_Req 0x2 - Pdelay_Req 0x3 - Pdelay_Resp 0x8 - Follow_Up 0x9 - Delay_Resp 0xA - Pdelay_Resp_Follow_Up 0xB - Announce 0xC - Signaling 0xD - Management
PTP Version	15(nibble)	0x1 or 0x2	0x1 - Supports PTP version 1 0x2 - Supports PTP version 2

Auxiliary Time Stamp Snapshot

The auxiliary snapshot feature stores snapshots of the system time whenever a rising edge is detected on the EMAC_PTPAUX_MCG_IN[n] pins.

The PTP stores 64 bits of the captured time stamp in a 16-deep FIFO. Only 64 bits of the time stamp are stored in the FIFO. The upper 16 bits of seconds can be read from the EMAC_TM_HISEC register when it is present. When a snapshot is stored, the PTP indicates this event to the EMAC with the auxiliary snapshot interrupt. The EMAC_TM_STAT.AUXTSTRIG bit is set (=1). The value of the snapshot is read through the EMAC_TM_AUXTS_SEC and EMAC_TM_AUXTS_NSEC registers. If the FIFO becomes full and an external trigger to take the snapshot is asserted, then the snapshot trigger missed (EMAC_TM_STAT.ATSSTM) bit is set (=1). The latest snapshot is not written to the FIFO when it is full.

When a host reads the 64-bit time stamp from the FIFO through the EMAC_TM_AUXTS_SEC and EMAC_TM_AUXTS_NSEC registers, the space becomes available to store the next snapshot.

NOTE: A space in the FIFO is created whenever the EMAC_TM_AUXTS_SEC register is read. Therefore, read the EMAC_TM_AUXTS_NSEC register before reading the EMAC_TM_AUXTS_SEC register.

The program can clear the FIFO by setting the EMAC_TM_AUX_CTL.ATSFC bit. When multiple snapshots are present in the FIFO, the EMAC_TM_STAT.ATSNS bits indicate the count.

NOTE: The minimum gap between two events on the EMAC_PTPAUX_MCG_IN[n] pin must be 4 cycles of PTP_CLK + 3 cycles of CLKO7). Otherwise, the logic misses the rising edge of the trigger.

System Time

To get a snapshot of the time, the EMAC requires a reference time in 64-bit format as defined in the IEEE 1588 specification. The PTP module maintains 80-bit time, known as system time. The PTP clock updates system time.

The 80-bit timing reference is split into the following registers:

- EMAC_TM_NSEC–32-bit nanoseconds register which provides time in nanoseconds
- EMAC_TM_SEC–32-bit seconds register which provides time in seconds
- EMAC_TM_HISEC–16-bit high seconds register which provides time beyond the seconds register. The IEEE 1588 standard does not include this register. Its use is application specific.

The 64-bit system time (seconds and nanoseconds) is the source for taking time stamps for Ethernet frames being transmitted or received at the RMII.

Since the PTP clock frequency does not correspond to a 1 ns period, the EMAC_TM_NSEC register is incremented with a value equal to the PTP clock period for every PTP clock cycle. The function uses the EMAC_TM_SUBSEC register. The EMAC_TM_NSEC value is incremented with the value programmed in EMAC_TM_SUBSEC register every PTP clock cycle.

Whenever the EMAC_TM_SEC register overflows from 0xFFFFFFFF to 0x00000000, the seconds overflow interrupt is triggered. The EMAC uses the EMAC_TM_STAT.TSSOVF bit to indicate the event. After a seconds overflow, the EMAC_TM_HISEC register increments by one.

The system time module supports the following types of rollover modes for the EMAC_TM_NSEC register.

- Digital rollover mode. The maximum value in the nanoseconds field is 0x3B9AC9FF, that is, 10^9 nanoseconds. After it reaches this value, the EMAC_TM_SEC register increments and the EMAC_TM_NSEC register restarts counting from zero. Accuracy in digital rollover mode it is 1 ns per bit.
- Binary rollover mode. The nanoseconds field rolls over and increments the seconds field after the value reaches 0x7FFFFFFF. Accuracy in binary rollover mode is ~0.466 ns per bit.

System Time Adjustment

The following sections describe the process for system time adjustment.

System Time Initialization

System time can be initialized with 64-bit time when the PTP module is enabled. The initial value is written to the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT system time update registers. The system time counter is written with the value in the registers when EMAC_TM_CTL.TSINIT =1.

Coarse Correction Method

If the complete system time has an offset based on the system time of the requester, the coarse correction method can correct it. The time offset value is written to the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers. The offset value is then added to or subtracted from the system time when the EMAC_TM_CTL.TSUPDT bit is set (=1). Use the EMAC_TM_NSECUPDT.ADDSUB bit to choose addition or subtraction. System time correction occurs in one clock cycle using the coarse correction method.

NOTE: During subtraction, the EMAC_TM_SECUPDT register value must be less than the value of the EMAC_TM_SEC register. Check the value prior to subtracting using coarse correction.

Fine Correction Method

If a target PTP clock frequency has a drift based on the requester PTP clock (as defined in IEEE 1588), it can be corrected using the fine correction method. Using this method, system time is corrected over a period (unlike coarse correction where it happens in one clock cycle). This correction helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP sync message intervals.

Using this method, an accumulator sums the contents of the EMAC_TM_ADDEND register. The ***System Time Update, Fine Correction Method*** figure shows the method. The arithmetic-carry that the accumulator generates acts as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency divider.

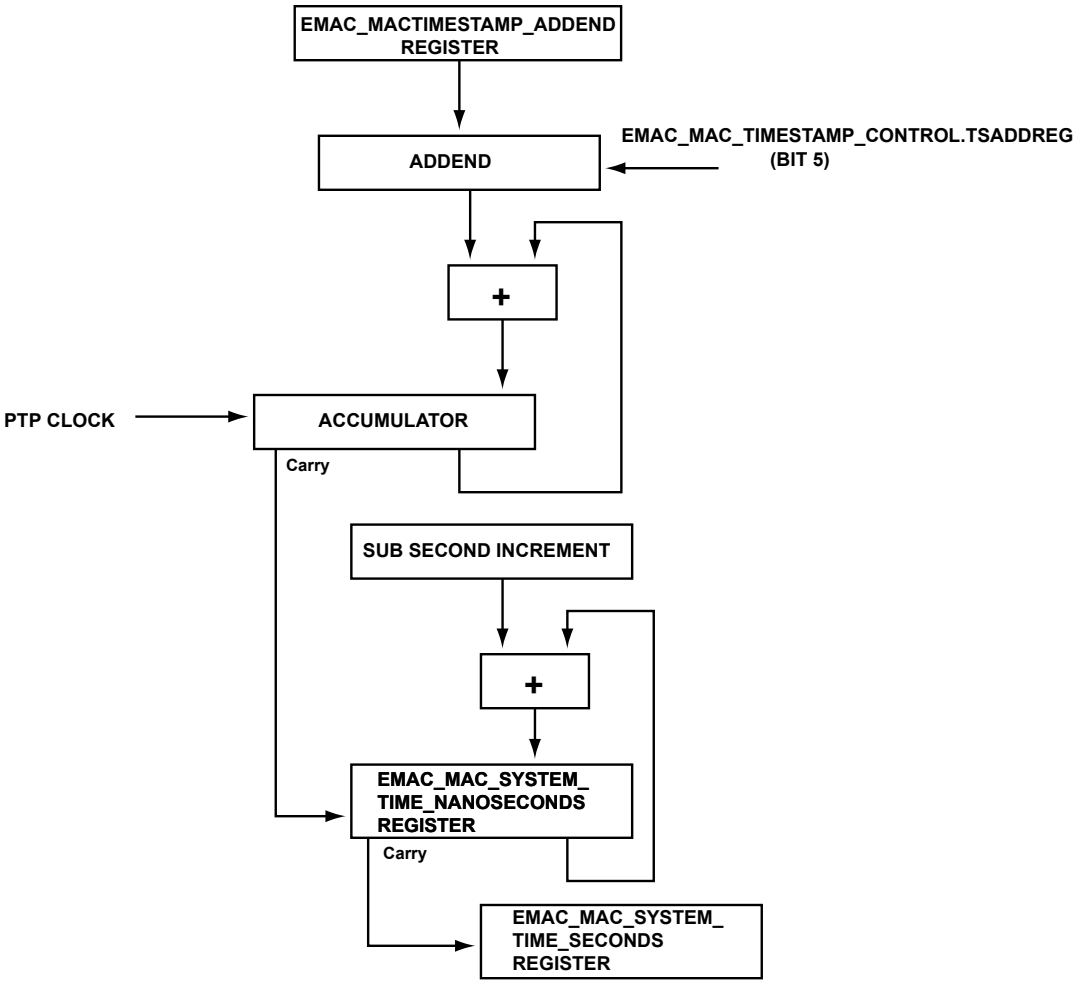


Figure 28-32: System Time Update, Fine Correction Method

Calculating Addend Value

The example in this section uses the fine correction process for system time.

In this example, the requester clock runs at 50 MHz and the completer clock has drifted to 66 MHz. The goal is to adjust the completer system time to 50 MHz, so that the completer PTP module synchronizes with the requester. Using the figure in Fine Correction Method, the nanoseconds increment signal runs at 50 MHz. The nanoseconds increment is the carry from accumulator register. The addend value increments the carry from accumulator at the rate of the completer clock (66 MHz).

The accumulator overflows and generates a carry every N addend value, so N × Addend = 2^{32}.

The accumulator increments at 66 MHz. This addition brings the carry to 50 MHz N = 66/50 = 1.32.

Hence, the addend = 2^{32}/1.32 = 0xC1F07C1F.

Therefore, if the addend is programmed with 0xC1F07C1F, the completer system time runs at 50 MHz, which synchronizes with the requester.

In the Fine Correction Method figure, the sub second increment is the value programmed in the EMAC_TM_SUBSEC register. It increments the EMAC_TM_NSEC register according to the frequency of the nanoseconds increment signal.

In the example, the sub second increment is 20 (for digital rollover) or 43 (for binary rollover). This addition increments the EMAC_TM_NSEC register by 20 ns (1/50 MHz).

The software must calculate the drift in frequency and update the EMAC_TM_ADDEND register accordingly.

NOTE: The PTP reference clock is the clock at which the system time is updated. When the EMAC_TM_CTL.TSCFUPDT bit is cleared (= 0), this clock equals the PTP clock. Using fine correction, the PTP reference clock is generated on the nanoseconds increment signal at which the system time updates.

Target Time Trigger (Alarm)

The PTP module provides an alarm function by triggering an alarm at a preset time. It sets the EMAC_TM_STAT.TSTARGT0 bit when the system time matches the value of the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers. This trigger can generate an interrupt and command the flexible PPS module to start or stop PPS output, depending on value programmed in the EMAC_PPS_CTL.TRGTMODSEL0 field.

The trigger is enabled by programming the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_SEC registers. Once an alarm has occurred, if the PTP needs another alarm, the software must:

- Clear the status bit,
- Reprogram the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers to a future value, and
- If the time programmed in the target time registers has elapsed, indicate a target time programming error by setting the EMAC_TM_STAT.TSTRGTERR0 bit.

The alarm time is represented in absolute units, not relative units. For example, if the software must generate an alarm after 10 seconds, it must read the current system time value. Then, the software must add the number corresponding to 10 seconds, and write the result back to the target time registers.

Pulse-Per-Second (PPS)

Pulse-per-second (PPS) is a physical representation of system time. It consists of a single pulse or train of pulses. The PTP uses PPS for extra synchronization or to monitor the synchronization performance between clocks. With proper configuration, the PTP module can generate PPS signals that are output on the EMAC_PTPPPS[n] pins. The PTP supports two kinds of PPS output: fixed and flexible.

Fixed Pulse-Per-Second Output

The EMAC module supports a fixed pulse-per-second (PPS) output that indicates 1 second intervals (default). Change the frequency of the PPS output by configuring the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits. The default value for these bits is 0000, which configures a 1 Hz signal with a pulse width equal to the period of the PTP clock.

The ***PPS Output Frequencies*** table shows various PPS output frequencies.

Table 28-65: PPS Output Frequencies

PPSCTL Bit Setting	Binary Rollover	Digital Rollover
0001	2 Hz	1 Hz
0010	4 Hz	2 Hz
0011	8 Hz	4 Hz
...	...	...
1111	32.768 kHz	16.384 kHz

In binary rollover mode, the PPS output (ETH)_PTPPS0) has a duty cycle of 50% with these frequencies.

In digital rollover mode, the PPS output frequency is an average number. The actual clock is a different frequency that is synchronized every second. PPS output pulses have different periods and duty cycles. This behavior is because of the non-linear toggling of the bits in digital rollover mode. For example:

- When `EMAC_PPS_CTL.PPSCTRL_PPSCMD` = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.
- When `EMAC_PPS_CTL.PPSCTRL_PPSCMD` = 0010, the PPS (2 Hz) is a sequence of:
  - One clock of 50 % duty cycle and 537 ms period
  - Second clock of 463 ms period (268 ms low and 195 ms high)
- When `EMAC_PPS_CTL.PPSCTRL_PPSCMD` = 0011, the PPS (4 Hz) is a sequence of:
  - Three clocks of 50 % duty cycle and 268 ms period
  - Fourth clock of 195 ms period (134 ms low and 61 ms high)

Flexible Pulse-Per-Second Output

The EMAC module also provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the PPS output. Enable this feature, called flexible PPS, by setting the `EMAC_PPS_CTL.PPSEN0` bit.

The flexible PPS output options are:

- Program the start point of the single pulse and start and stop points of the pulse train in terms of system time. The target time registers program the start and stop time.
- Program the stop time in advance. Programs can configure the stop time before the actual start time has elapsed.
- Program the width, between the rising edge and corresponding falling edge of the PPS signal output, in terms of number of units of subsecond increment. This value is configured in the `EMAC_TM_SUBSEC` register.

- Program the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment. This value is configured in the EMAC_TM_SUBSEC register.
- Cancel the programmed PPS start or stop request.
- Indicate an error if the start or stop time programmed has already elapsed.

PPS Start or Stop Time

Start time can initially be programmed in the target time registers. If necessary, the start or stop time can be programmed again, but, only after the earlier programmed value is synchronized to the PTP clock domain. The EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY bit indicates the status of synchronization. Programs can configure the start or stop time in advance, even before the earlier stop or start time has elapsed.

Program the start or stop time with advanced system time to ensure proper PPS signal output. If the application programs a start or stop time that has already elapsed, the EMAC module sets the EMAC_TM_STAT.TSTRGTERR0 bit indicating the error. If enabled, the EMAC module also sets the target time trigger (alarm) interrupt event. The application can cancel the start or stop request only when the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS Width and Interval

The PPS width and interval are programmed in terms of granularity of system time, that is, the number of the units of subsecond increment value. For example, with the PTP reference clock of 50 MHz, a PPS pulse width of 40 ns, and an interval of 100 ns, program the width and interval to 2 and 5, respectively.

Use a faster PTP reference clock to achieve smaller granularity. Before commanding to trigger a pulse or pulse train on the PPS output, programs must configure or update the interval and width of the PPS signal output.

PPS Command

When the PPS module has a flexible PPS output configuration, the PTP can use the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits to command the PPS module to use any of the flexible PPS features. Programming these bits with a non-zero value instructs the PPS module to initiate an event. Once the command transfers or synchronizes to the PTP clock domain, these bits clear automatically. Software must ensure that these bits are programmed only when they are all-zero.

The ***Flexible PPS Output Commands*** table describes the different commands and their functions.

Table 28-66: Flexible PPS Output Commands

PPSCTL (Bits 3–0)	Command	Description
0000	No Command	
0001	Start Single Pulse	Generates a single pulse rising at the start point defined in the target time registers and of the duration defined in the EMAC_PPS[n]_WID register.
0010	Start Pulse Train	Generates a train of pulses rising at the start time configured in the target time registers and of the duration configured in the EMAC_PPS[n]_WID register. The train of pulses repeats at the interval configured in the EMAC_PPS[n]_INTVL register. By

Table 28-66: Flexible PPS Output Commands (Continued)

PPSCTL (Bits 3–0)	Command	Description
		default, the PPS pulse train is runs freely unless shalted by a Stop Pulse Train at Time command or a Stop Pulse Train Immediately command.
0011	Cancel Start	Cancels the Start Single Pulse and Start Pulse Train commands when the system time has not crossed the programmed start time.
0100	Stop Pulse Train at Time	Stops the train of pulses initiated by the command for start pulse train after the time programmed in the target time registers elapses.
0101	Stop Pulse Train Immediately	Immediately stops the train of pulses initiated by the command for start pulse train.
0110	Cancel Stop Pulse Train	Cancels the stop pulse train at time command when the programmed stop time has not elapsed. The PPS pulse train switches to free-running on the successful execution of this command.
0111-1111	Reserved	

PTP Interrupts

Set the EMAC_MAC_IEN.TSIE bit to enable interrupts from the PTP module. The EMAC module uses the EMAC_MAC_ISTAT.TSIS bit to indicate the status of the interrupt. The PTP supports the following three types of interrupts.

Auxiliary Snapshot Trigger

When an external event occurs on the EMAC_PTPAUX_MCG_IN[n] pins and a time stamp snapshot occurs, an auxiliary snapshot interrupt is triggered. The EMAC module uses the EMAC_TM_STAT.AUXTSTRIG bit to indicate the interrupt.

Target Time Reached

This interrupt is triggered when the system time becomes equal to the value written in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers. This interrupt can be used as an alarm and is indicated on the EMAC_TM_STAT.TSTARGT0 bit.

System Time Seconds Register Overflow

The system time seconds register overflow interrupt is triggered when the EMAC_TM_SEC register overflows from 0xFFFF FFFF to 0x0000 0000. This interrupt is indicated on the EMAC_TM_STAT.TSSOVF bit. As soon as EMAC_TM_SEC register overflows, the EMAC_TM_HISEC register increments by one.

Media Clock Generation and Recovery (MCGR)

The EMAC module supports dedicated hardware for media clock recovery. This hardware generates a reference clock, which is multiplied to get the desired media clock. The hardware clock multiplier is external to the EMAC.

The media clock generation and recovery process is mainly hardware-based and involves insignificant software.

NOTE: The PPS and MCGR modes are mutually exclusive in that all the instances selected in the configuration must operate either in PPS mode or in MCGR mode. Mixed operation mode, where a few instances

operate in PPS mode and others in MCGR mode, is not supported. Therefore, program all the instances to the same mode, irrespective of whether they are used or not.

The media clock generation and recovery support involve a dedicated external hardware and a thin layer of software. The dedicated hardware generates the reference clock (for example, 6 kHz or 8 kHz for AVB class A stream) which is then multiplied to get the desired media clock (for example an audio MCLK of 12.288 MHz).

The media clock recovery process involves host software extracting the presentation time from the 1722 frames and programming them into a CSR register, which is read by the EMAC to generate a reference clock. When the timer expires, the previous value is either toggled or a pulse is sent. The output at the recovered clock pin is generally a wave of 750 Hz ranging up to 8 kHz depending on the application that the clock is synced with, at the other end of the network.

As shown in the *Media Clock Generation and Recovery Block Diagram*, the time stamp array and the DMA module are external to the EMAC block.

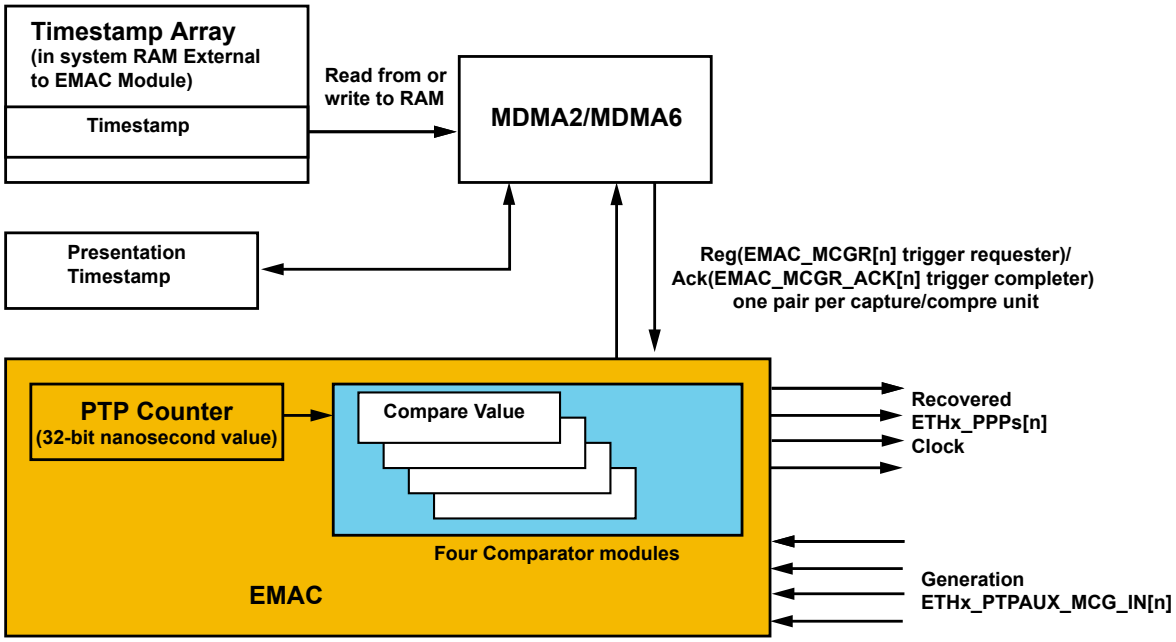


Figure 28-33: Media Clock Generation and Recovery Block Diagram

For the operation of the media clock generation and recovery, the EMAC supports:

- A 32-bit presentation time nanosecond counter that wraps at the full 32-bit value (and not after one second) to match up with the 1722 presentation time format.
- A mechanism to handshake with an external application (such as DMA) to program the presentation time into the CSR register. The hardware compares the programmed time stamp value with the presentation time and generates a wave based on the programmable settings.

Presentation Time Counter

The presentation time counter follows the 1722 presentation time format. In this format, time is represented as a 32-bit counter in nanoseconds that does a binary rollover on reaching the maximum value of 32'hFFFF_FFFF. This

counter is another view of the PTP system time (1588 timer). For a given PTP, system time (PTP[63:32] represents time in seconds; PTP[31:0] represents time in nanoseconds. There exists a corresponding presentation time (32-bit value in nanosecond). The presentation time computed is referred to as the Current Presentation Time (CPT).

The CPT can be viewed as derived from 64-bit PTP system time as:

PTPNS [63:0] = (PTP[63:32] × 32'd1,000,000,000) + PTP[31:0]

where,

PTPNS is the PTP system time converted into a 64-bit nanosecond value

CPT [31:0] = PTPNS [31:0]

Media clock recovery is possible when the system time is internally generated in the EMAC module. CPT is computed in the same way as the internal PTP system time generation. Because CPT and system time values are in nanoseconds, and must be synchronous, the increment cycle of both the timers are same.

The timer is updated at the same instance; however, the updated value could be different. The software must compute a separate 32-bit update value in nanoseconds for the CPT update. A CPT sampled at different edges of a triggering input results in the generation of media clock time stamps which are inserted in 1722-based AVBTP packets to recover the clock at the destination.

NOTE:
- The value of CPT can be obtained by reading the `EMAC_PRSNTM_NSEC` register.
- The EMAC module supports both fine grain and coarse grain correction modes.
- Only the digital rollover mode must be used for the PTP time. The CPT also uses the same increment value and CPT requires the time to be in nanoseconds.

Comparator Modules

As shown in Figure 28-33 Media Clock Generation and Recovery Block Diagram, the EMAC module supports four comparator modules to handle multiple media clocks that might be required.

The comparator modules handshake with the application to:

- Program the time stamps into the `EMAC_PPS[n]_TGTM_SEC` register when `EMAC_TM_CTL.PTGE` =1.
- Set the presentation time control bits of the instance for recovery mode

The time stamps received from the application are referred as Target Presentation Time (TPT).

The MCGR and PPS modes are mutually exclusive. When the `EMAC_PPS_CTL.MCGREN3` through `EMAC_PPS_CTL.MCGREN0` bits are set (= 1), the corresponding PPS instance[n] operates in MCGR mode. Otherwise, by default, the PPS instance[n] operates in PPS mode.

The comparator module sends up to two requests to the application for TPT write, when `EMAC_TM_CTL.PTGE` = 1, and the presentation time control bits of the instance are configured for MCGR mode. Subsequent requests are generated each time a presentation time match occurs. (CPT transitions from a value less than TPT, to a value greater than or equal to CPT). The first request is asserted when a specific comparator instance is configured to MCGR mode with a non-zero presentation time control; an additional request is made when the first data is

received into the comparator. This sequence allows the application to write the next TPT value while the EMAC module processes the previous TPT value for a match.

The TPT read from the EMAC_PPS[n]_TGTM_SEC register is considered as a future time. A toggle/pulse is generated in the next cycle when a match is detected. When a match is detected, the EMAC_PTPAUX_MCG_IN[n] trigger requester request for that comparator is asserted (to obtain next TPT) until the corresponding application acknowledgment is set.

The presentation control bits (for example, EMAC_PPS_CTL.PPSCMD1) determine the shape of the generated waveform. Also, these bits can be programmed to either toggle or generate a high/low pulse for one PTP clock cycle, upon match.

Clock Recovery

A match is when the free-running CPT value matches the received TPT value. An output signal, EMAC_PTPPPS[n], is asserted (toggle, low pulse, or high pulse) based on the programmed presentation control value.

Clock Generation

Based on the presentation control value programmed in the EMAC_PPS_CTL register, the comparator captures the presentation time and programs it into the EMAC_PPS[n]_TGTM_SEC register. A request is raised to the application to read the captured time stamp. Until the acknowledgment that the read operation is complete is received from the application, no new time stamps are captured.

Media Clock Generation and Recovery Flow

The *Media Clock Generation and Recovery Flow* figure shows the media clock generation and recovery flow.

For both media clock generation and recovery, set the EMAC_TM_CTL.PTGE field for CPT generation and set the EMAC_PPS_CTL.MCGREN3 through EMAC_PPS_CTL.MCGREN0 field for the corresponding instance to enable MCGR mode.

For media clock generation, the generated presentation time is sampled at positive, negative or at both the edges of trigger inputs (EMAC_PTPAUX_MCG_IN[n]) based on the mode programmed in the EMAC_PPS_CTL register. The sampled value is captured in the EMAC_PPS[n]_TGTM_SEC registers. After these registers are updated with the presentation time, the EMAC asserts the EMAC_PTPAUX_MCG_IN[n] trigger to request the application to read the presentation time. The application acknowledges that it has read the captured time stamp by asserting the corresponding EMAC_PTPAUX_MCG_IN[n] trigger completer.

For media clock recovery, the PPS instance asserts the EMAC_PTPAUX_MCG_IN[n] trigger to tell the application to program the target time seconds registers with the TPT value and acknowledge the corresponding request using the EMAC_PTPAUX_MCG_IN[n] trigger completer. When the application programs the target time seconds registers with the TPT value, the particular comparator compares the programmed TPT value with the CPT and generates the waveform on EMAC_PTPPPS[n] based on the presentation time control value programmed in the EMAC_PPS_CTL register.

NOTE:
- The consecutive triggers to sample the presentation time should always be asserted after a few PTP clock cycles to allow for synchronization delays. (This is not an issue when the input trigger maximum frequency is 8 kHz and the PTP clock is running at least at 1 MHz).
- DMA acknowledgment can be both posted as well as non-posted, in the media clock recovery mode.

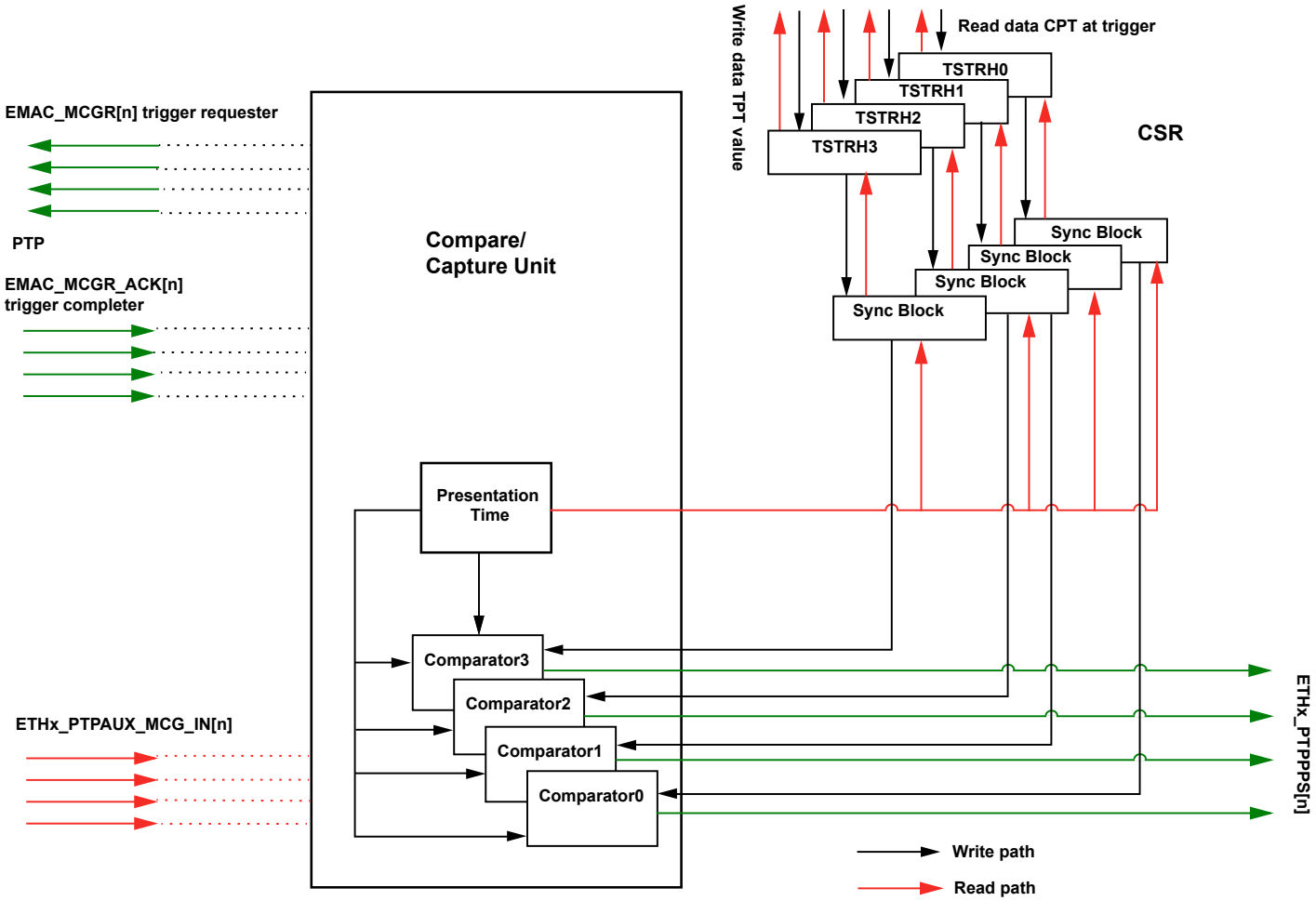


Figure 28-34: Media Clock Generation and Recovery Flow

PTP Time Stamp Offload Function

The PTP time stamp offload function enables the automatic generation of specific PTP packets when the MAC is working as a specific node in the PTP network. These packets may be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. This functionality helps in offloading certain functions of a PTP node with better accuracy and lower latency of responses.

Based on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application or based on reception of a particular PTP message.

The ***PTP Message Generation Criteria*** table indicates the PTP message generation criteria. Note that all other programming combinations are invalid for the PTP offload feature.

Table 28-67: PTP Message Generation Criteria

Programming (EMAC_TM_CTL)			Mode	Criteria for Generation of PTP Messages	PTP Message Type Generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
2'b00	0	1	Ordinary or boundary completer	SYNC message reception	Delay_Req
2'b00	1	1	Ordinary or boundary requester	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
2'b01	0	1	Transparent completer	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req
2'b01	1	1	Transparent requester	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
2'b11	X	X	Peer-to-peer transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp

NOTE: Clocks that support the peer delay mechanism must not generate delay request/delay response messages, according to the IEEE 1588-2008 specification. However, the EMAC supports this for flexibility, with a programmable control bit (`EMAC_PTO_CTL.DRRDIS`). Programs can use this bit to control the response generation for delay request/delay response message. For example, in transparent completer mode, delay request is generated in response to received sync only when the bit is reset.

For example, when the MAC is set as an ordinary or boundary completer clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in the *PTP Message Generation Criteria* table. The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the destination address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B19000000 for all except peer delay mechanism messages and 0x0180C200000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req, and Pdelay_Req PTP messages with a special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req, and Pdelay_Req PTP messages with a unicast destination address, it takes the SA field of the received packets and makes them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all received PTP messages are forwarded to the application along with the receive status, indicating whether the response was generated by the MAC and whether it satisfies the packet filtering logic of the MAC receiver.

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress time stamp of these two PTP messages are provided in the transmit TS status (`EMAC_TM_STAT`) register; an interrupt is generated.

In addition to matching messageType and versionPTP fields for basic PTP over Ethernet message detection, the following fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one step indication (1'b0).

The transportSpecific field is checked for the default PTP over Ethernet (4'h0) or 802.1AS mode (4'h1), when enabled.

Packet Generation

When enabled, the MAC can automatically generate PTP packets. The *Common PTP Message Header Fields* table describes the fields of specific PTP packets generated. The format and content of the packets are described in the following sections.

Table 28-68: Common PTP Message Header Fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30

Table 28-68: Common PTP Message Header Fields (Continued)

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

messageType

The following encoded values are used for PTP message types:

- SYNC – 4'h0
- Delay_Req – 4'h1
- Pdelay_Req – 4'h2
- Pdelay_Resp – 4'h3
- Delay_Resp – 4'h9

transportSpecific

The following transport protocol encoding is used:

- Default PTP over Ethernet – 4'h0
- 802.1AS mode – 4'h1

versionPTP

It is always configured to 2 because PTP version 2 is supported.

domainNumber

This contains the value from the `EMAC_PTO_CTL` register.

flagField

The following values are used:

- alternateMasterFlag (Octet 0 bit 0) – 1b0 for SYNC and Delay_Resp
- twoStepFlag (Octet 0 bit 1) – 1'b0 for SYNC and Pdelay_Resp
- unicastFlag (Octet 0 bit 2) – 1'b0 for Multicast Address, 1'b1 for Unicast Address

correctionField

For more information, see the Table 28-69 MAC Transmit PTP Modes table.

sourcePortIdentity

This field takes the value programmed in the EMAC_SRCPRT_IDNTY0 - EMAC_SRCPRT_IDNTY2 registers.

sequenceId

Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.

controlField

The following encoded values are used for controlField:

- SYNC – 8'h00
- Delay_Req – 8'h01
- Pdelay_Req – 8'h02
- Pdelay_Resp – 8'h05
- Delay_Resp – 8'h03

logMessageInterval

- SYNC – This contains logSyncInterval from the corresponding EMAC_LOG_MSG_INTVL register.
- Delay_Resp – This contains the sum of DRSYNCR and logSyncInterval value taken from the EMAC_LOG_MSG_INTVL register for a multicast PTP message and 8'h7F for unicast PTP message.
- Delay_Req, Pdelay_Req and Pdelay_Resp – 8'h7F where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of -15 to 15 for logSyncInterval fields, which translates to a range from 32.768 microsecond (2-15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- 2(30+N) ns, when N is negative (-1 to -15)
- 2N seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 21; therefore, the SYNC message is sent once every 2 seconds.

- When logSyncInterval is programmed to -1, the interval is 2-1 = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2-30 = 1 ns.
- When logSyncInterval is programmed to -5, the interval is 2-5 = 33.55 ms; therefore, the SYNC message is sent once every 33.55 milliseconds.

NOTE: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation 2(30+N), because of the non-binary nature of the nanoseconds field of the system time.

PTP Message Specific Fields

The PTP includes the following fields for messaging:

messageLength

There is no suffix supported; therefore, this field contains the length of the PTP message that includes 34 byte PTP common header and the body specific to the message type.

For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.

originTimestamp

This field is the captured egress time stamp for SYNC, Delay_Req, and Pdelay_Req PTP messages receiveTimestamp. For Delay_Resp PTP message, this field is the ingress time stamp of the corresponding received Delay_Req PTP message.

requestingPortIdentity

For Delay_Resp and Pdelay_Resp PTP messages, this field is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.

requestReceiptTimestamp

For the Pdelay_Resp PTP message, this field = 0.

One Step Time Stamp

The EMAC supports the one step time stamp feature. When the feature is enabled, the MAC identifies the offset in the packet and inserts the time stamp received from the application at that offset.

Depending upon the type of message and its mode, the MAC updates the following fields of transmit PTP packets:

- correctionField in the PTP header of messages.
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

The *MAC Transmit PTP Modes* table shows how the PTP mode is selected for the one step time stamp operation based on the configuration of EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA, and EMAC_TM_CTL.TSEVNTENA bits. The bits are updated for the incoming PTP packets based on the message type in that mode, during the one step time stamping operation.

Table 28-69: MAC Transmit PTP Modes

Programming			Mode	Per Packet Control*1			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE*2	OSTC*3	TTS*4	
X	X	X	N/A	1	X	X	Time stamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
2'b00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
							Delay_Req (correction field for residence time and Egress Asym Cor
2'b00	0	1	Ordinary or boundary completer	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym Cor)
2'b00	1	1	Ordinary or boundary requester	0	1	X	Sync (originTimestamp field) sync (correction field for sub nanosecond cor)

Table 28-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
2'b01	X	0	End-to-end transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym Cor)
						Ingress TS	Pdelay_Req (correction field for residence time and Egress Asym Cor)
						Ingress TS	Pdelay_Resp (correction field for residence time and Ingress Asym Cor)

Table 28-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
2'b01	0	1	Ordinary or boundary slave with support for peer delay mechanism or peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym Cor) (applicable only for peer-to-peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym Cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

Table 28-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
2'b01	1	1	Ordinary or boundary requestor with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) sync (correction field for sub nanosecond cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2'b10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress Asym cor)
						Ingress TS	Delay_Req (correction field for residence time and Egress Asym Cor)

Table 28-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control*1			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE*2	OSTC*3	TTS*4	
2'b11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym Cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

*1 The per packet control values provided here are the recommended settings used by devices in typical PTP operation, for the programmed mode.

*2 TTSE represents TTSE bit of transmit descriptor. The TTSE function is independent of the OST function and the programmed operation mode for OST. The MAC captures and returns the time stamp when the TTSE bit is set.

*3 OSTC represents OSTC bit of transmit descriptor.

*4 TTS represents the time stamp value provided in the TTSH, TTSL fields of transmit descriptor.

NOTE: Residence time/ turnaround time is calculated as the difference between the captured time stamp (egress time stamp) and the ingress time stamp. When the sub nanosecond feature is enabled, the residence time calculation includes sub nanosecond accuracy. Clocks supporting peer delay mechanism do not use delay request or response; however, it is included in OST for flexibility.

Enabling One Step Time Stamp

Programs can enable the one step time stamp feature for a packet by setting bit 20 (OSTC) in the transmit descriptor. For updating the correction field in certain PTP packets, the ingress time stamp must be given in the TSSL and TSSH fields. When the added IEEE 1588 higher word register option is selected, the time stamp is 80 bits. It also contains the content of the higher 16 bits of the `EMAC_TM_HISEC` register.

One Step Time Stamping for PTP Over UDP

The EMAC supports one step time stamp operation for PTP messages sent over UDP/IPv4 and UDP/IPv6. Because origin time stamp and/or correction field (based on the PTP message type and the programmed PTP mode) in the

PTP message are updated as part of the one-step time stamp operation, it also requires update to the checksum field of the PTP message sent over UDP/IP.

Checksum Update for One-Step Timestamping for PTP Over UDP

In the EMAC, checksum is updated for a packet, in the MTL before the packet is sent to the MAC where the PTP message is updated for one step time stamp operation. PTP requires the time stamp to be captured while the packet (SFD of the packet) is on the line and one step time stamp requires the origin time stamp and/or correction field to be updated in the PTP header while the packet is being transmitted on the line. Because the PTP message is present in the UDP payload, it is not possible to update the checksum field in the UDP header for modifications made to the PTP message, because it is already transmitted by then. Instead, the following options are supported that are in line with specification mentioned in Annex D and E of IEEE 1588-2008:

- When PTP message is sent over UDP/IPv4, the UDP checksum may be set to 0. The application must set the UDP checksum to 0 and set CIC to 0 or 1 while providing the packet to the MAC and the MAC does not update checksum for changes in origin time stamp or correction field.
- When PTP message is sent over UDP/IPv6, a transmitting node extends the UDP payload of all PTP messages by 2 octets beyond the end of the PTP message. The contents of the UDP checksum field or the final 2 octets of the UDP payload might be modified to ensure that the UDP checksum remains intact after modification of PTP fields. The application must add the two extra bytes at the end of the PTP message while providing the packet to the MAC and the MAC updates these two bytes to keep the checksum correct.

A programmable option (`EMAC_TM_CTL.CSC` field) supports UDP with a non-zero checksum for IPv4. When set, the two bytes at the end of the PTP message are updated to keep the UDP checksum correct as with PTP over UDP/IPv6. The application must add two bytes at the end of the PTP message sent to the EMAC and the EMAC updates these two bytes to keep the checksum correct.

Updating the pad bytes to keep the checksum correct requires the old value of correction field and/or origin time stamp in the packet. To get better latency and area, the application can set the origin time stamp field to zero, for PTP message in particular modes in which EMAC overwrites the origin time stamp. For example, EMAC operating in ordinary/boundary requester mode overwrites the origin time stamp field of the sync message, when OSTC is set. Because the sync message is originated by the port operating in ordinary/boundary requester mode, it is possible for the application generating the message to set the origin time stamp field to zero.

The transmit checksum engine is automatically enabled when this feature is selected. The EMAC uses the transmit checksum engine to identify the start of a PTP message in the UDP/IP packet. The checksum engine has the capability to detect errors in the header (L3) and payload (L4). These errors are not computed for some type of packets, which is indicated by "No" in the *Transmit Checksum Offload Engine Functions for Different Packet Types* table in the *Synopsis DesignWare® Cores Ethernet Quality-of-Service Databook*.

For one step, operation the packet is free of those errors and have "Yes/Not applicable" for the corresponding packet type in the *Transmit Checksum Offload Engine Functions for Different Packet Types* table. If an error is reported by the checksum engine or there is a "No" in For PTP messages sent over UDP/IP, no update is performed on the PTP message. See “Transmit Checksum Offload Engine” of the *Synopsis DesignWare® Cores Ethernet Quality-of-Service Databook* for more details about the errors detected by transmit checksum engine.

NOTE:
- The packet is sent with specific IP address and UDP port numbers and does not have IPv4 options/IPv6 extension headers. Applications must take care of these requirements before giving the packet to the EMAC. EMAC does not validate them. As PTP is identified by specific value in the UDP destination port and as EMAC does not investigate these fields, it assumes an IP packet with OSTC set as a PTP packet, and it is the responsibility of the application to set the OSTC bit only for IP packets which contains PTP messages.
- For PTP over UPD/IPv6 and PTP over UDP/IPv4 with CSC set, the EMAC does not verify the presence of the two pad bytes, instead it overwrites the two bytes after the PTP message.
- When the enable one step time stamp for PTP over UDP/IP feature is selected, the enable transmit TCP/IP checksum offload feature is automatically enabled, and this is enabled only for queue 0 in multiple transmit queue configurations. If the application requires OST support for PTP messages sent over UDP/IP for packets sent from queues other than 0, transmit checksum engine must be manually enabled for those transmit queues. If checksum engine is not enabled, no updates are done for PTP messages sent over UDP/IP even though OSTC is set.
- One step time stamp operation is not supported for tunneled packets

Multiple Channels and Queues Support

The EMAC supports up to eight queues and channels on transmit and receive paths. The EMAC Block Diagram and Interfaces figure shows the architecture of EMAC with multiple queues and channels.

Multiple Queues Support in the Transmit Path

The EMAC supports up to eight queues and channels on transmit and receive paths.

NOTE: EMAC0 supports eight transmit DMA channels and queues.

The number of transmit DMA channels is equal to the number of transmit queues and is direct one-to-one mapping. This enables the interleaving of packet/data transfer of multiple transmit DMA on the host/system bus and each transmit queue is reserved for the corresponding transmit DMA. Therefore, the integrity of packet transfer by each transmit DMA is maintained in the transmit queue and enables efficient use of the host bus for data transfers among multiple transmit DMAs without depending on the packet length, transmit buffer availability and so on.

When two transmit DMAs can transfer data into the same transmit queue, then a transmit DMA starts a packet transfer, the other transmit DMA cannot transfer data to the same transmit queue unless the previous packet transfer is complete. This means that the second DMA remains idle until the first packet transferred is complete, which is effectively a one-to-one mapping of the transmit DMA to the transmit queue.

This section describes how multiple queues and channels are operated in the EMAC.

Fixed Priority Scheme

The fixed priority scheme does not necessarily cause “starving” of lower priority channels for the SCB bus because of the capabilities of the SCB protocol and characteristics of the transmit DMA, described as follows:

- Read command requests from a transmit DMA for a burst of data transfers is driven on a separate SCB channel and is executed in one clock cycle itself. The actual data transfer transfers start on a separate SCB channel only after the request is accepted and it takes multiple clock cycles to complete with relatively large read access latencies.
- SCB requester is free to drive out subsequent requests even before the previous data transfers are completed.
- Transmit DMA requests for another data transfer only after the data transfer for its previous request is completed and transferred to the MTL TxQ, and space is available in TxQ to take in the next request.
- The EMAC can be programmed to control the maximum number of outstanding requests, requested burst sizes by DMA, and the burst sizes of each data transfer on the SCB with the help of the EMAC_DMA_SYSBMODE register.

When all transmit DMAs place a request simultaneously and are serviced by the fixed priority algorithm, all the requests are placed on the SCB bus in a window of 32 clocks (two clocks per SCB request, two SCB burst requests per transmit DMA request). The data transfer of the first request takes 32 clock cycles + the read-access latencies (when there are tens of cycles). Therefor the transmit DMA arbiter is already IDLE and has completed servicing all the transmit DMA requests before this data transfer completes.

Effectively, all the transmit DMA channels get equal access to SCB bus and the corresponding TxQs are filled up at the same rate. The data is read out from the TxQs as per the traffic class scheduler in the MTL. At a steady state (assuming that SCB bandwidth available to fetch data from system memory is always more than the operating line rate), the bandwidth for each channel is controlled by the priorities and settings of the traffic scheduler in the MTL.

Multiple Queues in the Receive Path

The EMAC supports up to eight receive channels. The number of receive channels is independent of the receive queues.

NOTE: EMAC0 supports eight receive DMA channels and queues.

The EMAC supports independent selection of the number of receive DMA channels and the number of receive queues in the MTL receive buffer. Therefore, the packets from any receive queue can be serviced by any of the receive DMA. As all the packets of receive queues are stored in a shared RXFIFO buffer in the MTL, the receive scheduler selects the receive queue whose packet data is to be transferred to the destination receive DMA, which then forwards the packet data to the host memory. The sequence of events in time are as follows:

1. Each receive DMA indicates whether it is ready to transfer data, that is, when the receive DMA has fetched the descriptor and has a receiver buffer ready, to accept the packet data. Each receive DMA also indicates the number of words that it can transfer, based on the space available in the receive buffer and the value programmed in the EMAC_DMA[n]_CTL.PBLX8 bit field.
2. The MTL receive scheduler matches the number of bytes/packets available for transfer at the top of all receive queues with the corresponding receive DMA ready status and the number of bytes requested by it.
3. If there are multiple successful matches, the scheduler selects the receive queue to be serviced, based on the arbitration schemes (strict priority or weighted strict priority) decided by the EMAC_MTL_OPMODE.RAA

bit field. In the weighted strict priority scheme, the weights of each receive queue is set in the EMAC_RQ[n]_CTL.RXQ_WEGT bit field.

4. The MTL receive controller fetches the data of the selected receive queue from the shared RXFIFO memory and starts the transfer of the requested PBL number of bytes to the DMA block or until the end of packet and the corresponding receive status is transferred (whichever is first).
5. After completing the current request, the scheduler repeats the process by starting a new arbitration cycle. However, it selects the same queues queue for servicing, if the end of packet is not transferred and the EMAC_RQ[n]_CTL.RXQ_FRM_ARBIT bits are 1, for the receive queue.

Receive Queue to DMA Mapping

The packets in the MTL receive queues can be routed to any one of the multiple DMA channels by programming the following registers:

- EMAC_RXQ_DMA_MAP0 register (for queues 0, 1, 2 and 3)
- EMAC_RXQ_DMA_MAP1 register (for queues 4, 5, 6 and 7)

The following types of receive queue to DMA mapping is possible EMAC_DMA[n]_IEN.RSE through programming:

- Static mapping
- Dynamic (per packet) mapping

Static Mapping

In static mapping mode, all packets of a receive queue are connected to a specific DMA channel. For example, all the packets from receive queue 0 can be routed to a DMA channel by programming the EMAC_RXQ_DMA_MAP0.Q0MDMACH and EMAC_RXQ_DMA_MAP0.Q0DDMACH bit fields.

Similarly, packets from other receive queues can be routed to any DMA channel by programming register fields corresponding to each queue.

Dynamic Mapping

In Dynamic (per packet) mapping mode, the destination DMA channel is chosen by the MAC core receiver for each packet.

In this mode, the destination DMA channel of a packet being read from a receive queue is not constant but decided independently for each packet. For example, if the EMAC_RXQ_DMA_MAP0.Q1MDMACH bit is set, the static mapping is disabled for receive queue 1 and the value in the EMAC_RXQ_DMA_MAP0.Q1MDMACH bit is ignored. The destination DMA channel is chosen by the MAC receiver for each packet, depending on the following in decreasing order of priority.

1. L3-L4 filter Based DMA Selection. When the enable layer 3 and layer 4 packet filter feature is present and enabled in the program configuration, the TCP/UDP and IP header fields of the received packet are matched against the corresponding values programmed and enabled for comparison in the EMAC_L3L4_CTL[i] register. If the match is successful, the DMA channel number programmed in the EMAC_L3L4_CTL[i].DMCHN bit field is selected as the destination DMA channel number provided

the EMAC_L3L4_CTL[i].DMCHEN bit of the same register is set. If none of the L3-L4 registers give a comparison match, the EMAC proceeds to the next step.

2. Extended VLAN Based DMA Selection. Extended routing is applicable only if the VLAN filter has passed. Routing is only based on perfect filter result. Each perfect filter has a programmable DMA channel enable and a DMA channel number field. Routing is applicable for a filter when the DMA channel enable bit is set.

   The frame is routed to the smallest matching filter's DMA channel provided it is enabled. If that filter's DMA channel number is not enabled, the frame gets routed to channel 0. For example, if a frame's VLAN tag matches filters 7, 3, and 1. Then the MAC checks if filter 1's DMA channel number is enabled through programming. If yes, the frame is routed to the programmed value; otherwise it is routed to channel 0. When the inverse filter is enabled, the frame is routed to the least mismatched filter's DMA channel number provided it is enabled. If the DMA channel enable bit is not set, then the frame is routed based on DA based addressing or to channel 0.

   If the hash filter is also enabled, it is used to determine the filter result only. Routing still depends on the enabled perfect filters. If none of the perfect filters are enabled or if all of them are bypassed, the VLAN filter based routing is not performed. The frame is routed through DA based addressing or to channel 0. If all the perfect filters give a fail result and the hash filter has passed, the VLAN filter result is a pass. However, routing is based on DA based addressing or to channel 0. The behavior is similar when inverse filtering is enabled.

3. Ethernet DA-Based DMA Selection. The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding EMAC_ADDR[i]_HI.DCS bit field (when enabled) determines the destination DMA channel number. If none of the previous operations can make a successful match/decision, then the packet is routed to DMA channel 0 by default.

Broadcast/Multicast Packet Duplication

The EMAC supports the Broadcast/Multicast Packet Duplication feature to send the received broadcast/multicast packets to multiple DMA channels.

In a virtual system that supports multiple guest OS's (for instance, each OS owning a DMA channel), where each DMA could act as an independent Ethernet node, it may be required to route the received multicast/broadcast packets to all/multiple DMA channels. This is implemented by using the MTL RX FIFO read controller to read the packet from the same queue multiple times and route the same packet to the specified multiple DMA channels.

To maintain backward compatibility, through software control, the DMA channel selection (EMAC_ADDR[i]_HI.DCS) field are redefined to denote multiple DMA channels.

The EMAC_ADDR[i]_HI.DCS bit field determines the DMA channel number to which the received packet (that matches the MAC address present in that register) must be routed.

The EMAC_ADDR[i]_HI.DCS is a programmable field that supports more than one DMA channel number to be selected.

The EMAC_ADDR[i]_HI.DCS field can hold any one of the following values:

- Binary value of the channel number when the EMAC_MAC_EXT_CFG.PDC bit is set to 0

- One-hot value of the channel number when the EMAC_MAC_EXT_CFG.PDC bit is set to 1

Use DA based routing to enable the multicast/broadcast packets to route to multiple DMA channels. Packet duplication is not supported in other DMA channel routing mechanisms such as extended VLAN, or L3-L4 based routing. In other words if the destination DMA channel is decided by the VLAN or L3-L4 filter result, it is routed only to the specific channel even if the DA of the packet matches the multicast address in the EMAC_ADDR[i]_HI/EMAC_ADDR[i]_LO registers.

The packet duplication feature is supported only on the highest MTL queue configured. Therefore, packet duplication for broadcast/multicast packets requires that the EMAC_RXQ_CTL1.MCBCQ bit field is programmed to the highest RxQ present in the configuration.

In packet duplication, the MTL read controller sequentially transfers the same packet to all the required receive DMA channels. This packet is removed from the MTL FIFO only after the packet transfer completes. Therefore, this feature is suitable for smaller packet sizes or very low bandwidth traffic. This ensures that the MTL RxQ does not overflow due to the long period when the packet remains in the queue during the packet duplication.

NOTE: When more than one bit is set for the DMA channel number, a packet that is not routed to the highest receive queue is routed to only one DMA channel as indicated by the lowest bit set in the EMAC_ADDR[i]_HI.DCS bit field.

Assigning Tag Priorities to Transmit and Receive Queues

The EMAC supports assigning tag priorities to transmit and receive queues through programming registers.

The VLAN tag priorities can be assigned to transmit queues by programming the EMAC_TXQ_PRTY_MAP0.PSTQ0 through EMAC_TXQ_PRTY_MAP1.PSTQ7 bit fields in the following registers:

- EMAC_TXQ_PRTY_MAP0 (for transmit queues 0 to 3)
- EMAC_TXQ_PRTY_MAP1 (for transmit queues 4 to 7)

For example, if the program needs to assign a VLAN tag priority of 3 to transmit queue 0, set bit [3] in the EMAC_TXQ_PRTY_MAP0.PSTQ0 bit field.

The same VLAN tag priority can be set for multiple transmit queues. The transmit packet association to transmit queue is governed by the application, so the MAC does not route the packet based on transmit queue priority mapping; it is only used for Pause Flow Control (PFC). The settings in the PSTQ[n] field determines the transmit queues blocked for transmission when the PFC packet is received with corresponding VLAN tag priorities enabled.

The VLAN tag priorities can be assigned to receive queues by programming the EMAC_RXQ_CTL2.PSRQ0 through EMAC_RXQ_CTL3.PSRQ7 bit fields. The bit corresponding to the VLAN tag priority can be set in the PSRQ field for assigning that priority to the receive queue. For example, to assign a VLAN tag priority of 3 to receive queue 0, set bit [3] in the EMAC_RXQ_CTL2.PSRQ0 bit field. The VLAN tag priority assigned to the receive queue must be unique, that is, more than one receive queue cannot be assigned the same VLAN tag priority. However, more than one VLAN tag priority can be assigned to same receive queue.

The settings in the PSRQ bit field is used for VLAN tagged receive packet routing to receive queues as well as for PFC based transmit flow control. The received VLAN tagged receive packet is routed to the receive queue that has the VLAN tag priority match. In PFC based transmit flow control, the PSRQ bit field corresponding to a particular receive queue is used for enabling VLAN tag priorities in the PFC packet transmitted when corresponding receive queue threshold levels are reached.

Distribution of Receive Packets from MAC to Receive Queues

The EMAC provides multiple receive queues to support classification and distribution of ingress packets. The MAC receiver pushes the ingress packets to the receive queues based on packet type (tagged, untagged, multicast, PTP, AVB, DCB, and so on), tag priorities, MAC packet filter results, preemptable or express packet, and so on. The distribution is controlled by the settings of the EMAC_RXQ_CTL0 through EMAC_RXQ_CTL4 and EMAC_MACPKT_FILT register fields.

EMAC additionally supports distribution of ingress packets based on the user-defined type field of the ingress packets when the EMAC_RXQ_CTL1.TBRQE bit is enabled. Up to 8 types for matching are programmed in the MAC_TMRQ_Regs(n) registers which are accessible through the EMAC_INDR_ACC_CTL and EMAC_INDR_ACC_DAT registers. Each MAC_TMRQ_Regs(n) contains

- TYP field: contains 16-bit type value of the packet that needs to be compared with the received Ethernet packet. This field is valid when programmed to a value greater than or equal to 0x0600.
- TMRQ field: contains the receive queue number to which ingress packet is pushed when its type field matches TYP.
- PFEX field: This bit indicates whether the TYP is applicable for express packets or preemptable packets.

Based on the packet filter settings, the MAC receiver decides to either drop or forward the ingress packet. However, the results of layer 3 and layer 4 filtering, mentioned in Layer 3 and Layer 4 Frame Filtering are not considered for the distribution of ingress packets into the receive queues. Only the results of DA/SA/VLAN filters mentioned in Source Address or Destination Address Filtering and VLAN Filtering are applicable.

Filtering of the broadcast packets is exclusively controlled by the EMAC_MACPKT_FILT.DBF bit. The filtering of multicast packets is controlled by the EMAC_MACPKT_FILT.PM bit, in addition to the DA/SA match.

By default, packets that fail any of the DA/SA filters, are dropped by the MAC receiver. However, packets that fail the DA/SA/VLAN filter can still be forwarded to the application when the EMAC_MACPKT_FILT.RA bit is 1. Even when the RA field is 0, VLAN filter fail packets can be forwarded if the EMAC_MACPKT_FILT.VTFE bit is 0.

The ingress packets go through the following operations in the decreasing order of priority.

RA = 0: Ingress packets are dropped when any one of the following conditions is true.

- DA/SA filter fail
- VLAN filter fail and EMAC_MACPKT_FILT.VTFE bit is 1

EMAC_MACPKT_FILT.RA = 1: Ingress packets that fail the DA/SA filter are also forwarded to the receive queue as mentioned in the *Routing DA/SA Failed Packets to Receive Queue* table.

Table 28-70: Routing DA/SA Failed Packets to Receive Queue

Packet Type	TSN Type	Destination Receive Queue Determined By
Broadcast/Multicast	Express	if MFFQE = 1, MFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
Unicast	Express	if MFFQE = 1, MFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
All	Preemptable	FPRQ field

`EMAC_MACPKT_FILT.RA` = 1 and `EMAC_MACPKT_FILT.VTFE` = 0: Ingress tagged packets that pass the DA/SA filter but fail the VLAN filter, are forwarded to the receive queue as mentioned in the *Routing DA/SA Pass But VLAN Filter Fail Packets to Receive Queue* table.

Table 28-71: Routing DA/SA Pass But VLAN Filter Fail Packets to Receive Queue

TSN Packet Type	Destination Receive Queue Determined By
Express	if VFFQE = 1, VFFQ; else RxQ0
Preemptable	RQ field

Ingress packets that pass the DA/SA/VLAN filters are forwarded to the receive queue. The *Priority Routing of Filter Pass Packets When OMCBCQ = 0* table shows the routing priority in the decreasing order when the `EMAC_RXQ_CTL1.OMCBCQ` bit field is 0.

Table 28-72: Priority Routing of Filter Pass Packets When OMCBCQ = 0

Packet Type	TSN Type	Tagged	Destination Receive Queue Determined By
Broadcast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Unicast AVTP Control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
Unicast PTP over Ethernet	Express	No	PTPQ
		Yes	PTPQ or PSRQ based on the value of TPQC
Unicast DCB/LLDP Control	Express	No	DCBPQ

Table 28-72: Priority Routing of Filter Pass Packets When OMCBCQ = 0 (Continued)

Packet Type	TSN Type	Tagged	Destination Receive Queue Determined By
Remaining Unicast	Express	No	If TBRQE = 1, TBRQ; else UPQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	If TBRQE = 1, TBRQ; else FPRQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ

The *Priority Routing of Filter Pass Packets When OMCBCQ = 1* table shows the routing priority in the decreasing order when the `EMAC_RXQ_CTL1.OMCBCQ` bit field is 1.

Table 28-73: Priority Routing of Filter Pass Packets When OMCBCQ = 1

Packet Type	TSN Type	Tagged	Destination Receive Queue Determined By
All AVTP Control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
All PTP over Ethernet	Express	No	PTPPQ
		Yes	PTPQ or PSRQ based on TPQC
Remaining Broad-cast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Unicast DCB/LLDP Control	Express	No	DCBPQ
Remaining Unicast	Express	No	If TBRQE = 1, TBRQ; else UPQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	If TBRQE = 1, TBRQ; else FPRQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ

NOTE: If the RxQ pointed to by the various fields in tables above is not enabled in the `EMAC_RXQ_CTL0` register as per the packet type, the packet is written to

- RxQ0 for express packets
- FPRQ field for preemptable packets
- If RxQ0/FPRQ is not enabled, the packet is dropped.

If the `EMAC_TM_CTL.AV8021ASMEN` bit is 1, the MAC receiver checks only for untagged PTP packets. Tagged packets with a PTP payload are considered as remaining (Unicast/Broadcast/Multicast) tagged packets in the above tables.

Traffic Shaper

Traffic shaping is the process of distributing frames/packets evenly in time to smooth out network traffic. This reduces traffic bursts which can overwhelm the network. The following sections describe how to implement and manage this feature.

Data Center Bridging

The data center bridging feature supports three types of algorithms: weighted round robin, deficit weighted round robin, and weighted fair queuing.

Weighted Round Robin

In the Weighted Round Robin algorithm used in the Data Center Bridging feature, all queues are serviced in round-robin order based on weight setting.

In Weighted Round Robin (WRR) algorithm, each queue is assigned a weight based on the percentage of configured bandwidth. All queues are serviced in the round-robin order according to the weight settings. This algorithm is less complex and requires less hardware as compared to other algorithms. Large packets get more bandwidth in this algorithm.

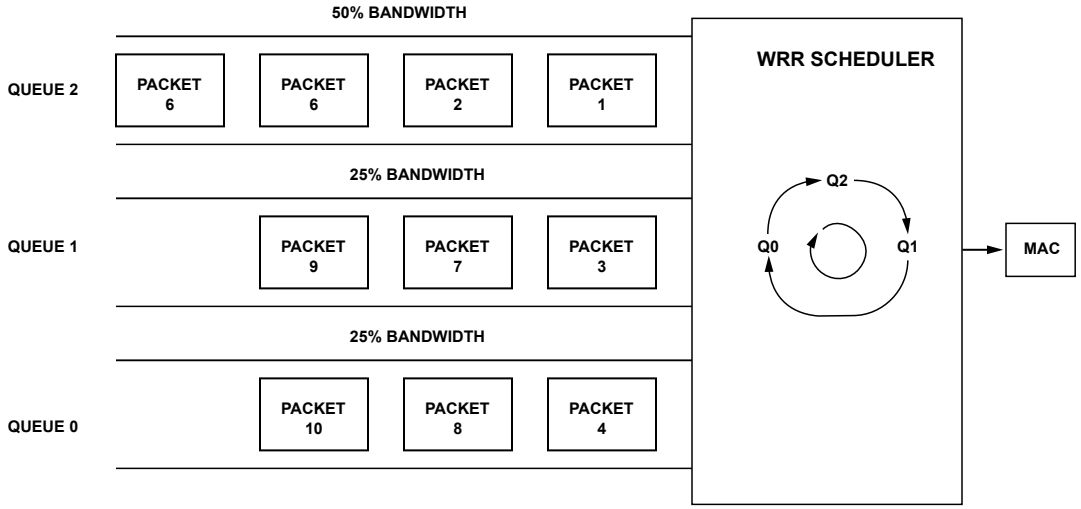


Figure 28-35: Weighted Round Robin Example

In the *Weighted Round Robin Example* figure, Queue 2, Queue 1, and Queue 0 are configured for 50%, 25%, and 25% bandwidth, respectively. The corresponding weights are two, one, and one. Before the WRR scheduler is started, Queue 2 has four packets; Queue 1 has three packets; and Queue 0 has three packets. The packets are transmitted in the following order:

Table 28-74: Weighted Round Robin Sequence Example

Packet Sequence	Queue Number
Packet 1	Queue 2
Packet 2	Queue 2
Packet 3	Queue 1
Packet 4	Queue 0

Table 28-74: Weighted Round Robin Sequence Example (Continued)

Packet Sequence	Queue Number
Packet 5	Queue 2
Packet 6	Queue 2
Packet 7	Queue 1
Packet 8	Queue 0
Packet 9	Queue 1
Packet 10	Queue 0

Deficit Weighted Round Robin

When the Data Center Bridging feature is supported, programs can use the deficit weighted round robin algorithm to assign a weight based on the percentage of configured bandwidth. A deficit counter holds the credit for transmission.

In Deficit Weighted Round Robin (DWRR) algorithm, each queue is assigned a weight based on the percentage of configured bandwidth. A deficit counter holds the credit for transmission. The quantum value, proportional to the bandwidth, is added to the deficit counter every time a queue is scanned. The packet in the queue is selected for transmission only if the credit in the deficit counter is less than or equal to the size of the packet available at the head of the queue. A selected queue is serviced until the outstanding credit is lesser than the packet size at the head of the queue during a particular queue scan cycle. After this, the next lower priority queue is serviced. When the packets in a queue are over, the credit becomes zero for that queue scan cycle. This algorithm is less complex, and it provides more fairness in scheduling as compared to other algorithms.

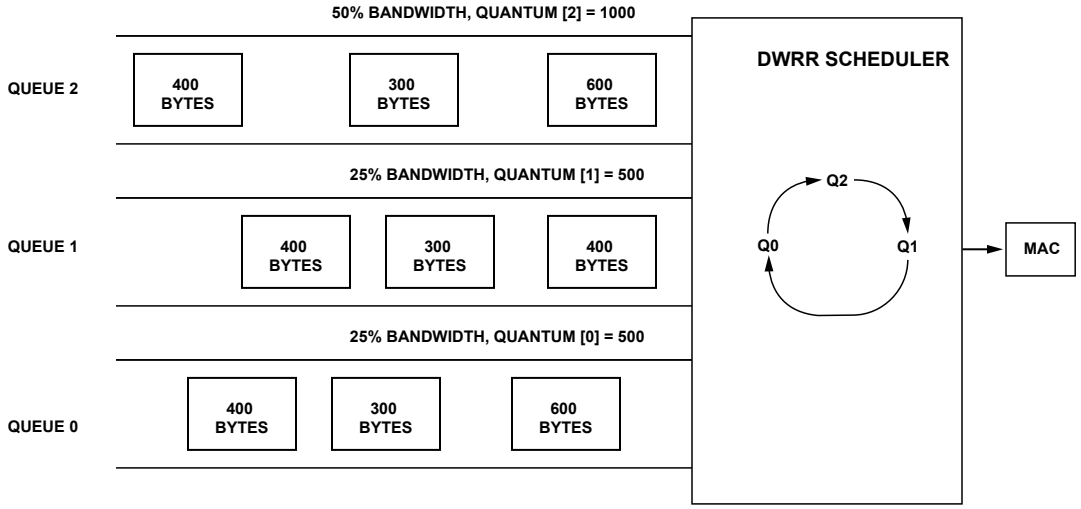


Figure 28-36: Deficit Weighted Round Robin Example

In the *Deficit Weighted Round Robin Example* figure, Queue 2, Queue 1, and Queue 0 are configured for 50%, 25%, and 25% bandwidth, respectively. The corresponding quantum values are 1000, 500, and 500. Before the DWRR scheduler is started, each queue has three packets. The *Deficit Weighted Round Robin* table shows the queues scan

cycle number and the packet size at the head of queues, deficit counter value before and after packet in selected queue is transmitted, and queue scheduled for transmission.

Table 28-75: Deficit Weighted Round Robin

Queues Scan Cycle Number	Packet Size at the Head of Queue			Deficit Counter Value Before the Packet is Transmitted in Selected Queue			Deficit Counter Value after the Packet is Transmitted in Selected Queue			Queue Scheduled for Transmission
Queue	Q2	Q1	Q0	Q2	Q1	Q0	Q2	Q1	Q0	
1	600	400	600	1000	500	500	400	500	500	Q2
	300	400	600	400	500	500	100	500	500	Q2
	400	400	600	100	500	500	100	100	500	Q1
	400	300	600	100	100	500	100	100	500	None
2	400	300	600	1100	600	1000	700	600	1000	Q2
	N/A	300	600	0	600	1000	0	300	1000	Q1
	N/A	400	600	0	300	1000	0	300	400	Q0
	N/A	400	300	0	300	400	0	300	100	Q0
	N/A	400	400	0	300	100	0	300	100	None
3	N/A	400	400	0	800	600	0	400	600	Q1
	N/A	N/A	400	0	N/A	600	0	0	200	Q0
	N/A	N/A	N/A	0	N/A	N/A	0	0	0	None

Weighted Fair Queuing

In the Weighted Fair Queuing (WFQ) algorithm used in the Data Center Bridging feature, each queue is assigned a weight based on the percentage of configured bandwidth. The algorithm computes the finish time of packets at the head of the queues based on the packet size and weights for the queue. The packets with earliest finish time is scheduled or transmission first. The algorithm is more complex and less scalable as compared to other algorithms.

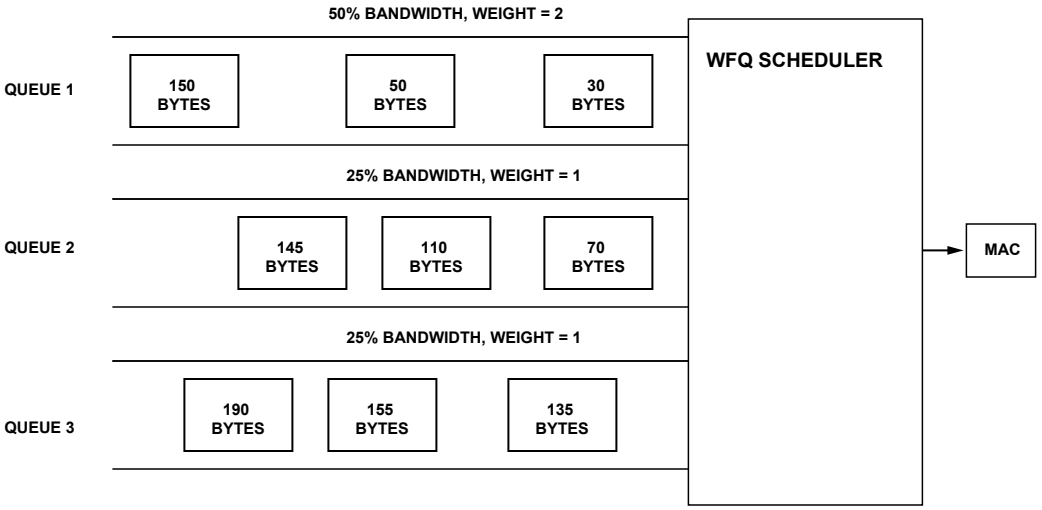


Figure 28-37: Weighted Fair Queuing Example

In the *Weighted Fair Queuing Example* figure, Queue 2, Queue 1, and Queue 0 are configured for 50%, 25%, and 25% bandwidth, respectively. The corresponding weights are 2, 1, and 1. Before the WFQ scheduler is started, each queue has three packets. The *Weighted Fair Queuing Algorithm Example Values* table shows the packet size at the head of the queue, computed finish time, and the queue scheduled for transmission.

Table 28-76: Weighted Fair Queuing Algorithm Example Values

Packet size at the Head of the Queue			Computed Finish Time in Clock Cycles (packet size or weights)			Queue Scheduled for Transmission
Q2	Q1	Q0	Q2	Q1	Q0	
30	70	135	15	70	135	Q2
50	70	135	25	70	135	Q2
150	70	135	75	70	135	Q1
150	110	135	75	110	135	Q2
N/A	110	135	N/A	110	135	Q1
N/A	145	135	N/A	145	135	Q0
N/A	145	155	N/A	145	155	Q1
N/A		155	N/A	N/A	155	Q0
N/A		190	N/A	N/A	190	Q0

Audio Video Bridging

The EMAC supports the AV data transfer in 100 Mbps and 1000 Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
- IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network.
- IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

While the AVB standards were originally designed for audio and video over standard Ethernet, extending these to control streams additionally benefits the industrial and automotive sectors. Time Sensitive Networking (TSN) is a set of standards developed by the Time Sensitive Networking Task Group with the intent to extend the future AVB enhancements to other sectors that could benefit.

Bridges are increasingly used to interconnect devices that support scheduled applications (for example, industrial automation, process control and vehicle control). The IEEE 802.1Qbv-2015 (Enhancements to Scheduling Traffic) provides performance assurances of latency and delivery variation to enable these applications in an engineered LAN while maintaining the existing guarantees for the credit-based shaper and best-effort traffic.

The IEEE 802.3br (Interspersing Traffic) and IEEE 802.1Qbu (Frame Preemption) enables the suspension of a large preemptable packet being transmitted by the MAC layer to allow one or more express packets to be transmitted before the transmission of the preemptable packet is resumed. This provides the capability to schedule express traffic packets with minimal delays/latencies or at predictable times with efficient utilization of the line bandwidth.

Transmit Path Functions

Transmit paths of queue 0 and the highest enabled queue are enabled by default.

The transmit path of queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for the MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of a size equal to the programmed threshold limit.

The transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets
- The credit for the queue is positive as per the algorithm

The credit-based shaper algorithm can be disabled for all queues or lower-priority queues. For example, the program can either disable the credit-based shaper algorithm for queue 1 and queue 2 or only for queue 1. The program

should not disable the credit-based shaper algorithm for queue 2 and enable it for queue 1. When the credit-based shaper algorithm for a queue is disabled, the channel uses the default strict priority algorithm.

Each transmit DMA has a separate descriptor chain for fetching the transmit data. The transmit channel that gets the access to the system bus depends on the DMA arbiter.

The transmit path has a shared FIFO (MTL layer) for each queue. The data fetched by the DMA is put in the respective part of the FIFO. The MTL transmit queue scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

- If the packet is available in the channel and has a positive or zero credit
- If the higher priority queue has no packet waiting in the FIFO

If the credit-based shaper algorithm is disabled for all queues, the packet transmitted from a queue is selected based on the priority scheme described in the following table.

Table 28-77: Weight for DMA Channels

`EMAC_DMA[n]_TXCTL.TCW` Field	**Transmit Channel Weight**
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

Receive Path Functions

The receive path of Queue 0 is enabled by default. All traffic is received on this channel. Programs can enable the receive paths of additional queues. By enabling the receive paths of multiple channels, the received data is demultiplexed and sends the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN priority tag value. This status is updated in the extended status field of the receive descriptor as explained in Receive Descriptor.

All received packets with the EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits[15:8] in the `EMAC_Q[i]_TXFLOW_CTL` register of the corresponding queue.

- AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on Queue 0, by default. To receive these packets on any other queue, programEMAC_RXQ_CTL1.AVCPQ bit field. Like the AV data packets, the tagged AV control packets are received based on the programmed priority value.

In addition to the AV packets, the untagged PTP packets on any queue can also be received. By default, the PTP packets (tagged or untagged) are received on queue 0. To receive these packets on any other queue, program the EMAC_RXQ_CTL1.PTPQ bits.

Credit-Based Shaper Algorithm

The MTL Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in queue 0. The additional queues can be programmed to use the credit-based shaper algorithm.

The following sections provide information for implementing the credit-based shaper algorithm:

- Credit value
- idleSlopeCredit and sendSlopeCredit values
- Bandwidth status

Credit Value

The credit value is part of the credit-based sharper algorithm used by the MTL queue scheduler. The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps and 8 ns for 1000 Mbps. The credit to add or subtract per cycle can be fractional based on the required idleSlope and sendSlope values, as described in the following table.

Table 28-78: Credit Value per Transmit Cycle Example

Mode	Values	Description
100 Mbps	portTransmitRate = 100 Mbps idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) sendSlope = 30 Mbps	credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the application bus.

The credit for a queue builds up only when the packet is available, but it cannot be transmitted because the MAC is sending a packet from another queue. The EMAC supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. You can enable this mode is enabled using the EMAC_TQ[n]_ETS_CTL.AVALG bit of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no

packet waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit Values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024, such as, 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229. In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values programmed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x1024, respectively.

Bandwidth Status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the `EMAC_TQ[n]_ETS_STAT` registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The `EMAC_TQ[n]_ETS_STAT` registers show the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in the `EMAC_TQ[n]_ETS_CTL.SLC` bit field of the respective queue. For example, when programming two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue that uses the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue that uses the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, queue 1 and queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and queue 0 uses the strict-priority algorithm. If queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

Slot Number Function

The slot number function is used to schedule the data fetch from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.

The slot number at which the DMA fetches the data from system memory is programmed in the Transmit Descriptor Word 3 (TDES3) register. This 4-bit field allows the application to schedule data up to 16 slots; a

programmable slot interval that ranges from 1µs to 4096 µs with a granularity of 1µs. This field is applicable only for the AV channels.

When the DMA fetches a transmit descriptor, it compares the slot number of the transmit descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated based on a programmable slot interval (with a range of from 1µs to 4096 µs) of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state until there is a match.

To enable the DMA to fetch the data only if it matches the current slot or the next two slots, program the `EMAC_DMA[n]_SFN_CTLSTAT.ASC` bit of the corresponding DMA channel.

NOTE: If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

The program also enables the check for slot number by setting `EMAC_DMA[n]_SFN_CTLSTAT.ASC` bit. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, the `EMAC_DMA[n]_SFN_CTLSTAT.RSN` bits indicate the value of the reference slot number in DMA.

Queue Modes

Programs can enable a transmit queue for generic DCB and AV traffic. When the program selects multiple transmit queues without selecting the DCB feature, the queues are enabled for generic queuing using the WRR or WSP algorithms.

When the program selects the enable data center bridging feature, DCB is enabled for all selected transmit queues and queuing is based on the WRR, WSP, DWRR, or WFQ algorithms. To enable a transmit queue for AV, select the enable audio video bridging option and then the enable support for AV in transmit Queue # option. The queuing is based on the CBS or SP algorithms.

The program can enable a receive queue for generic, DCB, or AV traffic. A particular queue enabled for generic DCB or AV based routing is determined by the RXQ#EN bit field (`EMAC_RXQ_CTL0` register) of the corresponding queue. The following sections explain how receive queues are enabled based on the selected features.

Multiple Receive Queues without the DCB Feature

When this mode is selected, all queues are enabled for generic queuing based on the VLAN tag priority. The VLAN tag priority should match the PSRQ field of the `EMAC_RXQ_CTL2` and `EMAC_RXQ_CTL3` registers. By default, untagged packets are routed to the receive queue specified in the `EMAC_RXQ_CTL1.UPQ` bit field. Queue 0 is the default value of the UPQ field. The program can override the default value with any other value for the UPQ field. The receive packets can also be routed to a particular DMA channel based on the `EMAC_ADDR[i]_HI.DCS` bit field.

Multiple Receive Queues with DCB Feature

When this mode is selected, DCB is enabled for all selected receive queues. The queuing is done based on the VLAN tag priority for DCB data packets. The VLAN tag priority should match the PSRQ field of the

EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The receive packets can also be routed to a particular DMA channel based on the EMAC_ADDR[i]_HI.DCS bit field. The DCB control packets are routed based on the EMAC_RXQ_CTL1.DCBCPQ bit field.

Multiple Receive Queues with AV Feature

When this mode is selected, AV is enabled for all selected receive queues. The queuing is done based on the VLAN tag priority for DCB data packets. The VLAN tag priority should match the PSRQ field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The receive packets can also be routed to a particular DMA channel based on the EMAC_ADDR[i]_HI.DCS bit field. The AV control packets (tagged or untagged) are routed based on the EMAC_RXQ_CTL1.AVCPQ bit field. The PTP over Ethernet packets are routed based on the EMAC_RXQ_CTL1.PTPQ bit field.

Queue Priorities

The transmit queue priorities are programmed using the EMAC_TXQ_PRTY_MAP0 and EMAC_TXQ_PRTY_MAP1 registers. The receive queue priorities are programmed using the EMAC_RXFLOW_CTL register.

The priority of a receive queue is programmed using the corresponding field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. Assign the priority in the following order:

1. AV queue (high priority)
2. DCB queue (medium priority)
3. Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programmed queue on transmit side. The MAC uses the programmed priorities for blocking the transmit queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

TCP/IP Offloading Features

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has an Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path.

The TCP Segmentation Offload (TSO) engine is useful in offloading the TCP segmentation functions to the hardware.

This section contains the following sub-sections:

- Transmit Checksum Offload Engine
- Receive Checksum Offload Engine
- TCP/IP Segmentation Offload (TSO) Engine

- UDP/IPv4 Fragment Offload (UFO) Engine
- IPv4 ARP Offload Engine

Transmit Checksum Offload Engine

The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path where the software can offload the checksum insertion task to the hardware. In the transmit path the MAC calculates the checksum and inserts it in the transmit packet. This feature reduces the load on the software and can improve the overall throughput of the system. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 Bits[17:16]).

NOTE: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, when this function is enabled, the transmit FIFO automatically operates in the store-and-forward mode even if the EMAC is configured for threshold (cut-through) mode.

NOTE: The application must ensure that the transmit FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason being that when space is not available to accept the programmed burst length of data, then the MTL transmit FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the program must enable the checksum insertion only in the packets that are less than the number of bytes, given by the following equation:

Packet size < TxQSize() – (PBL + 7) x 4,

Where, TxQSize is indicated by the `EMAC_TQ[n]_OPMODE.TQS` bit field and PBL by the `EMAC_DMA[n]_TXCTL.TXPBL` bit field.

IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit header checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the type field of IPv4 datagrams. The integrity of the header fields is indicated by the 16-bit header checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the type field of Ethernet packet has the value 0x0800 and the version field of an IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

NOTE: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP header error status bit in the transmit status (bit 0 in RDES3 Normal Descriptor (Write-Back Format)). This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the version field of IP header is not equal to 0x4.

- The IPv4 header length field indicates a value less than 0x5 (20 bytes).
- The total packet length is less than the value given in the IPv4 header length field.

- For IPv6 datagrams:
  - The Ethernet type is 0x86dd but the IP header version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding header length field in an extension header) is completely received.

TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The transmit COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the checksum field of the input packet. This engine includes the checksum field in the checksum calculation, and then replaces the checksum field with the final calculated checksum.
- The engine ignores the checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

NOTE: For ICMP-over-IPv4 packets, the checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16’h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the payload checksum error status bit in the transmit status vector (bit 12 of TDES3). This engine sets the payload checksum error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the payload length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

The following table describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Table 28-79: Transmit Checksum Offload Engine Functions for Different Packet Types

Packet Type	Hardware IP Header Checksum Insertion	Hardware TCP/UDP Checksum Insertion
Non-IPv4 or IPv6 packet	No	No

Table 28-79: Transmit Checksum Offload Engine Functions for Different Packet Types (Continued)

Packet Type	Hardware IP Header Checksum Insertion	Hardware TCP/UDP Checksum Insertion
IPv4packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Yes	Yes
IPv6packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Not Applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6packet with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 main header) • Hop-by-hop options (in IPv6 extension header) • Destinationsoptions • Routing (with segment left = 0) • Routing (with segment left > 0) • TCP, UDP, orICMP • Authentication • Any other next header field in main or extension headers	 Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable	 Yes No Yes No No Yes No No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	 Yes (IPv4 tunnel header) Yes (IPv4 tunnel header)	 No No
IPv6 Tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 Not applicable Not applicable	 No No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive Checksum Offload Engine

The EMAC provides the Checksum Offload Engine that is used to detect any error in an IPv4 or IPv6 packet in the receive path. The MAC verifies the checksum field of the received packet with the internally calculated checksum and provides the status. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

Here, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the enable double VLAN processing option is selected and the `EMAC_VLANTAG_CTL.EDVLP` bit is set.

The receive COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when the `EMAC_RQ0_OPMODE.DIS_TCP_EF` bit is reset and the `EMAC_RQ0_OPMODE.FEP` bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a payload checksum error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

The following table describes the functions supported by the receive COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

NOTE: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 28-80: Transmit Checksum Offload Engine Functions for Different Packet Types

Packet Type	Hardware IP Header Checksum Checking	Hardware TCP/UDP Checksum Checking
Non-IPv4 or IPv6 packet	No	No
IPv4packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC)	Yes	Yes
IPv6packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC)	Not Applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes

Table 28-80: Transmit Checksum Offload Engine Functions for Different Packet Types (Continued)

Packet Type	Hardware IP Header Checksum Checking	Hardware TCP/UDP Checksum Checking
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 mainheader) • Hop-by-hop options (in IPv6 extensionheader) • Destinations options • Routing (with segment left 0) • Routing (with segment left >0) • TCP, UDP, or ICMP • Any other next header field in main or extension headers	 Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable	 Yes No Yes Yes No Yes No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	Yes (IPv4 tunnel header) Yes (IPv4 tunnel header)	No No
IPv6 Tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	Not applicable Not applicable	No No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Header-Payload Split

The MAC can identify the boundary between header and payload of the received packet and store them into separate buffers. This feature is useful for multiple reasons:

- Header buffers can be lin faster memory/cache while payload buffers are in slower DRAM memory. This enables the software stack to process the headers faster.

- Buffers used for payload can be directly forwarded to the application layer without the need to copy the payload into application buffer.
- When the Large Receive Offload (LRO) function is implemented in the software/driver layer, it is easy to link the buffers of multiple and contiguous payload data and form a bigger packet with a new header. This reduces the number of packets forwarded to the upper layer and thus improves system software performance.

The EMAC supports multiple methods and packet types for header-payload splitting. This is controlled by the setting the `EMAC_SPLM_OFST_CFG.SPLM` bit field and described in the following table.

Table 28-81: Split Header Support Depending on the Packet Type

Packet Type	SPLM	Description
TCP or UDP packet	00 (L3/L4 split)	The DMA writes the Ethernet header + IP header + TCP or UDP header into the header buffer.
IP packet (not TCP/ UDP)		The DMA writes the Ethernet header + IP header into the header buffer.
Non-IP packet		The DMA does not split the header and payload
Any packet	01 (L2 split)	The DMA writes the Ethernet header • For AV type packets using AV type Split Offset (SAVO) when split AV Enable (SAVE) field is set to 1. • For non-AV type packets using SPLOFST
IP packet	10 (combination of L2 or L3/L4 split)	L3/L4 split
Non-IP packet		L2 split
N/A	11	Reserved

NOTE:
- L3/L4 split is applicable for IP packets that are either untagged or VLAN stripped. If VLAN tag is retained in the packet forwarded to the DMA, L3-L4 split is not performed. When SPLM field is set to 2, L2 Split is performed for IP packets which are VLAN tagged.
- For AV packets, program the `EMAC_SPLM_OFST_CFG.SAVE` and `EMAC_SPLM_OFST_CFG.SAVO` bit fields to specify an alternative L2 split value.

The IP header includes IPv4 options in case of a IPv4 packet, and IPv6 extension headers in case of IPv6 frames. The points at which the L3/L4 header is split are shown in the following figure.

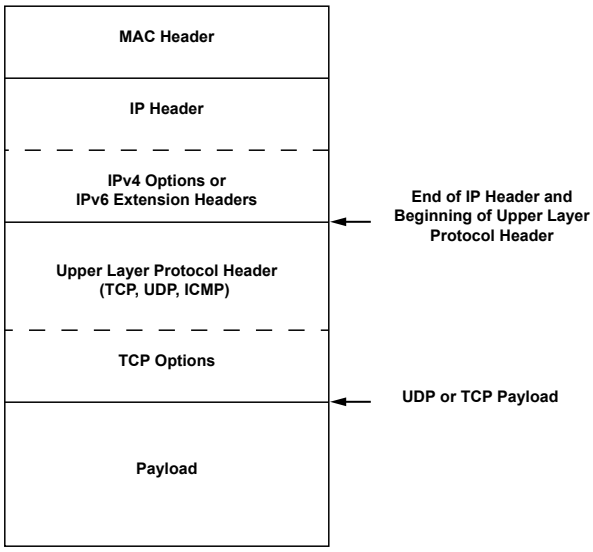


Figure 28-38: L3/L4 Header Split Points

The Figure 28-39 L2 Header Split figure shows the L2 split boundary. In this mode, the header payload boundary is determined for received packets at a fixed length offset starting from the length/type field, as indicated by the `EMAC_SPLM_OFST_CFG.SPLOFST` bit field. This mode is applicable for all received packets when `EMAC_SPLM_OFST_CFG.SPLM` = 01 and only for non-IP packets when SPLM = 10.

The following table shows the split offset for various packet types, when SPLOFST field is set to 2.

Table 28-82: Split Offset Calculation Example: SPLOFST = 2

Packet Type	Split Offset (From the Beginning of LT Field)
Untagged packet	DA + SA + LT = 14
Single VLAN tagged packet with VTAG stripped	
Double VLAN tagged packet with both VTAGs stripped	
Double VLAN tagged packet with both VTAGs stripped	DA + SA + VTAG + LT = 18
Double VLAN tagged packet with only one VTAG stripped	
Double VLAN tagged packet with no stripping	DA + SA +2 x VTAG + LT = 22

The split offset reference is always from the beginning of the LT field, whether the VLAN tag is present or stripped. For more details about the descriptor structure for the split header feature, see

Descriptor Structure for Split Header Support

.

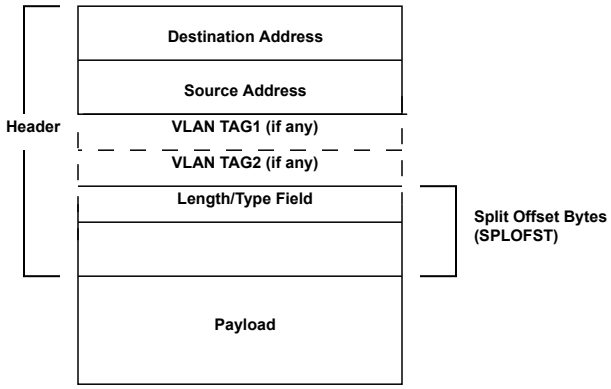


Figure 28-39: L2 Header Split

For programming details see the Programming Header Payload Split Receive section.

TCP/IP Segmentation Offload (TSO) Engine

The TCP Segmentation Offload (TSO) engine is useful in offloading the TCP segmentation functions to the hardware.

It also supports UDP Segmentation Offload (USO) in which the UDP payload is segmented in the hardware. There are no sequencing/ordering controls available or updated in the segments, so it can be used in point to point applications in which out of order packets are not expected by the receiver. The description and flow of TSO mentioned in this section is same as USO, any deviation is provided as notes.

In the TCP segmentation offload (TSO) feature, the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in the TCP Segmentation Offload Overview figure.

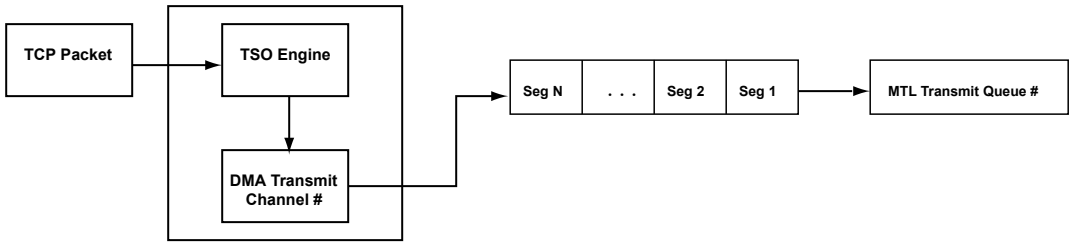


Figure 28-40: TCP Segmentation Offload Overview

DMA Operation with TSO Feature

The TCP Segmentation Offload Flow figure shows the TSO flow.

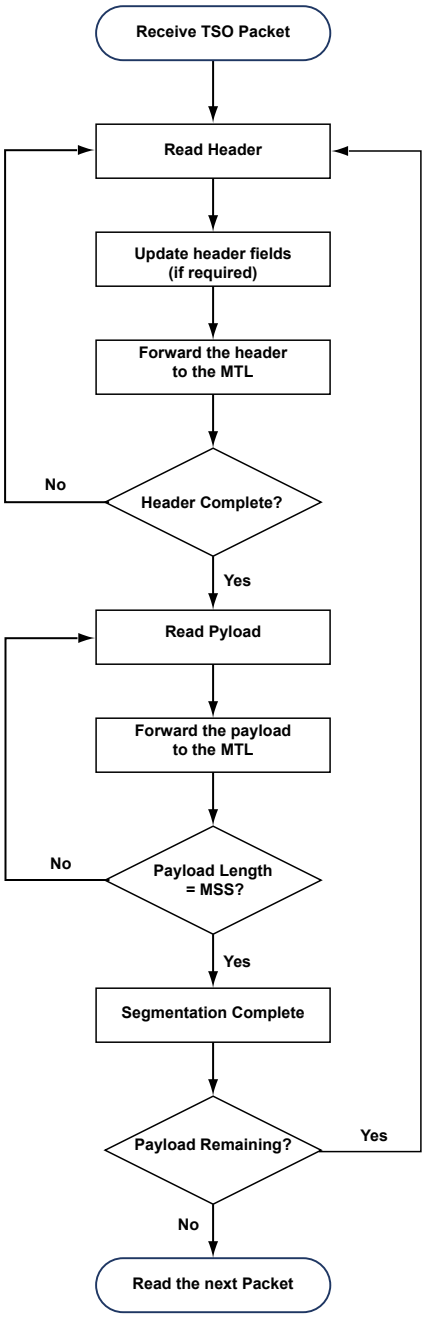


Figure 28-41: TCP Segmentation Offload Flow

For the TSO feature, the transmit DMA operation is as follows:

1. The application sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA transmit channel.
3. While in the Run state, the DMA fetches the next available descriptor and does one of the following:

a. If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.

b. If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a context descriptor error (TDES3[23]) and fetches the next descriptor.

c. If the descriptor is a normal descriptor, the DMA checks for the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, the DMA fetches the header from the local TSO memory. Otherwise, it fetches the header buffer in system memory again, as done for the first segment. In such cases (header not available is TSO memory), the DMA does not close the first descriptor containing the header buffer, until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL TxQ.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory and directly pushes it into the MTL TxQ. In case the buffer(s) in the descriptor do not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the TxQ.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory) and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no descriptor is available, the DMA enters the suspend state.

NOTE:
- The TSO memory size on the processor is 256 bytes per channel.
- The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first normal transmit descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

The DMA repeats all these steps if more descriptors are available. When no descriptor is available, the DMA enters the suspend state.

TCP/IP Header Fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. The following table describes how the TCP and IP headers are updated.

Table 28-83: TSO: TCP and IP Header Fields

Packet Sequence	TCP Header	IP Header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. The TCP checksum is calculated again. 3. If set, the FIN and PSH flags are cleared.	IPv4 Header: • Total Length = MSS + TCP Header Length + IP Header Length • Identification field is not modified. It is sent as per the header provided by the software • IPv4 Header Checksum is recalculated. IPv6 Header: Payload Length = MSS + TCP Header Length + IPExtension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header: • Total Length = MSS + TCP Header Length + IP Header Length • Identification field = Previous Identification Field + 1 • IPv4 Header Checksum is recalculated. IPv6 Header: Payload Length = MSS + TCP Header Length + IPExtension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header: • Total Length = Remaining Payload + TCP Header Length + IP Header Length • Identification Field = Previous Identification Field + 1 • IPv4 Header Checksum is recalculated. IPv6 Header: Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

NOTE: In case of USO, the engine updates only the following fields in the IP-UDP headers:

- Total length, identification field, and header checksum field in IPv4 header
- Payload length field in IPv6 header
- Length and checksum field in UDP header

Header and Payload Fields of Segmented Packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in the Table 28-83 TSO: TCP and IP Header Fields table. The *Header and Payload Fields of Segmented Packets* figure shows how the same header is used for the header fields of segmented packets.

The application must create the header in buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

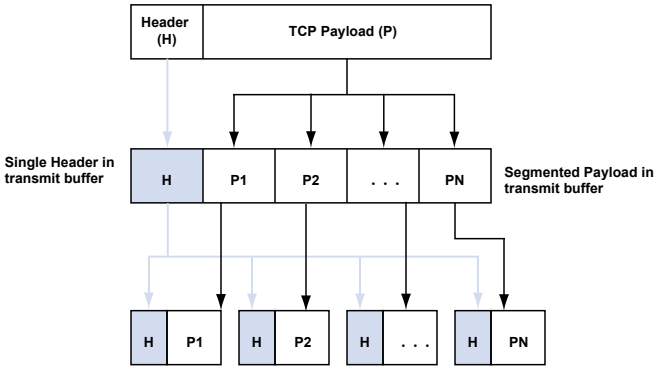


Figure 28-42: Header and Payload Fields of Segmented Packets

Context Descriptor Sequence

Programs can use the context descriptor to provide the MSS value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application wants to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset.

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this MSS value for all subsequent packets that have the TSO Enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor
2. The DMA indicates the error in descriptor status
3. The DMA generates an interrupt if the `EMAC_DMA[n]_IEN.CDEE` bit is set corresponding to a DMA channel

The application can read the interrupt status through the CDE bit of the status register corresponding to a DMA channel.

Building the Descriptor and the Packet for the TSO Feature

Set the TSE bit of TDES3 of the first normal descriptor. If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 KB.

NOTE: The TDES3[22:19] field is used as slot number in case of AV configurations. If both TSO and AV are enabled for the same DMA transmit channel (the `EMAC_DMA[n]_TXCTL.TSE` bit is set), then slot checking for that DMA transmit channel must not be enabled (do not set the `EMAC_DMA[n]_SFN_CTLSTAT.ESC` bit).

The header of the packet including the Ethernet header, L3 header and L4 header should be provided in buffer 1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer 1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to and be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the maximum segment size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the `EMAC_DMA[n]_CTL.MSS` value or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes, otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in the `EMAC_DMA[n]_TXCTL` register should be lesser than the Tx Queue size programmed in the `EMAC_TQ0_OPMODE.TQS` bit field. It is recommended that MSS + header is equal to half the programmed transmit queue size.

The DMA also supports segmentation of VLAN tagged TCP/IP frames, so if the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the double VLAN feature is selected, then the DMA passes both the tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits for both tags is applicable for all segments.

If the double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 2'b00 (MAC does CRC and padding) and checksum insertion control bits are forced to 2'b11 (hardware does the checksum calculation for both header and payload).

See the Programming TSO section for more details on how to program the TSO engine.

UDP/IPv4 Fragment Offload (UFO) Engine

The EMAC supports breaking a large UDP packet into smaller and multiple packets for transmission, using IPv4 Fragmentation. Each IPv4 fragment contains the following header or payload information:

- Ethernet Header
- IP Header
- UDP Header (only in the first fragment)
- UDP Payload

NOTE:
- Fragmentation is supported only for UDP over IP. Fragmentation is not supported for TCP packets.
- Fragmentation of UDP is supported only over IPv4. This support is not available over IPv6.

The UDP Fragment Offload (UFO) Engine is an add-on function of the TSO engine. Therefore, both UFO and TSO functions can be enabled on the same transmit DMA channel. In this mode, the UDP packets are fragmented and TCP packets are segmented. If the UFO function is not enabled, then both TCP packets and UDP packets are segmented into smaller packets as described in TCP/IP Segmentation Offload (TSO) Engine.

The UFO breaks down a large UDP packet into smaller packets using IPv4 fragmentation. In this mode, the MSS field in the context Descriptor (TDES2[17:0]) and the `EMAC_DMA[n]_CTL.MSS` bit are interpreted as Maximum Fragmentation Size (MFS). This is the maximum size of the payload after the IPv4 header in each fragment. When UFO is enabled, the MSS field must be programmed in multiples of 8 bytes because the IP fragments offset is defined in terms of 8 bytes.

The UFO functions in two modes:

- UFO with valid UDP checksum
- UFO with no UDP checksum

UFO with Valid UDP Checksum

In this mode, the UFO engine calculates the UDP payload checksum and inserts it in the respective UDP header field. However, as the checksum must be calculated over the complete payload ending in the last fragment and then inserted in the UDP header encapsulated in the first IPv4 fragment, the fragments are not transmitted in order. The first IPv4 fragment containing the UDP header is sent last. The *Fragmentation Flow With Checksum* figure shows this sequence.

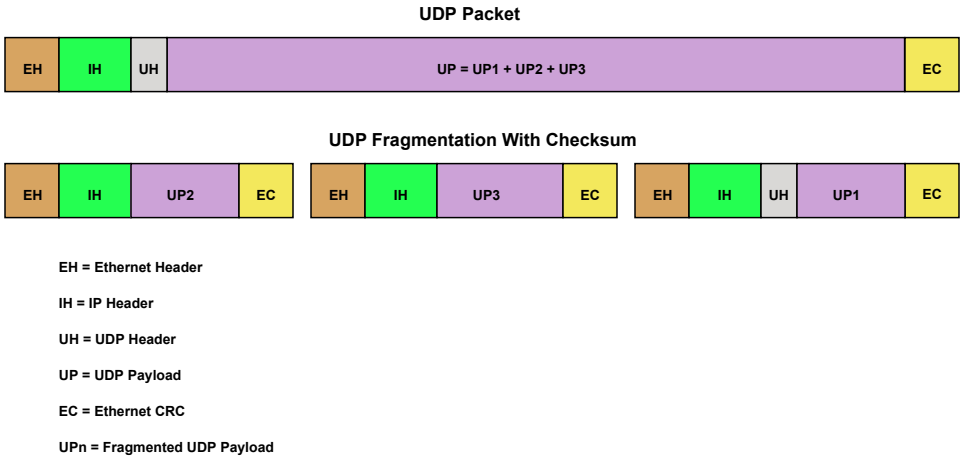


Figure 28-43: Fragmentation Flow With Checksum

The following table shows the details of the IP/UDP Header updates performed in each fragment by the UFO Engine in this mode.

Table 28-84: Details of IPv4 Fragmentation with UDP Checksum

Packet Sequence	UDP Header	IPv4 Header	Comments
First fragment (This segment is sent last)	Calculate the UDP header checksum	1. Total length = 32 bytes + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = 0 5. Update the IPv4 header checksum	The header (and 32 bytes IPv4 fragment payload) data is stored in TSO memory in this step.
Second fragment	NA (No UDP header)	1. Total length = MFS + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = previous fragment offset + (32/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.

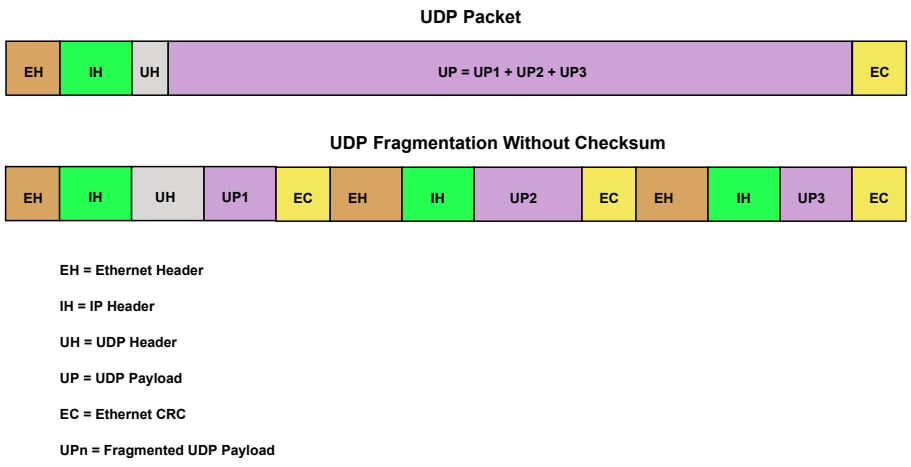
Table 28-84: Details of IPv4 Fragmentation with UDP Checksum (Continued)

Packet Sequence	UDP Header	IPv4 Header	Comments
Intermediate (non-first and non-last) fragment	NA (No UDP header)	1. Total length = MFS + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = previous fragment offset + (MFS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.
Last fragment	NA (No UDP header)	1. Total length = remaining payload + IPv4 header Length 2. DF = 0 (do not Fragment) 3. MF = 0 4. Fragment offset = previous fragment offset + (MFS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.

UFO with no UDP Checksum

In this mode, the UDP payload checksum is not calculated and an all-zero value is inserted in the corresponding checksum field of the UDP header and all the IP fragment packets are transmitted in sequence. the *Fragmentation Flow Without UDP Checksum* figure shows the sequence of fragmented packets created and transmitted by the UFO engine.

Figure 28-44: Fragmentation Flow Without UDP Checksum



The following table shows the details of IPv4/UDP header updates performed in each fragment of the UFO engine, in this mode.

Table 28-85: Details of IPv4 Fragmentation without UDP Checksum

Fragment Sequence	UDP Header	IPv4 Header	Comments
First fragment	Replace the UDP checksum field with all 0s	1. Total length = MFS (MSS field) + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = 0 5. Update the IPv4 header checksum	The header data is stored in the TSO memory in this step.
Intermediate (non-first and non-last) fragment	NA (No UDP header)	1. Total length = MFS (MSS field) + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = previous fragment offset + (MSS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.
Last fragment	NA (No UDP header)	1. Total length = remaining payload + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 0 4. New fragment offset = previous fragment offset + (MSS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.

Segmentation Versus Fragmentation

The following table shows the differences between UDP packet fragmentation performed by the UFO and UDP segmentation performed by the TSO engine.

Table 28-86: Segmentation Versus Fragmentation

Segmentation	Fragmentation
The UDP header is replicated for each segment. The length field in the UDP header is updated for each segment	The UDP header is not updated; however, checksum field in the UDP header is updated.
MSS field indicates the size of the maximum segment after the L4(TCP/UDP) header	MFS field indicates the size of the fragment after the L3 (IPv4) header
The TCP packet is broken down into multiple chunks by keeping L2+L3+L4 header	The UDP packet is broken down into multiple chunks by keeping the L2 + L3 header

IPv4 ARP Offload Engine

The EMAC supports the Address Recognition Protocol (ARP) offload for IPv4 packets. This feature allows the processing of the IPv4 ARP request packet in the receive path and generating corresponding ARP response packet in

the transmit path. EMAC generates the ARP reply packets for appropriate ARP request packets. The ARP packet for IPv4 is a L2 layer packet with length/type of 0x0806.

The ARP offloading process is as follows:

1. The MAC receiver gets an ARP request if the target protocol address of request matches the IPv4 address programmed in the EMAC_ARP_ADDR register of the MAC.
2. The MAC generates an ARP reply packet.
3. The MAC copies the sender hardware address field in the ARP request to the following fields:
   - DA field of the Ethernet packet header
   - Target hardware address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the target protocol address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   - SA field of the Ethernet packet header
   - Sender Hardware Address field of the ARP reply packet
6. The MAC copies the target protocol address field in the ARP request to the sender protocol address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for generated ARP reply packet.
9. The MAC transmitter sends the ARP reply.

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to application with ARP reply not generated (bit 34) status bit set. However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request.

If the disable CRC check bit of the MAC extension configuration bit is set, then the MAC does not check for valid CRC of a ARP request packet. It can generate an ARP response packet if the other conditions are valid.

NOTE: When the received ARP request is less than 64 bytes packet length, EMAC does not send a ARP response. It is treated as normal packet and forwarded to the application based on the EMAC filter settings.

Energy Efficient Ethernet (EEE) Mode

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps, 1000 Mbps, and 10 Gbps. The EMAC supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functions to save power during the periods of low-link use. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit Path Functions

The transmit path functions include tasks that the MAC must perform to make the PHY to enter the LPI state.

In the transmit path, the software must set the EMAC_LPI_CTLSTAT.LPIEN bit to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the EMAC_LPI_TMRSCTL.LST bit field. The PHY link status (EMAC_LPI_CTLSTAT.PLS bit) indicates the link status of the PHY.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. Deasserts TXEN
2. Asserts TX_ER
3. Sets EMAC_TXD[n][3:0] to 0x1 (for 100 Mbps) or TXD[7:0] to 0x01 (for 1,000 Mbps)
4. Updates the status (EMAC_LPI_CTLSTAT.TLPIEN bit) and generates an interrupt.

To bring the PHY out of the LPI state, that is, when the software resets the TLPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern
2. Starts the LPI TW TIMER. The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the EMAC_LPI_TMRSCTL.TWT bit field.
3. Updates the LPI exit status (EMAC_LPI_CTLSTAT.TLPIEX bit) and generates an interrupt

The *LPI Transitions (Transmit)* figure shows the behavior of the TXEN, TX_ER, and EMAC_TXD[n] [3:0] signals during the LPI mode transitions.

NOTE:
- Gating of EMAC_TXCLK is not supported on EMAC.
- If the MAC is in the LPI mode and the application issues a soft or hard reset, the MAC transmitter comes out of the LPI mode.

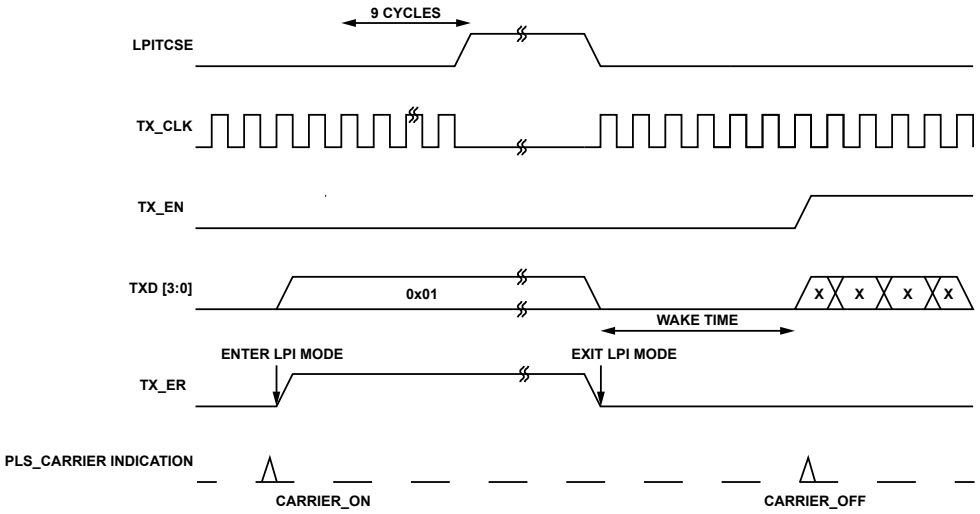


Figure 28-45: LPI Transitions (Transmit)

Automated Entry/Exit of LPI Mode

The MAC transmitter can be programmed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period or has a packet to transfer. These modes are enabled and controlled by the `EMAC_LPI_CTLSTAT` register.

When the `EMAC_LPI_CTLSTAT.LPITXA` and `EMAC_LPI_CTLSTAT.LPIEN` are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI IDLE state and clear the LPIEN bit as soon as any of functions in the transmit path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when (`EMAC_LPI_CTLSTAT.LPIATE`) is also set, the MAC transmitter enters LPI IDLE state only when the Transmit path remains in idle state (no activity) for the period indicated by the value in the `EMAC_LPI_ENTR_TMR` register. In this mode also, the MAC transmitter exits the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both the `EMAC_LPI_CTLSTAT.LPIATE` and `EMAC_LPI_CTLSTAT.LPITXA` bits are cleared, the program can directly control the entry and exit of LPI IDLE state by programming the LPIEN bit.

Receive Path Functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts `EMAC_RXERR`
2. The PHY sets `EMAC_RXD[n]`[7:0] to 0x01
3. The PHY deasserts RX_DV

4. The MAC updates the EMAC_LPI_CTLSTAT.RLPIEN bit and immediately generates an interrupt

NOTE:
- The PHY maintains the same state of the EMAC_RXERR, EMAC_RXD[n], and RX_DV signals for the entire duration during which it remains in the LPI state.
- If the LPI pattern is detected for a very short duration (that is, less than two cycles of receive clock), the MAC does not enter the receive LPI mode.
- If the duration between end of the current receive LPI pattern and start of the next receive LPI pattern, is very short (that is, less than two cycles of receive clock), then the MAC exits and again enters the receive LPI mode. The MAC does not give the receive LPI exit and entry interrupts

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY deasserts EMAC_RXERR and returns to a normal inter-packet state.
2. The MAC updates the EMAC_LPI_CTLSTAT.RLPIEX bit and generates an interrupt immediately.

The following figure shows the behavior of the EMAC_RXERR, RX_DV, and EMAC_RXD[n][3:0] signals during the LPI mode transitions.

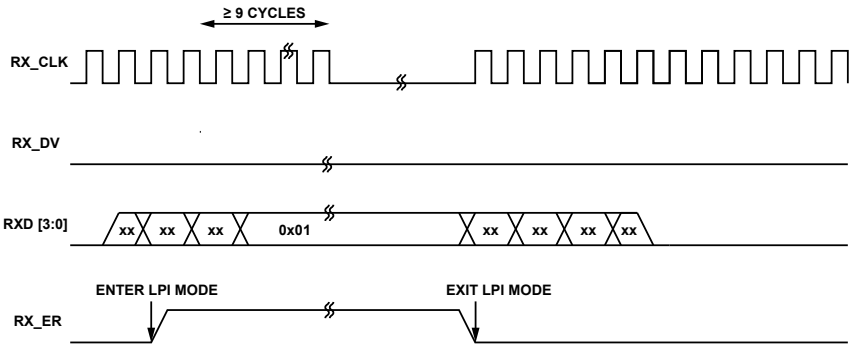


Figure 28-46: LPI Transitions (Receive)

NOTE:
- If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in the *LPI Transitions (Receive)* figure.
- If the MAC is in the LPI mode and the application issues a soft or hard reset, the MAC receiver comes out of the LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver again enters the LPI state.
- If the receive clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the receive clock domain.
- When the PHY sends the LPI pattern and if the EMAC_MAC_CFG.RE bit is set, the MAC automatically enters the LPI state. There is no other software control to prevent the MAC from entering the LPI state.

LPI Timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers. The following sections describe the LPI timers that are loaded with the respective values from the EMAC_LPI_TMRSCTL and EMAC_LPI_ENTR_TMR registers:

LPI LS Timer

The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. Programs can enable the monitoring of the EMAC_PHYIF_CTLSTAT.LUD bit by setting the EMAC_LPI_CTLSTAT.PLSEN bit.

The link status is indicated to the MAC by the EMAC_PHYIF_CTLSTAT.LNKSTS bit or the value programmed by the software in the EMAC_LPI_CTLSTAT.PLS bit. If the link status is not available in the EMAC_PHYIF_CTLSTAT, the software can get the PHY link status by reading the PHY register and accordingly update the PLS bit.

This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the EMAC_LPI_TMRSCTL.LST bits. The RGMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1023 milliseconds.

LPI TW Timer

The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count is programmed using the EMAC_LPI_TMRSCTL.TWT bit field. The terminal count of the timer is the value of resolved transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count. The MAC supports the LPI TW TIMER in units of microseconds. The LPI TW TIMER is 16 bits wide. Therefore, the software can program up to 65535 µs.

LPI Auto Entry Timer

This timer counts in steps of eight microseconds, the time for which the MAC transmit path must remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when the EMAC_LPI_CTLSTAT.LPIATE bit is set.

NOTE: Program the EMAC_LPI_CTLSTAT.PLS bit to 1'b0 before switching between the GMII and MII modes. This resets the internal timers. If the mode is changed after the LPI LS TIMER or LPI TW TIMER starts, the change in the transmit clock frequency can result in incorrect timeout

Low Power Interrupt (LPI)

The MAC generates the LPI interrupt when the transmit or receive side enters or exits the LPI state. The LPI interrupt can be cleared by reading the EMAC_LPI_CTLSTAT register.

For detailed programming guidelines see Programming Energy Efficient Ethernet.

Time-Sensitive Networking

The Time-Sensitive network (TSN) is a set of standards developed by the IEEE 802.1 work group. The standards define mechanisms for the time-sensitive transmission of data over the Ethernet.

This section contains the following topics:

- Enhancements to Scheduled Traffic (EST)
- Frame Preemption (FPE)
- Time-Based Scheduling

Enhancements to Scheduled Traffic (EST)

The IEEE 802.1Qbv-2015 specification defines the schedule for each of the queues on every egress port which makes the implementation aware of traffic arrival schedule. This information can be used to block the lower priority traffic from transmission in this time window/slot. This ensures that scheduled traffic is forwarded from sender to receiver through all the network nodes with a deterministic delay. The ***Time Aware Shaper Implementation*** figure shows the scheduled traffic windows for periodic priority data.

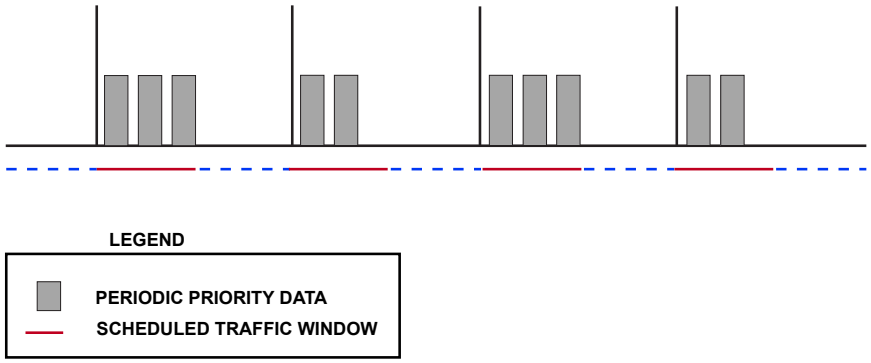


Figure 28-47: Time Aware Shaper Implementation

To achieve a low latency, ensure there are no interfering frames during the scheduled windows that are reserved for high priority traffic. The use of scheduled traffic imposes limitations while starting a transmission.

Refer to the ***Implementing a Guard Band*** figure below. When an interfering frame begins transmission just before the start of a reserved period, transmission can be extended into the reserved window, and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

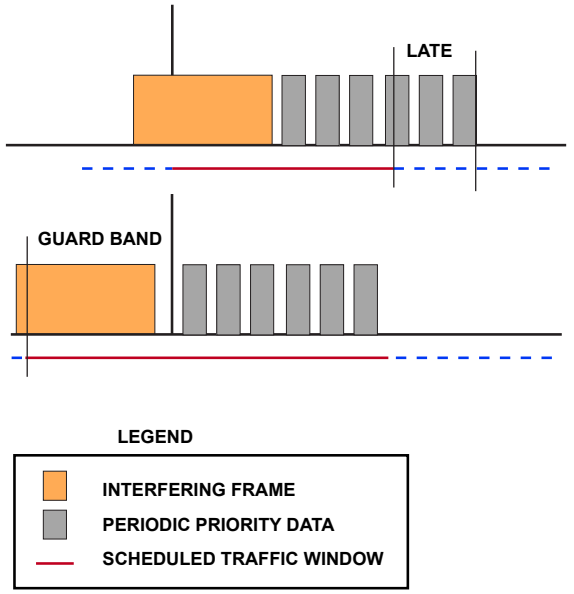


Figure 28-48: Implementing a Guard Band

A larger guard band equates to a less efficient use of the network bandwidth. However, the implementation of IEEE802.1Qbu (frame preemption) addresses this inefficiency. Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band must only be as large as the largest possible interfering fragment (instead of the largest possible interfering frame).

During the guard band, only the frames that can complete the transmission of the entire frame before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the specified window .

Definitions

The following terms are frequently used in the EST support:

- *Gate Control List* - The list in the hardware memory that holds the gate controls and the associated time intervals.
- *Gate Controls* – For a given schedule (row in gate control list), there is a gate open (O) and a gate close (C) state associated with each traffic control (TC). Gate controls are the set of O or C values, whose width is same as the configured TCs. For example, CCOCOOCO means TC7 = C, TC6 = C, TC5 = O and so forth.
- *Time Interval* – Time interval (in nano seconds). It is a 16-, 20-, or 24-bit configured field in the gate control list that indicates the time for which the associated gate controls are valid.
- *Base Time Register* – Each gate control list is associated with a 64-bit base time register that holds the start time (in PTP format) for the list.

Transmit Scheduling Updates

To support EST, the following updates are required for transmit scheduling:

- Implementation of the Gate Control List (GCL)
- Enforcing gate controls in the scheduler
- Accounting for gate open duty cycle in the computation of idleSlope (CBS)

Gate Control List (GCL) Implementation

The ***Gate Close and Open Events*** figure shows a block diagram based on the IEEE 802.1Qbv specification. It illustrates how the gate control list governs the gate close (C) and open (O) events based on the schedule provided for each event. GCL includes a time interval and gate control.

Time Interval – Defines the time in nano seconds for which the gate controls are valid. It should be applied before reading the next gate control from the list. It has a configurable width of 16, 20 or 24 to represent a maximum of 64 us, 1 ms, or 16 ms schedule intervals, respectively. The GLC implementation supports a left shift of a time interval up to 7 bits to apply a multiplication factor from 1 to 128 ns (in steps of 2^n). The maximum value (post shifting) of this field must be 999,999,999 ns.

Gate Control – Defines the open event (represented by a logic 1) or the close event (represented by logic 0) state for the gate of each TC.

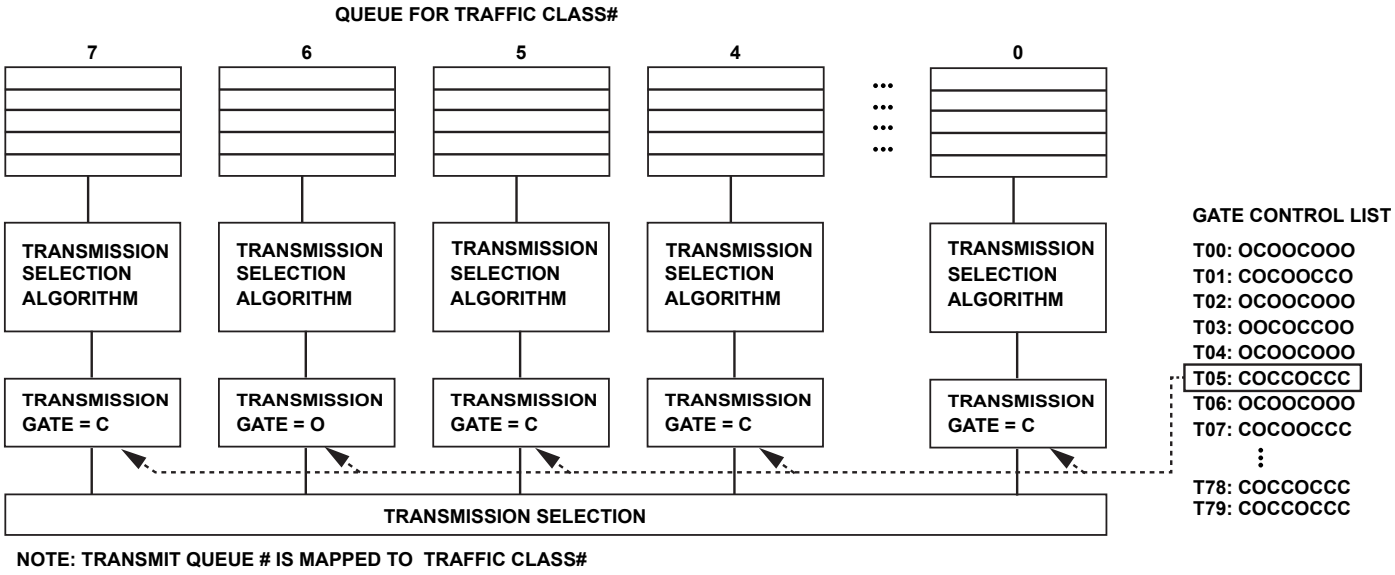


Figure 28-49: Gate Close and Open Events

The implementation of GCL consists of the following gate control lists:

- Hardware Owned List (HWOL) - a list for hardware access
- Software Owned List (SWOL) - a list for software access

The access to these lists is mutually exclusive. Hardware configures the ownership to the list in the `EMAC_EST_STAT` register. The ***External Memory Used for Holding the GCL*** table provides the implementation details of GCL.

Table 28-87: External Memory Used for Holding the GCL

Gate Control (up to 8 Bits)	Time Interval (ns) 16, 20 or 24 Bits	
OOCCCCCC	T0	HWOL or SWOL
OOOOCCCC	T1	
CCCCOOCC	T2	
CCCCCCOO	T3	
…	…	
…	…	
OCOCOCOO		
OOOOCCCC	T last	
OOOOCCCC	T0	SWOL or HWOL
OOOOCCCC	T1	
OOCCCCCC	T2	
OOOOCCCC	T3	
…	…	
…	…	
OCOCCCCC		
OOCCCCCC	T last	

GCL uses the following registers (one for each GCL). These registers are implemented through indirect addressing using the `EMAC_EST_GCL_CTL` and `EMAC_EST_GCL_DAT` registers.

Base Time Register (BTR) – A 64-bit register that holds the start time to execute the GCL. The format of the BTR is the same as the PTP format (the upper 32-bits holds time in seconds and the lower 32 bits hold time in nano seconds). Once the execution of a given list begins, the implementation can update the value in the BTR to indicate the next list execution time.

Cycle Time Register (CTR) – A 40-bit register that holds the time at which the execution of the GCL should be repeated. The cycle time register consists of an 8-bit value in seconds, and a 32-bit value in nano seconds (similar to the PTP time format with a truncated seconds register).

For a given gate control list:

Start time = Base Time + N * Cycle Time

where N is an integer value representing the iteration number starting with 0 for first iteration

If the gate control list execution takes longer than the cycle time, the list is truncated at the cycle time and the subsequent loop begins at cycle time.

Time Extension Register (TER) – An m-bit (where m = configured time interval width + 7) register that holds the amount of time (in nano seconds) that the current gate control list can be extended before switching to the new gate control list. This is useful to avoid small fragments of the current list before switching to a new list.

List Length Register (LLR) – An n-bit gate control register (when n is 7, 8, 9, 10, or 11 for a GCL configured depth of 64, 128, 256, 512, or 1024, respectively) that holds the integer value of the length of the GCL (the number of valid rows in each GCL). The processing of the GCL stops after the number of rows read equals the LLR value.

Transmission Gating Implementation

A bridge or an end station can be enhanced to allow transmission from each TC that is yet to be scheduled relative to a known timescale. To achieve this, a transmission gate is associated with each TC; the state of the transmission gate determines whether the queued frames can be selected for transmission.

For a TC, the transmission gate can be in one of the following states:

1. Open-queued frames are selected for transmission in accordance with the definition of the transmission selection algorithm associated with the TC
2. Closed-queued frames are not selected for transmission

A frame on a traffic class queue is not available for transmission when the transmission gate is in the closed state or when there is not enough time to transmit the entire frame before the next gate-close event associated with the queue.

The implementation has visibility into the current schedule of gate controls and the immediate next schedule of the gate controls. So, the maximum gate open period does not exceed the sum of the two time intervals. A frame is selected for transmission only when the gate is currently open and the duration of gate open interval is greater than the time taken to transfer the entire frame.

The implementation must have the frame size before the transmission to avoid transmission overruns and schedule only frames that can complete. To implement transmission gating:

- Program the `EMAC_EST_EXT_CTL.OVHD` bit field with a fixed overhead of the IPG or EIPG, preamble, and scheduler delay
- Enable the automatic prediction of variable overhead due to the offloads for a packet in the MAC transmitter. Consider the following while programming the EST-related fields for correct scheduling of packets:
    - Speeds: 1Gpbs/100 Mbps
    - PTOV in ns: 6 PTP clock cycles
    - CTOV in ns: 3 PTP + 3 transmit clock cycles
    - Per packet slot interval in ns: (packet size + overhead + IPG or EIPG + preamble) * (time to transmit 1 byte)
    - Overhead, scheduler delay in bytes is calculated as:

      17 Tx + (6 + X + Y + Z) CLKO7 clock cycles

where

- If Tx COE or TSO is selected, X = 4

  If TBS or OST is selected, Y = 4

  If VLAN tag insertion is enabled, Z = 2

The implementation compensates for the data transfer delays from the buffer to the line by offsetting the current time with all the relevant delays (provided by the EMAC_EST_CTL.CTOV field). This configuration ensures that the schedule provided is always accurately implemented at the line.

NOTE:
- An overhead is added to the packet size to account for the scheduler delay, IPG or EIPG, and preamble overheads on the line. Therefore, for a 128 byte frame to be transmitted, the gate open window must be able to accommodate at least 128 overhead bytes.
- In 100 Mbps mode of operation, the rounding down error is about 0.4%. For example, the gate open duration must be at least 1035 bytes for transmitting 1000 byte frames at a speed of 100 Mbps (31 bytes for scheduler delay, line overheads for two-port SRAM configurations, and 4 bytes for rounding down error)
- The minimum time interval for GCL entry must be equal to the time required to transmit fragment size + scheduler delay + IPG/EIPG + preamble overheads

Idle Slope Computation

When EST is enabled, credit is accumulated only when the gate is open; therefore, the effective data rate of the idle slope must be increased to reflect the duty cycle for the transmission gate associated with the queue.

The idle slope is computed based on GateOpenTime and OperCycleTime values. Program the idleSlope registers (implemented one per CBS-enabled TC) based on the following equation. The existing MTL register has sufficient bit width to accommodate the new values for idleSlope.

idleSlope = (operIdleSlope(N) * OperCycle/GateOpenTime)

Operational Details of GCL

Set (=1) the switch to the software list using the EMAC_EST_CTL.SSWL bit. Hardware can subsequently access the programmed gate control list. The first configuration of gate controls are applied when the current time is equal to the value in the BTR. It is held for the programmed time interval value.

To avoid transmission overruns, one additional gate control event is read ahead from the list. This operation permits the GCL to determine the next gate close events (if any) for the TCs that are open.

The scheduling is based on the gate open state and time interval of only the current and subsequent schedule. An internal accumulator is used to add the time intervals when gate controls are applied. The BTR and accumulator hold the time at which the next set of gate controls will be applied.

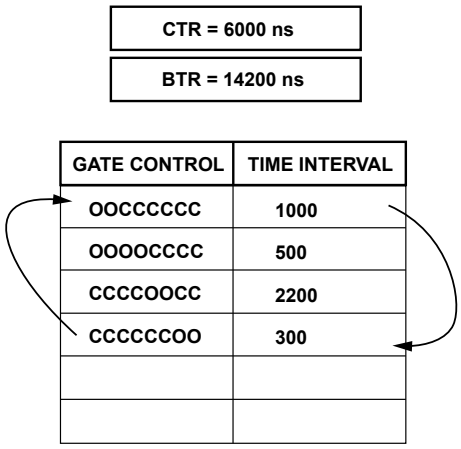


Figure 28-50: GCL Example - Base Time and Cycle Time

Refer to the ***GCL Example - Base Time and Cycle Time*** figure. The GCL is read progressively from the first row adhering to the schedule. The read operations continue until the list length (from LLR register) is reached and the execution of the list restarts at BTR + CTR time. At this point, the BTR is incremented by value in CTR to mark the beginning of a new cycle. In the absence of any gate controls, all gates are deemed to be in the open state during the execution of the list.

In cases where the execution time of the list is greater than the cycle time, the list is truncated and the next iteration starts when the current time equals BTR + CTR.

Table 28-88: GCL Execution Example - BTR and CTR

Current Time	Gate Control Applied	Accumulator Value (ns)	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17200	CCCCCCOO	4000	14200
18200	OOOOOOOO	0	20200
20200	OOCCCCCC	1000	20200
21200	OOOOCCCC	1500	20200

Refer to the ***GCL Execution Example - BTR and CTR*** table. The execution starts at 14200 and the first set of gate controls OOCCCCCC are applied immediately. When the time interval is 1000 ns, the next set of gate controls are applied at 14200 (BTR) + 1000 (Accumulator) = 15200 ns. Since there are no gate controls available after the execution of the last gate control and before the next iteration of the loop, the gates are deemed to be in open state during that period as depicted at time 18200.

Refer to the ***GCL Example - Base Time and Cycle Time, Execution Time > Cycle Time*** figure. Because the list execution takes longer than the allocated cycle time, the list is truncated.

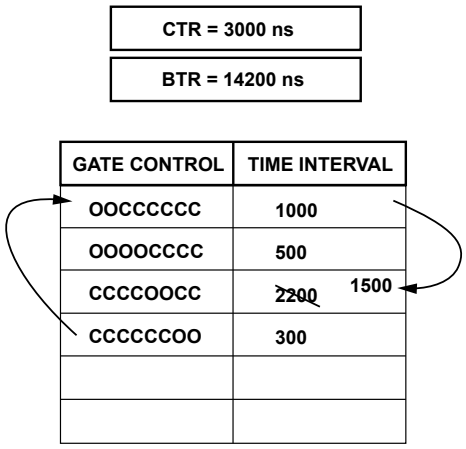


Figure 28-51: GCL Example - Base Time and Cycle Time, Execution Time > Cycle Time

Refer to the ***GCL Execution Example - BTR and CTR , Execution Time > Cycle Time*** table. The list is started from the BTR + CTR. While applying the third set of gate controls, the BTR + Cycle Time (17200) < BTR + Accumulator (17900). Therefore, the list is truncated and execution switches to a new iteration at 17200.

Table 28-89: GCL Execution Example - BTR and CTR, Execution Time > Cycle Time

Current Time	Gate Control Applied	Accumulator Value (ns)	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17200	OOCCCCCC	1000	17200
18200	OOOOCCCC	1500	17200
18700	CCCCOOCC	3700	17200

Installing a New GCL

When a new software programmed GCL is available and must be executed at the new BTR value, the switch to the new GCL can happen as a:

- New base time aligned with current schedule
- New base time unaligned with current schedule

New Base Time Aligned with Current Schedule

If the cycle time for the new gating cycle is the same as the cycle time for the current gating cycle, and if the BTR chosen for the new gating cycle is an integer multiple of the current cycle time (+ current BTR), the new and old gating cycle are aligned. This would be the ideal case; it allows the new gating cycle to be installed and executed with no timing issues. The implementation completes the execution of an iteration of the current list, and switches to the new list at the beginning of the base time listed in the new list.

When (New BaseTime ≥ Current Time), then ConfigChangeTime = New BaseTime, Else If (New BaseTime < Current Time):

1. Set the BTRError.
2. ConfigChangeTime = (New BaseTime + N* New CycleTime).

   where N is the smallest integer for which the relationship ConfigChangeTime ≥ CurrentTime and (N =< 8) = TRUE.

When N > 8, the hardware cannot auto recover and the loop count value in BTR error reporting is set to 1111. The software must reprogram the new base time.

The ***Switching to a New List Example - Aligned*** figure illustrates the installation of a new GCL that is aligned with the existing configuration. In the example, after the sixth iteration of the first GCL, the BTR values of the old and new GCL are equal. At that point, the new GCL is processed as a natural extension to the existing GCL.

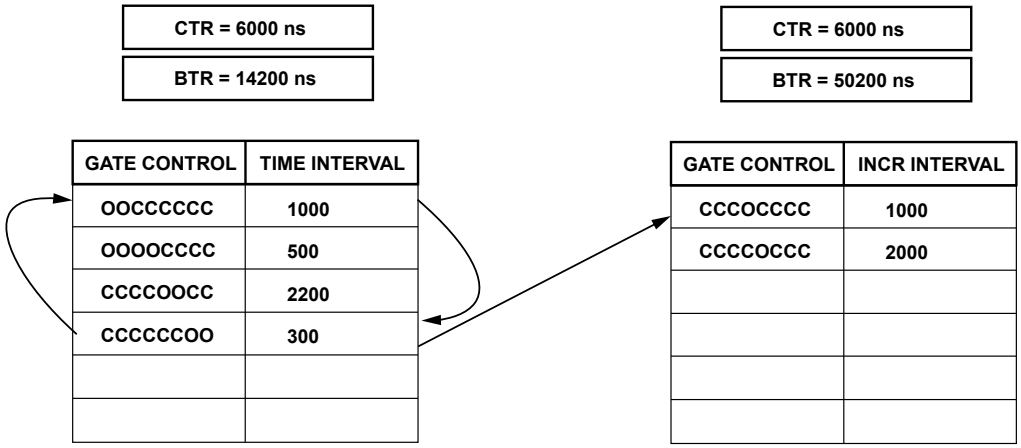


Figure 28-52: Switching to a New List Example - Aligned

Table 28-90: GCL Execution Example - BTR and CTR

Current Time	Gate Control Applied	Accumulator Value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47900	CCCCCCOO	4000	44200
48200	OOOOOOOO	0	50200
50200	CCCOCCCC	1000	50200
51200	CCCCOCCC	3000	50200
53200	OOOOOOOO	0	56200

New Base Time Unaligned with Current Schedule

If a new cycle time differs from current cycle time or the new base time is in the future and not an integer multiple of the current cycle time, the old and new cycles are unaligned. When new base time is reached (when the new configuration is installed and starts to execute), the last old cycle is normally truncated to start the first new cycle. This could be undesirable if it results in a very short last old cycle. Arguably, it would be better to extend the old cycle by that small amount, rather than start a short cycle. The cycle time extension register (related to the current configuration list) allows this extension of the last old cycle. If the last complete old cycle ends normally in less than current Cycle Time Extension (TER) ns before the new base time, then the last complete cycle (before new base time is reached) is extended so that it ends at the new base time.

The ***Switching to a New List Example - Unaligned*** figure shows how the installation of a new GCL that is unaligned with the current schedule truncates the current list.

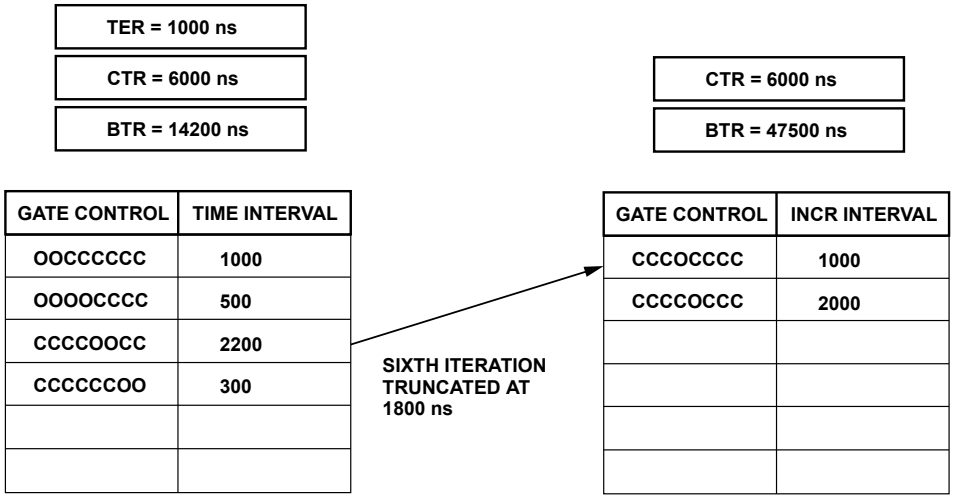


Figure 28-53: Switching to a New List Example - Unaligned

At the end of the fifth iteration the Current time + Cycle time extension (TER) < New BTR. So, the sixth iteration of current configuration is started. During the sixth iteration of the current list when the new BTR value is smaller than the next schedule in the current list, it switches to the new list.

Table 28-91: Extending to New List by Truncating the Current List

Current Time	Gate Control Applied	Accumulator Value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47500	CCCOCCCC	4000	44200
48500	CCCCOCCC	0	50200
50500	OOOOOOOO	1000	50200

The *Switching to a New List Example - Extended* figure shows where the current configuration list is extended (instead of starting a new iteration). The extension time of 800 ns is less than the allowed cycle extension time (TER) of 1000 ns.

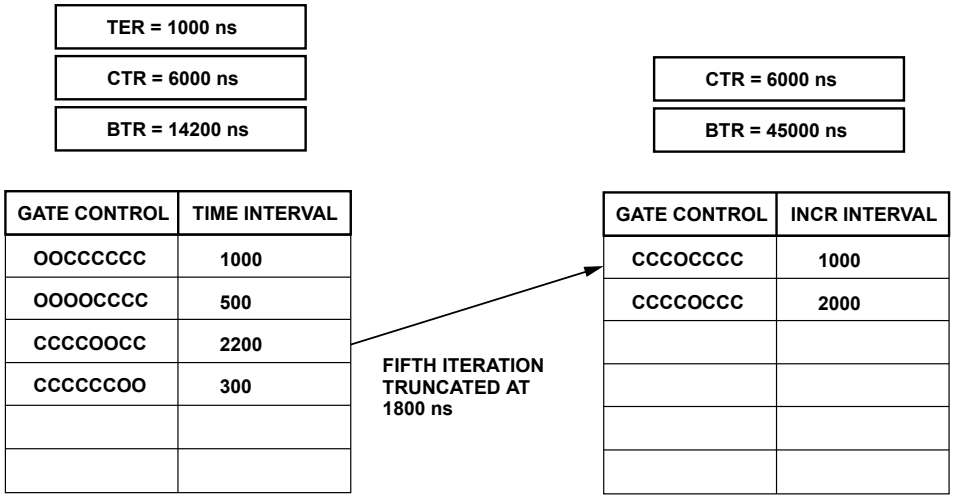


Figure 28-54: Switching to a New List Example - Extended

Table 28-92: Switching to New List by Extending the Current List

Current Time	Gate Control Applied	Accumulator Value	BTR (with updates)
38200	OOCCCCCC	1000	38200
39200	OOOOCCCC	1500	38200
39700	CCCCOOCC	3700	38200
41900	CCCCCCOO	4000	38200
42200	OOOOOOOO	0	44200
45000	CCCOCCCC	1000	45000
46000	CCCCOCCC	3000	45000
48000	OOOOOOOO	0	51000

Impact of Transmit Scheduling Algorithms on EST

When EST is used in isolation, the gate control list manages the final open/close state of the queues along with the checks carried out by the transmission selection algorithm in MTL. Because the gate controls operate on a predefined repetitive schedule, use the strict priority or Credit Based Shaper (CBS) scheduling algorithms.

Other algorithms such as the Weighted Round Robin (WRR), Deficit Weighted Round Robin (DWRR) and Weighted Fair Queuing (WFQ) implement masking of the queues based on the current winning queue. The algorithm is applied only among the group of queues that open simultaneously. To ensure queues whose gates are open get priority, these algorithms are modified to treat gate open queues and gate closed queues as separate groups giving priority to gate open queues.

For example, consider four queues (Q3, Q2, Q1, Q0) with weights 4:3:2:1; Q3 and Q2 are in open state in slot one slot, while Q1 and Q0 are in open state in another slot. In this case, the scheduler works as follows:

1. In the first slot, the Q3 and Q2 are serviced in the ratio of 4:3 for the duration that the slot is open.
2. In the second slot, the queues Q1 and Q0 are serviced in the ratio of 2:1.
3. Fresh arbitration is started every time a slot is opened.

The traffic does not get distributed in the intended ratio of 4:3:2:1, but rather as two groups with different ratios and only for the duration of the slot when the gates are open continuously.

Frame Preemption (FPE)

The IEEE 802.1Qbv-2015 specification defines the schedule for each queue on every egress port that makes the implementation aware of the traffic arrival schedule. This information can be used to block the lower priority traffic from transmission in this time window. Scheduled traffic is forwarded from the sender to the receiver through all the network nodes with a deterministic delay.

To achieve a low latency, ensure that there are no interfering frames during the scheduled windows that are reserved for high priority traffic. The use of scheduled traffic imposes limitations while starting a transmission.

If an interfering frame begins transmission just before the start of a reserved time period, it can extend transmission into the reserved window, and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

A larger guard band equates to a less efficient use of network bandwidth. However, with the implementation of IEEE802.1Qbu (frame preemption), this issue is addressed. Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

During the guard band, only the frames that can complete the transmission of the entire frame before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the window reserved for it.

NOTE: Internal loopback is not supported for preempted frames.

Preemption allows one or more higher priority (express) frames to interrupt the transmission of a lower priority (preemptable) frame; the preemptable frame transmission is resumed and completed after the express frame transmission is complete. To support frame preemption, the following abstractions of the MAC are used:

- A preemptable MAC, called pMAC, which carries the preemptable traffic
- An express MAC, called eMAC, which carries the express traffic

In the implementation, only the parts of the MAC that hold the state during preemption are replicated and represented as pMAC and eMAC. The MAC uses the following ways to halt the transmission of preemtable traffic in the presence of express traffic:

- The MTL scheduler interrupts the preemptable traffic that is currently being transmitted. When the preemption capability is active, the MAC interrupts the transmission and reception of preemptable frames. A preempted fragment can be followed by zero or more express frames before the continuation fragments. The preemptable frame can be fragmented any number of times. However, the minimum final and non-final fragment length criterion must be met.
    - Interleaving of more than one preemptable packet is not permitted. IWhen a preemptable packet is fragmented by an express packet, another preemptable packet cannot be transferred until all the remaining fragments of the first preempted packet are transferred.
- The MTL scheduler prevents starting the transmission of preemptable traffic. When the preemption capability is inactive, the pMAC entity is disabled and only express traffic is transmitted or received.

NOTE: Queue 0 can be an express queue when FPE is enabled, and EST is disabled.

Enabling Frame Preemption

Enable the frame preemption feature by setting the `EMAC_FPE_CTLSTAT.EFPE` bit.

GCL Modification to Support FPE

In the EST only configuration, the GCL entry has up to 24 bits of time interval, and up to 8 high order bits representing the gate open and close state requirements as shown in the ***Gate Control*** table.

Table 28-93: Gate Control - FPE Disabled

Gate Control (up to 8 Bits)	Time Interval (ns) 16, 20 or 24 Bits
OOCCCCCC	T0
OOOOCCCC	T1
CCCCOOCC	T2
CCCCCCOO	T3
…	…
…	…
OCOCOCOO	T126
OOOOCCCC	T127

EST only supports the *SetGate* operation, which implies that the gates are set to either open or close at a given time interval.

However, when both FPE and EST are enabled, the GCL also supports *Set-and-Hold-MAC* and *Set-and-Release-MAC* operations, in addition to the *SetGate* operation. To enable the support of hold and release operations, the format of the GCL is slightly changed while configuring and enabling the FPE. The first queue (Q0) is always programmed to carry preemption traffic. Therefore, it is always open. The bit corresponding to Q0 is renamed as hold and release MAC control.

The TX queues whose packets are preemptable are indicated by the EMAC_FPE_CTRLSTS.PEC field. The GCL bit of the corresponding queue is ignored and considered as always open. So, even if the software writes a 0 (C), it is ignored for such queues.

Table 28-94: Gate Control, FPE Enabled

Gate Control (up to 7 Bits)	Release/Hold Indication	Time Interval (ns) 16, 20 or 24 Bits
CCCOOOO	0	T0
CCCCOOO	0	T1
OCCCOOO	1	T2
COCCOOO	1	T3
CCOCOOO	1	T4
CCCOOOO	0	T5
CCCCOOO	0	T6

When the hold and release bit transitions from a 0 to 1, a *Set-and-Hold-MAC* operation is performed. This operation indicates the halting of the preemptable traffic. A hold request is sent to MTL in advance (the time interval is given in the EMAC_FPE_ADV.HADV field). When the hold and release bit transitions from a 1 to 0, a *Set-and-Release-MAC* operation is performed. This operation marks the resumption of the preemptable traffic. A release request is sent to MTL in advance (the time interval is given in the EMAC_FPE_ADV.RADV field). The preemptable traffic is blocked for the duration that the hold and release bit is set to a 1 in the GCL.

NOTE: The EMAC_FPE_ADV.HADV field is the advance time to initiate the hold state. Ideally, it should be programmed to a value of minimum fragment size. The size should be smaller than the time interval (minimum fragment size + IPG + preamble overheads in nanoseconds).

Impact of Preemption on CBS

In CBS, the definition of *transmit* is:

- True for the duration of frame transmission from the queue.
- False when frame transmission from the queue is complete.

When the CBS algorithm is used along with frame preemption, the value of *transmit* is true only while the MAC is transmitting the frame. If the frame transmission is delayed or interrupted, the value of *transmit* is false until transmission of the frame commences or is resumed. For example, a preemptable frame transmission is interrupted to allow the transmission of an express frame from a different queue. Or the start of express frame is delayed because a preemptable frame is being transmitted)

The value of *transmit* is false during the transmission of any overhead that is a consequence of frame preemption. For example, an additional frame overhead (mCRC, fragment count) that is added to the preemptable frame.

At any given time, if there are no frames in the queue, and the value of *transmit* is false, and credit is a positive value. The credit value is set to zero if there is no preemptable frame from the queue for which transmission is in-progress but has been interrupted.

mPacket Format

When the preemption capability is active, the MAC sends mPackets to the PHY. An mPacket can be one of the following:

1. An express packet
2. A preemptable packet
3. An initial fragment of a preemptable packet
4. A continuation fragment of a preemptable packet

The ***mPacket Format*** figure shows the format of the mPacket. It contains an express packet, a complete preemptable packet, or the initial fragment of a preemptable packet. Format B shows the format of an mPacket containing a continuation fragment of a packet.

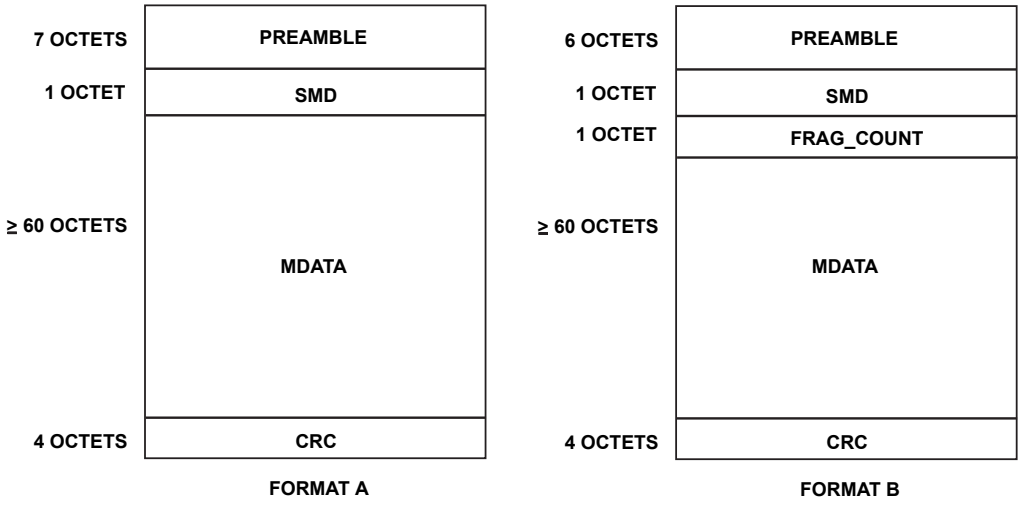


Figure 28-55: mPacket Format

Preamble

The preamble in the mPacket shown in ***mPacket Format*** contains seven octets. The preamble in the mPacket shown in ***mPacket Format*** (Format B) contains six octets. Each octet contains the value of 0x55 (transmitted in order from left to right 10101010).

Start mPacket Delimiter (SMD)

The value of the SMD indicates whether the mPacket contains an express packet, the start of a preemptable packet (initial fragment or complete packet), or any of continuation fragments of a preemptable packet. The ***SMD Values*** table shows the valid SMD values for different mPacket types.

Table 28-95: SMD Values

mPacket Type	Notation	Frame Count	Value
Verify packet	SMD-V	N/A	0x07
Respond packet	SMD-R	N/A	0x19
Express packet	SMD-E	N/A	0xD5
Preemptable packet start	SMD-S0	0	0xE6
	SMD-S1	1	0x4C
	SMD-S2	2	0x7F
	SMD-S3	3	0xB3
Continuation fragment	SMD-C0	0	0x61
	SMD-C1	1	0x52
	SMD-C2	2	0x9E
	SMD-C3	3	0x2A

frag_count

A frag_count is a modulo 4 counter that increments for each continuation fragment of the preemptable packet. The frag_count protects against mPacket reassembly errors by enabling the detection of the loss of up to three packet fragments.

The frag_count field is present only in mPackets with SMD-C notation (continuation fragment). The frag_count is zero in the first continuation fragment of each preemptable packet. The ***frag_count Values*** table shows value associated with the frag_count.

Table 28-96: frag_count Values

frag_count	Value
0	0xE6
1	0x4C
2	0x7F
3	0xB3

mData

The contents of the packet from the MAC. It includes the first byte after the SFD to the last byte before the FCS. The contents are sent in the mData fields of one or more mPackets for that frame. The minimum size of the mData field is 60 bytes.

CRC

The CRC field contains a Cyclic Redundancy Check (CRC) and has an indication of the final mPacket of a frame. In the final mPacket of a frame, the CRC field contains the last four octets of the MAC frame (the FCS field).

For other mPackets, the CRC field contains an mCRC value. The mCRC is calculated on the octets of the packet from the first octet of the frame (the octet following the SFD of preemption frames) to the last octet of the packet transmitted in that mPacket. The check by performing an XOR of the calculated 32 bit CRC value of the fragment and a value of 0x0000FFFF.

Summary of Packet Formats

- Express packet – 7 bytes of preamble, SMD-E, data, and CRC
- Complete preemptable packet – 7 bytes of preamble, current preemptable packet SMD, data, CRC
- Initial fragment (non-final) of preemptable packet– 7 bytes of preamble, current preemptable packet SMD, data, mCRC
- Continuation fragments (non-final) of preemptable packet – 6 bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, mCRC
- Final fragment of preemptable packet – 6 bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, CRC

When frame preemption is enabled, the MAC receiver communicates the exceptions related to received preempted fragments (incorrect fragment sequencing, missed fragment, and so on) to MTL layer by reporting it as a CRC error. Software should not set (=1) the `EMAC_MAC_EXT_CFG.DCRCC` bit. Otherwise, the MTL layer might forward the unintended preempted fragments to the application.

Table 28-97: Current and Previous SMD Values

Previous Preemptable Packet SMD	Current Preemptable Packet SMD
SMD-S0	SMD-S1
SMD-S1	SMD-S2
SMD-S2	SMD-S3
SMD-S3	SMD-S0

Table 28-98: Current and Previous SMD Values

Previous Preemptable Fragment SMD	Previous Preemptable Fragment FC	Current Preemptable Continuation Fragment SMD	Current Preemptable Continuation Fragment FC
SMD-S0	N/A	SMD-C0	FC0
SMD-S1	N/A	SMD-C1	FC0
SMD-S2	N/A	SMD-C2	FC0

Table 28-98: Current and Previous SMD Values (Continued)

Previous Preemptable Fragment SMD	Previous Preemptable Fragment FC	Current Preemptable Continuation Fragment SMD	Current Preemptable Continuation Fragment FC
SMD-S3	N/A	SMD-C3	FC0
SMD-C0	FC0	SMD-C0	FC1
SMD-C0	FC1	SMD-C0	FC2
SMD-C0	FC2	SMD-C0	FC3
SMD-C0	FC3	SMD-C0	FC0
SMD-C1	FC0	SMD-C1	FC1
SMD-C1	FC1	SMD-C1	FC2
SMD-C1	FC2	SMD-C1	FC3
SMD-C1	FC3	SMD-C1	FC0
SMD-C2	FC0	SMD-C2	FC1
SMD-C2	FC1	SMD-C2	FC2
SMD-C2	FC2	SMD-C2	FC3
SMD-C2	FC3	SMD-C2	FC0
SMD-C3	FC0	SMD-C3	FC1
SMD-C3	FC1	SMD-C3	FC2
SMD-C3	FC2	SMD-C3	FC3
SMD-C3	FC3	SMD-C3	FC0

Transmit Preemption

To support preemption, the MAC should have more than one transmit queue with at least one queue designated as express queue (and one queue designated as preemption queue). When FPE is enabled (`EMAC_FPE_CTLSTAT.EFPE` =1), the MTL preempts a preemptable frame. When a hold request is asserted (EST/Qbv configured and enabled), express frames are available for transmission. The frame is present in the MTL FIFO and qualified for arbitration after ensuring that the minimum mPacket mData field size is met.

Therefore, preemption occurs only when at least 60 bytes of the preemptable frame have been transmitted and at least 64 bytes (including the frame CRC) remain.

The earliest starting position of preemption is controlled by the `EMAC_FPE_CTRLSTS.AFSZ` bit. Preemption does not occur until at least 64 x (1 + AFSZ) - 4 bytes of the preemptable frame have been sent.

When preemption occurs, all of the preemptable queues are blocked; only the express queues can arbitrate (if more than one express queue has traffic) and transmit.

The continuation fragment of the preempted frame is the first frame to be transmitted after the release request is asserted (EST/Qbv configured and enabled) and all the express traffic transmission completes.

NOTE: All the PTP packets should be transmitted as express packets.

MTL communicates the following frame-type information to the MAC using a 2-bit preemption control signal on the MTI interface (qualified with SoF and EoF). Refer to the ***MTI Preemption Control Values*** table.

- Express frame
- Preemption frame (full or fragment)
- Continuation fragment (non-final or final)

Table 28-99: MTI Preemption Control Values

Qualifier	Preemption Control Values	Frame Types
SoF	00	Start express
SoF	01	Start preemption
SoF	10	Continuation fragment
SoF	11	Reserved
EoF	00	End of frame
EoF	01	End of fragment

The MTL must wait for the previous fragment status to be received before resuming the continuation fragments of a preempted frame.

MAC Transmit Preemption

The MAC supports preemption by implementing the functionality needed to generate the mPackets. The functionality is based on the preemption control value received on the MTI interface (qualified with SoF and EoF). The MAC determines the frame type (see the ***MTI Preemption Control Values*** table) and generates mPackets accordingly. When the preemption capability is active, the MAC replaces the SFD of a preemption packet with an SMD-S value. A 2-bit rolling frame count is encoded in the SMD-S value. The SMD-E value is the same as the SFD value. So, the SFD of an express packet does not need to be replaced.

Transmit Fragment Status

The MAC sends the transmit fragment status to indicate a successful transmission of fragmented mPackets. To indicate a fragment status, bit 31 of the mti_txstatus_o field is set (=1) and all other values in the status field are zero.

For a transmission error (for example, an underflow or jabber), the frame status is sent (and not a fragment status) with an error indication along with all other relevant status fields. For a receiving an error status, the MTL drops the remaining fragments and does not send any more continuation fragments.

Receive Preemption

When FPE is enabled, the MAC receiver passes the incoming packets and differentiates between express packets and preemptable packets. An SMD containing an SMD-E indicates an express packet; an SMD containing an SMD-S indicates the first mPacket of a preemptable packet.

If an mPacket containing an SMD-S is received when the MAC has not completely received the previous preempted packet, MAC sets a CRC error status for the previously received partial packet.

When an SMD-S is detected, the MAC records the frame count indicated by the SMD and then begins sending data on the MRI interface.

The MAC checks the last four bytes of the mPacket. If the last four bytes of the mPacket do not match CRC, it indicates the end of the packet with or without a CRC error (CRC check result). If the last four octets of the mPacket match, it indicates that the packet was preempted.

An SMD containing an SMD-C indicates an mPacket that continues the data for a preempted packet. Upon receiving an SMD value of SMD-C, the MAC checks the following:

1. A preempted packet is in progress
2. The frame count indicated by the SMD matches the frame count of the packet in progress
3. The frag_count value indicates the next fragment count

If any of these checks fail, the mPacket is discarded and MAC sets a CRC error status for the partially received packet.

If all the checks pass, the next fragment count is incremented modulo 4.

When a packet is preempted, the MAC saves the state of the partially received packet (filter check status, time stamp, length fields and so on) and can process any received express packets before the continuation fragment is received.

The MAC receiver sends a dummy status for all of the mPacket fragments that are successfully received. It sends the Rx status with the final fragment. If an error is detected during any of the fragments, the Rx status is sent, and the fragment is marked as final fragment. All subsequent continuation fragments received for this packet are dropped in the MAC.

The MAC communicates the following frame type information to the MTL using a 2-bit preemption control signal on the MRI interface. Refer to the ***MRI Preemption Control Values*** table.

1. Express frame
2. Preemption frame (full or fragment)
3. Continuation fragment (non-final or final)

Table 28-100: MRI Preemption Control Values

Qualifier	Preemption Control Values	Frame Types
SoF	00	Start express
SoF	01	Start preemption
SoF	10	Continuation fragment
SoF	11	Reserved
EoF	00	End of frame
EoF	01	End of fragment

Data Alignment

When a received frame cannot be fragmented on any byte boundary, the MAC retains the unaligned bytes of data in the previous fragment and resends them with the next fragment. See the ***MAC Data Alignment*** figure.

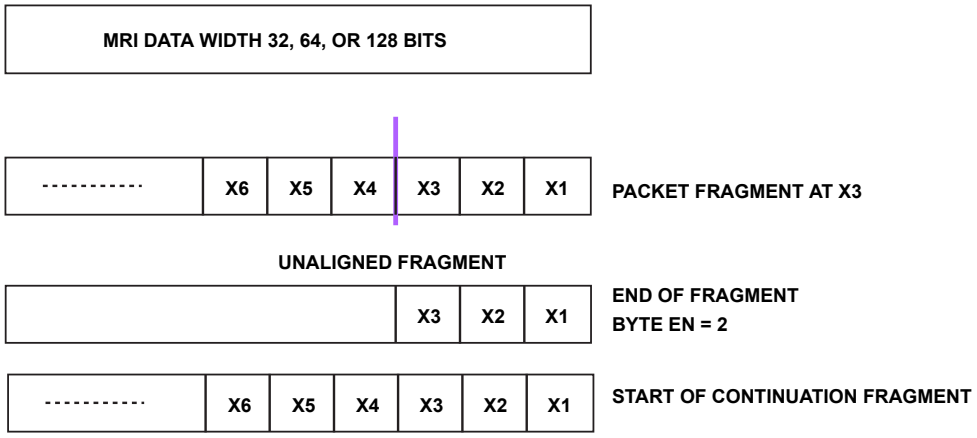


Figure 28-56: MAC Data Alignment

The MTL can choose to ignore the partial data (based on byte enable value) that comes with the end of the fragment and instead use the data width aligned data received in the first beat of the next continuation fragment.

MTL Receive Preemption

The MTL Rx must have at least two receive queues to support FPE function because the preemptable packets and the express packets must be routed to separate queues. The `EMAC_RXQ_CTL0` registers control the destination Rx queue of a received packet. Program these registers such that preemptable traffic and express traffic are not routed to the same receive queue. The queue mapping for tagged packets is based on VLAN user-priority field. So, the priority of preemptable and express packets are mutually exclusive. Packets of a certain priority (traffic class) are either express or preemptable, but not both.

For the preemptable frames, in addition to the Priority Selected in Rx Queue (PSRQ) routing, a programmable Frame Preemption Residue Queue (FPRQ) is supported. This feature routes all other preemption packets received (untagged, SA/ DA or VLAN filter fails forwarded due to RA being set or VTFE being reset). Refer to the ***MTI Preemption Routing*** table.

Table 28-101: MRI Preemption Routing

Preemption Packet Type	Queue Routing
AV tagged packets passing the SA/DA and VLAN filters	PSRQ*
DCB/generic tagged packets passing the SA/DA and VLAN filters	PSRQ**
Tagged packets failing the SA/DA and VLAN filters without setting the receive all (RA = 0) or setting VTFE =1	Dropped
All other packets or when RA = 1	FPRQ

The express frames use queue 0 as a default queue. In case of filter failures, queue 0 must not be used for receiving preemption frames.

When a fragment is received, it is pushed into its destination receive queue. When this packet is preempted, and is followed by an express packet, it is pushed/stored in a different receive queue. The MTL saves the context of the preempted frame, and processes one or more express frames. When the continuation fragment arrives, it restores the saved context and pushes the remainder of the fragmented frame into its receive queue.

MTL Receive Arbitration

On the ARI interface, data is fetched from the MTL receive FIFO based on the arbitration selected in the `EMAC_MTL_OPMODE` register. Frame-based arbitration can be used only when all the MTL preemption queues operate in store-and-forward mode. Otherwise, there is a loss of bandwidth on the ARI interface because all of the fragments of a preemptable packet are not available. Therefore, express packets received between the fragments are blocked until all of the fragments are received and transferred; this defeats the purpose of express traffic.

When operating in either threshold (cut through) or store-and-forward modes, PBL-based arbitration is recommended over frame-based arbitration. For PBL-based arbitration, the watermark check is always performed. The arbitration/transfer of data is in chunks of PBL size of data (similar to the concepts of fragments). Therefore, the express queue packets are blocked for less time and the ARI interface transfers the data without a loss of efficiency.

Frames Support in Offload Engines

When frame preemption is enabled, all the packets processed by the offload engines (ARP, PTO) and PTP, PAUSE, and PFC packets must be received as express/normal packets. These offload engine logic and functions do not recognize preemption frames.

Verify and Respond mPackets

When FPE function is present, the MAC can receive and detect the verify and respond mPackets, even when FPE is not enabled by software. When the MAC detects valid verify and respond mPackets, it notifies the software by

configuring the EMAC_FPE_CTLSTAT.RVER and EMAC_FPE_CTLSTAT.RRSP fields, respectively. Optionally, an interrupt can be generated. These packets have empty (all-zero) data payload; they are dropped inside the MAC and not forwarded to the MRI.

Software can configure the EMAC_FPE_CTLSTAT.SRSP and EMAC_FPE_CTLSTAT.SVER fields to request the MAC to transmit verify and respond mPackets, respectively. Upon successful transmission of these frames, the MAC clears the EMAC_FPE_CTLSTAT.SRSP / EMAC_FPE_CTLSTAT.SVER bit and sets the EMAC_FPE_CTLSTAT.TVER/EMAC_FPE_CTLSTAT.TRSP bits. Optionally, an interrupt can be generated when these events occur.

NOTE: 1. For preemption frames received with a CRC error, ignore the following status fields:
   - Packet length
   - Dribble error indication
   - Runt frame indication
2. When FPE is configured, the EMAC_RQ0_OPMODE.FUP (RXQ(n)) bit should not be set.
3. When FPE is configured, do not disable CRC checking upon receive. Internally, FPE error fragments are identified using CRC checks. For details, see the EMAC_MAC_EXT_CFG.DCRCC field.
4. Express and preemption traffic must be routed to different DMAs.

MMC Counter and Interrupt Registers

The following MMC counters and associated interrupt registers are instantiated/present in the MAC when the frame preemption feature and any one of the existing MMC counters are selected during RTL configuration.

Table 28-102: MMC Counters and Interrupt Registers

Frame Assembly Error Counter	Description	Associated MMC Counter
Frame assembly error counter	A 32-bit counter that provides the number of MAC frames with reassembly errors on the receiver, due to a mismatch in the fragment count value	EMAC_RXPKT_ASERR_CNT
Frame SMD error counter	A 32-bit counter that provides the number of MAC frames rejected due to arriving with an SMD-C when there was no preceding preempted frame	EMAC_RXPKT_SMDERR_CNT
Frame assembly OK counter	A 32-bit counter that provides the number of MAC frames that were successfully reassembled	EMAC_RXPKT_ASOK_CNT
MAC Rx fragment counter	A 32-bit counter that provides the number of additional mPackets received due to preemption.	EMAC_RXFPE_FRGMNT_CNT

Table 28-102: MMC Counters and Interrupt Registers (Continued)

Frame Assembly Error Counter	Description	Associated MMC Counter
MAC Tx fragment counter	A 32-bit counter that provides the number of additional mPackets transmitted due to preemption.	EMAC_TXFPE_FRGMNT_CNT
Hold request counter	A 32-bit counter that maintains the count of the number of hold requests given to the MAC	EMAC_TXHLDRQ_CNT

Additional Registers Associated with MMC Interrupts

Following are the additional registers associated with MMC interrupts for the MMC error counters:

- EMAC_MMC_FPE_TXINT
- EMAC_MMC_FPE_TXIMSK
- EMAC_MMC_FPE_RXINT
- EMAC_MMC_FPE_RXIMSK

Time-Based Scheduling

The time-based scheduling feature is suitable for traffic whose periodicity and rate are predictable. This feature is optional. To improve the quality-of-service of such traffic:

- The transmit DMA fetches the packet from the host memory for transmission at designated time. This method helps the software to set up the transmit descriptors before the packet is ready/available. It reduces the overhead on the software and avoids constant monitoring of the time. It prepares the descriptors for the targeted transmission time of the packet.
- The MAC transmits the packet only at the designated/pre-determined time even when the packets are fetched in advance. This method helps in maintaining a constant transmission rate that can be consumed by the receiver station; therefore, it avoids congestion and excessive buffering in the network.

The time-based scheduling feature supports fetching and launching an Ethernet packet at (or after) a pre-determined time. The time-based scheduling is supported only in the following modes:

- Full duplex mode
- Link speed is 100 Mbps or higher

NOTE:
- Do not enable time-based scheduling for the channel for which TSO feature is enabled.
- In EST enabled configurations, the EMAC_TBS_CTRL.ESTM bit impacts the time-based scheduling feature only when the DUT processes the GCL list.
- Slot number is meaningful only after GCL is active. So, start the GCL list before enabling the TBS and EST features. If GCL is not started, all gates are deemed open. If packets do not make progress, the Tx queue creates back pressure.

TBS Definitions

The following definitions apply to time-based scheduling:

- Launch time - the time beyond which MTL can schedule the packet for transmission
- Fetch time - the time beyond which the Tx DMA can schedule a packet-fetch from the host memory
- Expiry time - the time beyond which the packet is dropped by MTL

Launch Time

The launch time is specified in the enhanced normal descriptor in the DMA configurations. For more details, see Enhanced Descriptor for Time-Based Scheduling. The launch time can also be specified as a control word for each packet. When the launch time is specified in the DMA configuration, it is valid only for the specific frame that is scheduled for transmission. The launch time is reset after the transmission of one frame.

The formats of the launch time include:

- Normal/absolute format
- EST/offset format

In normal/absolute format, the launch time is an absolute time value at which the packet is launched for transmission. The launch time is interpreted in the normal/absolute format when `EMAC_TBS_CTRL.ESTM` = 0.

The launch time is a 32-bit value, where the most-significant 8-bits represent the time in seconds and the rest 24 bits represent the time in 256 ns. The launch time is compared against the IEEE 1588 based system/PTP time (bits [39:8]) and rolls over after 256 seconds.

The maximum value of the lower 32-bits of system time is 999,999,999 decimal (0x3B9AC9FF). It wraps to zero when reaching this value (representing a full second). Therefore, the maximum value of the lower 24 bits of the launch time (after multiplying by 256) must be 0x3B9AC9.

Because the maximum value of launch time is 256 seconds:

- Launch time is greater than current system time when its value is between system time[39:8] and system time[39:8] + 128 s
- Launch time is less than current system time when its value is between system time[39:8] + 128 s and system time[39:8] + 256 s. This computation is modulo 256.

In EST/offset format, the launch time is an offset value relative to the time indicated by the BTR of the GCL list provided in the GCL Slot Number (GSN). The value in the BTR is always updated to the start time of the current loop of the GCL. For each packet, the GSN value and the launch time value are specified in the enhanced transmit descriptor.

The launch time offset is a 32-bit value; the upper 8-bits represent the time in seconds and the rest 24 bits represent the time in 256 ns, which is added to the BTR value corresponding to GSN. The value of the launch time offset must be smaller than the value of the cycle time (specified in the CTR register that is implemented when EST is enabled). If the CTR is greater than or equal to 1s, the maximum value of the lower-24 bits of launch time offset should be 999,999,999 decimal (0x3B9AC9FF).

In EST/offset format, the launch time is a 64-bit value, which is interpreted as:

Launch time = Launch GSN BTR[63:0] + (Launch time offset[31:0] << 8), which is compared with the system time [63:0].

where:

GSN BTR – the base time value at which the launch GSN loop started execution.

GSN – A modulo 16 count of the GCL loop count. The count is incremented for every new GCL loop; installation of new GCL list does not impact the count. The current GCL slot number can be obtained by reading the EMAC_EST_STAT.CGSN field. The maximum value of GSN is 15. Therefore, GSN values between the EMAC_EST_STAT.CGSN field and CGSN+8 represent current or future slots; all other GSN values are interpreted as elapsed slots or past slots. So, for the correct interpretation of time, GSN value must be between EMAC_EST_STAT.CGSN and CGSN+8 for the correct interpretation of time.

Launch Expiry Time

In the normal/absolute format, when EMAC_TBS_CTRL.LEOV = 1, the EMAC_TBS_CTRL.LEOS bit determines the maximum amount of time a frame is eligible for launch, starting from the time the frame becomes eligible for launch.

The launch expiry offset is a 24-bit value defined in 256 ns units, with a maximum possible value of 999,999,999 ns (0x3B9AC9FF).

Launch expiry time = (launch time[39:8] + EMAC_TBS_CTRL.LEOS [32:8]) * 256 ns

The packet with a specific launch time is considered eligible for transmission when the launch time is less than the system time and (if EMAC_TBS_CTRL.LEOV = 1) the system time is less than the launch expiry time.

When the system time is greater than the launch expiry time the frame is categorized as expired and is dropped from the MTL FIFO.

NOTE:
1. For correct interpretation and meaningful operation, the fetch, launch, and launch expiry time should never be set to a value larger than current system time + 128 s; such a value is interpreted as time that has already elapsed.
2. In full duplex mode, the frames dropped from the MTL FIFO have an error summary (bit 15) and Excessive Deferral (bit 3) of the TxStatus set.

In the EST/offset format, when EMAC_TBS_CTRL.LEOV = 1, the launch expiry GSN offset (EMAC_TBS_CTRL.LEGOS) and launch expiry offset (EMAC_TBS_CTRL.LEOS bits determine the maximum amount of time a frame remains eligible for launch, starting from the time the frame becomes eligible for launch.

The launch expiry offset is computed as:

Launch expiry offset = (EMAC_TBS_CTRL.LEGOS: EMAC_TBS_CTRL.LEOS)

The launch expiry offset is a 24-bit value defined in the units of 256 ns, with a maximum possible value of the smaller of 999,999,999 ns or CTR-1 ns. The EMAC_TBS_CTRL.LEGOS field holds the GSN offset (multiples of CTR time). The EMAC_TBS_CTRL.LEOS field holds maximum value of CTR (sub-CTR values) in ns.

The launch expiry GSN is computed as:

Launch expiry GSN = (launch GSN + EMAC_TBS_CTRL.LEGOS + CCMA)

where,

CCMA - the CTR carry due to modulo addition. This value is 1 if ((launch time offset + EMAC_TBS_CTRL.LEOS) << 8) ≥ CTR.

- When CCMA = 1, launch expiry offset = (launch time offset+ EMAC_TBS_CTRL.LEOS) - CTR
- When CCMA = 0, launch expiry offset = (launch time offset + EMAC_TBS_CTRL.LEOS).

Launch expiry time = launch expiry GSN BTR[63:0] + launch expiry time offset

- When EMAC_TBS_CTRL.LEOV = 0, the launch expiry time is not checked.
- When EMAC_TBS_CTRL.LEOV =1, and
  - The system time is greater than the launch expiry time, the frame is dropped from MTL FIFO. The frame is considered as expired.
  - The launch time is smaller than the system time and launch expiry time is greater than system time, the frame is considered eligible for launch.

NOTE:

1. The max value of EMAC_TBS_CTRL.LEGOS is 7. This implies that when EMAC_TBS_CTRL.LEOV = 1, the frame has a maximum lifetime of < 8 GCL loop iterations after it becomes eligible for launch.
2. The slot number of the first GCL list executed each time after EST is enabled, is zero.

Fetch Time

Software can indicate the fetch time for each packet (EMAC_DMA_TBS_CTL0.FTOV =1). When EMAC_DMA_TBS_CTL0.FTOV = 0, the fetch time offset is not valid and the DMA fetches packets without any time constraints. The fetch time accounts for all possible delays in the DMA fetch operation and ensures that the frame is present in the MTL FIFO before the launch time.

In normal/absolute mode, fetch time is derived/calculated by reducing the time specified in the EMAC_DMA_TBS_CTL0.FTOS field from the given launch time. In a normal mode of operation, the fetch launch time is computed as:

Fetch time[39:8] = (launch time[39:8] - EMAC_DMA_TBS_CTL0.FTOS[31:8])

The fetch time is 32 bits. It is compared against system time[39:8] to determine the eligible for fetching the frame:

- The fetch time is defined as greater than system time when the fetch time is in the range of system time[39:8] and system time[39:8] + 128 s. The frame is considered ineligible for fetch.
- The fetch time is defined as smaller than the system time when the fetch time is in the range of system time[39:8] + 128 s and system time[39:8] + 256 s. The frame is considered as eligible for fetch.

This computation is modulo 256 .

In the EST/offset mode, theEMAC_DMA_TBS_CTL0.FGOS field provides the slot number offset to be deducted from the launch GSN. In this case, the EMAC_DMA_TBS_CTL0.FTOS value should be the smaller of 999,999,999 ns or CTR-1 ns.

When (launch time offset ≥ EMAC_DMA_TBS_CTL0.FTOS):

- Fetch time offset = ((launch time offset - EMAC_DMA_TBS_CTL0.FTOS) x 256 ns)
- CBFS (CTR borrow for fetch subtraction) = 0

When (launch time offset < EMAC_DMA_TBS_CTL0.FTOS):

- Fetch time offset = CTR + ((launch time offset - EMAC_DMA_TBS_CTL0.FTOS) x 256 ns)
- CBFS = 1

The fetch GSN is computed as follows:

Fetch GSN = launch GSN - EMAC_DMA_TBS_CTL0.FGOS - CBFS

Fetch time = Fetch GSN BTR[63:0] + Fetch time offset

The frame is marked eligible for DMA fetch when the fetch time is smaller than the system time.

The max value of EMAC_DMA_TBS_CTL0.FGOS is 7. This implies that when EMAC_DMA_TBS_CTL0.FTOV =1, the frame can be fetched at a maximum of <8 GCL loop iterations before it becomes eligible for launch.

DMA Operations Sequence

When EMAC_DMA_TBS_CTL0.FTOV = 1, the following DMA operations sequence occurs:

1. Fetch the first enhanced normal descriptor. (FD is set).
2. If LTV is set and fetch enabled, compute the fetch time based on the launch time. Wait for the system time to be greater than fetch time.
3. Read the frame (data) from the host memory and transfer to MTL FIFO.
4. Close the normal descriptor.
5. Fetch the next normal descriptor (if the previous descriptor was not the last).
6. Repeat steps 4 to 6, until the last descriptor of the frame (LD is set).

After the last descriptor of a frame, the next enhanced normal descriptor should be programmed with a new launch time and with the LTV bit set. Otherwise, the subsequent frames are processed without any time restrictions.

Control Word

Time-based scheduling ensures that a packet is allowed to arbitrate for transmission only when the system time is greater than the launch time.

- Launch time and GSN use the same field that holds the 64-bit time stamp value for One Step Timestamping Correction (OSTC). When the LTV bit is set and OSTC bit is not set (bit 19 of the first control word), the time stamp field is interpreted as the launch time. Launch time uses the fields meant for OSTC time stamp[39:8]; GSN uses the OSTC time stamp[43:40] field.
- Bit 31 of control word 0 is the equivalent of the LTV field. The last two Control D words provide the launch time and GST.
- The last two control D words are the time stamp for OSTC when bit 19 of control word 0 (OSTC) is set.

When the MTL TxFIFO read controller reads the first control word, it checks bit 31 to determine if the frame has a valid launch time. If bit 31 is set, it reads the subsequent control words to get the launch time.

Impact on Transmission Selection Algorithm

The time-based scheduler does not directly influence the transmission selection algorithm. However, it has an indirect influence as it determines whether a queue is eligible to participate in the scheduling.

When a frame of a specific queue has a valid launch time, the time-based scheduler ensures that the frame participates in transmit scheduling (TRC scheduler) only after the launch time has elapsed.

The gating done by time-base scheduler is only for the scheduling (picking frame for transmission). It might not alter any other characteristics and behavior of the frame. For example, the frame waiting for the launch time to lapse continues to gain credits in CBS. The CBS credits are not lost because the frame is marked as available, but not ready. Strict priority or CBS are the recommended transmit scheduling algorithms for the TBS implementation. Scheduling for TBS-enabled queues is more predictable.

The ***Time-Based Scheduling Flow*** figure shows the time-based scheduling flow.

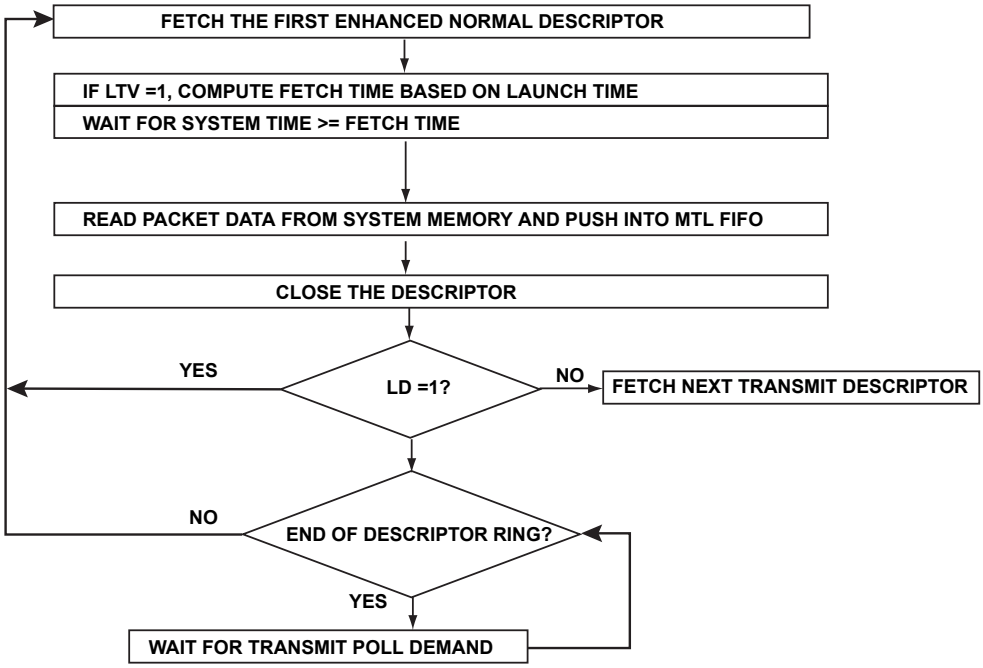


Figure 28-57: Time-Based Scheduling Flow

Flow Control

This section describes the flow control feature of EMAC for transmit and receive paths.

Transmit Flow Control

The transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

Flow Control in Full-Duplex Mode

In full-duplex mode, the EMAC uses one of the following packet types for flow control:

- IEEE 802.3x pause packets
- Priority flow control (PFC) packets

The `EMAC_RXFLOW_CTL.PFCE` bit determines whether PFC packets or IEEE 802.3x pause control packets are used for flow control. If the PFCE bit is set, the MAC sends the PFC packets.

Pause Packet Structure

The *Pause Packet Fields* table describes the fields of a pause packet.

Table 28-103: Pause Packet Fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control Opcode	Contains 0001 for IEEE 802.3x Pause Control packets; 0101 for PFC packets
PT	Contains Pause time specified in the PT field of the MAC_Q#_Tx_Flow_Ctrl register

Pause Packet Control

When the `EMAC_Q[i]_TXFLOW_CTL.FCB_BPA` bit is set, the MAC generates and transmits a single pause packet. If the FCB bit is set again after the pause packet transmission is complete, the MAC sends another pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously transmitted pause packet, the application should program the pause time register with appropriate value and then again set the FCB_BPA bit.

Similarly, when the flow control signal from FIFO layer (mti_flowctrl_i) is asserted, the MAC generates and transmits a single pause packet. If the mti_flowctrl_i signal remains asserted at a configurable number of slot times before the pause time runs out, the MAC transmits a second pause packet. This process is repeated if the mti_flowctrl_i signal remains active. If the mti_flowctrl_i signal goes inactive prior to the sampling time, the MAC transmits a pause packet with zero pause time (if the bit is set to 0) to indicate to the remote end that the receive buffer is ready to receive new data packets. For PFC packets, the priority, pause time, and other controls can be specified for a queue in the `EMAC_Q[i]_TXFLOW_CTL` register. The MAC supports independent flow control for each receive queue. There is one trigger input for each receive queue. Therefore, when multiple triggers come simultaneously, the MAC sends one PFC packet for each trigger. Based on the hardware trigger or software trigger through respective queues, the MAC sends a PFC packet with programmed pause time and priority. If multiple priorities are programmed in same receive queue, multiple priorities are set for the PFC packet with same pause time values. A separate pause timer is available for each receive queue.

Flow Control in Half-Duplex Mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

The *Transmit MAC Flow Control* table describes the flow control in the transmit path for queue 0 based on the setting of the following bits:

- `EMAC_RQ0_OPMODE.EHFC`

- EMAC_Q[i]_TXFLOW_CTL.TFE
- EMAC_MAC_CFG.DM

Flow control is similar for all queues.

Table 28-104: Transmit MAC Flow Control

EHFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs back-pressure when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i = 1.
1	1	0	The MAC transmitter performs back-pressure when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx performs back-pressure when the Rx queue level crosses the threshold set by bits[10:8] of the EMAC_RQ0_OPMODE register.
0	1	1	The MAC transmitter sends the pause packet when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i = 1.
1	1	1	The MAC transmitter sends the pause packet when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i = 1. In addition, the MAC Tx sends a pause packet when the Rx queue level crosses the threshold set by bits[10:8] of the EMAC_RQ0_OPMODE register.

Enabling Transmit Flow Control

To independently enable transmit flow control for each receive queue, set the EMAC_Q[i]_TXFLOW_CTL.TFE bit.

Triggering Transmit Flow Control

Transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end. The application can request the MAC to send a pause packet or initiate back-pressure by using either of the following methods:

- Software Trigger: In this method, the application sets the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit.
- Hardware Trigger: In this method, the application triggers the flow control by asserting the mti_flowctrl_i signal based on the receive queue threshold.
- Receive Queue Threshold: The flow control operation of the MAC is enabled when the EMAC_RQ0_OPMODE.EHFC bit is set. The flow control signal to the MAC is asserted when the fill level of the receive queue crosses the threshold configured in the EMAC_RQ0_OPMODE.RFA bit field. This flow control signal is de-asserted when the fill-level of the queue is lower than the threshold configured in the EMAC_RQ0_OPMODE.RFD field.

The hardware flow control generated based on the receive queue threshold crossing condition is applicable only when the receive queue size is 4,096 bytes or more.

NOTE: The RFA sets the threshold at which the flow control is triggered and the MAC schedules to send a PAUSE frame to be sent by the local transmitter.

The remote transmitter can continue to send packets until it receives this PAUSE packet. So, the RXQ needs space to take in this data even after the flow-control is triggered, to avoid overflow and loss of packets.

The max-space required that is calculated theoretically can be 2 max-sized packets + a little more when the read from the RXFIFO is not occurring during this whole time.

Theoretically, this makes RFA = 4 or more, which implies that RxQ must have at least a size of 4KB.

Practically, the system works with RFA = 2 (Full – 2KB, with little probability of overflow). When the application has a FIFO size of less than 4K, with RFA = 2, the flow control is triggered with 1000B in the RxQ, which is less than maximum packet size.

The `EMAC_RQ0_OPMODE.RFA` bit field is used for activating the flow control. These bits control the threshold (fill-level of Rx queue) at which the flow control is activated.

For example, if the RxQ size is 2K and the program sets the `EMAC_RQ0_OPMODE.RFA` bit field to 1K; as soon as 1K is received, flow control is activated. If the program sends a packet of 1500 bytes (which is allowed as per spec), flow-control is triggered even before a complete packet is received. This reduces the throughput considerably.

When the RxQ size is 4K, the program can set the RFA to 2K, and the whole packet can be received.

NOTE: Receive pause packets should have a frame size of 64 bytes.

Receive Flow Control

Receive flow control is implemented by the MAC based on the bit value of the respective register, and the destination address and different fields of the received packet.

The *Receive MAC Flow Control* table describes the flow control in the receive path based on the setting of the `EMAC_RXFLOW_CTL.RFE` and `EMAC_MAC_CFG.DM` bits.

Table 28-105: Receive MAC Flow Control

RFE	DM	Description
0	X	The MAC receiver does not detect the received pause packets
1	0	The MAC receiver does not detect the received pause packets but recognizes such packets as control packets.
1	1	The MAC receiver detects or processes the pause packets and responds to such packets by stopping the MAC transmitter.

For EMAC0, the program can enable the PFC packet detection by setting the EMAC_RXFLOW_CTL.PFCE bit. If PFC packet detection is enabled, the MTL transmit queue corresponding to the received priority is blocked when the PFC packet is received. If PFC packet detection is not enabled in configurations with multiple queues and the RFE bit is enabled, the MAC transmitter is blocked when the 802.3x pause packet is received.

The following list describes the receive flow control.

1. The MAC checks the destination address of the received pause packet for either of the following:
    - Multicast destination address: The DA matches the unique multicast address specified for the control packet (48'h0180C2000001).
    - Unicast destination address: The DA matches the content of the MAC address register 0 and the EMAC_RXFLOW_CTL.UP bit. If the UP bit is set, the MAC processes pause packets with a unicast destination address in addition to the unique multicast address.
2. The MAC decodes the following fields of the received packet:
    - Type field: This field is checked for 16'h8808.
    - Opcode field: This field is checked for 16'h0001 (pause packet) or 16'h0101 (PFC packet).
    - Pause Time or Pause Time Vector field: The pause time (for pause packet) is captured to determine the time for which the transmitter needs to be blocked. The Pause Time Vector field (for a PFC packet) is captured to determine the time for which the MTL transmit queue corresponding to the received priority needs to be blocked.
    - Priority Enable Vector field: This field is valid only for PFC packets. It is captured to determine the MTL transmit queue corresponding to the received priority.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter does one of the following:
    - For 802.3x Pause packets, the MAC pauses the transmission of any data packet for the duration of the decoded pause time value multiplied by the slot time (64-byte times).
    - For PFC packets, the MAC blocks the transmit queues corresponding to the priority. Based on the priorities assigned to the transmit queues in the EMAC_TXQ_PRTY_MAP0 and EMAC_TXQ_PRTY_MAP1 registers, the MAC asserts the respective bit in the mti_disable_txq_o signal and loads the pause timer corresponding to the priority. The transmit queue corresponding to the priority is blocked until the pause timer expires.
    - In addition to the PFC-based flow control operation, when the PFCE bit is set, all bits of the mti_disable_txq_o signal are asserted while in LPI mode so that the ETS or CBS scheduler can start over after exiting the LPI mode.
4. The MAC transfers the received control packet to the application based on the setting of the EMAC_MACPKT_FILT.PCF bit field.

If subsequent pause or PFC packets are received before the earlier pause time expires, the MAC updates the pause timer with new value.

Loopback Mode

The MAC supports Loopback of transmitted packets to its receiver.

Guidelines for Using Loopback Mode

The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result in issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip (for example, in a FPGA setup), the program should externally generate the transmit and receive clocks and provide these clocks to the MAC.
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO.

The transmit and receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the `EMAC_TXD[n]` data to the receive path. The asynchronous FIFO is 10-bits (6-bits in 10/100 Mbps mode) wide to accommodate `EMAC_TXD[n]`, TXEN, and TX_ER. The depth of the FIFO is five in 1000 Mbps mode and nine in 10/100 Mbps mode. The FIFO is free running to write on the write clock TX_CLK and read on every read clock RX_CLK.

At the start of each packet read out of the FIFO, the write and read pointers are re-initialized to have an offset of 2 (4 in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer. This also ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between MII/RMII/RGMII transmit and receive clock frequencies. Therefore, bigger packets should not be looped back because they may become corrupt in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP packets that are looped back.

Enabling Loopback Mode

To enable this feature, program the `EMAC_MAC_CFG.LM` bit.

Programs can enable loopback for all PHY interfaces. The data is always looped back through an internal asynchronous FIFO on to the internal receive MII/RMII/RGMII interface, irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding transmit PHY interface block, on to the Ethernet line.

NOTE:
- During loopback, the data/packet is reflected on TXD[n] signal.
- Preemption is not supported in loopback mode.

Endian Support

The EMAC DMA supports both little endian and big endian modes. The following table illustrates the order of bytes on 32-bit data bus for little endian and big endian modes, respectively.

Table 28-106: Endian Support

Transfer Size	Address OffsetA[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Little-Endian Mode					
Always word transfer	0	MSB	MSB - 1	LSB + 1	LSB
Big Endian Mode					
Always word transfer	0	LSB	LSB + 1	MSB - 1	MSB

The endian mode can be programmed using the `PADS_PCFG0.EMAC0_ENDIANNESS` register. This bit should be set for big-endian mode and cleared for little endian mode. The reset value of this bit is '1' (big endian mode).

EMAC Event Control

The EMAC has registered with the system event controller (SEC) module. Various interrupt sources within the EMAC peripheral are shared through this interrupt line. Refer to the SEC chapter for details on how interrupts work in this product and how to configure them.

EMAC Interrupt Signals

Interrupts from the EMAC are triggered from the EMAC DMA layer MTL, or the EMAC CORE (MAC) layer. The following table shows the EMAC interrupt signals.

Table 28-107: EMAC Interrupt Signals

Interrupt Type	Description
`EMAC_STAT`	Combined interrupt signal includes MAC, MTL, and DMA interrupts.
`EMAC_MAC`	Interrupts from MAC and MTL
`EMAC_DMA[n]_TX`	DMA Transmit Per Channel Interrupts
`EMAC_DMA[n]_RX`	DMA Receive Per Channel Interrupts

Interrupts can be generated because of various events in the EMAC. These events are captured in status registers, and interrupt enables are provided for each source of an interrupt such that the interrupt signal (`EMAC_STAT`) is asserted for an event only when the corresponding interrupt enable is set.

On EMAC, the interrupt status and corresponding enable registers are organized in an hierarchical manner so that it is easier for software to traverse and identify the source of an interrupt event quickly. When `EMAC_STAT` is asserted, the `EMAC_DMA_ISTAT` register is the first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (transmit and receive

pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- EMAC_MAC_ISTAT
- EMAC_MTL_ISTAT
- EMAC_DMA[n]_STAT

Interrupts from the MAC

Interrupts can be generated from the MAC because of various events in the MAC Receiver, Transmitter, or the modules/functions such as RMON counters, EEE and so on. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software. The EMAC_MAC_ISTAT register describes the events that can cause an interrupt from the MAC. The MAC interrupts are disabled by default. Each event can assert the interrupt on the EMAC_STAT or EMAC_MAC signals when the corresponding bit is set in the EMAC_MAC_IEN register.

The interrupt register bits only indicate the block from which the event is reported. The program must read the corresponding status registers and other registers to clear the interrupt. For example, when set high, bit 0 (EMAC_MAC_ISTAT.RGSMIIIS) indicates that the link status on the RGMII, SGMII, or SMII interface has changed. The program must read the EMAC_PHYIF_CTLSTAT register to clear this interrupt event.

NOTE: By default, the MAC interrupt status bits are cleared when the register that contains the source of the interrupt is read. If the EMAC_CSR_SW_CTL.RCWE bit is programmed to 1, the MAC interrupt status bits are cleared when the bit that contains the source of the interrupt is explicitly written to 1.

NOTE: Both the EMAC_STAT and the EMAC_MAC interrupt signals are generated for interrupts from the MAC. The EMAC_MAC interrupt is always generated one cycle ahead of the EMAC_STAT interrupt. If both the EMAC_STAT and EMAC_MAC interrupts are enabled:

- For GIC/ARM, only the EMAC_MAC interrupt service routine is called
- For SEC/SHARC, both the EMAC_STAT and EMAC_MAC interrupt services routines is called

Interrupts from the MTL

The EMAC can generate interrupts because of events in the MAC layer or in the MTL modules. The EMAC_STAT output is the interrupt signal, which is a level signal (asserted until the interrupt source is read and cleared). In the MTL, the interrupts are mainly related to exception events in the TxQ or RxQ in the transmit and receive paths, respectively. The interrupt status is captured and organized in a hierarchical manner to identify the root cause quickly.

The EMAC_MTL_ISTAT register identifies the top-level modules that can cause the interrupt to be asserted. Bits[7:0] identify 8 Queues (transmit or receive). They are read-only bits, and the application should read the corresponding EMAC_Q[n]_INT_CTLSTAT register to identify the exact cause and set the corresponding bits to 1'b1 to clear the interrupt event. The assertion of EMAC_MAC due to these events is enabled by the corresponding enable bits in the EMAC_Q[n]_INT_CTLSTAT register.

Interrupts from DMA

The EMAC_STAT interrupt is a level signal and gets de-asserted only when all the enabled interrupt events are cleared in their respective status registers and correspondingly all the bits in the EMAC_DMA_ISTAT register are cleared.

The EMAC_DMA[n]_STAT register captures all the interrupt events of that transmit DMA and receive DMA channel pair. The EMAC_DMA[n]_IEN register contains the corresponding enable bits for each interrupt event. There are two groups of interrupts in the DMA channel, normal and abnormal. They are indicated by the EMAC_DMA[n]_STAT.NIS and EMAC_DMA[n]_STAT.AIS bits. The normal group is for events that happen during the normal transfer of packets (EMAC_DMA[n]_STAT.TI, EMAC_DMA[n]_STAT.RI, EMAC_DMA[n]_STAT.TBU) while the abnormal interrupt events are for error events. Interrupt events are cleared by writing 1'b1 to the corresponding bit position. When all the enabled interrupt events are cleared (including the NIS and AIS), the interrupt source from the DMA channel is cleared and the corresponding bit in the EMAC_DMA_ISTAT register is also cleared.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. For example, the EMAC_DMA[n]_STAT.RI bit indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received.

An interrupt is generated only once for multiple events. The driver must scan the EMAC_DMA_ISTAT register for the cause of the interrupt and clear the source in the respective status register. The EMAC_STAT is cleared only when all the bits of the EMAC_DMA_ISTAT register are cleared.

Periodic Scheduling of Transmit and Receive Interrupt

To improve the throughput and performance, EMAC supports interrupt timer and transmit descriptor to generate interrupts periodically, instead of every DMA transfer.

It is not preferable to generate interrupts for every packet transferred by DMA (EMAC_DMA[n]_STAT.RI and EMAC_DMA[n]_STAT.TI) for system throughput performance reasons. The EMAC gives the flexibility to schedule the interrupt at regular intervals using two methods:

- Set the interrupt on completion bit in transmit descriptor (TDES2[31]) once for every "required" number of packets to be transmitted.
- Similarly, set the IOC (see RDES3[30]) bit only at some specific intervals of receive descriptors. This way, whenever a received packet transfer to system memory is complete, and any of the descriptors used for that packet transfer has the IOC bit set, only then is the event is generated.

In addition, an interrupt timer (EMAC_DMA[n]_RXINTWDTMR) is given for flexible control and periodic scheduling of receive interrupts. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the receive DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (see Table 28-26 Receive Descriptor Fields 3 (RDES3)) is not set. When this timer runs out as per the programmed value, the bit is set, and the interrupt is asserted when the corresponding EMAC_DMA[n]_IEN.RIE bit is enabled. The timer is stopped and cleared

before it expires, iwhen the RI bit is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Per Channel Transfer Complete Interrupt

The transmit transfer complete interrupt (TI) and receive transfer complete interrupt (RI) are reflected in theEMAC_DMA[n]_STAT register. The EMAC_DMA[n]_STAT.TI bit is set whenever the transmit DMA channel closes the transmit descriptor in which the IOC (interrupt on completion - TDES2[31]) bit is set. Similarly, the EMAC_DMA[n]_STAT.RI bit is set whenever the receive DMA channel closes the receive descriptor with LD bit set and in any of the descriptors used for transferring that packet, IOC (interrupt enable on completion - RDES3[30]) bit is set.

The common EMAC_STAT output signal is asserted for the transfer complete interrupts only when the corresponding interrupts are enabled in the EMAC_DMA[n]_IEN register.

EMAC also supports the following per channel transfer complete interrupt signals.

- EMAC_DMA[n]_TX (transmit per channel interrupts)
- EMAC_DMA[n]_RX (receive per channel interrupts)

The behavior of the RI/TI/EMAC_DMA[n]_TX/EMAC_DMA[n]_RX changes depending on the settings of the EMAC_DMA_MODE.INTM bit field [17:16]. The following explains the transfer complete interrupt behavior.

Table 28-108: Transfer Complete Interrupt Behavior

S. No.	Interrupt Mode (INTM)	Behavior of EMAC_DMA[n]_TX and EMAC_DMA[n]_RX	Behavior of EMAC_STAT
1	0	A pulse is asserted on these output signals when corresponding TX/RX transfer complete event (for which IOC descriptor bits are enabled) is detected, irrespective of the corresponding interrupt status.	The TI/RI status signals are set whenever the "Transfer complete" event is detected. The bits get cleared whenever the software driver writes '1' to these bits. The EMAC_STAT is asserted whenever the corresponding interrupts are also enabled in the EMAC_DMA[n]_IEN register. The EMAC_DMA[n]_STAT.NIS status bit is asserted for RI/TI events.
2	1	These signals reflect the value of corresponding EMAC_DMA[n]_STAT.TI/ EMAC_DMA[n]_STAT.TI bits when the corresponding interrupt enable is set. Therefore, they are level signals and are cleared by the application by writing 1'b1 to the RI/TI status bits. This signal is not asserted when the corresponding interrupt enable bit is not set.	The EMAC_STAT signal and EMAC_DMA[n]_STAT.NIS status bit are not asserted for RI/TI events.

Table 28-108: Transfer Complete Interrupt Behavior (Continued)

S. No.	Interrupt Mode (INTM)	Behavior of EMAC_DMA[n]_TX and EMAC_DMA[n]_RX	Behavior of EMAC_STAT
3	2	In this mode, RI/TI interrupts are queued. These signals reflect the value of corresponding EMAC_DMA[n]_STAT.TI/ EMAC_DMA[n]_STAT.TI bits when the corresponding interrupt enable is set. They are level signals and cleared by software by writing 1'b1 to the RI/TI status bits. However, it is set again if another TI/RI event(s) is detected before the TI/RI bits are cleared for the previous event.	The RI/TI status bits are set whenever the Transfer Complete event is detected and gets reset whenever software driver clears those bits by writing 1. However, if another transfer complete event is detected before it is cleared (serviced) by the software, then EMAC automatically set these status bits again. However, the EMAC_STAT signal is not generated based on TI/RI. The EMAC_DMA[n]_STAT.NIS status bit is not asserted for RI/TI events.

EMAC Triggers

EMAC0 contains:

- 12 trigger requesters – 8 for receive DMA channels (EMAC_DMA[n]_RX) and 4 for MCGR DMA request (EMAC0_MCGR[n])
- 20 trigger completers – 8 for receive DMA channels (EMAC_DMA[n]_RX_STOP), 8 for transmit DMA channels (EMAC_DMA[n]_TX) and 4 for MCGR DMA Ack (EMAC0_MCGR[n]_ACK)

NOTE:
- Each of the DMA trigger completers at EMAC interface is controller by 2 trigger signals from the TRU – 1 start trigger, 1 stop trigger. When the TRU gives a start trigger, the trigger completer at EMAC interface is asserted. When the TRU gives a stop trigger, the trigger completer at EMAC interface is de-asserted.
- An error condition occurs when the TRU generates a 2nd start trigger that occurs before a stop trigger or a 2nd stop trigger before a start trigger. This error condition does not affect the trigger signal at EMAC. It is just to check the status and is stored as a W1C status bit in ECO_REG9 of the MISC REG block.

EMAC Programming Model

This section provides the programming model and programming procedures for the Ethernet MAC peripheral.

Initializing DMA

Use the following procedure to initialize DMA.

1. Perform a software reset by setting the EMAC_DMA_MODE.SWR bit. This action resets all of the EMAC internal registers and logic.

2. Wait for the completion of the reset process by polling the EMAC_DMA_MODE.SWR bit which is only cleared (automatically) after the reset operation completes.
3. Program the following fields to initialize the EMAC_DMA_SYSBMODE register:
    a. EMAC_DMA_SYSBMODE.AAL
    b. Fixed burst or undefined burst (EMAC_DMA_SYSBMODE.FB)
    c. Maximum SCB outstanding request limit (EMAC_DMA_SYSBMODE.WR_OSR_LMT and EMAC_DMA_SYSBMODE.RD_OSR_LMT for both write and read)
    d. If fixed length value is enabled (EMAC_DMA_SYSBMODE.FB bit is set), select the maximum burst length possible on the SCB bus (EMAC_DMA_SYSBMODE.BLEN16, EMAC_DMA_SYSBMODE.BLEN8 and EMAC_DMA_SYSBMODE.BLEN4 bits).
4. Create a descriptor list for transmit and receive. In addition, ensure that the descriptors are owned by DMA (set bit 31 of descriptor TDES3/RDES3). For more information about descriptors, see DMA Descriptors.
5. Program the transmit and receive ring length registers (EMAC_DMA[n]_TXDSC_RLEN and EMAC_DMA[n]_RXCTL2.RDRL). The ring length programmed must be at least four.
6. Initialize the receive and transmit descriptor list address with the base address of the transmit and receive descriptor (EMAC_DMA[n]_TXDSC_ADDR and EMAC_DMA[n]_RXDSC_ADDR). Also, program transmit and receive tail pointer registers indicating to the DMA about the available descriptors (EMAC_DMA[n]_TXDSC_TLPTR and EMAC_DMA[n]_RXDSC_TLPTR).
7. Program the settings of the following registers for the parameters like maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSP in case of TxDMA, RBSZ in case of RxDMA, and so on:
    a. EMAC_DMA[n]_CTL
    b. EMAC_DMA[n]_TXCTL
    c. EMAC_DMA[n]_RXCTL
8. Enable the interrupts by programming the EMAC_DMA[n]_IEN register.
9. Start the receive and transmit DMAs by setting the EMAC_DMA[n]_RXCTL.SR and the EMAC_DMA[n]_TXCTL.ST bits.
10. Repeat steps 4 to 9 for all the transmit DMA and receive DMA channels selected in the hardware.

Initializing the MTL Registers

The Transaction layer (MTL) registers must be initialized to establish the transmit and receive operating modes and commands. Use the following steps to initialize the MTL registers:

1. Program the transmit scheduling (EMAC_MTL_OPMODE.SCHALG) and receive arbitration algorithm (EMAC_MTL_OPMODE.RAA) fields to initialize the MTL operation in case of multiple transmit and receive queues.
2. Program the receive queue to DMA mapping in the EMAC_RXQ_DMA_MAP0 and EMAC_RXQ_DMA_MAP1 registers.
3. Program the following fields to initialize the mode of operation in the EMAC_TQ0_OPMODE register.
    a. Transmit store and forward (EMAC_TQ0_OPMODE.TSF) or transmit threshold control (EMAC_TQ0_OPMODE.TTC) in case of threshold mode.
    b. Transmit queue enable (EMAC_TQ0_OPMODE.TXQEN) to value 2‘b10 to enable transmit queue0.
    c. Transmit queue size (EMAC_TQ0_OPMODE.TQS).
4. Program the following fields to initialize the mode of operation in the EMAC_RQ0_OPMODE register:
    a. Receive store and forward (EMAC_RQ0_OPMODE.RSF) or EMAC_RQ0_OPMODE.RTC in case of threshold mode.
    b. Flow control activation and de-activation thresholds for MTL receive FIFO (EMAC_RQ0_OPMODE.RFA and EMAC_RQ0_OPMODE.RFD).
    c. Error Packet and undersized good packet forwarding enable (EMAC_RQ0_OPMODE.FEP and EMAC_RQ0_OPMODE.FUP).
    d. Receive queue size (EMAC_RQ0_OPMODE.RQS).
5. Repeat previous two steps for all MTL transmit and receive queues selected in the configuration.

Initializing the MAC

The MAC configuration registers establish the operating mode of the MAC. These registers must be initialized before initializing the DMA.

The following MAC initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (the last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the receive FIFO and overflow.

1. Provide the required number of MAC address registers:EMAC_ADDR0_HI and EMAC_ADDR0_LO.
2. Program the following fields to set the appropriate filters for the incoming frames in the packet filter register:
    a. Receive all (EMAC_MACPKT_FILT.RA)
    b. Promiscuous mode (EMAC_MACPKT_FILT.PR)
    c. Hash or perfect filter (EMAC_MACPKT_FILT.HPF)
    d. Unicast, multicast, broadcast, and control frames filter settings

3. Program the following fields for proper flow control in the flow control register:
    a. Pause time and other pause frame control bits (EMAC_Q[i]_TXFLOW_CTL.PT, EMAC_Q[i]_TXFLOW_CTL.PLT)
    b. Transmit flow control bits (EMAC_Q[i]_TXFLOW_CTL.TFE)
    c. Flow control busy (EMAC_Q[i]_TXFLOW_CTL.FCB_BPA)
4. Program the EMAC_MAC_IEN register, as required, and if applicable, for the configuration.
5. Program the appropriate fields in the EMAC_MAC_CFG register. For example: Inter-packet gap (EMAC_MAC_CFG.IPG) while transmission and jabber disable (EMAC_MAC_CFG.JD).
6. Set bit 0 (EMAC_MAC_CFG.RE) and 1 (EMAC_MAC_CFG.TE) to start the MAC transmitter and receiver.
7. To support jumbo transmit/receive packets, configure the MAC configuration register as follows:
    a. Set the EMAC_MAC_CFG.JE field to 1
    b. Set the EMAC_MAC_CFG.JD and EMAC_MAC_CFG.WD fields to 1 to avoid giant packet error reporting
    c. Set the EMAC_MAC_CFG.GPSLCE field to 1
    d. Set the EMAC_MAC_EXT_CFG.GPSL field to a value > 9026
8. To support transmit/receive packets up to 16K, configure the MAC configuration register as follows:
    a. Set the EMAC_MAC_CFG.JD and EMAC_MAC_CFG.WD fields to 1 to avoid giant packet error reporting
    b. Set the EMAC_MAC_CFG.GPSLCE field to 1
    c. Set the EMAC_MAC_EXT_CFG.GPSL field to 16383

Performing Normal Transmit and Receive Operations

During normal operation of the EMAC, normal and transmit interrupts are read, descriptors polled, the DMA is suspended (when it does not own descriptors), and the values of the current host transmitter or receiver descriptor pointers are read for debugging.

For normal operations, use the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the host (either transmit or receive).
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.

    ADDITIONAL INFORMATION: When the descriptors are not DMA (or no descriptor is available), the DMA goes into the SUSPEND state. The transmission or reception can be resumed by freeing

the descriptors and writing the descriptor tail pointer to the transmit/receive tail pointer registers (EMAC_DMA[n]_TXDSC_TLPTR and EMAC_DMA[n]_RXDSC_TLPTR).

3. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process (EMAC_DMA[n]_TXDSC_CUR and EMAC_DMA[n]_RXDSC_CUR).
4. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (EMAC_DMA[n]_TXBUF_CUR and EMAC_DMA[n]_RXBUF_CUR).

Stopping and Starting Transmission

Use the following procedure to stop and start EMAC transfers.

1. Disable the MAC transmitter and receiver by clearing the EMAC_MAC_CFG.RE and EMAC_MAC_CFG.TE bits.
2. Ensure that both the transmit and receive queues are empty (EMAC_TQ0_DBG.TXQSTS and EMAC_RQ0_DBG.RXQSTS is 0).
3. Disable the receive DMA (whenapplicable), after ensuring that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of EMAC_RQ0_DBG register, EMAC_RQ0_DBG.PRXQ = 0 and EMAC_RQ0_DBG.RXQSTS = 00).
4. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

   ADDITIONAL INFORMATION: Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active. Similarly, do not change the DMA-related configuration when transmit and receive DMA are active.

Switching to New Descriptor List in Receive DMA

Switching to a new descriptor list is different in the receive DMA compared to the transmit DMA. Switching to a new descriptor list is permitted when the receive DMA is in SUSPEND state, as clarified by the following points:

- Generally, receive DMA prepares the descriptors in advance.
- If the receive DMA goes to the SUSPEND state due to the descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of a receive FIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the receive DMA to start using it immediately, without going into the STOP state.

Multi-Channel Multi-Queuing

Transmit

Use the following procedure for a transmit operation.

1. Program the transmit queue size in the EMAC_TQ0_OPMODE.TQS bit field. Based on the value programmed in this bit field, the size of the queue is determined. In the transmit operation, the number of channels is equal to the number of the queues. Due to this reason, the channel-to-queue mapping is fixed.
2. For a queue to be used, enable the queue using the EMAC_TQ0_OPMODE.TXQEN bit. Enable the EMAC_DMA[n]_TXCTL.ST bit and the corresponding EMAC_TQ0_OPMODE.TXQEN bit.
3. Program the scheduling method in the EMAC_MTL_OPMODE.SCHALG bit field.
4. Program the EMAC_TQ0_QTMWGT register for DCB queues as per the selected algorithm. For CBS algorithms in AVB queues, the EMAC_TQ[n]_ETS_CTL, EMAC_TQ[n]_SSCRDT, EMAC_TQ[n]_HICRDT and EMAC_TQ[n]_LOCRDT registers also need to be programmed as required.
5. If DCB is enabled and the PFC function is required, program the EMAC_TXQ_PRTY_MAP0 register to assign a fixed priority to the queue. This priority is used for determining if the corresponding queue should stop transmitting a packet based on the received PFC packet.

Receive

Use the following procedure for the receive operation.

1. Program the receive queue size in the EMAC_RQ0_OPMODE.RQS bit field. Based on the value programmed in this bit field, the size of the queue is determined.
2. Enable the receive queues 0 to 7 in the EMAC_RXQ_CTL0.RXQ0EN to EMAC_RXQ_CTL0.RXQ7EN bit fields for AV or DCB. In DMA configurations, enable the EMAC_DMA[n]_RXCTL.SR bit of statically or dynamically mapped EMAC_DMA[n]_RXCTL register and the corresponding EMAC_RXQ_CTL0.RXQ0EN to EMAC_RXQ_CTL0.RXQ7EN bits.
3. The MAC routes the receive packets to the receive queues based on following packet types:
   - AV PTP packets – based on the programming of the EMAC_RXQ_CTL1.PTPQ bit.
   - AV untagged control packets – based on the programming of the EMAC_RXQ_CTL1.AVCPQ bit.
   - Data Center Bridging (DCB) related Link Layer Discovery Protocol (LLDP) packets. Program the EMAC_RXQ_CTL1.DCBCPQ bits to indicate to the MAC which queue receives the DCB packets.
   - VLAN Tag Priority field in VLAN tagged packets – program the PSRQ7-0 bits in the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers for the routing of tagged packets based on the USP (user priority) field of the received packets to the receive queues 0 to 7.
   - The AV tagged control and data packets are also routed based on PSRQ field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The priorities set in PSRQ7-0 should be unique.
4. If multiple receive DMA channels are enabled, the following programming is required for proper arbitration and mapping:
   a. Program the EMAC_MTL_OPMODE.RAA bit field to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory.

b. Program the EMAC_RQ0_CTL register to decide the weights and the packet arbitration for each RxQ.

c. If static mapping is programmed in the EMAC_RXQ_DMA_MAP0/EMAC_RXQ_DMA_MAP1 registers (the DDMACH bits are reset to 0), the RXQx2DMA and others need to be programmed to select the channel for which each queue is mapped.

d. Set the EMAC_RXQ_DMA_MAP0 DDMACH bits to select dynamic mapping of packets in each receive queue.

e. In dynamic channel mapping, the routing of a packet to a specific receive DMA channel is decided by the value of the DCS field in the lowest MAC address register (for example EMAC_ADDR0_HI.DCS).

Recovering from DMA Channel Failure

When the DMA channel issues a bus error, follow these steps to recover from the failure.

Recovering from the Receive DMA Channel Failure

Follow these steps a bus error occurs in the receive DMA channel:

1. Set the EMAC_DMA[n]_RXCTL.RPF bit to 1. This flushes all the packets one after the other. This step is optional. However, setting this bit prevents HOL (head-of-line) blocking in the receive queues when packets sent to the receive DMA are stopped due to bus error.
2. Re-program the specific registers of the DMA channel.
3. Start the DMA channel.

Recovering from the Transmit DMA Channel Failure

Follow these steps if a bus error occurs in the transmit DMA channel:

1. Stop the specific DMA channel, even if it is in an active state.
2. Flush the corresponding MTL queue.
3. Re-program the specific registers of the DMA channel.
4. Start the DMA channel.

NOTE: Reprogramming the DMA channel registers might not always be successful in recovering from a bus error. If EMAC is not fully functional after reprogramming the DMA, as a workaround, issue a soft reset to recover from the bus error.

Disabling the Receive Queue

Follow these steps to disable specific receive queues

1. Disable the receiver by setting the EMAC_MAC_CFG.RE bit to 0.

2. Read the debug register (ensure that the EMAC_MAC_DBG.RPESTS and EMAC_MAC_DBG.RFCFCSTS fields are 0) and the EMAC_RQ0_DBG registers of all the receive queues.
3. When the EMAC_RQ0_DBG registers are read as 0, disable the intended receive queue by programming the corresponding fields in the EMAC_RXQ_CTL0 register.
4. Enable the receiver by setting the EMAC_MAC_CFG.RE bit to 1.

IEEE 1588 Time Stamping (PTP)

The following sections provide programming information for using for IEEE 1588 Time Stamping.

Initialization Guidelines for System Time Generation

Programs can enable the time stamp feature by setting the EMAC_TM_CTL.TSENA bit. However, it is essential that the time stamp counter be initialized after this bit is set. Complete the following steps during EMAC initialization:

1. Mask the time stamp trigger interrupt by clearing the EMAC_MAC_IEN.TSIE bit.
2. Set the EMAC_TM_CTL.TSENA bit 0 to enable time stamping.
3. Program the EMAC_TM_SUBSEC register based on the PTP clock frequency.
4. If the Fine Correction method is required, program the EMAC_TM_ADDEND register and set the EMAC_TM_CTL.TSADDREG bit.
5. Poll the EMAC_TM_CTL register until the EMAC_TM_CTL.TSADDREG bit is cleared.
6. Program the EMAC_TM_CTL.TSCFUPDT bit to select the Fine Update method (if required).
7. Program the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers with the appropriate time value.
8. Set the EMAC_TM_CTL.TSINIT bit.

   ADDITIONAL INFORMATION: The time stamp counter starts operation as soon as it is initialized with the value written in the time stamp update registers. If one-step time stamping is enabled:

   a. To enable one-step time stamping, program bit 27 of the TDES3 Context Descriptor.

   b. Program the EMAC_TM_INGASYM_CORR and EMAC_TM_EGASYM_CORR registers to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper time stamping.

If a time stamp operation is disabled by clearing the EMAC_TM_CTL.TSENA bit, repeat all these steps to restart the time stamp operation.

System Time Correction

The following sections provide programming information for system time correction.

Coarse Correction Method

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT).
2. Set the EMAC_TM_CTL.TSUPDT bit. The value in the time stamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

Fine Correction Method

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm explained in Fine Correction Method, calculate the rate by which to make the system time increments slower or faster.
2. Update the EMAC_TM_ADDEND register with the new value and set the EMAC_TM_CTL.TSADDREG bit.
3. Wait for the time for the new value of the Addend register to be active. You can do this by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers.
5. Enable the time stamp interrupt using the EMAC_MAC_IEN.TSIE bit.
6. Set bit 4 in the EMAC_TM_CTL register.
7. When this trigger causes an interrupt, read the EMAC_MAC_ISTAT register.
8. Reprogram the EMAC_MAC_ISTAT register with the old value and set the EMAC_TM_CTL.TSADDREG bit again.

PTP Offloading

The following sections provide programming information for using PTP offload.

Automatic Periodic Generation of PTP Sync Messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA, and EMAC_TM_CTL.TSEVNTENA bit fields to zero, one, and one respectively, to configure the node as ordinary or boundary requester (one, one, and one for transparent requester).
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to send in egress PTP Sync message.
3. Program the EMAC_PTO_CTL.ASYNCEN bit to enable periodic generation of PTP Sync messages.

4. Program the 80-bit Source Port Identity in theEMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to send in egress PTP Sync message.
5. Program the EMAC_LOG_MSG_INTVL.LSI bit field to program the periodicity of the PTP Sync messages. For example, value of 1 corresponds to 21 which translates to PTP Sync message every 2 seconds and value of 0xFF (twos complement of -1) corresponds to 2-1 which translates to PTP Sync message every 0.536 seconds.
6. Program the EMAC_MAC_IEN.TSIE bit to enable generation of time stamp interrupt.
7. Wait for a EMAC_STAT or EMAC_MAC interrupt generated when the EMAC_TM_STAT.TXTSSIS bit is set, which indicates that the time stamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers.

Periodic Generation of PTP Pdelay_Req Messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages.

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA, and EMAC_TM_CTL.TSEVNTENA bit fields to 1, 0, and 1 respectively to configure the node as transparent completer (1, 1, and 1 for transparent requester OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable PTP Offload feature and domain Number to send in egress PTP Pdelay_Req message.
3. Program the EMAC_PTO_CTL.APDREQEN bit to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in theEMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to send in egress PTP Pdelay_Req message.
5. Program the EMAC_LOG_MSG_INTVL.LMPDRI bit field to program the periodicity of the PTP Pdelay_Req messages. For example, value of 1 corresponds to 21 which translates to PTP Pdelay_Req message every 2 seconds and value of 0xFF (twos complement of -1) corresponds to 2-1 which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the EMAC_MAC_IEN.TSIE bit to enable generation of Timestamp Interrupt.
7. Wait for EMAC_STAT or EMAC_MAC interrupt generated on setting of the EMAC_TM_STAT.TXTSSIS bit is set, which indicates that the time stamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers.

Response Generation - Ordinary or Boundary Requester Mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Requester mode (periodic PTP sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA and EMAC_TM_CTL.TSEVNTENA bit fields to zero, one, and one respectively.

2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN field to enable the PTP offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR and EMAC_LOG_MSG_INTVL.LSI bit fields. The sum of both these fields is updated in the logMinDelayReqInterval field of the PTP Delay_Resp message.

Response Generation - Ordinary or Boundary Completer Mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary completer mode (PTP Delay_Req message generated in response to PTP Sync message).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA and EMAC_TM_CTL.TSEVNTENA bit fields to zero, zero, and one respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR bit field to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Response Generation - Transparent Completer Mode

Follow these steps to enable generation of PTP response messages for transparent completer mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA and EMAC_TM_CTL.TSEVNTENA bit fields to one, zero, and one respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.

4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR bit field and EMAC_LOG_MSG_INTVL.LMPDRI bit fields to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the EMAC_MAC_IEN.TSIE bit to enable generation of the timestamp interrupt.
6. Wait for the EMAC_STAT or EMAC_MAC interrupt that is generated when the EMAC_TM_STAT.TXTSSIS bit is set. This indicates that the timestamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers for egress PTP Pdelay_Req and Pdelay_Resp messages.

Response Generation - Transparent Requester Mode

Follow these steps to enable generation of PTP response messages for transparent requester mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA and EMAC_TM_CTL.TSEVNTENA bit fields to one, one, and one respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR, EMAC_LOG_MSG_INTVL.LSI and EMAC_LOG_MSG_INTVL.LMPDRI bit fields. The sum of EMAC_LOG_MSG_INTVL.DRSYNCR and EMAC_LOG_MSG_INTVL.LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the EMAC_MAC_IEN.TSIE bit to enable generation of the timestamp interrupt.
6. Wait for the EMAC_STAT or EMAC_MAC interrupt that is generated when the EMAC_TM_STAT.TXTSSIS bit is set. This indicates that the timestamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Response Generation - Peer-to-Peer Transparent Mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSREQENA and EMAC_TM_CTL.TSEVNTENA bit fields to 3, X, and X respectively.

2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
4. Program the EMAC_LOG_MSG_INTVL.LMPDRI bit field to indicate periodicity of the PTP Pdelay_Req messages.
5. Program the EMAC_MAC_IEN.TSIE bit to enable generation of the timestamp interrupt.
6. Wait for the EMAC_STAT or EMAC_MAC interrupt that is generated when the EMAC_TM_STAT.TXTSSIS bit is set. This indicates that the timestamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers for egress PTP Pdelay_Req and Pdelay_Resp messages.

Using the Audio Video Feature

The following sections provide procedures to program audio video features.

Initializing the DMA in Audio Video Feature

1. Provide a software reset to reset all internal registers and logic (EMAC_DMA_MODE.SWR bit).
2. Wait for the completion of the reset process. Poll the EMAC_DMA_MODE.SWR bit, which is cleared only after the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in the EMAC_DMA_MODE register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the transmit and receive descriptors.

   ADDITIONAL INFORMATION: When OSF mode is used, at least two transmit descriptors are required. For more information about descriptors, see the DMA Descriptors section.
5. Ensure that the software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the transmit and receive ring length registers (EMAC_DMA[n]_TXDSC_RLEN and EMAC_DMA[n]_RXCTL2.RDRL). The programmed ring length must be at least 4.
7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (EMAC_DMA[n]_TXDSC_ADDR and EMAC_DMA[n]_RXDSC_ADDR). In addition, program the transmit and receive tail pointer registers that indicate to the DMA the available descriptors (EMAC_DMA[n]_TXDSC_TLPTR and EMAC_DMA[n]_RXDSC_TLPTR).
8. Program the following fields to initialize the mode of operation in the EMAC_TQ0_OPMODE register:
   a. Transmit store and forward (EMAC_TQ0_OPMODE.TSF)

b. Transmit threshold control (EMAC_TQ0_OPMODE.TTC)

c. Transmit queue enable (EMAC_TQ0_OPMODE.TXQEN) to value 2'b10 to enable transmit queue0

d. Transmit queue size (EMAC_TQ0_OPMODE.TQS)

9. Enable the interrupts by programming the EMAC_DMA[n]_IEN register.
10. Repeat steps 4 through 9 for all additional channels of AV feature.
11. Program the EMAC_TQ[n]_ETS_CTL register, idleSlope (EMAC_TQ0_QTMWGT), sendSlope (EMAC_TQ[n]_SSCRDT, hiCredit (EMAC_TQ[n]_HICRDT), and loCredit (EMAC_TQ[n]_LOCRDT) registers of the AVQueues.
12. Start the receive and transmit DMA by setting the EMAC_DMA[n]_TXCTL.ST bit and the EMAC_DMA[n]_RXCTL.SR bit.

Enabling Slot Number Checking

The program enables the ability to report the average bits that are transmitted in a slot. The EMAC_TQ0_ETS_STAT register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting.

1. Enable time stamping by following the steps described in the "Initialization Guidelines for System Time Generation."
2. Program the EMAC_TQ[n]_ETS_CTL.SLC bits [6:4] of a channel with the number of slots over which the average transmitted bits per slot need to be computed.

   ADDITIONAL INFORMATION: The frequency of this interrupt depends on the value programmed in this step. For example, when the value 0 is used in the EMAC_TQ[n]_ETS_CTL.SLC field, the interrupt is generated at every 125 microseconds. When not required, disable this interrupt to stop the interrupt flooding.
3. Enable the EMAC_Q0_INT_CTLSTAT.ABPSIE bit of a channel to generate the average bits per slot interrupt.
4. Read the EMAC_TQ0_ETS_STAT.ABS bits [16:0] of a channel on each interrupt.

   ADDITIONAL INFORMATION: The software can read the EMAC_TQ0_ETS_STAT.ABS bits in polling mode even if the EMAC_Q0_INT_CTLSTAT.ABPSIE bit is not enabled. When high, bit 1 (EMAC_Q0_INT_CTLSTAT.ABPSIS) indicates that a new value is updated in the ABS field.

Disabling Flow Control for AV Enabled Queues

Receive Flow

Program the EMAC_RQ0_OPMODE.EHFC (enable hardware flow control) bit of the corresponding receive queue's operation mode register to 0.

Transmit Flow

Program the EMAC_TXQ_PRTY_MAP0.PSTQ0 through EMAC_TXQ_PRTY_MAP1.PSTQ7 bit fields corresponding to the AV enabled transmit queue (using the EMAC_TXQ_PRTY_MAP0 and EMAC_TXQ_PRTY_MAP1 registers) to 0.

Programming Energy Efficient Ethernet

The following sections provide procedures to program energy efficient Ethernet.

Entering and Exiting the Transmit LPI Mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of Physical layers to operate in the Low-Power Idle (LPI) mode. In the transmit path, the software must set the EMAC_LPI_CTLSTAT.LPIEN bit to indicate to the MAC to stop transmission and initiate the LPI protocol. Follow these steps for entering/exiting the transmit LPI Mode:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop receive clock in LPI mode.)
3. Program the EMAC_LPI_TMRSCTL.LST and EMAC_LPI_TMRSCTL.TWT bit fields.
4. Read the link status of the PHY chip by using the MDIO interface and update the EMAC_LPI_CTLSTAT.PLS bit accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program the EMAC_ONEUS_TIC_CNT register as per the frequency of SCLK0.
6. Program the EMAC_LPI_ENTR_TMR.LPIET bit field with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set the EMAC_LPI_CTLSTAT.LPIATE and EMAC_LPI_CTLSTAT.LPITXA to enable the auto-entry into LPI and auto-exit of MAC from LPI state.
8. Set the EMAC_LPI_CTLSTAT.LPIEN bit to make the MAC transmitter enter the LPI state.

   The MAC enters the LPI mode after completing all scheduled packets and remains IDLE for the time indicated by the EMAC_LPI_ENTR_TMR.LPIET bit field. It sets the EMAC_LPI_ENTR_TMR.LPIET bit [0] after entry to the LPI state.

   When a packet is scheduled for transmission (when the transmit DMA comes out of the IDLE state), the MAC transmitter exits the LPI state automatically. It waits for TWT time before setting the EMAC_LPI_CTLSTAT.TLPIEX interrupt status bit and then resumes the packet transmission.
9. The MAC transmitter re-enters the LPI state if it remains IDLE for EMAC_LPI_ENTR_TMR.LPIET time and sets the EMAC_LPI_CTLSTAT.LPIEN bit and the entry-exit cycle continues.

10. Reset the EMAC_LPI_CTLSTAT.TLPIEX bit in case the application wants to over-ride the auto-entry/exit modes and make the MAC transmitter exit the LPI state directly.

To ensure MAC enter the LPI state only after completion of transmission of all the queued frames in the Tx FIFO, set the EMAC_LPI_CTLSTAT.LPITXA bit [19].

Programming Flexible Pulse-Per-Second Output

The following sections provide procedures to program Flexible Pulse-Per-Second Output.

Generating Single Pulse on PPS

To program single pulse on PPS, use the following procedure.

1. Program 11 or 10 (for interrupt) using the EMAC_PPS_CTL.TRGTMODSEL0 bits [6:5]. This instructs the MAC to use the target time registers (EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC) for the start time of PPS signal output.
2. Program the start time value in the target time registers (EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC).
3. Program the width of the PPS signal output in the EMAC_PPS[n]_WID register.
4. Program the EMAC_PPS_CTL.PPSCMD1 bits [3:0] to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the target time registers.

Generating Next Pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), the program can cancel the pulse generation by giving the cancel start command (PPSCMD = 0011) before the programmed start time elapses. The behavior of the next pulse can be programmed in advance. To program the next pulse, follow this procedure.

1. Program the start time for the next pulse in the target yime registers. This time should be more than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in the EMAC_PPS[n]_WID register.
3. Program the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits [3:0], PPSCMD to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs the MAC to generate single pulse on the PPS signal output, at the time programmed in target time registers. When this command executes before the previous pulse becomes low, then the new command overwrites the previous command and the EMAC may generate only one extended pulse.

Generating a Pulse Train on PPS

The EMAC_PPS_CTL.TRGTMODSEL0 bits program the target time registers to do any one of the following:

- Generate only interrupts.

- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

Complete the following steps to program the target time registers to generate only interrupt event.

1. Program the EMAC_PPS_CTL.TRGTMODSEL0 bits = 00 (for interrupt). This instructs the MAC to use the target time registers for target time interrupt.
2. Program a target time value in the target time registers. This instructs the MAC to generate an interrupt when the target time elapses. If the mode select bits [6:5] are changed (for example, to control the PPS), then the interrupt generation is over-written with the new mode and new programmed target time register value.

The TSTARGTERR[n] interrupt is generated when the programmed target time is smaller (that is, time in the past) compared to system time in the EMAC_TM_SEC and EMAC_TM_NSEC registers. Therefore, to avoid unwanted TSTARGTERR[n] interrupt, the correct writing order is as follows:

1. EMAC_PPS[n]_TGTM_NSEC
2. EMAC_PPS[n]_TGTM_SEC
3. EMAC_PPS[n]_INTVL
4. EMAC_PPS[n]_WID
5. EMAC_PPS_CTL.PPSCTRL_PPSCMD and EMAC_PPS_CTL.PPSEN0

Programming TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet. Use the following steps to program TSO.

1. Program the EMAC_DMA[n]_TXCTL.TSE bit to enable TCP packet segmentation in that DMA channel.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a. Enable the TSE in bit 18 of the TDES3 register
   b. Program the length of the un-segmented TCP/IP packet payload in bits [17:0] of the TDES3 register and the TCP header in bits [22:19] of TDES3.
   c. Program the maximum size of the segment using the EMAC_DMA[n]_CTL.MSS bits or MSS in the context descriptor. If the MSS field is programmed in both the DMA channel control register and in the context descriptor, the latest software programmed sequence is considered.
3. The header of the unsegmented TCP/IP packet should be in buffer 1 of the first descriptor and this buffer must not hold any payload bytes. The payload is allocated to buffer 2 and the buffers of the subsequent descriptors.

ADDITIONAL INFORMATION: If the TSE bit is enabled in the TDES3 register for a non-TCP-IP packet, the result is unpredictable.

Programming UFO

EMAC supports fragmentation of UDP packets into smaller IPv4 fragments. Enable the IP fragmentation by programming the `EMAC_DMA[n]_TXCTL.TSE_MODE` bits [14:13] as follows:

- Set to 2'b01 to enable fragmentation of UDP over IPv4 with checksum
- Set to 2'b10 to enable fragmentation of UDP over IPv4 without checksum

NOTE: In both the cases, if the `EMAC_DMA[n]_TXCTL.TSE` bit [12] is set to 1 for a TCP packet, DMA enables TCP segmentation; fragmentation is not enabled, because fragmentation is supported only for UDP packets.

Software Guidelines

When the software creates Transmit Descriptor of the UDP/IPv4 packet that is to be fragmented, the software must:

1. Set the desired fragment size (MFS) in the MSS field of TDES2 of the Context Descriptor. The MFS must be set only once, unless the MFS needs to be changed (for TCP packet targeted for segmentation). The MFS must be a multiple of 8 bytes for IP fragmentation.
2. Set the TSE bit of TDES3 of normal descriptor.
3. Program the THL = 2 in TDES3 of normal descriptor.
4. Program the correct value in the TPL field of TDES3 of normal descriptor. This value should be equal to length of UDP Header + UDP payload.

NOTE:
- Enable fragmentation only for IPv4; not for IPv6.
- Set the DF flag (do not fragment) in the IPv4 header packet to 0.
- Set the fragmentation offset in the IPv4 header packet to 0.

When the input packet is UDP over IPv6, do not enable IP fragmentation. Set the TSE bit to 0 in the TDES3 of normal descriptor and let the software perform the fragmentation.

Programming Header Payload Split Receive

The Header-Payload Split support can be enabled by

1. Setting the `EMAC_DMA[n]_CTL.SPH` bit field.
2. Selecting the Header-Payload split option by setting the `EMAC_SPLM_OFST_CFG.SPLM` bit field.

The DMA can process the header and payload of the received packets separately.

NOTE:
- In case of L3/L4 header-payload split, if the IPv4 header length or IPv6 payload length fields are corrupt, the split might not always happen at the correct boundary.
- When the EMAC_DMA[n]_CTL.SPH bit field is set to 1, buffer 1 of only the first descriptor is used with buffer size indicated by the EMAC_MAC_EXT_CFG.HDSMS bit field.
- Buffer 2 of the first descriptor and subsequent descriptors are used with buffer size indicated by the EMAC_DMA[n]_RXCTL.RBSZ_13_Y/EMAC_DMA[n]_RXCTL.RBSZ_X_0 bit fields.
- When the HL field of the first descriptor (RDES2) is zero, it indicates packet is not split into header and payload.
- When the HL field of the first descriptor (RDES2) is non-zero, it indicates that the packet is split into header and payload.

Programming VLAN Filtering on Receive

Use the following steps to program VLAN filtering on receive:

1. Program the EMAC_VLANTAG_CTL register as follows to select the filtering method.
   a. EMAC_VLANTAG_CTL.ETV – Enable 12-bit VLAN tag comparison or 16-bit VLAN tag comparison.
   b. EMAC_VLANTAG_CTL.VTHM – VLAN tag hash table match enable.
   c. EMAC_VLANTAG_CTL.ERIVLT – Enable inner VLAN tag or outer VLAN tag (to enable the inner or outer VLAN tag filtering, enable double VLAN processing by setting the EMAC_VLANTAG_CTL.EDVLP bit)
   d. EMAC_VLANTAG_CTL.ERSVLM – Enable receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set EMAC_VLANTAG_CTL.ESVL ESVL)
   e. EMAC_VLANTAG_CTL.DOVLTC – Ignores VLAN type for tag match
   f. EMAC_VLANTAG_CTL.VTIM – Enable VLAN tag inverse match instead of the normal VLAN tag matching
2. Program the EMAC_VLANTAG_CTL.ETV bit field to select 12-bit or 16-bit VLAN tag.
3. If hash filtering of VLAN tag is enabled, program the EMAC_VLAN_HASHTBL register.

   ADDITIONAL INFORMATION: When the EMAC_VLANTAG_CTL.ETV bit is reset, upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the EMAC_VLAN_HASHTBL register. When the ETV bit is set, upper 4 bits of calculated CRC-32 of VLAN tag are used to index the content of the EMAC_VLAN_HASHTBL register. For example, when the EMAC_VLANTAG_CTL.ETV bit is set, a hash value of 4b'1000 selects Bit 8 of the VLAN hash table. When the ETV bit is reset a hash value of 4'b1000 selects bit 7 of the VLAN hash table.

Programming Extended VLAN Filtering and Routing on Receive

For the indirect access of the per VLAN tag registers, follow these steps:

Write

1. Write the required data into the EMAC_VLANTAG_DAT register.
2. Program the EMAC_VLANTAG_CTL.OFS bit field with the required filter register's offset and command type to the EMAC_VLANTAG_CTL.CT bit field. For a write command, set this bit to 0.
3. Write 1 to the EMAC_VLANTAG_CTL.OB bit field and wait until the bit is reset before performing the next write. This is to ensure that the appropriate VLAN tag filter register has been programmed.

Read

1. Program the EMAC_VLANTAG_CTL.OFS bit field with the required register's offset and command type to the EMAC_VLANTAG_CTL.CT bit field. For a read command, set this bit to 1.
2. Write 1 to the EMAC_VLANTAG_CTL.OB bit field and wait until the bit is reset. The appropriate VLAN tag filter register's value is available in the EMAC_VLANTAG_DAT register.

Programming Queue/Channel Based VLAN Inclusion Registers

The following sections provide the procedures for reading and writing the Queue/Channel based VLAN inclusion register.

Writing to the Indirect Registers

To write to the queue/channel based VLAN inclusion register, use the following steps.

1. Set the EMAC_VLAN_INCL.CBTI bit to 1, to enable queue/channel based VLAN tag insertion on all transmitted packets.

   ADDITIONAL INFORMATION: This bit must be set before any indirect access to the queue/channel specific MAC_VLAN_INCL[n] register.
2. Set the EMAC_VLAN_INCL.RDWR bit field to 1, to indicate a write access.
3. Program the VLAN tag and VLAN type to be inserted in packets from a particular queue/channel using the EMAC_VLAN_INCL.VLT and EMAC_VLAN_INCL.CSVL bit fields. The corresponding offset address in the EMAC_VLAN_INCL.ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on) must be set.
4. Set the EMAC_VLAN_INCL.RDWR bit to 1 to indicate write access.

   ADDITIONAL INFORMATION: The write to byte 0 (byte 3 in Big Endian mode) of the EMAC_VLAN_INCL register initiates access to indirect access MAC_VLAN_INCL[n] register.
5. The EMAC_VLAN_INCL.BUSY bit is set by EMAC to indicate the progress of access to indirect access MAC_VLAN_INCL[n] register. On completion of the access, the EMAC_VLAN_INCL.BUSY bit is cleared.

The application must not attempt subsequent accesses to the MAC_VLAN_INCL[n] register when this bit is one.

6. Repeat step 3 through 5 to insert VLAN tag and VLAN type in packets from the remaining queues/channels. The application must ensure that the required VLAN tag and VLAN type for all the queues/channels are programmed; otherwise unintended VLAN tag and VLAN type might be inserted.

Reading the Indirect Registers

To read the queue/channel based VLAN inclusion register, complete the following steps:

1. Set the EMAC_VLAN_INCL.CBTI bit to 1, to enable queue/channel based VLAN tag insertion on all the transmitted packets. This bit must be set before indirect access to the queue/channel specific MAC_VLAN_INCL[n] register.
2. Program the read offset address in the EMAC_VLAN_INCL.ADDR ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on). Set the EMAC_VLAN_INCL.RDWR bit to 0, to indicate read access.
3. The EMAC_VLAN_INCL.BUSY field is set by EMAC to indicate the progress of access to the indirect access MAC_VLAN_INCL[n] register. On completion of the access, the BUSY field is cleared. The bits [15:0] and bit [19] of the EMAC_VLAN_INCL register contains the VLAN tag and VLAN type respectively, from the corresponding queue/channel/offset address.
4. Repeat step 2 and step 3 to read VLAN tag and VLAN type from the remaining queues/channels.

Programming EST

Program the gate control values and time intervals in the Software Owned Gate Control List (SWOL) along with the other EST related registers to appropriate values. The following sub-sections provide step by step details for programming the GCL and the other EST related registers.

GCL and GCL Linked Registers

Follow these steps to program the Gate Control List (GCL) and the four other registers implemented per GCL.

The GCL and the four other GCL-linked registers are accessed through indirect addressing using the EMAC_EST_GCL_CTL and EMAC_EST_GCL_DAT registers. The EMAC_EST_STAT.SWOL bit field indicates if GCL0 or GCL1 is owned by software.

1. Write the 32-bit write data to EMAC_EST_GCL_DAT register. Then program the EMAC_EST_GCL_CTL register to write the write address and other control information.

   ADDITIONAL INFORMATION: In the EMAC_EST_GCL_DAT register, write data consists of up to 8 bits (configurable) of gate controls and up to 24 bits (configurable) of time interval. Gate close is indicated by programming a 0 and gate open is indicated by programming a 1. For a 4-TC and 20-bit time interval configuration, the data width is 24 bits and the remaining 8 bits are reserved/read-only. Write the data in the following format: {8'h0, TC3, TC2, TC1, TC0, 20-bit Time Interval} where TCx = 0 or 1.

2. Program the EMAC_EST_GCL_CTL.SRWO bit to 1 (to start a write op) and program the address and R/W fields appropriately.
3. Poll and check for the EMAC_EST_GCL_CTL.SRWO bit to be cleared by hardware to indicate the completion of the previous operation before initiating a new R/W operation using the same indirect addressing mode.
4. Repeat steps 3, 4, 5 until the programming of the GCL is completed.
5. Using the indirect addressing method, program the BTR, CTR, TER and LLR registers. Set the EMAC_EST_GCL_CTL.GCRR bit appropriately. This bit interprets the address field as belonging to these registers (instead of the GCL).
6. After programming of the GCL and the related registers, program the EMAC_EST_CTL register to allow hardware to own and process the GCL.

   ADDITIONAL INFORMATION: When the list length (as indicated in LLR) is 1, the associated time interval should be smaller than the value of the cycle time register. Otherwise, an error is reported (as detailed in the Error handling section) as a single set of gate controls add no value in the TSN context.

   ADDITIONAL INFORMATION: The time unit in all the GCL related registers is seconds and nanoseconds. In cases where internally generated PTP System Time is used, the nano-seconds field must be programmed to use the digital rollover mode (EMAC_TM_CTL.TSCTRLSSR bit field must be set to 1).

EST Registers

After completing the steps described in GCL and GCL Linked Registers, program the EMAC_EST_CTL register.

1. Set the current time offset value (EMAC_EST_CTL.CTOV and time internal left shift fields (EMAC_EST_CTL.TILS appropriately. Also, set the enable EST (EMAC_EST_CTL.EEST) and switch to SWOL (EMAC_EST_CTL.SSWL) bits.

   This enables EMAC to own and process the new GCL and switch to the new GCL at the BTR value. If enabled, EMAC generates an interrupt when the switch to the new list occurs.

   Software must address any other interrupts received during the hardware execution of the GCL
2. Set the EMAC_EST_CTL.SSWL bit field to handoff to the controller.

   The EMAC_EST_CTL.SSWL is reset/cleared by the controller when it successfully switches to the new list. The controller also flips the EMAC_EST_CTL.SSWL bit to indicate the new GCL that the software owns. To install a new GCL, program the GCL it owns (indicated by SSWL bit) as described in GCL and GCL Linked Registers.
3. Program the EMAC_EST_CTL register as described. Ensure that the new BTR is set to an appropriate value to avoid BTR error that might need software intervention in some cases.

   ADDITIONAL INFORMATION: To avoid transmission overruns, the packet length (frame size) information should always be available. Therefore, in the DMA configurations, program the packet length in the first

descriptor of every transmit frame. Similarly, in the MTL configuration, provide the packet length in the control word.

Programming Launch Time in Time-Based Scheduling

The launch time is programmed in the enhanced normal transmit descriptors in DMA configurations and is driven as a control word in MTL configurations as follows.

The OSTC and launch time features are mutually exclusive and should not be used together. If a context descriptor is received with a valid OSTC value immediately before receiving a first normal descriptor with the LTV bit set, then the LTV is ignored.

Programming Media Clock Generation and Recovery

The following sections provide procedures for programming the media clock.

Media Clock Generation

1. Program the appropriate presentation time control (supported generation modes "1001-1011") to the `EMAC_PPS_CTL.PPSCTRL_PPSCMD` (for 0th instance) and `EMAC_PPS_CTL.PPSCMD1` through `EMAC_PPS_CTL.PPSCMD3` bits. Program all the `EMAC_PPS_CTL.MCGREN0` through `EMAC_PPS_CTL.MCGREN3` bits to enable MCGR mode.

   ADDITIONAL INFORMATION: The application must drive the appropriate trigger signal to the corresponding `EMAC_PTPAUX_MCG_IN[n]` pin, for the instances enabled for media clock generation.

2. Based on the programmed mode, MCGR instance[n] captures the time stamp and programs it in the `EMAC_PPS[n]_TGTM_SEC` register. Then, the trigger requester event EMAC_MCGR[n] is generated.

   For every request generated by the MCGR instance n, EMAC reads the corresponding `EMAC_PPS[n]_TGTM_SEC` register and acknowledges the read request by accepting the corresponding completer trigger request.

Media Clock Recovery

1. Enable the Current Presentation Time (CPT) counter by setting the `EMAC_TM_CTL.PTGE` bit field. In addition to programming the initialization values for the system time, update the `EMAC_PRSNTM_UPDT` bits with the equivalent presentation time initial value. This is a requirement for the `EMAC_TM_CTL.TSINIT` bit field to be set.

   ADDITIONAL INFORMATION: The increment values used for the system time is also used for the current presentation time. This is because, the increment value is in Sub-seconds and Sub-nanoseconds.

2. Along with programming the update values for the system time, update the `EMAC_PRSNTM_UPDT` register with the equivalent presentation time update value (32-bit ns) when an update is needed. Finally, set the `EMAC_TM_CTL.TSUPDT` bit field ensure update is completed.

3. Program the appropriate presentation time control (supported recovery modes "0001 - 0011") to the EMAC_PPS_CTL.PPSCTRL_PPSCMD bit field (for 0th instance) and EMAC_PPS_CTL.PPSCMD1 through EMAC_PPS_CTL.PPSCMD3 bits. Program all the EMAC_PPS_CTL.MCGREN0 through EMAC_PPS_CTL.MCGREN3 bits to enable MCGR mode.
4. For instances enabled for media clock recovery, the corresponding EMAC_MCGR[n] trigger signal is set by the EMAC. For every request generated by EMAC, program the target presentation time into the EMAC_PPS[n]_TGTM_SEC register and acknowledge the request by asserting the corresponding trigger completer signal. Acknowledgment can be given before/after programming the target presentation time.
5. Observe the recovered media clock on the corresponding EMAC_PTPPPS[n] pin.

EMAC Programming Concepts

The following sections provide basic information and guidelines to help with programming the EMAC module.

IEEE 802.3 Ethernet Packet Structure

The ***IEEE 802.3 Frame Structure*** table provides typical frame format of an Ethernet packet. Refer to the IEEE standards for detailed information on Ethernet packets and their format.

Table 28-109: IEEE 802.3 Frame Structure

Parameter	Description	Position in Ethernet Packet	Total Bytes
PREAMBLE	This parameter is a 56-bit (7-byte) pattern of alternating 1 and 0 bits (#10101010), which allows devices on the network to detect a new incoming frame for synchronization.	1	7
SFD	The SFD (#10101011) is a 1-byte pattern designed to break the preamble pattern, and signal the start of the actual frame.	2	1
DA	48-bit destination address. This parameter can be a unicast, multicast, or broadcast address.	3	6
SA	48-bit long source address, typically a unicast, multicast, or broadcast address.	4	6
LT	Typically this field is the length, in terms of the number of bytes, and can be anywhere between 0–1500. When the value is greater than or equal to 0x0600, this field also indicates the type of special payload carried by the frame. Examples include 0x8808 for flow control and 0x0800 for IPv4.	5	2
DATA	Actual application data payload, usually between 0–1500.	6	0–1500
PAD	This field compensates for data frames that are shorter than 64 bytes long, not including the preamble.	7	0–46
FCS	The frame check sequence is a 32-bit cyclic redundancy check that detects corrupted data within the entire frame. This parameter is generated from a CRC-32 polynomial code (CRC-32-IEEE): G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1.	8	4

Frame Size Statistics for Application Software

Table 28-110: Ethernet Frame Size Statistics

Frame size statistics	VLAN-specific change Comments	
Information bytes/Header	4 byte 802.1Q header inserted after source address and before Type/LAN in 802.3 packets = 22 bytes.	6 x 2 + 2 + 4 = 18 bytes (DA+SA+LT+FCS)
Minimum Frame Size (typical)	If DATA is NULL, 42-byte padding makes 64 bytes (42 +22)	64 bytes. If DATA is NULL, 46-byte padding makes 64 bytes (46 +18)
Maximum Frame Size (typical)	1522 bytes	1518 bytes (1500 bytes DATA and 18-bytes header)
Jumbo Frame Size	9022 bytes	Typical industry standard. Ethernet jumbo frame size treated as 9018 bytes.

Software Visualization of Programmable Packet Size

The *Visualization of Programmable Packet Size* table provides the byte sizes of packets with various configurations.

Table 28-111: Visualization of Programmable Packet Size

Size in Bytes	Comments
16384	Receive watchdog and transmit jabber disabled, jumbo frames enabled.
10240	Receive watchdog and transmit jabber disabled, jumbo frames disabled.
2048	Receive watchdog and transmit jabber enabled.
1518	Typical max size of Ethernet frame. Receive watchdog and transmit jabber enabled.
64	Typical minimum size of Ethernet frame.
< 64	Runt frames requiring Zero-PAD.

Ethernet Packet Structure in C

The following is an example for Ethernet packet structure in the C language.

```
typedef struct ETHER_PACKET
{
    char  dst_addr[6];             //destination address
    char  src_addr[6];             //source address
    char  length[2];               //length of actual data
    char  data[DATA_SIZE];         //application data
    char  fdlimit[DELIMIT_SIZE];   //32-bit delimit (if manual appending)
    char  fcs[4];                  //crc frame checksum, used by RX buffer.
} ETHER_PACKET;
```

PTP Header Structure in C

The following code is an example of the PTP message format.

```
/* PTP Message Format (Refer to PTP Frame Over IPv4)*/
    typedef struct EMAC_PTP_HEADER
    {
    unsigned char    messageType:4,    //PTP Version 2 message type
    transportSpecific:4;
    unsigned char    versionPTP;       //PTP Version (1 or 2)
    unsigned short    messageLength;
    unsigned char    domainNumber;
    unsigned char    RESERVED1;
    unsigned short    flagField;
    unsigned char    correctionField[8];
    unsigned char    RESERVED2[4];
    unsigned char    sourcePortIdentity[10];
    unsigned short    sequenceid;
    unsigned char    controlField;     //PTP Version 1 message type
    unsigned char    logMessageInterval;
    }EMAC_PTP_HEADER;
```

ADSP-2184x EMAC Register Descriptions

RGMII, RMII, MII Ethernet Controller (EMAC) contains the following registers.

Table 28-112: ADSP-2184x EMAC Register List

Name	Description
EMAC_DMA[n]_CTL	DMA Channel n Control Register
EMAC_DMA[n]_RXBUF_CUR	DMA Channel n Current Application Receive Buffer Address Register
EMAC_DMA[n]_RXDSC_CUR	DMA Channel n Current Application Receive Descriptor Register
EMAC_DMA[n]_TXBUF_CUR	DMA Channel n Current Application Transmit Buffer Address Register
EMAC_DMA[n]_TXDSC_CUR	DMA Channel n Current Application Transmit Descriptor Register
EMAC_DMA[n]_IEN	DMA Channel n Interrupt Enable Register
EMAC_DMA[n]_MISSFRM_CNT	DMA Channel n Missed Frame Counter Register
EMAC_DMA[n]_RXDSC_ADDR	DMA Channel n Receive Descriptor List Address Register
EMAC_DMA[n]_RXDSC_TLPTR	DMA Channel n Receive Descriptor Tail Pointer Register
EMAC_DMA[n]_RXCTL	DMA Channel n Receive Control Register
EMAC_DMA[n]_RXCTL2	DMA Channel n Extended Receive Control Register
EMAC_DMA[n]_RXINTWDTMR	DMA Channel n Receive Interrupt Watchdog Timer Register
EMAC_DMA[n]_SFN_CTLSTAT	DMA Channel n Slot Function Control and Status register
EMAC_DMA[n]_STAT	DMA Channel n Status Register
EMAC_DMA[n]_TXDSC_ADDR	DMA Channel n Transmit Descriptor List Address Register
EMAC_DMA[n]_TXDSC_RLEN	DMA Channel n Transmit Descriptor Ring Length Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_DMA[n]_TXDSC_TLPTR	DMA Channel n Transmit Descriptor Tail Pointer Register
EMAC_DMA[n]_TXCTL	DMA Channel n Transmit Control Register
EMAC_DMA_DBG_STAT0	DMA Debug Status 0 Register
EMAC_DMA_DBG_STAT1	DMA Debug Status 1 Register
EMAC_DMA_DBG_STAT2	DMA Debug Status 2 Register
EMAC_DMA_ISTAT	DMA Interrupt Status Register
EMAC_DMA_MODE	DMA Mode Register
EMAC_DMA_SYSBMODE	DMA System Bus Mode Register
EMAC_DMA_TBS_CTL0	DMA TBS Control 0 Register
EMAC_DMA_TBS_CTL1	DMA TBS Control 1 Register
EMAC_DMA_TBS_CTL2	DMA TBS Control 2 Register
EMAC_DMA_TBS_CTL3	DMA TBS Control 3 Register
EMAC_ONEUS_TIC_CNT	LPI One Microsecond Tic Counter Register
EMAC_ARP_ADDR	Address Resolution Protocol (ARP) Register
EMAC_ADDR0_HI	MAC Address 0 High Register
EMAC_ADDR0_LO	MAC Address 0 Low Register
EMAC_ADDR[i]_HI	MAC Address n High Register
EMAC_ADDR[i]_LO	MAC Address n Low Register
EMAC_TM_AUX_CTL	Auxiliary Time Stamp Control Register
EMAC_TM_AUXTS_NSEC	Auxiliary Time Stamp Nanoseconds Register
EMAC_TM_AUXTS_SEC	Auxiliary Time Stamp Seconds Register
EMAC_CSR_SW_CTL	CSR Software Control Register
EMAC_MAC_CFG	MAC Configuration Register
EMAC_MAC_DBG	Debug Register
EMAC_SPLM_OFST_CFG	MAC Split Mode and Offset Configuration Register
EMAC_MAC_EXT_CFG	MAC Extended Configuration Register
EMAC_FPE_CTLSTAT	Frame Preemption Control and Status Register
EMAC_HW_FTR_PSNT0	Hardware Feature Presence Register 0
EMAC_HW_FTR_PSNT1	Hardware Feature Presence Register 1
EMAC_HW_FTR_PSNT2	Hardware Feature Presence Register 2
EMAC_HW_FTR_PSNT3	Hardware Feature Presence Register 3

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_MAC_HW_FEATURE4	MAC Hardware Feature Presence Register 4
EMAC_HASHTBL_REG0	Hash Table Register 0
EMAC_HASHTBL_REG1	Hash Table Register 1
EMAC_HASHTBL_REG2	Hash Table Register 2
EMAC_HASHTBL_REG3	Hash Table Register 3
EMAC_HASHTBL_REG4	Hash Table Register 4
EMAC_HASHTBL_REG5	Hash Table Register 5
EMAC_HASHTBL_REG6	Hash Table Register 6
EMAC_HASHTBL_REG7	Hash Table Register 7
EMAC_INDR_ACC_CTL	Indirect Access Control Register
EMAC_INDR_ACC_DAT	Indirect Access Data Register
EMAC_INVLAN_INCL	Inner VLAN Tag Inclusion or Replacement Register
EMAC_MAC_IEN	MAC Interrupt Enable Register
EMAC_MAC_ISTAT	MAC Interrupt Status Register
EMAC_L3L4_CTL7	Layer 3 and Layer 4 Control Register
EMAC_L3L4_CTL[i]	Layer 3 and Layer 4 Control Register
EMAC_LPI_CTLSTAT	LPI Control and Status Register
EMAC_LPI_ENTR_TMR	Transmit LPI Entry Timer Register
EMAC_LPI_TMRSCTL	LPI Timers Control Register
EMAC_L3_ADDR0_REG7	Layer 3 Address 0 Register 7
EMAC_L3_ADDR0_REG[i]	Layer 3 Address 0 Register [i]
EMAC_L3_ADDR1_REG7	Layer 3 Address 1 Register 7
EMAC_L3_ADDR1_REG[i]	Layer 3 Address 1 Register [i]
EMAC_L3_ADDR2_REG7	Layer 3 Address 2 Register 7
EMAC_L3_ADDR2_REG[i]	Layer 3 Address 2 Register [i]
EMAC_L3_ADDR3_REG7	Layer 3 Address 3 Register 7
EMAC_L3_ADDR3_REG[i]	Layer 3 Address 3 Register [i]
EMAC_L4_ADDR7	Layer 4 Address Register [i]
EMAC_L4_ADDR[i]	Layer 4 Address Register [i]
EMAC_LOG_MSG_INTVL	PTP Log Message Interval Register
EMAC_MDIO_ADDR	MDIO Address Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_MDIO_DATA	MDIO Data Register
EMAC_PHYIF_CTLSTAT	PHY Interface Control and Status Register
EMAC_PPS[n]_INTVL	PPS n Interval Register
EMAC_PPS[n]_TGTM_NSEC	PPS n Target Time Nanoseconds Register
EMAC_PPS[n]_TGTM_SEC	PPS n Target Time Seconds Register
EMAC_PPS[n]_WID	PPS n Width Register
EMAC_PPS_CTL	PPS Control Register
EMAC_PTO_CTL	PTP Time Stamp Offload Control Register
EMAC_MACPKT_FILT	MAC Packet Filter Register
EMAC_PRSNTM_UPDT	Presentation Time Update Register
EMAC_PRSNTM_NSEC	Presentation Time Nanosecond Register
EMAC_Q0_TXFLOW_CTL	Queue 0 Transmit Flow Control Register
EMAC_Q[i]_TXFLOW_CTL	Queue [i] Transmit Flow Control Register
EMAC_RXQ_CTL0	Receive Queue Control 0 Register
EMAC_RXQ_CTL1	Receive Queue Control 1 Register
EMAC_RXQ_CTL2	Receive Queue Control 2 Register
EMAC_RXQ_CTL3	Receive Queue Control 3 Register
EMAC_RXQ_CTL4	Receive Queue Control 4 Register
EMAC_MAC_RX_DOMAIN_TIME_INCR	MAC Receive Time Domain Increment Register
EMAC_RXFLOW_CTL	Receive Flow Control Register
EMAC_RXTX_STAT	Receive Transmit Status Register
EMAC_SRCPRT_IDNTY0	Source Port Identity 0 Register
EMAC_SRCPRT_IDNTY1	Source Port Identity 1 Register
EMAC_SRCPRT_IDNTY2	Source Port Identity 2 Register
EMAC_TM_SUBSEC	Subsecond Increment Register
EMAC_TM_HISEC	System Time Higher Word Seconds Register
EMAC_TM_NSEC	System Time Nanoseconds Register
EMAC_TM_NSECUPDT	System Time Nanoseconds Update Register
EMAC_TM_SEC	System Time Seconds Register
EMAC_TM_SECUPDT	System Time Seconds Update Register
EMAC_TM_ADDEND	Time Stamp Addend Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_TM_CTL	Time Stamp Control Register
EMAC_TM_EGASYM_CORR	Time Stamp Egress Asymmetry Correction Register
EMAC_TM_EGCORR_NSEC	Time Stamp Egress Correction Nanosecond Register
EMAC_TM_EGCORR_SNSEC	Time Stamp Egress Correction Sub Nanosecond Register
EMAC_TM_EG_LTNCY	Time Stamp Egress Latency Register
EMAC_TM_INGASYM_CORR	Time Stamp Ingress Asymmetry Correction Register
EMAC_TM_INGCORR_NSEC	Time Stamp Ingress Correction Nanosecond Register
EMAC_TM_INGCORR_SNSEC	Time Stamp Ingress Correction Sub Nanosecond Register
EMAC_TM_ING_LTNCY	Time Stamp Ingress Latency Register
EMAC_TM_STAT	Time Stamp Status Register
EMAC_TXQ_PRTY_MAP0	Transmit Queue Priority Mapping 0 Register
EMAC_TXQ_PRTY_MAP1	Transmit Queue Priority Mapping 1 Register
EMAC_MAC_TX_DOMAIN_TIME_INCR	MAC Transmit Time Domain Increment Register
EMAC_TXTMSTAT_NSEC	Transmit Time Stamp Status Nanoseconds Register
EMAC_TXTMSTAT_SEC	Transmit Time Stamp Status Seconds Register
EMAC_VLAN_HASHTBL	VLAN Hash Table Register
EMAC_VLAN_INCL	VLAN Tag Inclusion or Replacement Register
EMAC_VLANTAG_CTL	VLAN Tag Control Register
EMAC_VLANTAG_DAT	VLAN Tag Indirect Access Data Register
EMAC_WDT_JB_TMOUT	Watchdog and Jabber Timeout Register
EMAC_MMC_CTRL	MMC Control Register
EMAC_MMC_FPE_RXINT	MMC Receive FPE Interrupt Status Register
EMAC_MMC_FPE_RXIMSK	MMC Receive FPE Interrupt Mask Register
EMAC_MMC_FPE_TXINT	MMC Transmit FPE Interrupt Status Register
EMAC_MMC_FPE_TXIMSK	MMC Transmit FPE Interrupt Mask Register
EMAC_MMC_IPC_RXINT	MMC Receive IPC Interrupt Status Register
EMAC_MMC_IPC_RXIMSK	MMC Receive IPC Interrupt Mask Register
EMAC_RXFPE_FRGMNT_CNT	MMC Receive FPE Fragment Counter Register
EMAC_MMC_RXINT	MMC Receive Interrupt Status Register
EMAC_MMC_RXIMSK	MMC Receive Interrupt Mask Register
EMAC_RXPKT_ASERR_CNT	MMC Receive FPE Packet Assembly Error Counter Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_RXPKT_ASOK_CNT	MMC Receive FPE Packet Assembly Okay Counter Register
EMAC_RXPKT_SMDERR_CNT	MMC Receive FPE Packet SMD Error Counter Register
EMAC_TXFPE_FRGMNT_CNT	MMC Transmit FPE Packet Fragment Counter Register
EMAC_TXHLDRQ_CNT	MMC Transmit FPE Hold Request Counter Register
EMAC_MMC_TXINT	MMC Transmit Interrupt Status Register
EMAC_MMC_TXIMSK	MMC Transmit Interrupt Mask Register
EMAC_EST_CTL	EST Control Register
EMAC_EST_EXT_CTL	EST Extended Control Register
EMAC_EST_FRMSZ_CAP	EST Frame Size Capture Register
EMAC_EST_FRMSZ_ERR	EST Frame Size Error Register
EMAC_EST_GCL_CTL	EST Gate Control List Control Register
EMAC_EST_GCL_DAT	EST Gate Control List Data Register
EMAC_EST_IEN	EST Interrupt Enable Register
EMAC_EST_SCH_ERR	EST Scheduling Error Register
EMAC_EST_STAT	EST Status Register
EMAC_FPE_ADV	Frame Preemption Advance Register
EMAC_FPE_CTRLSTS	MTL Frame Preemption Control and Status Register
EMAC_MTL_ISTAT	MTL Interrupt Status Register
EMAC_MTL_OPMODE	MTL Operation Mode Register
EMAC_Q0_INT_CTLSTAT	Queue 0 Interrupt Control and Status Register
EMAC_Q[n]_INT_CTLSTAT	Queue n Interrupt Control and Status Register
EMAC_RQ0_CTL	Receive Queue Control Register
EMAC_RQ0_DBG	Receive Queue 0 Debug Register
EMAC_RQ0_MSPKTOF_CNT	Receive Queue 0 Missed Packet and Overflow Counter Register
EMAC_RQ0_OPMODE	Receive Queue 0 Operation Mode Register
EMAC_RQ[n]_CTL	Receive Queue n Control Register
EMAC_RQ[n]_DBG	Receive Queue n Debug Register
EMAC_RQ[n]_MSPKTOF_CNT	Receive Queue n Missed Packet and Overflow Counter Register
EMAC_RQ[n]_OPMODE	Receive Queue n Operation Mode Register
EMAC_RXQ_DMA_MAP0	Receive Queue to DMA Channel Mapping Register 0
EMAC_RXQ_DMA_MAP1	Receive Queue to DMA Channel Mapping Register 1

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_TBS_CTRL	MTL Time Based Scheduling Control Register
EMAC_TQ0_DBG	Transmit Queue 0 Debug Register
EMAC_TQ0_ETS_STAT	Transmit Queue 0 ETS Status Register
EMAC_TQ0_OPMODE	Transmit Queue 0 Operation Mode Register
EMAC_TQ0_QTMWGT	Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ0_URFL	Transmit Queue 0 Underflow Packet Counter Register
EMAC_TQ[n]_DBG	Transmit Queue n Debug Register
EMAC_TQ[n]_ETS_CTL	Transmit Queue n ETS Control Register
EMAC_TQ[n]_ETS_STAT	Transmit Queue n ETS Status Register
EMAC_TQ[n]_HICRDT	Transmit Queue n CBS HiCredit Register
EMAC_TQ[n]_LOCRDT	Transmit Queue n CBS LoCredit Register
EMAC_TQ[n]_OPMODE	Transmit Queue n Operation Mode Register
EMAC_TQ[n]_QTMWGT	Transmit Queue n IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ[n]_SSCRDT	Transmit Queue n SendSlopeCredit Register
EMAC_TQ[n]_URFL	Transmit Queue n Underflow Packet Counter Register
EMAC_RXICMP_ERR_OCNT	Receive ICMP Error Octets Counter Register
EMAC_RXICMP_ERR_PCNT	Receive ICMP Error Packets Counter Register
EMAC_RXICMP_OCNT_G	Receive ICMP Good Octets Counter Register
EMAC_RXICMP_PCNT_G	Receive ICMP Good Packets Counter Register
EMAC_RXIPV4_FRGMT_OCNT	Receive IPv4 Fragmented Octets Counter Register
EMAC_RXIPV4_FRGMT_PCNT	Receive IPv4 Fragmented Packets Counter Register
EMAC_RXIPV4_OCNT_G	Receive IPv4 Good Octets Counter Register
EMAC_RXIPV4_PCNT_G	Receive IPv4 Good Packets Counter Register
EMAC_RXIPV4_HDRERR_OCNT	Receive IPv4 Header Error Octets Counter Register
EMAC_RXIPV4_HDRERR_PCNT	Receive IPv4 Header Error Packets Counter Register
EMAC_RXIPV4_PYLD_OCNT	Receive IPv4 Payload Octets Counter Register
EMAC_RXIPV4_PYLD_PCNT	Receive IPv4 Payload Packets Counter Register
EMAC_RXIPV4_UDPCSD_OCNT	Receive IPv4 UDP Checksum Disable Octets Counter Register
EMAC_RXIPV4_UDPCSD_PCNT	Receive IPv4 UDP Checksum Disable Packets Counter Register
EMAC_RXIPV6_OCNT_G	Receive IPv6 Good Octets Counter Register
EMAC_RXIPV6_PCNT_G	Receive IPv6 Good Packets Counter Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_RXIPV6_HDRERR_OCNT	Receive IPv6 Header Error Octets Counter Register
EMAC_RXIPV6_HDRERR_PCNT	Receive IPv6 Header Error Packets Counter Register
EMAC_RXIPV6_PYLD_OCNT	Receive IPv6 Payload Octets Counter Register
EMAC_RXIPV6_PYLD_PCNT	Receive IPv6 Payload Packets Counter Register
EMAC_RXTCP_ERR_OCNT	Receive TCP Error Octets Counter Register
EMAC_RXTCP_ERR_PCNT	Receive TCP Error Packets Counter Register
EMAC_RXTCP_OCNT_G	Receive TCP Good Octets Counter Register
EMAC_RXTCP_PCNT_G	Receive TCP Good Packets Counter Register
EMAC_RXUDP_ERR_OCNT	Receive UDP Error Octets Counter Register
EMAC_RXUDP_ERR_PCNT	Receive UDP Error Packets Counter Register
EMAC_RXUDP_OCNT_G	Receive UDP Good Octets Counter Register
EMAC_RXUDP_PCNT_G	Receive UDP Good Packets Counter Register
EMAC_RX1024TOMAX_GB	Receive 1024 - Maximum Octets Packets Good Bad Counter Register
EMAC_RX128TO255_GB	Receive 128 - 255 Octets Packets Good Bad Counter Register
EMAC_RX256TO511_GB	Receive 256 - 511 Octets Packets Good Bad Counter Register
EMAC_RX512TO1023_GB	Receive 512 - 1023 Octets Packets Good Bad Counter Register
EMAC_RX64_GB	Receive 64 Octets Packets Good Bad Counter Register
EMAC_RX65TO127_GB	Receive 65 - 127 Octets Packets Good Bad Counter Register
EMAC_RX_ALGNERR_PCNT	Receive Alignment Error Packets Counter Register
EMAC_RX_BCERR_PCNT_G	Receive Broadcast Packets Good Counter Register
EMAC_RX_CRCERR_PCNT	Receive CRC Error Packets Counter Register
EMAC_RX_CTL_PCNT_G	Receive Control Packets Good Counter Register
EMAC_RX_FIFOOVFL_PCNT	Receive FIFO Overflow Packets Counter Register
EMAC_RX_JBERR_PCNT	Receive Jabber Error Packets Counter Register
EMAC_RX_LPITRAN_CNT	Receive LPI Transition Counter Register
EMAC_RX_LPIUSEC_CNT	Receive LPI Microseconds Counter Register
EMAC_RX_LENERR_PCNT	Receive Length Error Packets Counter Register
EMAC_RX_MCST_PCNT_G	Receive Multicast Packets Good Counter Register
EMAC_RX_OCNT_G	Receive Octet Good Counter Register
EMAC_RX_OCNT_GB	Receive Octet Good Bad Counter Register
EMAC_RX_ORTYPE_PCNT	Receive Out of Range Type Packets Counter Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_RX_OVRSZ_PCNT	Receive Oversize Packets Good Counter Register
EMAC_RX_PCNT_GB	Receive Packets Good Bad Counter Register
EMAC_RX_PAUSE_PCNT	Receive Pause Packets Counter Register
EMAC_RX_ERR_PCNT	Receive Error Packets Counter Register
EMAC_RX_RNTERR_PCNT	Receive Runt Error Packets Counter Register
EMAC_RX_UNDRSZ_PCNT_G	Receive Undersize Packets Good Counter Register
EMAC_RX_UCST_PCNT_G	Receive Unicast Packets Good Counter Register
EMAC_RX_VLAN_PCNT_GB	Receive VLAN Packets Good Bad Counter Register
EMAC_RX_WDTERR_PCNT	Receive Watchdog Error Packets Counter Register
EMAC_TX1024TOMAX_GB	Transmit 1024-Max Octets Packets Good Bad Counter Register
EMAC_TX128TO255_GB	Transmit 128-255 Octets Packets Good Bad Counter Register
EMAC_TX256TO511_GB	Transmit 256-511 Octets Packets Good Bad Counter Register
EMAC_TX512TO1023_GB	Transmit 512-1023 Octets Packets Good Bad Counter Register
EMAC_TX64_GB	Transmit 64 Octets Packets Good Bad Counter Register
EMAC_TX65TO127_GB	Transmit 65-127 Octets Packets Good Bad Counter Register
EMAC_TX_BCST_PCNT_G	Transmit Broadcast Packets Good Counter Register
EMAC_TX_BCST_PCNT_GB	Transmit Broadcast Packets Good Bad Counter Register
EMAC_TX_CRERR_PCNT	Transmit Carrier Error Packets Counter Register
EMAC_TX_DFRD_PCNT	Transmit Deferred Packets Counter Register
EMAC_TX_EXCOL_PCNT	Transmit Excessive Collision Packets Counter Register
EMAC_TX_EXDFRL_PCNT	Transmit Excessive Deferral Error Counter Register
EMAC_TX_LPITRAN_CNT	Transmit LPI Transition Counter Register
EMAC_TX_LPIUSEC_CNT	Transmit LPI Microseconds Counter Register
EMAC_TX_LTCOL_PCNT	Transmit Late Collision Packets Counter Register
EMAC_TX_MCST_PCNT_G	Transmit Multicast Packets Good Counter Register
EMAC_TX_MCST_PCNT_GB	Transmit Multicast Packets Good Bad Counter Register
EMAC_TX_MCOL_PCNT_G	Transmit Multiple Collision Packets Good Counter Register
EMAC_TX_OSIZE_PCNT_G	Transmit Osize Packets Good Counter Register
EMAC_TX_OCNT_G	Transmit Octet Good Counter Register
EMAC_TX_OCNT_GB	Transmit Octet Good Bad Counter Register
EMAC_TX_PCNT_G	Transmit Packet Good Counter Register

Table 28-112: ADSP-2184x EMAC Register List (Continued)

Name	Description
EMAC_TX_PCNT_GB	Transmit Packet Good Bad Counter Register
EMAC_TX_PAUSE_PCNT	Transmit Pause Packets Counter Register
EMAC_TX_SNGLCOL_PCNT_G	Transmit Single Collision Packets Good Counter Register
EMAC_TX_URFL_PCNT	Transmit Underflow Error Packets Counter Register
EMAC_TX_UCST_PCNT_GB	Transmit Unicast Packets Good Bad Counter Register
EMAC_TX_VLAN_PCNT_G	Transmit VLAN Packets Good Counter Register

DMA Channel n Control Register

The EMAC_DMA[n]_CTL register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

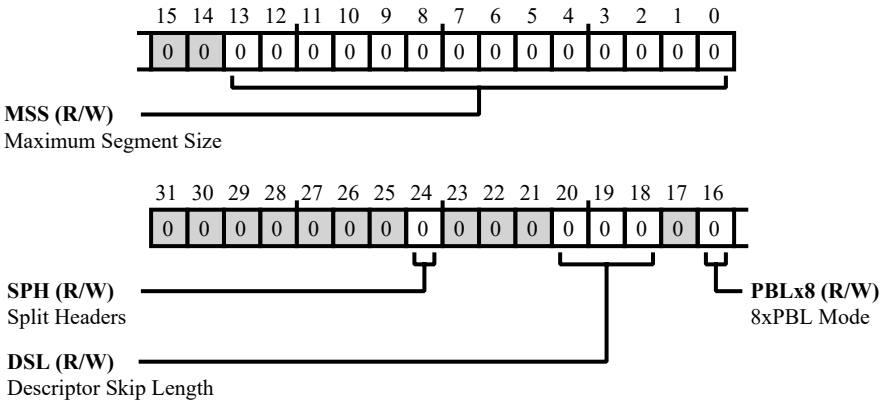


Figure 28-58: EMAC_DMA[n]_CTL Register Diagram

Table 28-113: EMAC_DMA[n]_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SPH	Split Headers. When the EMAC_DMA[n]_CTL.SPH bit is set (=1), the DMA splits the header and payload in the receive path. The DMA writes the header to Buffer Address1 of RDES0. The DMA writes the payload to the buffer to which the Buffer Address2 is pointing. The software must ensure that the header fits into the receive buffers. If the header length exceeds the receive buffer size, the DMA does not split the header and payload. The EMAC_DMA[n]_CTL.SPH bit is available only when the split header feature is enabled (EMAC_HW_FTR_PSNT1.SPHEN =1).	
		0	Split Headers Feature is Disabled
		1	Split Headers Feature is Enabled
20:18 (R/W)	DSL	Descriptor Skip Length. The EMAC_DMA[n]_CTL.DSL bit specifies the number of 32-bit words to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When EMAC_DMA[n]_CTL.DSL = 0, the DMA takes the descriptor table as contiguous.	

Table 28-113: EMAC_DMA[n]_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	PBLX8	8xPBL Mode. When the EMAC_DMA[n]_CTL.PBLX8 bit is set, the PBL value programmed in bits[21:16] in EMAC_DMA[n]_TXCTL and bits[21:16] in EMAC_DMA[n]_RXCTL is multiplied by eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.	
		0	8xPBL Mode is Disabled
		1	8xPBL Mode is Enabled
13:0 (R/W)	MSS	Maximum Segment Size. The EMAC_DMA[n]_CTL.MSS field specifies the maximum segment size to be used while segmenting the packet. The EMAC_DMA[n]_CTL.MSS field is valid only when EMAC_DMA[n]_TXCTL.TSE =1. The value programmed in this field must be more than the configured data width in bytes. Use a EMAC_DMA[n]_CTL.MSS value of 64 bytes or more.	

DMA Channel n Current Application Receive Buffer Address Register

The EMAC_DMA[n]_RXBUF_CUR register points to the current Rx buffer address read by the DMA.

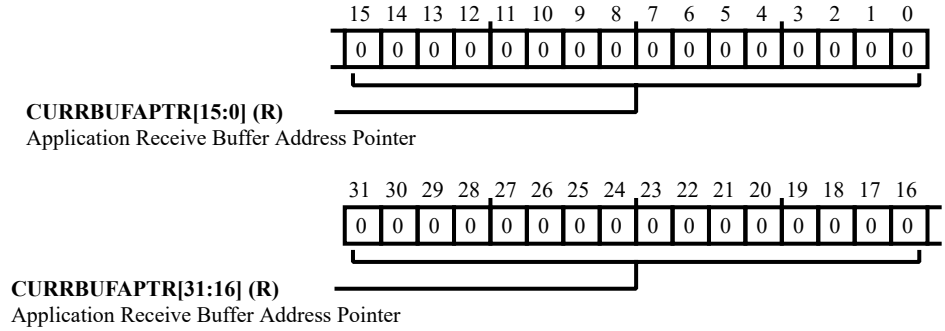


Figure 28-59: EMAC_DMA[n]_RXBUF_CUR Register Diagram

Table 28-114: EMAC_DMA[n]_RXBUF_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURRBUFAPTR	Application Receive Buffer Address Pointer. The DMA updates the EMAC_DMA[n]_RXBUF_CUR.CURRBUFAPTR field during the Rx operation. This pointer is cleared on reset.

DMA Channel n Current Application Receive Descriptor Register

The EMAC_DMA[n]_RXDSC_CUR register points to the current receive descriptor read by the DMA.

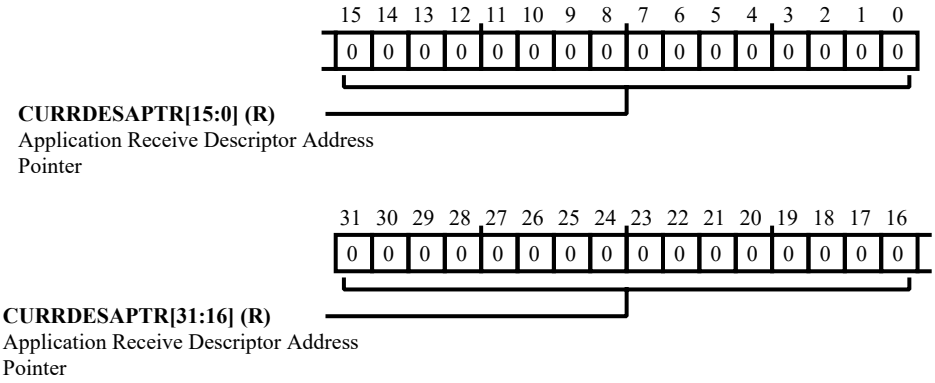


Figure 28-60: EMAC_DMA[n]_RXDSC_CUR Register Diagram

Table 28-115: EMAC_DMA[n]_RXDSC_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURRDESAPTR	Application Receive Descriptor Address Pointer. The DMA updates the EMAC_DMA[n]_RXDSC_CUR.CURRDESAPTR field during the Rx operation. This pointer is cleared on reset.

DMA Channel n Current Application Transmit Buffer Address Register

The EMAC_DMA[n]_TXBUF_CUR register points to the current Tx buffer address read by the DMA.

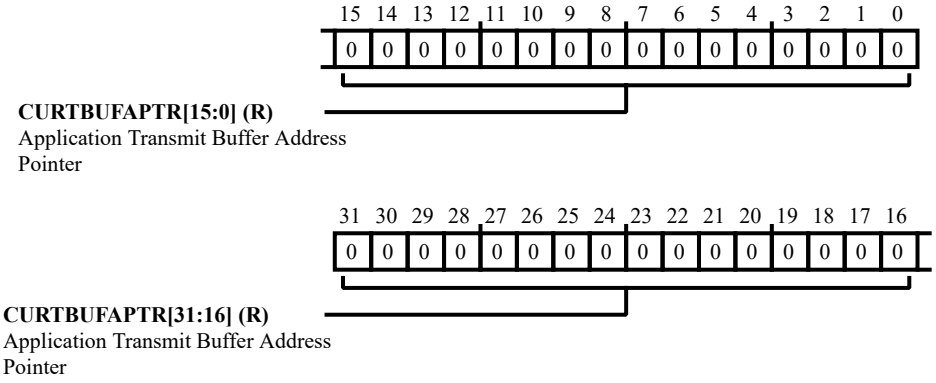


Figure 28-61: EMAC_DMA[n]_TXBUF_CUR Register Diagram

Table 28-116: EMAC_DMA[n]_TXBUF_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURTBUFAPTR	Application Transmit Buffer Address Pointer. The DMA updates the EMAC_DMA[n]_TXBUF_CUR.CURTBUFAPTR field during the Tx operation. This pointer is cleared on reset.

DMA Channel n Current Application Transmit Descriptor Register

The EMAC_DMA[n]_TXDSC_CUR register points to the current transmit descriptor read by the DMA.

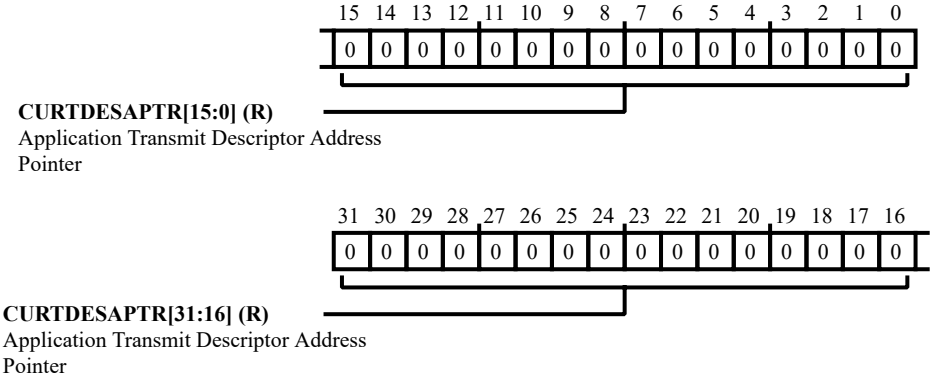


Figure 28-62: EMAC_DMA[n]_TXDSC_CUR Register Diagram

Table 28-117: EMAC_DMA[n]_TXDSC_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURTDESAPTR	Application Transmit Descriptor Address Pointer. The DMA updates the EMAC_DMA[n]_TXDSC_CUR.CURTDESAPTR field during the Tx operation. This pointer is cleared on reset.

DMA Channel n Interrupt Enable Register

The EMAC_DMA[n]_IEN register enables the interrupts reported by the status register.

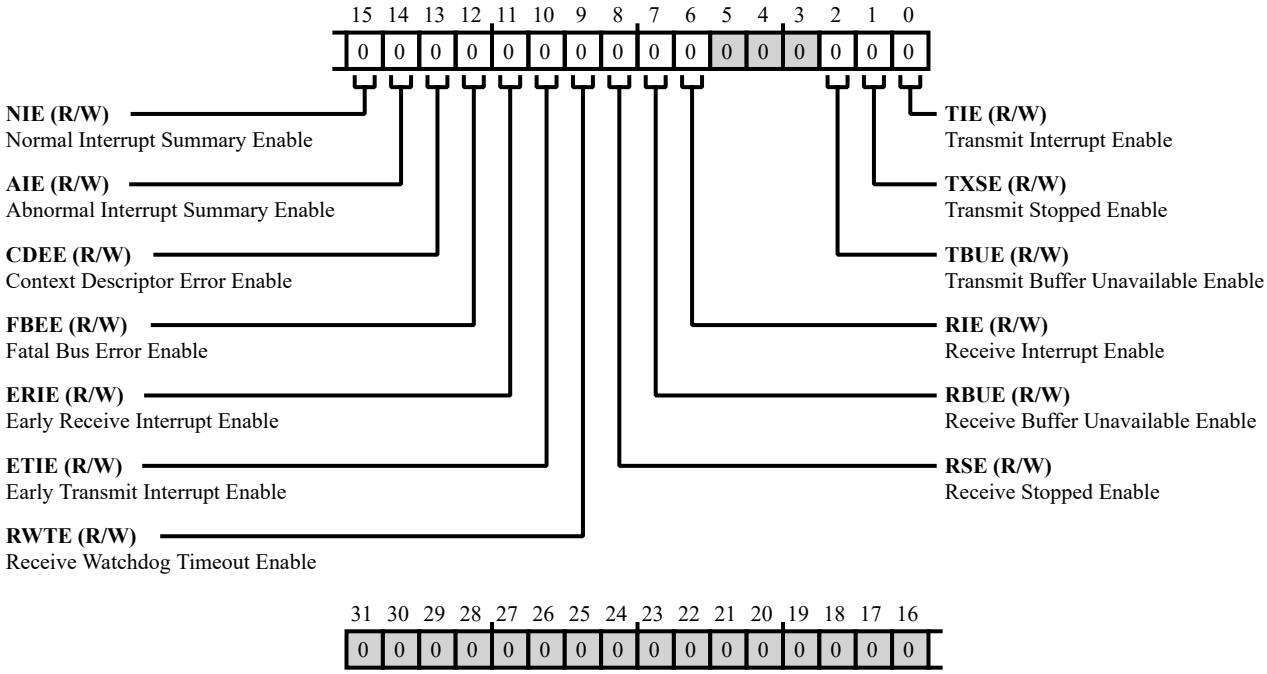


Figure 28-63: EMAC_DMA[n]_IEN Register Diagram

Table 28-118: EMAC_DMA[n]_IEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	NIE	Normal Interrupt Summary Enable. When the EMAC_DMA[n]_IEN.NIE bit is set (=1), the normal interrupt summary is enabled. The EMAC_DMA[n]_IEN.NIE bit enables the following interrupts in the EMAC_DMA[n]_STAT register: - Bit 0: Transmit Interrupt - Bit 2: Transmit Buffer Unavailable - Bit 6: Receive Interrupt - Bit 11: Early Receive Interrupt When EMAC_DMA[n]_IEN.NIE = 0, the normal interrupt summary is disabled.	
		0	Disable
		1	Enable

Table 28-118: EMAC_DMA[n]_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	AIE	Abnormal Interrupt Summary Enable. When the EMAC_DMA[n]_IEN.AIE bit is set, the abnormal interrupt summary is enabled. The EMAC_DMA[n]_IEN.AIE bit enables the following interrupts in the EMAC_DMA[n]_STAT register: - Bit 1: Transmit Process Stopped - Bit 7: Rx Buffer Unavailable - Bit 8: Receive Process Stopped - Bit 9: Receive Watchdog Timeout - Bit 10: Early Transmit Interrupt - Bit 12: Fatal Bus Error - Bit 13: Context Descriptor Error When EMAC_DMA[n]_IEN.AIE = 0, the abnormal interrupt summary is disabled.	
		0	Disable
		1	Enable
13 (R/W)	CDEE	Context Descriptor Error Enable. When EMAC_DMA[n]_IEN.CDEE =1 and EMAC_DMA[n]_IEN.AIE =1, the descriptor error interrupt is enabled. When EMAC_DMA[n]_IEN.CDEE = 0, the descriptor error interrupt is disabled.	
		0	Disable
		1	Enable
12 (R/W)	FBEE	Fatal Bus Error Enable. When EMAC_DMA[n]_IEN.FBEE =1 and EMAC_DMA[n]_IEN.AIE =1, the fatal bus error interrupt is enabled. When EMAC_DMA[n]_IEN.FBEE = 0, the fatal bus error interrupt is disabled.	
		0	Disable
		1	Enable
11 (R/W)	ERIE	Early Receive Interrupt Enable. When EMAC_DMA[n]_IEN.ERIE =1 and EMAC_DMA[n]_IEN.NIE =1, the early receive interrupt is enabled. When EMAC_DMA[n]_IEN.ERIE = 0, the early receive interrupt is disabled.	
		0	Disable
		1	Enable

Table 28-118: EMAC_DMA[n]_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	ETIE	Early Transmit Interrupt Enable. When EMAC_DMA[n]_IEN.ETIE =1 and EMAC_DMA[n]_IEN.AIE =1, the early transmit interrupt is enabled. When EMAC_DMA[n]_IEN.ETIE = 0, the early transmit interrupt is disabled.	
		0	Disable
		1	Enable
9 (R/W)	RWTE	Receive Watchdog Timeout Enable. When EMAC_DMA[n]_IEN.RWTE =1 and EMAC_DMA[n]_IEN.AIE =1, the receive watchdog timeout interrupt is enabled. When EMAC_DMA[n]_IEN.RWTE = 0, the receive watchdog timeout interrupt is disabled.	
		0	Disable
		1	Enable
8 (R/W)	RSE	Receive Stopped Enable. When EMAC_DMA[n]_IEN.RSE =1 and EMAC_DMA[n]_IEN.AIE =1, the receive stopped interrupt is enabled. When EMAC_DMA[n]_IEN.RSE = 0, the receive stopped interrupt is disabled.	
		0	Disable
		1	Enable
7 (R/W)	RBUE	Receive Buffer Unavailable Enable. When EMAC_DMA[n]_IEN.RBUE =1 and EMAC_DMA[n]_IEN.AIE =1, the receive buffer unavailable interrupt is enabled. When EMAC_DMA[n]_IEN.RBUE = 0, the receive buffer unavailable interrupt is disabled.	
		0	Disable
		1	Enable
6 (R/W)	RIE	Receive Interrupt Enable. When EMAC_DMA[n]_IEN.RIE =1 and EMAC_DMA[n]_IEN.NIE =1, the receive interrupt is enabled. When EMAC_DMA[n]_IEN.RIE = 0, the receive interrupt is disabled.	
		0	Disable
		1	Enable

Table 28-118: EMAC_DMA[n]_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	TBUE	Transmit Buffer Unavailable Enable. When `EMAC_DMA[n]_IEN.TBUE` =1 and EMAC_DMA[n]_IEN.NIE =1, the transmit buffer unavailable interrupt is enabled. When `EMAC_DMA[n]_IEN.TBUE` = 0, the transmit buffer unavailable interrupt is disabled.	
		0	Disable
		1	Enable
1 (R/W)	TXSE	Transmit Stopped Enable. When `EMAC_DMA[n]_IEN.TXSE` =1 and EMAC_DMA[n]_IEN.AIE =1, the transmission stopped interrupt is enabled. When `EMAC_DMA[n]_IEN.TXSE` = 0, the transmission stopped interrupt is disabled.	
		0	Disable
		1	Enable
0 (R/W)	TIE	Transmit Interrupt Enable. When `EMAC_DMA[n]_IEN.TIE` =1 and EMAC_DMA[n]_IEN.NIE =1, the transmit interrupt is enabled. When `EMAC_DMA[n]_IEN.TIE` = 0, the transmit interrupt is disabled.	
		0	Disable
		1	Enable

DMA Channel n Missed Frame Counter Register

The EMAC_DMA[n]_MISSFRM_CNT register indicates the number of frames that the DMA dropped due to a bus error or due to programming the EMAC_DMA[n]_RXCTL.RPF bit.

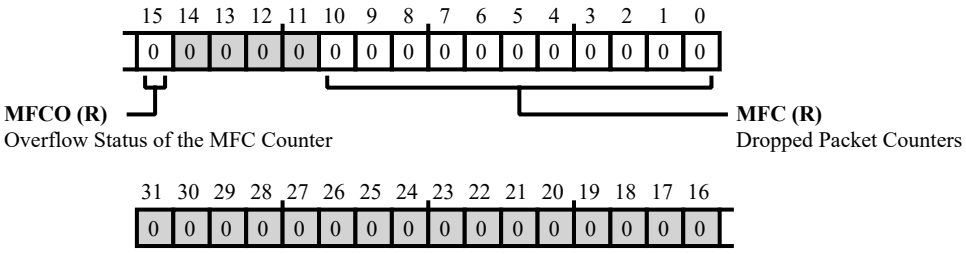


Figure 28-64: EMAC_DMA[n]_MISSFRM_CNT Register Diagram

Table 28-119: EMAC_DMA[n]_MISSFRM_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	MFCO	Overflow Status of the MFC Counter. When EMAC_DMA[n]_MISSFRM_CNT.MFCO bit is set (=1), the MFC counter does not get incremented further. The bit is cleared (=0) when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Miss Frame Counter Overflow Not Occurred
		1	Miss Frame Counter Overflow Occurred
10:0 (R/NW)	MFC	Dropped Packet Counters. The EMAC_DMA[n]_MISSFRM_CNT.MFC field indicates the number of frames that are dropped by the DMA either because of a bus error or because of programing the EMAC_DMA[n]_RXCTL.RPF bit. The counter is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.	

DMA Channel n Receive Descriptor List Address Register

The EMAC_DMA[n]_RXDSC_ADDR register points the DMA to the start of the receive descriptor list.

The descriptor lists reside in the physical memory space of the application. It must be 32-bit aligned. The DMA internally converts it to bus width aligned address by making the corresponding least significant bits low.

Writing to the EMAC_DMA[n]_RXDSC_ADDR register is permitted only when reception is stopped. When stopped, this register must be written to before the receive start command is given.

Write to this register only when Rx DMA has stopped (EMAC_DMA[n]_RXCTL.SR = 0). When stopped, this register can be written with a new descriptor list address.

When EMAC_DMA[n]_RXCTL.SR = 1, the DMA takes the newly programmed descriptor base address.

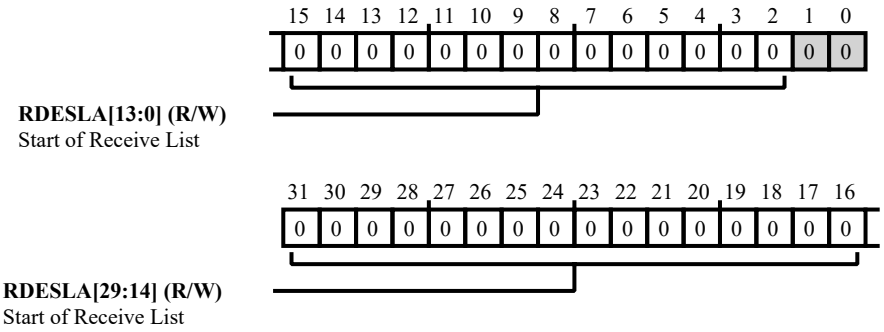


Figure 28-65: EMAC_DMA[n]_RXDSC_ADDR Register Diagram

Table 28-120: EMAC_DMA[n]_RXDSC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	RDESLA	Start of Receive List. The EMAC_DMA[n]_RXDSC_ADDR.RDESLA field contains the base address of the first descriptor in the Rx descriptor list. The DMA ignores the last two bits (1:0) and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

DMA Channel n Receive Descriptor Tail Pointer Register

The EMAC_DMA[n]_RXDSC_TLPTR register indicates an offset from the base and indicates the location of the last valid descriptor.

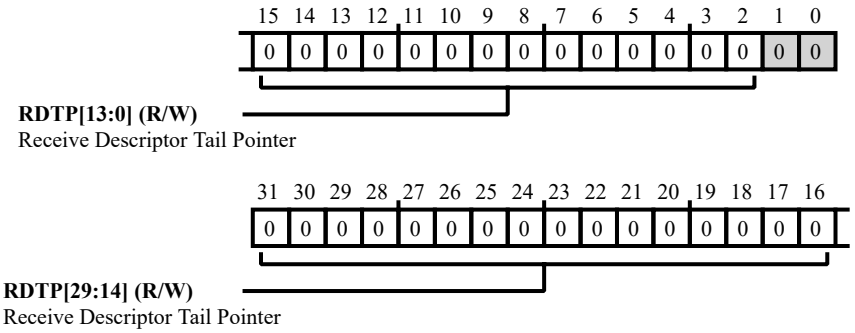


Figure 28-66: EMAC_DMA[n]_RXDSC_TLPTR Register Diagram

Table 28-121: EMAC_DMA[n]_RXDSC_TLPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	RDTP	Receive Descriptor Tail Pointer. The EMAC_DMA[n]_RXDSC_TLPTR.RDTP field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

DMA Channel n Receive Control Register

The EMAC_DMA[n]_RXCTL register controls the Rx features such as PBL, buffer size, and extended status.

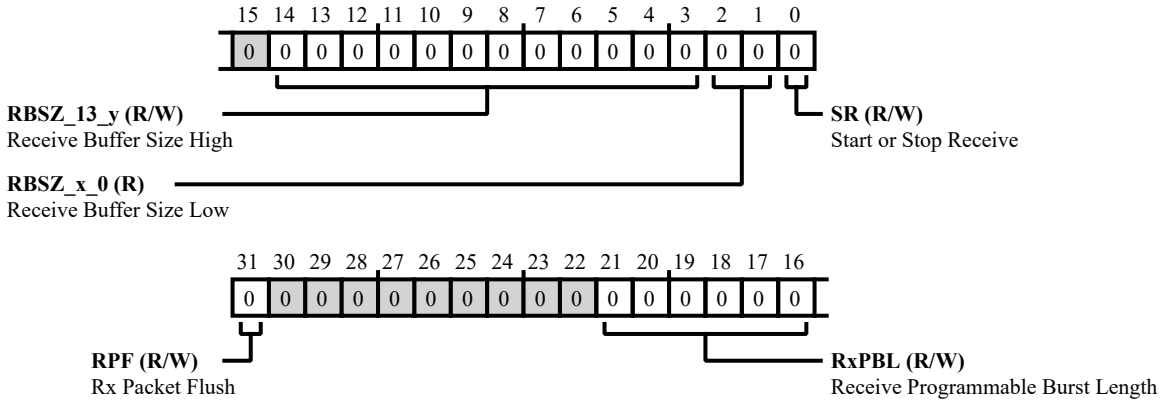


Figure 28-67: EMAC_DMA[n]_RXCTL Register Diagram

Table 28-122: EMAC_DMA[n]_RXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	RPF	Rx Packet Flush. The EMAC_DMA[n]_RXCTL.RPF bit enables the Rx packet flush feature. When set (= 1), the EMAC module automatically flushes the packet from the Rx queues designated to this DMA Rx channel, when it is stopped. When the EMAC_DMA[n]_RXCTL.RPF bit remains set and the DMA is restarted by the software driver, the packets residing in the Rx queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is restarted are routed to the RxDMA. The flushing is done on the read side of the Rx queue. When the EMAC_DMA[n]_RXCTL.RPF bit is cleared (= 0), the EMAC module does not flush the packet in the Rx queue destined to this RxDMA channel when it is in STOP state. This operation can cause head-of-line blocking in the corresponding Rx queue. Note: The stopping of packet flow from a Rx DMA channel to the application by setting RPF works only when there is one-to-one mapping of Rx queue to Rx DMA channels. In dynamic mapping mode, setting this bit can flush packets from unintended Rx queues which are destined to the stopped Rx DMA channel.	
		0	Rx Packet Flush is Disabled
		1	Rx Packet Flush is Enabled

Table 28-122: EMAC_DMA[n]_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21:16 (R/W)	RXPBL	Receive Programmable Burst Length. The EMAC_DMA[n]_RXCTL.RXPBL field indicates the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. Program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in the EMAC_DMA[n]_CTL register. 2. Set the RxPBL. Note: The maximum value of RxPBL must be less than or equal to half the Rx queue size (EMAC_RQ[n]_OPMODE.RQS) in terms of beats. This is required so that the Rx queue has space to store at least another Rx PBL worth of data while the Rx DMA is transferring a block of data. For example, in the total locations in Rx queue of size 512 bytes is 128, so RxPBL and 8xPBL needs to be programmed to less than or equal to 64.
14:3 (R/W)	RBSZ_13_Y	Receive Buffer Size High. RBSZ[13:0] is split into two fields: higher EMAC_DMA[n]_RXCTL.RBSZ_13_Y and lower RBSZ_x_0. The RBSZ[13:0] field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when split headers are enabled. Note: The buffer size must be a multiple of 4 bytes. This is required even if the value of buffer address pointer is not aligned to data bus width. Hence, the lower RBSZ_x_0 bits are read-only and the value is considered as all-zero. Thus, the EMAC_DMA[n]_RXCTL.RBSZ_13_Y indicates the buffer size in terms of locations (with the width same as bus width).
2:1 (R/NW)	RBSZ_X_0	Receive Buffer Size Low. RBSZ[13:0] is split into two fields RBSZ_13_y and RBSZ_x_0. The EMAC_DMA[n]_RXCTL.RBSZ_X_0 field is the lower field whose width is based on data bus width of the configuration. The EMAC_DMA[n]_RXCTL.RBSZ_X_0 field is 2 bits wide. The EMAC_DMA[n]_RXCTL.RBSZ_X_0 field is read-only (RO).

Table 28-122: EMAC_DMA[n]_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	SR	Start or Stop Receive. When the EMAC_DMA[n]_RXCTL.SR bit is set (=1), the DMA tries to acquire the descriptor from the receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: The current position in the list - This is the address set by the EMAC_DMA[n]_RXDSC_ADDR register. The position at which the Rx process was previously stopped - If the DMA does not own the current descriptor, the reception is suspended and the EMAC_DMA[n]_STAT.RBU bit is set (=1). The start receive command is effective only when the reception is stopped. If the command is issued before configuring the EMAC_DMA[n]_RXDSC_ADDR register, the DMA behavior is unpredictable. When EMAC_DMA[n]_RXCTL.SR bit = 0, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the receive list is saved, and it becomes the current position after the Rx process is restarted. The stop receive command is effective only when the Rx process is in the running (waiting for Rx packet) or suspended state.	
		0	Stop Receive
		1	Start Receive

DMA Channel n Extended Receive Control Register

The EMAC_DMA[n]_RXCTL2 register controls the Rx features such as Rx descriptor ring length and alternate Rx buffer size.

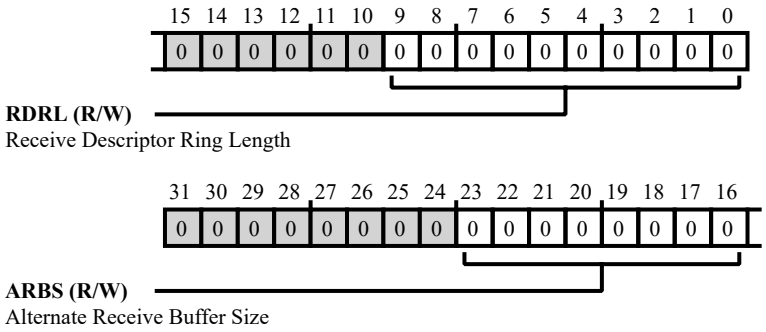


Figure 28-68: EMAC_DMA[n]_RXCTL2 Register Diagram

Table 28-123: EMAC_DMA[n]_RXCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	ARBS	Alternate Receive Buffer Size. The EMAC_DMA[n]_RXCTL2.ARBS field indicates the size in bytes for buffer 1, when programmed to a non-zero value (when the split header feature is not enabled). When the split header feature is enabled, the EMAC_DMA[n]_RXCTL2.ARBS field indicates the buffer size for header data. The maximum alternate buffer is limited to 1020 bytes for the 32-bit data bus width. When EMAC_DMA[n]_RXCTL2.ARBS= 0, the Rx Buffer1 and Rx Buffer2 sizes are based on EMAC_DMA[n]_RXCTL.RBSZ.
9:0 (R/W)	RDRL	Receive Descriptor Ring Length. The EMAC_DMA[n]_RXCTL2 register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. For example, program any value up to 0x3FF in this field. The EMAC_DMA[n]_RXCTL2.RDRL field is 10 bits wide. When programming 0x3FF, there are 1024 descriptors. For 10 descriptors, program it to a value of 0x9.

DMA Channel n Receive Interrupt Watchdog Timer Register

The EMAC_DMA[n]_RXINTWDTMR register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the EMAC_DMA[n]_STAT.RI bit.

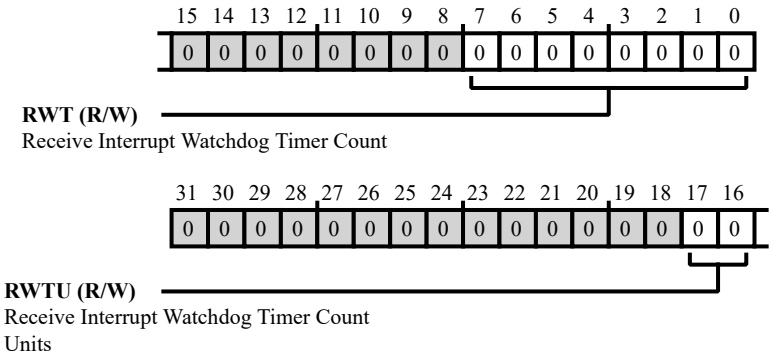


Figure 28-69: EMAC_DMA[n]_RXINTWDTMR Register Diagram

Table 28-124: EMAC_DMA[n]_RXINTWDTMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	RWTU	Receive Interrupt Watchdog Timer Count Units. The EMAC_DMA[n]_RXINTWDTMR.RWTU field indicates the number of SCLK0 cycles corresponding to one unit in the EMAC_DMA[n]_RXINTWDTMR.RWT field. For example, when EMAC_DMA[n]_RXINTWDTMR.RWT=2 and EMAC_DMA[n]_RXINTWDTMR.RWTU = 1, the watchdog timer is configured for 2 * 512 = 1024 SCLK0 cycles.	
		0	256
		1	512
		2	1024
		3	2048
7:0 (R/W)	RWT	Receive Interrupt Watchdog Timer Count. The EMAC_DMA[n]_RXINTWDTMR.RWT field indicates the number of SCLK0 cycles, multiplied by factor indicated in EMAC_DMA[n]_RXINTWDTMR.RWTU field, for which the watchdog timer is set. The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which EMAC_DMA[n]_STAT.RI = 0 register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set (=1) because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.	

DMA Channel n Slot Function Control and Status register

The EMAC_DMA[n]_SFN_CTLSTAT register contains the control bits for slot function and the status for transmit path.

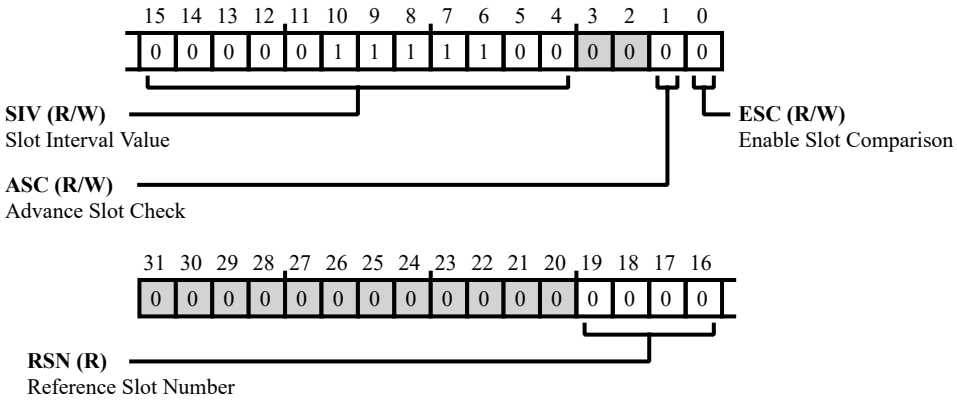


Figure 28-70: EMAC_DMA[n]_SFN_CTLSTAT Register Diagram

Table 28-125: EMAC_DMA[n]_SFN_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/NW)	RSN	Reference Slot Number. The EMAC_DMA[n]_SFN_CTLSTAT.RSN field gives the current value of the reference slot number in the DMA. It is used for slot comparison.	
15:4 (R/W)	SIV	Slot Interval Value. The EMAC_DMA[n]_SFN_CTLSTAT.SIV field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096us. The default/reset value is 0x07C which corresponds to a slot interval of 125 us.	
1 (R/W)	ASC	Advance Slot Check. When set, the EMAC_DMA[n]_SFN_CTLSTAT.ASC bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is: - equal to the reference slot number given in the RSN field or - ahead of the reference slot number by up to two slots. The EMAC_DMA[n]_SFN_CTLSTAT.ASC bit is applicable only when the ESC bit is set.	
		0	Advance Slot Check is Disabled
		1	Advance Slot Check is Enabled

Table 28-125: EMAC_DMA[n]_SFN_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	ESC	Enable Slot Comparison. When set (=1), the `EMAC_DMA[n]_SFN_CTLSTAT.ESC` bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is: - equal to the reference slot number or - ahead of the reference slot number by one slot. When cleared (=0), the `EMAC_DMA[n]_SFN_CTLSTAT.ESC` bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed. Note: The UFO (UDP Fragmentation over IPv4)/TSO/USO must not be enabled with the TBS/AVB slot number check. The UFO/TSO/USO involves multiple packets/segments/fragments transmission for single packet received from application, and the slot number checks are applicable for the fetching of only the first segment/fragment. As a result, it can be difficult for software to specify a slot number for subsequent packets.	
		0	Slot Comparison is Disabled
		1	Slot Comparison is Enabled

DMA Channel n Status Register

The software driver (application) reads the EMAC_DMA[n]_STAT register during interrupt service routine or polling to determine the status of the DMA.

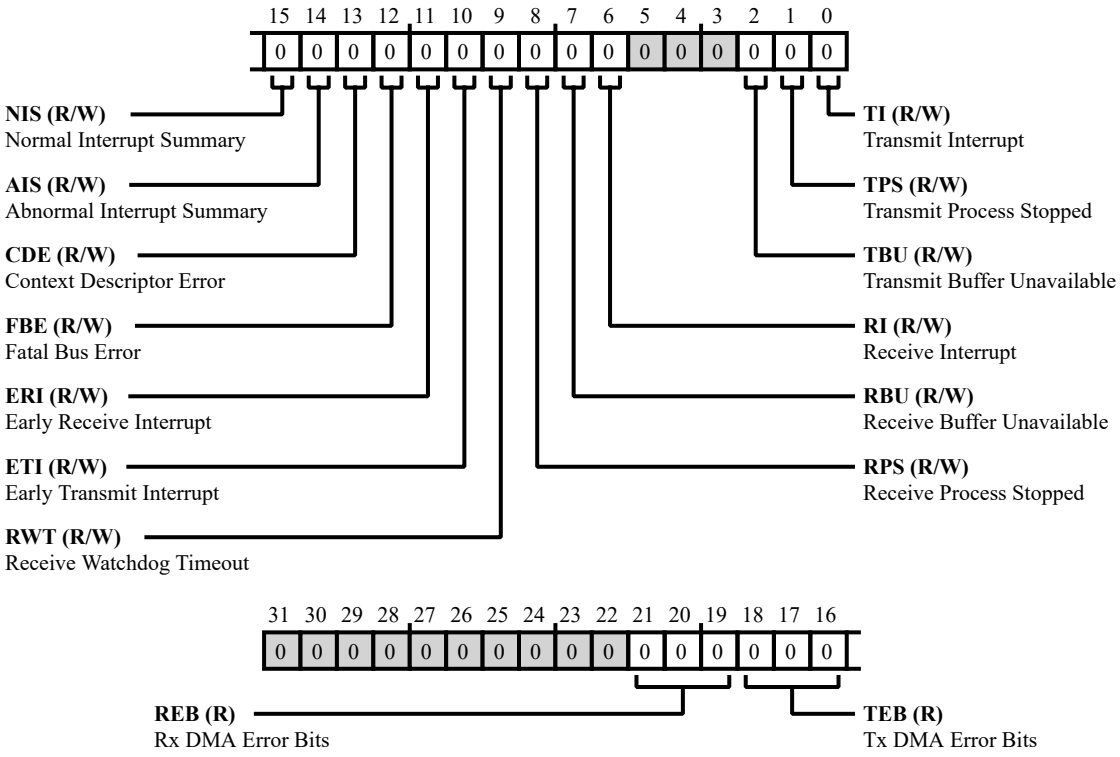


Figure 28-71: EMAC_DMA[n]_STAT Register Diagram

Table 28-126: EMAC_DMA[n]_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:19 (R/NW)	REB	Rx DMA Error Bits. The EMAC_DMA[n]_STAT.REB field indicates the type of error that caused a bus error. For example, an error response on the SPB or SCB interface. Bit 21 -- 1'b1: Error during data transfer by Rx DMA -- 1'b0: No Error during data transfer by Rx DMA Bit 20 -- 1'b1: Error during descriptor access -- 1'b0: Error during data buffer access Bit 19 -- 1'b1: Error during read transfer -- 1'b0: Error during write transfer The EMAC_DMA[n]_STAT.REB field is valid only when the FBE bit is set. It does not generate an interrupt.
18:16 (R/NW)	TEB	Tx DMA Error Bits. The EMAC_DMA[n]_STAT.TEB field indicates the type of error that caused a bus error. For example, error response on the SPB or SCB interface. Bit 18 -- 1'b1: Error during data transfer by Tx DMA -- 1'b0: No Error during data transfer by Tx DMA Bit 17 -- 1'b1: Error during descriptor access -- 1'b0: Error during data buffer access Bit 16 -- 1'b1: Error during read transfer -- 1'b0: Error during write transfer The EMAC_DMA[n]_STAT.TEB field is valid only when the FBE bit is set. It does not generate an interrupt.

Table 28-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	NIS	Normal Interrupt Summary. The EMAC_DMA[n]_STAT.NIS bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the EMAC_DMA[n]_IEN register: Bit 0: Transmit Interrupt Bit 2: Transmit Buffer Unavailable Bit 6: Receive Interrupt Bit 11: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in EMAC_DMA[n]_IEN register) affect the Normal Interrupt Summary bit. This is a sticky bit; the EMAC_DMA[n]_STAT.NIS bit must be cleared (by writing 1) each time a corresponding bit that causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Normal Interrupt Summary Status Not Detected
		1	Normal Interrupt Summary Status Detected
14 (R/W)	AIS	Abnormal Interrupt Summary. The EMAC_DMA[n]_STAT.AIS bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the EMAC_DMA[n]_IEN register: Bit 1: Transmit Process Stopped Bit 7: Receive Buffer Unavailable Bit 8: Receive Process Stopped Bit 10: Early Transmit Interrupt Bit 12: Fatal Bus Error Bit 13: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. Clear the EMAC_DMA[n]_STAT.AIS bit (by writing 1 to the EMAC_DMA[n]_STAT.AIS bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Abnormal Interrupt Summary Status Not Detected
		1	Abnormal Interrupt Summary Status Detected

Table 28-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	CDE	Context Descriptor Error. The EMAC_DMA[n]_STAT.CDE bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one's descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Context Descriptor Error Status Not Detected
		1	Context Descriptor Error Status Detected
12 (R/W)	FBE	Fatal Bus Error. The EMAC_DMA[n]_STAT.FBE bit indicates that a bus error occurred (as described in the EB field). When the EMAC_DMA[n]_STAT.FBE bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Fatal Bus Error Status Not Detected
		1	Fatal Bus Error Status Detected
11 (R/W)	ERI	Early Receive Interrupt. The EMAC_DMA[n]_STAT.ERI bit when set indicates that the RxDMA has completed the transfer of packet data to the memory. In configurations supporting ERIC, ERIC = 0: The EMAC_DMA[n]_STAT.ERI bit is set only after the Rx DMA has completely filled up a receive buffer with packet data. ERIC = 1: The EMAC_DMA[n]_STAT.ERI bit is set after every burst transfer of data from the Rx DMA to the buffer. The setting of RI bit automatically clears the EMAC_DMA[n]_STAT.ERI bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Early Receive Interrupt Status Not Detected
		1	Early Receive Interrupt Status Detected

Table 28-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	ETI	Early Transmit Interrupt. The EMAC_DMA[n]_STAT.ETI bit when set indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory. In configurations supporting ETIC: ETIC = 0: The EMAC_DMA[n]_STAT.ETI bit is set only after the Tx DMA has transferred a complete packet to MTL. ETIC = 1: The EMAC_DMA[n]_STAT.ETI bit is set after completion of (partial) packet data transfer from buffers in the transmit descriptor in which IOC = 1. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Early Transmit Interrupt Status Not Detected
		1	Early Transmit Interrupt Status Detected
9 (R/W)	RWT	Receive Watchdog Timeout. The EMAC_DMA[n]_STAT.RWT bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1) is received.	
		0	Receive Watchdog Timeout Status Not Detected
		1	Receive Watchdog Timeout Status Detected
8 (R/W)	RPS	Receive Process Stopped. The EMAC_DMA[n]_STAT.RPS bit is asserted when the Rx process enters the STOP state. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Process Stopped Status Not Detected
		1	Receive Process Stopped Status Detected

Table 28-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RBU	Receive Buffer Unavailable. The EMAC_DMA[n]_STAT.RBU bit indicates that the application owns the next descriptor in the receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the EMAC_DMA[n]_RXDSC_TLPTR register of a channel. The EMAC_DMA[n]_STAT.RBU bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Buffer Unavailable Status Not Detected
		1	Receive Buffer Unavailable Status Detected
6 (R/W)	RI	Receive Interrupt. The EMAC_DMA[n]_STAT.RI bit indicates that the packet reception is complete. When packet reception is complete, bit 31 of RDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the running state. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Interrupt Status Not Detected
		1	Receive Interrupt Status Detected
2 (R/W)	TBU	Transmit Buffer Unavailable. The EMAC_DMA[n]_STAT.TBU bit indicates that the application owns the next descriptor in the transmit list, and the DMA cannot acquire it. Transmission is suspended. The EMAC_DMA_DBG_STAT0.TPS0 field indicates the transmit process state transitions. To resume processing the transmit descriptors, the application must: 1. Change the ownership of the descriptor by setting bit 31 of TDES3. 2. Issue a Transmit Poll Demand command. 3. For ring mode, the application should advance the EMAC_DMA[n]_TXDSC_TLPTR register of a channel. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Buffer Unavailable Status Not Detected
		1	Transmit Buffer Unavailable Status Detected

Table 28-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	TPS	Transmit Process Stopped. The `EMAC_DMA[n]_STAT.TPS` bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Process Stopped Status Not Detected
		1	Transmit Process Stopped Status Detected
0 (R/W)	TI	Transmit Interrupt. The `EMAC_DMA[n]_STAT.TI` bit indicates that the packet transmission is complete. When transmission is complete, bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Interrupt Status Not Detected
		1	Transmit Interrupt Status Detected

DMA Channel n Transmit Descriptor List Address Register

The EMAC_DMA[n]_TXDSC_ADDR register points the DMA to the start of transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be 32-bit aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Write to the EMAC_DMA[n]_TXDSC_ADDR register only when the Tx DMA has stopped (EMAC_DMA[n]_TXCTL.ST = 0). When stopped, this register can be written with a new descriptor list address. When EMAC_DMA[n]_TXCTL.ST = 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when EMAC_DMA[n]_TXCTL.ST = 0, the DMA takes the descriptor address where it was stopped earlier.

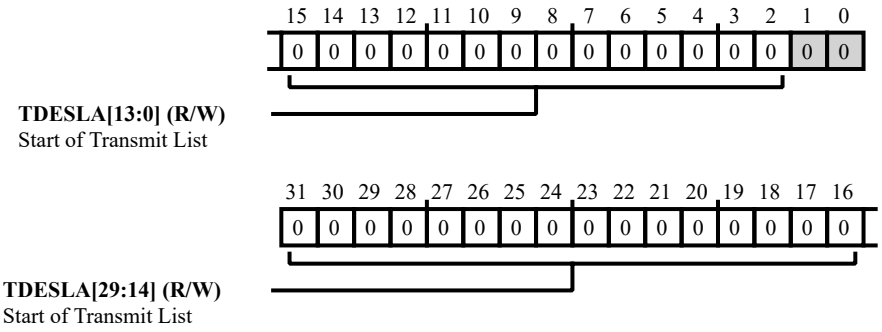


Figure 28-72: EMAC_DMA[n]_TXDSC_ADDR Register Diagram

Table 28-127: EMAC_DMA[n]_TXDSC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	TDESLA	Start of Transmit List. The EMAC_DMA[n]_TXDSC_ADDR.TDESLA field contains the base address of the first descriptor in the transmit descriptor list. The DMA ignores the LSB bits (1:0) and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

DMA Channel n Transmit Descriptor Ring Length Register

The EMAC_DMA[n]_TXDSC_RLEN register contains the length of the transmit descriptor ring.

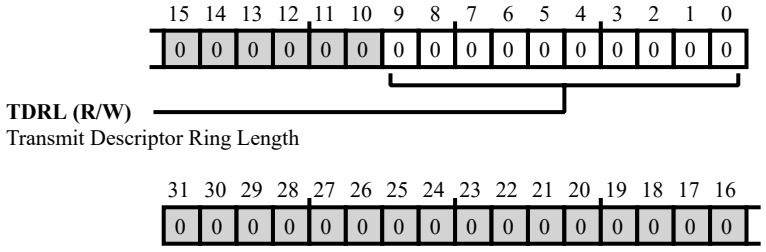


Figure 28-73: EMAC_DMA[n]_TXDSC_RLEN Register Diagram

Table 28-128: EMAC_DMA[n]_TXDSC_RLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	TDRL	Transmit Descriptor Ring Length. The EMAC_DMA[n]_TXDSC_RLEN.TDRL field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. A minimum ring descriptor length of 4 is recommended. For example, program any value up to 0x3FF in this field. The EMAC_DMA[n]_TXDSC_RLEN.TDRL field is 10 bits wide. To program 0x3FF, 1024 descriptors are available. For 10 descriptors, program it to a value of 0x9.

DMA Channel n Transmit Descriptor Tail Pointer Register

The EMAC_DMA[n]_TXDSC_TLPTR register points to an offset from the base and indicates the location of the last valid descriptor.

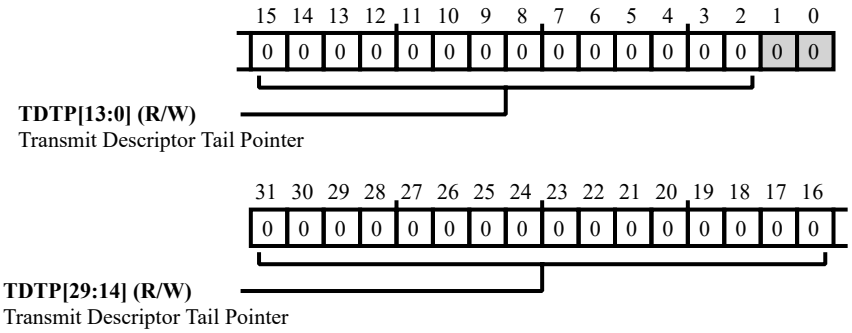


Figure 28-74: EMAC_DMA[n]_TXDSC_TLPTR Register Diagram

Table 28-129: EMAC_DMA[n]_TXDSC_TLPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	TDTP	Transmit Descriptor Tail Pointer. The EMAC_DMA[n]_TXDSC_TLPTR.TDTP field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

DMA Channel n Transmit Control Register

The EMAC_DMA[n]_TXCTL register controls the Tx features such as PBL, TCP segmentation, and Tx channel weights.

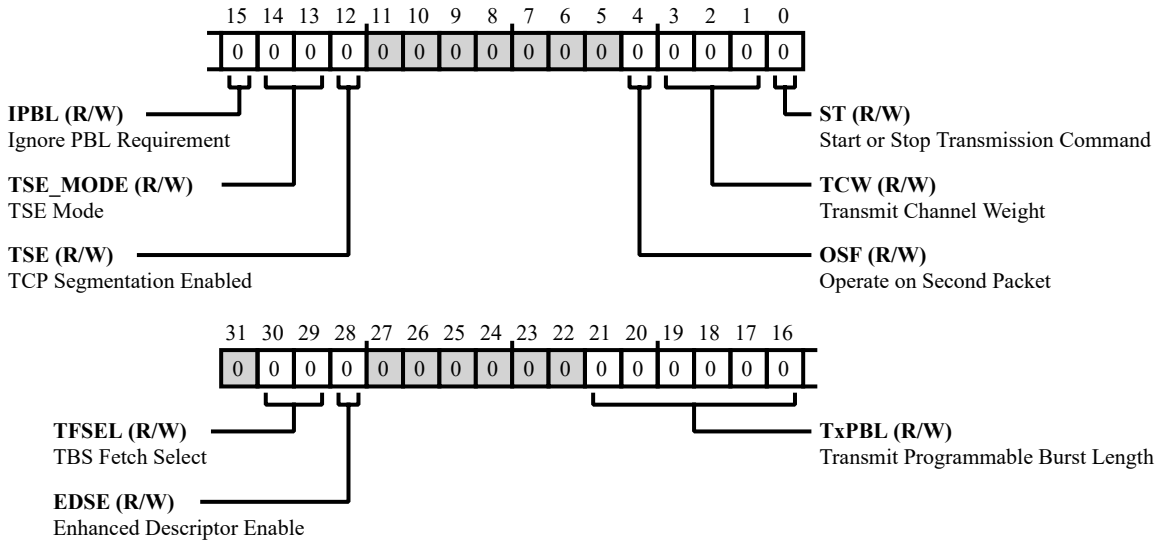


Figure 28-75: EMAC_DMA[n]_TXCTL Register Diagram

Table 28-130: EMAC_DMA[n]_TXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/W)	TFSEL	TBS Fetch Select. The EMAC_DMA[n]_TXCTL.TFSEL field select the EMAC_DMA_TBS_CTRL0 - EMAC_DMA_TBS_CTRL3 register fields (FTOS, FGSN, FTOV) for the channel.	
		0	EMAC_DMA_TBS_CTL0
		1	EMAC_DMA_TBS_CTL1
		2	EMAC_DMA_TBS_CTL2
		3	EMAC_DMA_TBS_CTL3
28 (R/W)	EDSE	Enhanced Descriptor Enable. When EMAC_DMA[n]_TXCTL.EDSE = 1, the corresponding channel uses enhanced descriptors that are 32 bytes for both normal and context descriptors. When cleared (=0), the corresponding channel uses the descriptors that are 16 bytes.	
		0	Enhanced Descriptor is Disabled
		1	Enhanced Descriptor is Enabled

Table 28-130: EMAC_DMA[n]_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/W)	TXPBL	Transmit Programmable Burst Length. The `EMAC_DMA[n]_TXCTL.TXPBL` field indicates the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. Program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in EMAC_DMA[n]_CTL register. 2. Set the TxPBL. Note: The maximum value of TxPBL must be less than or equal to half the Tx queue size (EMAC_TQ[n]_OPMODE.TQS) in terms of beats. This is required so that the Tx queue has space to store at least another Tx PBL worth of data while the MTL Tx queue controller is transferring data to MAC. For example, in 64-bit data width configurations the total locations in Tx queue of size 512 bytes is 64, TxPBL and 8xPBL must be programmed to less than or equal to 32.	
15 (R/W)	IPBL	Ignore PBL Requirement. When `EMAC_DMA[n]_TXCTL.IPBL` = 1, the DMA does not check for PBL number of locations in the MTL before initiating a transfer. If space is not available, the MTL can use handshaking to slow the DMA. Note: The `EMAC_DMA[n]_TXCTL.IPBL` bit/mode must not be used when multiple transmit DMA channels are enabled as it might block other transmit and receive DMA channels from accessing the read data channel of SCB bus until space is available in transmit queue for the current transfer.	
		0	Ignore PBL Requirement is Disabled
		1	Ignore PBL Requirement is Enabled
14:13 (R/W)	TSE_MODE	TSE Mode. The `EMAC_DMA[n]_TXCTL.TSE_MODE` field indicates the TSE mode.	
		0	TSO/USO. Segmentation functionality is enabled.
		1	UFO with Checksum. Enables the UDP fragmentation functionality with checksum for all the UDP packets.
		2	UFO Without Checksum. Enables the UDP fragmentation functionality without Checksum for all the UDP packets.
		3	Reserved

Table 28-130: EMAC_DMA[n]_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	TSE	TCP Segmentation Enabled. When EMAC_DMA[n]_TXCTL.TSE = 1, the DMA performs the TCP segmentation or UDP segmentation/fragmentation for packets in this channel. The TCP segmentation or UDP packet's segmentation/fragmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When EMAC_DMA[n]_TXCTL.TSE = 1, the TxPBL value must be greater than 4.	
		0	TCP Segmentation is Disabled
		1	TCP Segmentation is Enabled
4 (R/W)	OSF	Operate on Second Packet. When EMAC_DMA[n]_TXCTL.OSF = 1, the DMA processes the second packet of the transmit data, even before the status for the first packet is obtained.	
		0	Operate on Second Packet Disabled
		1	Operate on Second Packet Enabled
3:1 (R/W)	TCW	Transmit Channel Weight. The EMAC_DMA[n]_TXCTL.TCW field indicates the weight assigned to the corresponding transmit channel. When reset is complete, EMAC_DMA[n]_TXCTL.TCW = 0 for all channels by default, resulting in equal weights to all channels.	

Table 28-130: EMAC_DMA[n]_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	ST	Start or Stop Transmission Command. When the EMAC_DMA[n]_TXCTL.ST bit is set, transmission is placed in the running state. The DMA checks the transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: The current position in the list - This is the base address of the transmit list configured by the EMAC_DMA[n]_TXDSC_ADDR register. The position at which the transmission was previously stopped - If the DMA does not own the current descriptor, the transmission enters the suspended state and the EMAC_DMA[n]_STAT.TBU =1. The start transmission command is effective only when the transmission is stopped. If the command is issued before configuring the EMAC_DMA[n]_TXDSC_ADDR register, the DMA behavior is unpredictable. When EMAC_DMA[n]_TXCTL.ST = 0, the transmission process is placed in the stopped state after completing the transmission of the current packet. The Next Descriptor position in the transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, program EMAC_DMA[n]_TXDSC_ADDR register with a new value when EMAC_DMA[n]_TXCTL.ST = 0. The new value is considered when EMAC_DMA[n]_TXCTL.ST =1 again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the suspended state.	
		0	Stop Transmission Command
		1	Start Transmission Command

DMA Debug Status 0 Register

The EMAC_DMA_DBG_STAT0 register gives the receive and transmit process status for DMA channels 0-2 for debugging purpose.

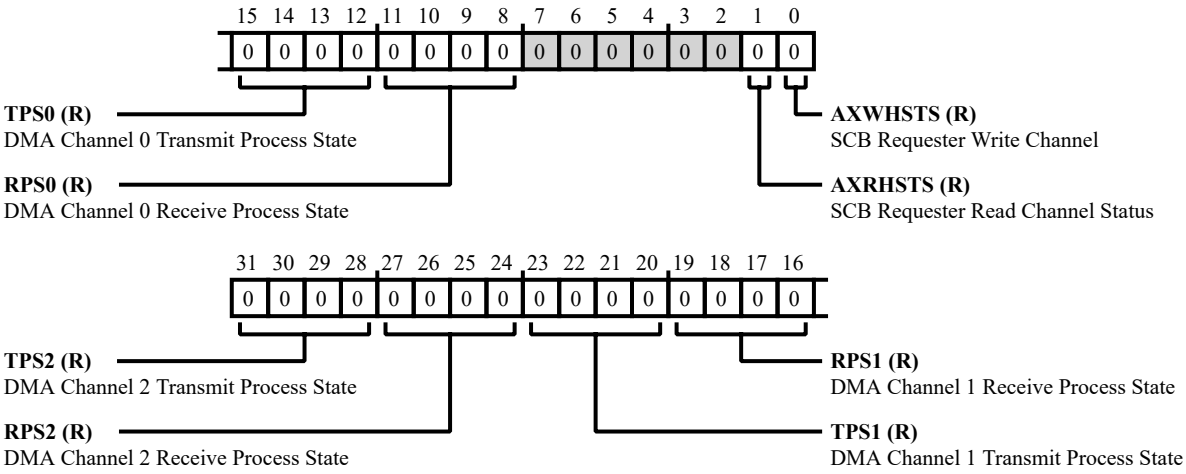


Figure 28-76: EMAC_DMA_DBG_STAT0 Register Diagram

Table 28-131: EMAC_DMA_DBG_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	TPS2	DMA Channel 2 Transmit Process State. The EMAC_DMA_DBG_STAT0.TPS2 field indicates the Tx DMA FSM state for channel 2. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT0.TPS2 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)

Table 28-131: EMAC_DMA_DBG_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27:24 (R/NW)	RPS2	DMA Channel 2 Receive Process State. The EMAC_DMA_DBG_STAT0.RPS2 field indicates the Rx DMA FSM state for channel 2. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT0.RPS2 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
23:20 (R/NW)	TPS1	DMA Channel 1 Transmit Process State. The EMAC_DMA_DBG_STAT0.TPS1 field indicates the Tx DMA FSM state for channel 1. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT0.TPS1 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
19:16 (R/NW)	RPS1	DMA Channel 1 Receive Process State. The EMAC_DMA_DBG_STAT0.RPS1 field indicates the Rx DMA FSM state for channel 1. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT0.RPS1 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved

Table 28-131: EMAC_DMA_DBG_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
15:12 (R/NW)	TPS0	DMA Channel 0 Transmit Process State. The `EMAC_DMA_DBG_STAT0.TPS0` field indicates the Tx DMA FSM state for channel 0. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT0.TPS0` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
11:8 (R/NW)	RPS0	DMA Channel 0 Receive Process State. The `EMAC_DMA_DBG_STAT0.RPS0` field indicates the Rx DMA FSM state for channel 0. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT0.RPS0` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)

Table 28-131: EMAC_DMA_DBG_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	AXRHSTS	SCB Requester Read Channel Status. When high, the `EMAC_DMA_DBG_STAT0.AXRHSTS` bit indicates that the read channel of the SCB requester is active, and it is transferring the data.	
		0	Status Not Detected
		1	Status Detected
0 (R/NW)	AXWHSTS	SCB Requester Write Channel. When high, the `EMAC_DMA_DBG_STAT0.AXWHSTS` bit indicates that the write channel of the SCB requester is active, and it is transferring data.	
		0	Status Not Detected
		1	Status Detected

DMA Debug Status 1 Register

The EMAC_DMA_DBG_STAT1 register gives the receive and transmit process status for DMA channels 3-6.

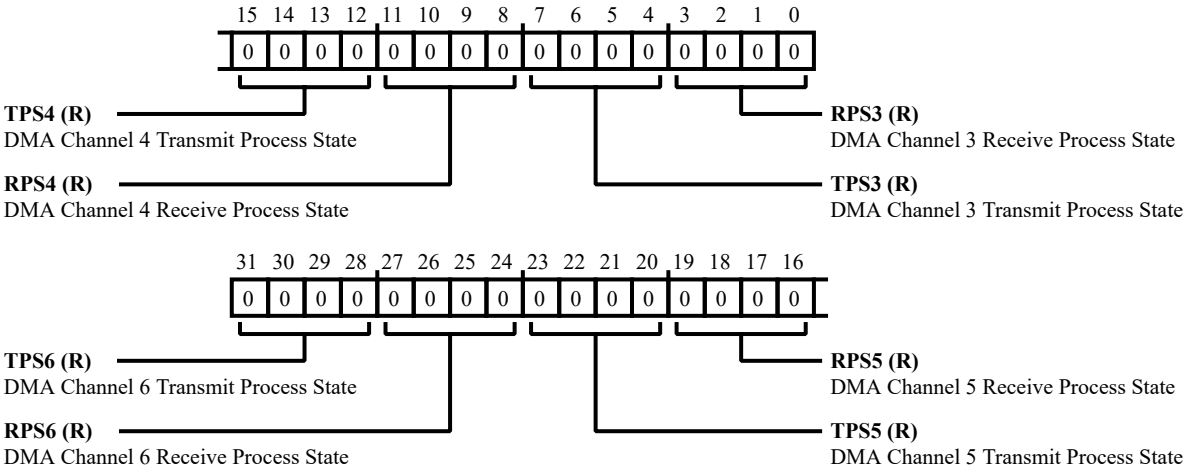


Figure 28-77: EMAC_DMA_DBG_STAT1 Register Diagram

Table 28-132: EMAC_DMA_DBG_STAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	TPS6	DMA Channel 6 Transmit Process State. The EMAC_DMA_DBG_STAT1.TPS6 field indicates the Tx DMA FSM state for channel 6. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT1.TPS6 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
27:24 (R/NW)	RPS6	DMA Channel 6 Receive Process State. The EMAC_DMA_DBG_STAT1.RPS6 field indicates the Rx DMA FSM state for channel 6. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT1.RPS6 field does not generate an interrupt.	

Table 28-132: EMAC_DMA_DBG_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
23:20 (R/NW)	TPS5	DMA Channel 5 Transmit Process State. The `EMAC_DMA_DBG_STAT1.TPS5` field indicates the Tx DMA FSM state for channel 5. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.TPS5` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
19:16 (R/NW)	RPS5	DMA Channel 5 Receive Process State. The `EMAC_DMA_DBG_STAT1.RPS5` field indicates the Rx DMA FSM state for channel 5. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.RPS5` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)

Table 28-132: EMAC_DMA_DBG_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
15:12 (R/NW)	TPS4	DMA Channel 4 Transmit Process State. The `EMAC_DMA_DBG_STAT1.TPS4` field indicates the Tx DMA FSM state for channel 4. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.TPS4` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
11:8 (R/NW)	RPS4	DMA Channel 4 Receive Process State. The `EMAC_DMA_DBG_STAT1.RPS4` field indicates the Rx DMA FSM state for channel 4. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.RPS4` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
7:4 (R/NW)	TPS3	DMA Channel 3 Transmit Process State. The `EMAC_DMA_DBG_STAT1.TPS3` field indicates the Tx DMA FSM state for channel 3. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.TPS3` field does not generate an interrupt.	

Table 28-132: EMAC_DMA_DBG_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
3:0 (R/NW)	RPS3	DMA Channel 3 Receive Process State. The `EMAC_DMA_DBG_STAT1.RPS3` field indicates the Rx DMA FSM state for channel 3. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.RPS3` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)

DMA Debug Status 2 Register

The EMAC_DMA_DBG_STAT2 register gives the receive and transmit process status for DMA channel 7.

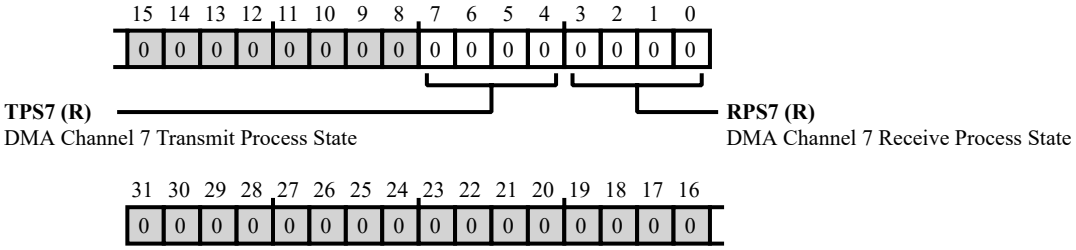


Figure 28-78: EMAC_DMA_DBG_STAT2 Register Diagram

Table 28-133: EMAC_DMA_DBG_STAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	TPS7	DMA Channel 7 Transmit Process State. The EMAC_DMA_DBG_STAT2.TPS7 field indicates the Tx DMA FSM state for channel 7. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT2.TPS7 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
3:0 (R/NW)	RPS7	DMA Channel 7 Receive Process State. The EMAC_DMA_DBG_STAT2.RPS7 field indicates the Rx DMA FSM state for channel 7. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT2.RPS7 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)

Table 28-133: EMAC_DMA_DBG_STAT2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)

DMA Interrupt Status Register

The application reads the EMAC_DMA_ISTAT register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

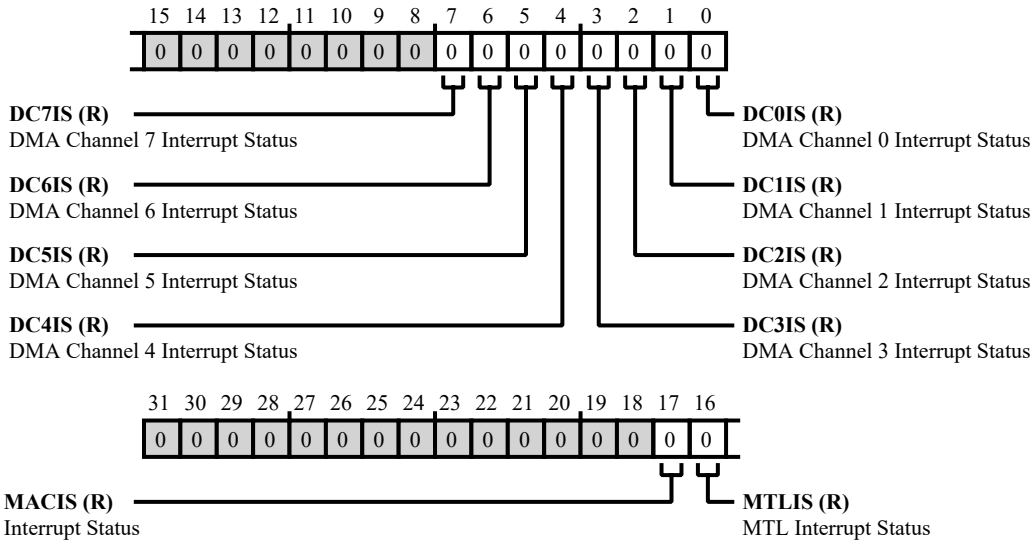


Figure 28-79: EMAC_DMA_ISTAT Register Diagram

Table 28-134: EMAC_DMA_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	MACIS	Interrupt Status. The EMAC_DMA_ISTAT.MACIS bit indicates an interrupt event in the MAC. To clear the EMAC_DMA_ISTAT.MACIS bit (=0), the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.	
		0	MAC Interrupt Status Not Detected
		1	MAC Interrupt Status Detected
16 (R/NW)	MTLIS	MTL Interrupt Status. The EMAC_DMA_ISTAT.MTLIS bit indicates an interrupt event in the MTL. To clear the EMAC_DMA_ISTAT.MTLIS bit (=0), the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.	
		0	MTL Interrupt Status Not Detected
		1	MTL Interrupt Status Detected

Table 28-134: EMAC_DMA_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	DC7IS	DMA Channel 7 Interrupt Status. The EMAC_DMA_ISTAT.DC7IS bit indicates an interrupt event in DMA channel 7. To clear the EMAC_DMA_ISTAT.DC7IS bit (=0), the software must read the corresponding register in DMA channel 7 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 7 Interrupt Status Not Detected
		1	DMA Channel 7 Interrupt Status Detected
6 (R/NW)	DC6IS	DMA Channel 6 Interrupt Status. The EMAC_DMA_ISTAT.DC6IS bit indicates an interrupt event in DMA channel 6. To clear the EMAC_DMA_ISTAT.DC6IS bit (=0), the software must read the corresponding register in DMA channel 6 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 6 Interrupt Status Not Detected
		1	DMA Channel 6 Interrupt Status Detected
5 (R/NW)	DC5IS	DMA Channel 5 Interrupt Status. The EMAC_DMA_ISTAT.DC5IS bit indicates an interrupt event in DMA channel 5. To clear the EMAC_DMA_ISTAT.DC5IS bit (=0), the software must read the corresponding register in DMA channel 5 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 5 Interrupt Status Not Detected
		1	DMA Channel 5 Interrupt Status Detected
4 (R/NW)	DC4IS	DMA Channel 4 Interrupt Status. The EMAC_DMA_ISTAT.DC4IS bit indicates an interrupt event in DMA channel 4. To clear the EMAC_DMA_ISTAT.DC4IS bit (=0), the software must read the corresponding register in DMA channel 4 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 4 Interrupt Status Not Detected
		1	DMA Channel 4 Interrupt Status Detected
3 (R/NW)	DC3IS	DMA Channel 3 Interrupt Status. The EMAC_DMA_ISTAT.DC3IS bit indicates an interrupt event in DMA channel 3. To clear the EMAC_DMA_ISTAT.DC3IS bit (=0), the software must read the corresponding register in DMA channel 3 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 3 Interrupt Status Not Detected
		1	DMA Channel 3 Interrupt Status Detected

Table 28-134: EMAC_DMA_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	DC2IS	DMA Channel 2 Interrupt Status. The `EMAC_DMA_ISTAT.DC2IS` bit indicates an interrupt event in DMA channel 2. To clear the `EMAC_DMA_ISTAT.DC2IS` bit (=0), the software must read the corresponding register in DMA channel 2 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 2 Interrupt Status Not Detected
		1	DMA Channel 2 Interrupt Status Detected
1 (R/NW)	DC1IS	DMA Channel 1 Interrupt Status. The `EMAC_DMA_ISTAT.DC1IS` bit indicates an interrupt event in DMA channel 1. To clear the `EMAC_DMA_ISTAT.DC1IS` bit (=0), the software must read the corresponding register in DMA channel 1 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 1 Interrupt Status Not Detected
		1	DMA Channel 1 Interrupt Status Detected
0 (R/NW)	DC0IS	DMA Channel 0 Interrupt Status. The `EMAC_DMA_ISTAT.DC0IS` bit indicates an interrupt event in DMA channel 0. To clear the `EMAC_DMA_ISTAT.DC0IS` bit (=0), the software must read the corresponding register in DMA channel 0 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 0 Interrupt Status Not Detected
		1	DMA Channel 0 Interrupt Status Detected

DMA Mode Register

The EMAC_DMA_MODE register establishes the bus operating modes for the DMA.

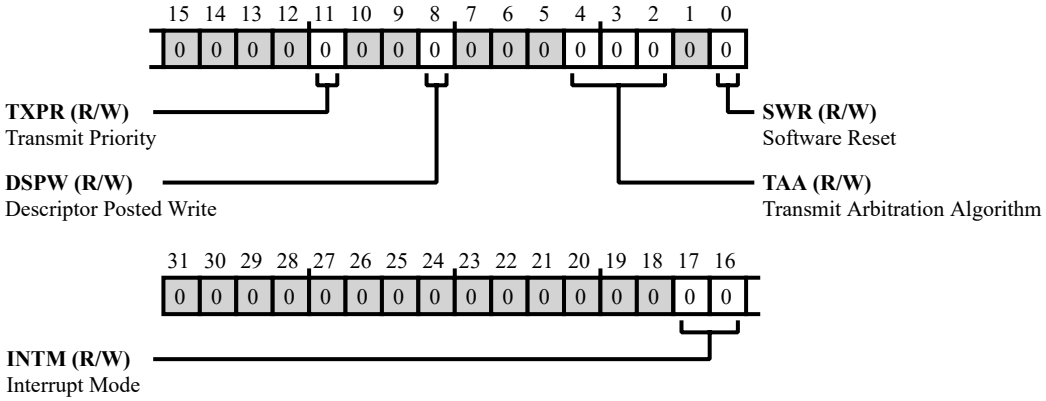


Figure 28-80: EMAC_DMA_MODE Register Diagram

Table 28-135: EMAC_DMA_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	INTM	Interrupt Mode. The `EMAC_DMA_MODE.INTM` field defines the interrupt mode of the EMAC module. The behavior of the following outputs change depending on the following settings: - EMAC_DMA[n]_TX (Transmit Per Channel Interrupt) - EMAC_DMA[n]_RX (Receive Per Channel Interrupt) - EMAC_STAT (Common Interrupt) It also changes the behavior of the EMAC_DMA[n]_STAT.RI and EMAC_DMA[n]_STAT.TI bits. - 00: EMAC_DMA[n]_TX/RX are pulse signals for each TX/RX packet transfer completion events (irrespective of whether corresponding interrupts are enabled) for which IOC bits are enabled in descriptor. EMAC_STAT is also asserted when corresponding interrupts are enabled and cleared only when software clears the corresponding RI/TI status bits. - 01: EMAC_DMA[n]_TX/RX are level signals asserted on TX/RX packet transfer completion event when corresponding interrupts are enabled and deasserted when the software clears the corresponding RI/TI status bits. The EMAC_STAT is not asserted for these TX/RX packet transfer completion events. - 10: EMAC_DMA[n]_TX/RX are level signals asserted on TX/RX packet transfer completion event when corresponding interrupts are enabled and deasserted when the software clears the corresponding RI/TI status bits. However, the signal is asserted again if the same event occurred again before it was cleared. The EMAC_STAT is not asserted for these TX/RX packet transfer completion events. - 11: Reserved For more details, see Transfer Complete Interrupt Behavior.	
		0	See Above Description
		1	See Above Description
		2	See Above Description
		3	Reserved
11 (R/W)	TXPR	Transmit Priority. When set, the `EMAC_DMA_MODE.TXPR` bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.	
		0	Transmit Priority is Disabled
		1	Transmit Priority is Enabled

Table 28-135: EMAC_DMA_MODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	DSPW	Descriptor Posted Write. When the EMAC_DMA_MODE.DSPW bit is cleared (=0), it indicates that the descriptor writes are always non-posted. When set (=1), the EMAC_DMA_MODE.DSPW bit indicates that the descriptor is non-posted only when the IOC (Interrupt on Completion) is set in last descriptor. Otherwise, the descriptor writes are always posted.	
		0	Disable
		1	Enable
4:2 (R/W)	TAA	Transmit Arbitration Algorithm. The EMAC_DMA_MODE.TAA field is used to select the arbitration algorithm for the transmit side when multiple Tx DMAs are selected.	
		0	Fixed Priority (Channel 0 Has the Lowest Priority and the Last Channel Has the Highest Priority)
		1	Weighted Strict Priority (WSP)
		2	Weighted Round-Robin (WRR)
		3	Reserved
0 (R/W)	SWR	Software Reset. When the EMAC_DMA_MODE.SWR bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. The EMAC_DMA_MODE.SWR bit is automatically cleared after the reset operation is complete in all the EMAC clock domains. Before reprogramming any the EMAC register, a value of zero should be read in the EMAC_DMA_MODE.SWR bit. The EMAC_DMA_MODE.SWR bit must be read at least 4 SCLK0 clock cycles after it is written to 1. Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Software Reset is Disabled
		1	Software Reset is Enabled

DMA System Bus Mode Register

The EMAC_DMA_SYSBMODE register controls the behavior of the SPB or SCB requester. It mainly controls burst splitting and number of outstanding requests.

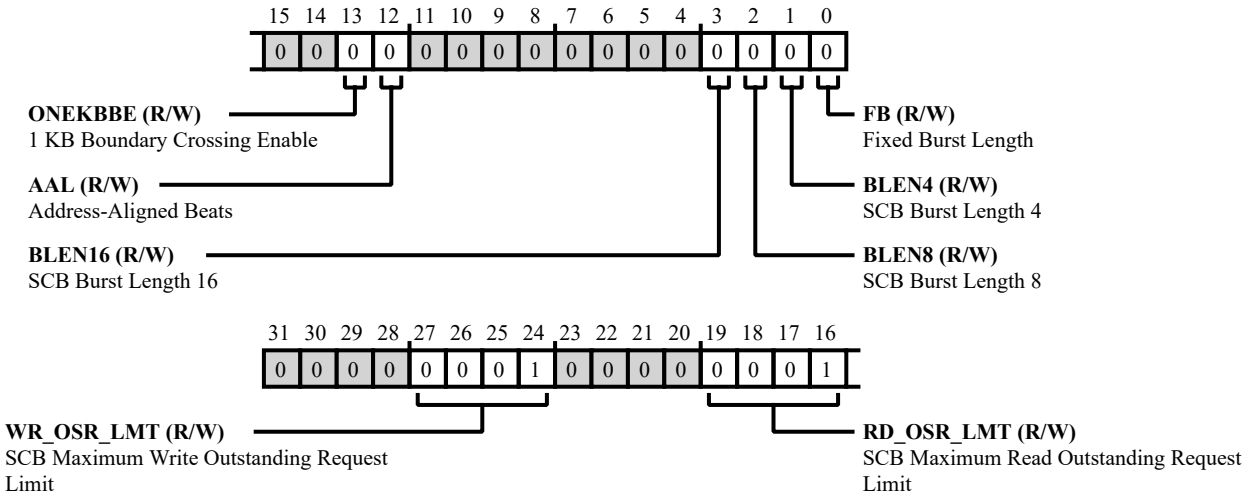


Figure 28-81: EMAC_DMA_SYSBMODE Register Diagram

Table 28-136: EMAC_DMA_SYSBMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27:24 (R/W)	WR_OSR_LMT	SCB Maximum Write Outstanding Request Limit. This value limits the maximum outstanding request on the SCB write interface. Maximum outstanding requests = EMAC_DMA_SYSBMODE.WR_OSR_LMT + 1	
19:16 (R/W)	RD_OSR_LMT	SCB Maximum Read Outstanding Request Limit. This value limits the maximum outstanding request on the SCB read interface. Maximum outstanding requests = EMAC_DMA_SYSBMODE.RD_OSR_LMT + 1.	
13 (R/W)	ONEKBBE	1 KB Boundary Crossing Enable. The EMAC_DMA_SYSBMODE.ONEKBBE bit indicates whether the burst transfers performed by the SCB requester do not cross a 1 KB or (the default) 4 KB boundary.	
		0	Disable
		1	Enable

Table 28-136: EMAC_DMA_SYSBMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	AAL	Address-Aligned Beats. When the EMAC_DMA_SYSBMODE.AAL = 1, the SCB or SPB requester performs address-aligned burst transfers on read and write channels. When the EMAC_DMA_SYSBMODE.AAL = 0, the SCB or SPB requester performs burst transfers on read and write channels without aligning to address boundaries.	
		0	Address-Aligned Beats is Disabled
		1	Address-Aligned Beats is Enabled
3 (R/W)	BLEN16	SCB Burst Length 16. When EMAC_DMA_SYSBMODE.BLEN16 = 1 or EMAC_DMA_SYSBMODE.FB = 1, the SCB requester can select a burst length of 16 on the SCB interface. When EMAC_DMA_SYSBMODE.FB = 0, writing to the EMAC_DMA_SYSBMODE.BLEN16 bit has no effect.	
		0	Disable
		1	Enable
2 (R/W)	BLEN8	SCB Burst Length 8. When EMAC_DMA_SYSBMODE.BLEN8 = 1 or EMAC_DMA_SYSBMODE.FB = 1, the SCB requester can select a burst length of 8 on the SCB interface. When EMAC_DMA_SYSBMODE.FB = 0, writing to the EMAC_DMA_SYSBMODE.BLEN8 bit has no effect.	
		0	Disable
		1	Enable
1 (R/W)	BLEN4	SCB Burst Length 4. When EMAC_DMA_SYSBMODE.BLEN4 = 1 or EMAC_DMA_SYSBMODE.FB = 1, the SCB requester can select a burst length of 4 on the SCB interface. When EMAC_DMA_SYSBMODE.FB = 0, writing to the EMAC_DMA_SYSBMODE.BLEN4 bit has no effect.	
		0	Disable
		1	Enable

Table 28-136: EMAC_DMA_SYSBMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	FB	Fixed Burst Length. The EMAC_DMA_SYSBMODE.FB bit enables a fixed burst length. When EMAC_DMA_SYSBMODE.FB= 1, the SCB requester initiates burst lengths as indicated by the BLEN16, BLEN8, or BLEN4 field -- Burst transfers of length 1. When EMAC_DMA_SYSBMODE.FB = 0, the SCB requester initiates burst transfers that are equal to or less than the maximum allowed burst length programmed in bits[7:4].	
		0	Fixed Burst Length is Disabled
		1	Fixed Burst Length is Enabled

DMA TBS Control 0 Register

The EMAC_DMA_TBS_CTL0 register is used to control the Time Based Scheduling (TBS) attributes.

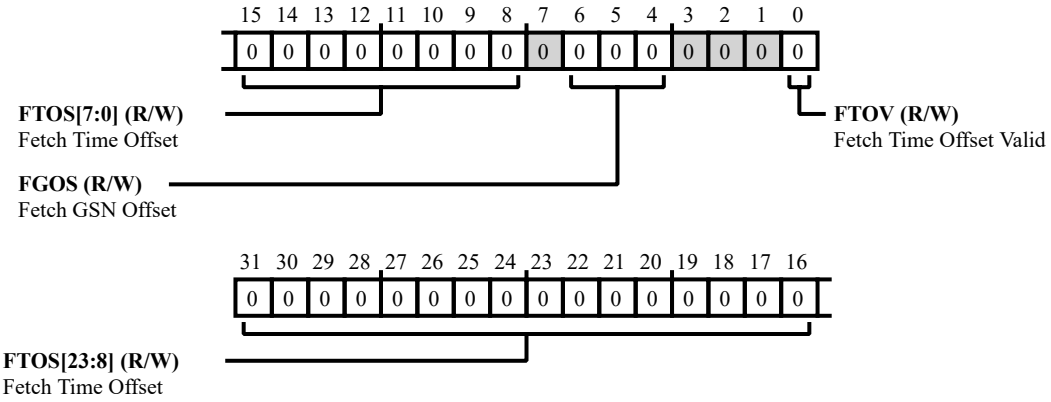


Figure 28-82: EMAC_DMA_TBS_CTL0 Register Diagram

Table 28-137: EMAC_DMA_TBS_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The EMAC_DMA_TBS_CTL0.FTOS field indicates the value in units of 256 nano-seconds, that has to be deducted from the launch time to compute the fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The EMAC_DMA_TBS_CTL0.FGOS field indicates the number of GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL0.FTOV is set.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL0.FTOV bit indicates that the EMAC_DMA_TBS_CTL0.FTOS field is valid. When cleared (=0), the EMAC_DMA_TBS_CTL0.FTOV bit indicates that the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

DMA TBS Control 1 Register

The EMAC_DMA_TBS_CTL1 register is used to control the Time Based Scheduling (TBS) attributes.

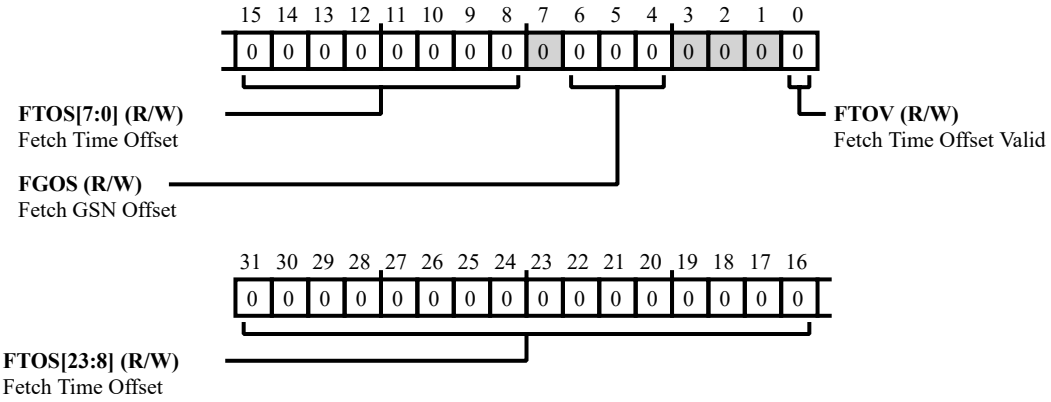


Figure 28-83: EMAC_DMA_TBS_CTL1 Register Diagram

Table 28-138: EMAC_DMA_TBS_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The value in units of 256 nanoseconds, that has to be deducted from the Launch time to compute the Fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The number GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL1.FTOV= 1.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL1.FTOV bit indicates that the EMAC_DMA_TBS_CTL1.FTOS field is valid. When not set, indicates the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

DMA TBS Control 2 Register

The EMAC_DMA_TBS_CTL2 register is used to control the Time Based Scheduling (TBS) attributes.

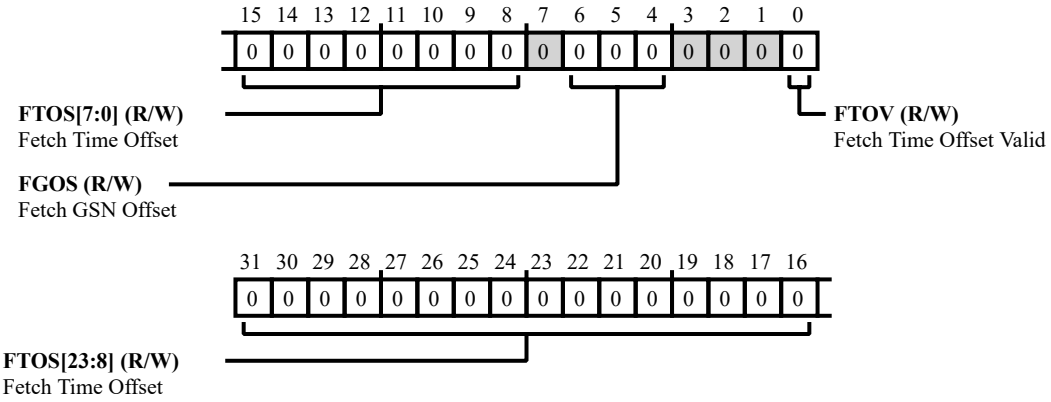


Figure 28-84: EMAC_DMA_TBS_CTL2 Register Diagram

Table 28-139: EMAC_DMA_TBS_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The value in units of 256 nanoseconds, that has to be deducted from the Launch time to compute the Fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The EMAC_DMA_TBS_CTL2.FGOS field indicates the number GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL2.FTOV= 1.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL2.FTOV bit indicates that the EMAC_DMA_TBS_CTL2.FTOS field is valid. When not set, indicates the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

DMA TBS Control 3 Register

The EMAC_DMA_TBS_CTL3 register is used to control the TBS attributes.

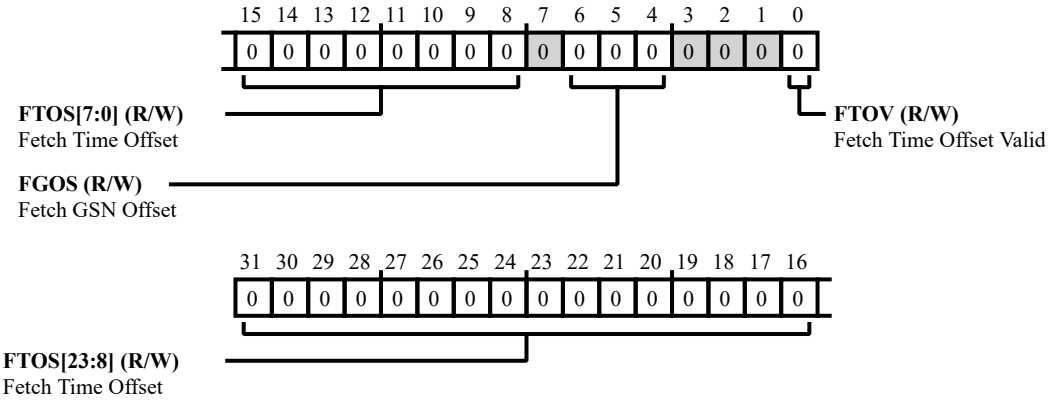


Figure 28-85: EMAC_DMA_TBS_CTL3 Register Diagram

Table 28-140: EMAC_DMA_TBS_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The value in units of 256 nanoseconds, that has to be deducted from the Launch time to compute the Fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The EMAC_DMA_TBS_CTL3.FGOS field indicates the number of GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL3.FTOV is set.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL3.FTOV bit indicates that the EMAC_DMA_TBS_CTL3.FTOS field is valid. When cleared (=0), the EMAC_DMA_TBS_CTL3.FTOV bit indicates that the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

LPI One Microsecond Tic Counter Register

The EMAC_ONEUS_TIC_CNT register controls the generation of the reference time (1 microsecond tic) for all of the LPI timers. This timer has to be programmed by the software initially.

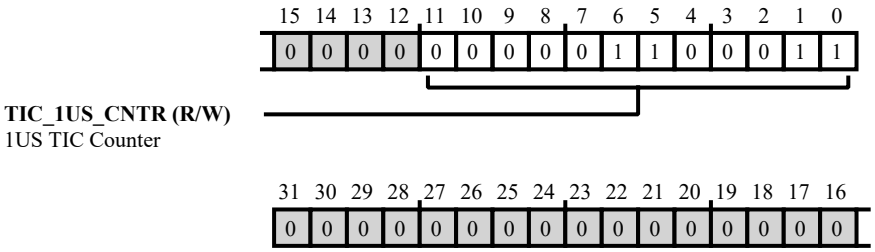


Figure 28-86: EMAC_ONEUS_TIC_CNT Register Diagram

Table 28-141: EMAC_ONEUS_TIC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	TIC_1US_CNTR	1US TIC Counter. The application must program the EMAC_ONEUS_TIC_CNT.TIC_1US_CNTR field so that the number of clock cycles of SCLK0 is 1us. (Subtract 1 from the value before programming). For example, if the SCLK0 is 100MHz, this field must be programmed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1 us events that are used to update some of the EEE-related counters.

Address Resolution Protocol (ARP) Register

The EMAC_ARP_ADDR register contains the IPv4 destination address of the MAC.

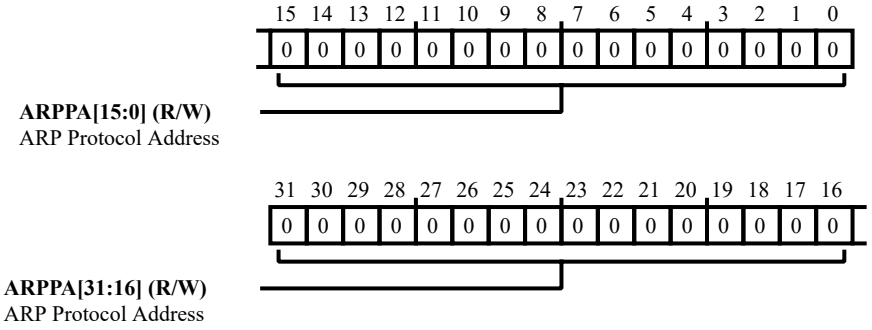


Figure 28-87: EMAC_ARP_ADDR Register Diagram

Table 28-142: EMAC_ARP_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ARPPA	ARP Protocol Address. The EMAC_ARP_ADDR.ARPPA field contains the IPv4 destination address of the MAC. This address is used for perfect match with the protocol address of the target field in the received ARP packet.

MAC Address 0 High Register

The EMAC_ADDR0_HI register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the EMAC_ADDR0_LO register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MAC Address 0 Register [47:0] is compared with 0x665544332211.

The MAC address registers are configured to be double-synchronized to the (G)MII clock domains. Hence, the synchronization is triggered only when bits[31:24] (in little endian mode) or bits[7:0] (in big endian mode) of the EMAC_ADDR0_LO register are written. For proper synchronization updates, the consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

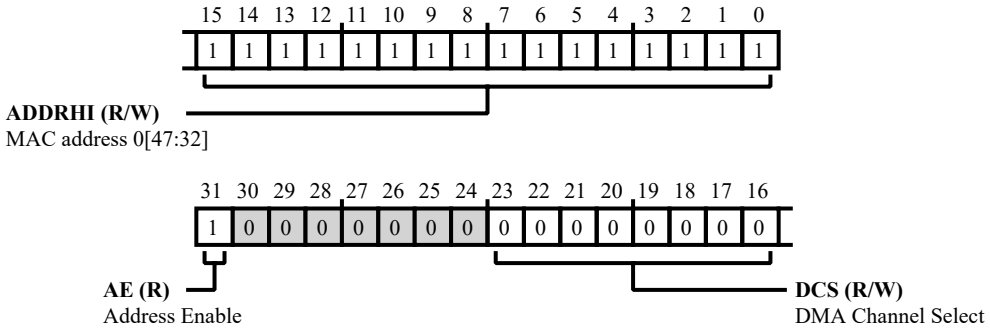


Figure 28-88: EMAC_ADDR0_HI Register Diagram

Table 28-143: EMAC_ADDR0_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	AE	Address Enable. The EMAC_ADDR0_HI.AE bit is always set (=1).	
		0	Invalid
		1	The bit is always set
23:16 (R/W)	DCS	DMA Channel Select. When EMAC_MAC_EXT_CFG.PDC = 0, the EMAC_ADDR0_HI.DCS field contains the binary representation of the DMA channel number to which an Rx packet whose DA matches the MAC Address 0 content is routed. When EMAC_MAC_EXT_CFG.PDC = 1, the EMAC_ADDR0_HI.DCS field contains the one-hot representation of one or more DMA channel numbers to which an Rx packet whose DA matches the MAC Address 0 content is routed.	
15:0 (R/W)	ADDRHI	MAC address 0[47:32]. The EMAC_ADDR0_HI.ADDRHI field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.	

MAC Address 0 Low Register

The EMAC_ADDR0_LO register holds the lower 32 bits of the 6-byte first MAC address of the station.

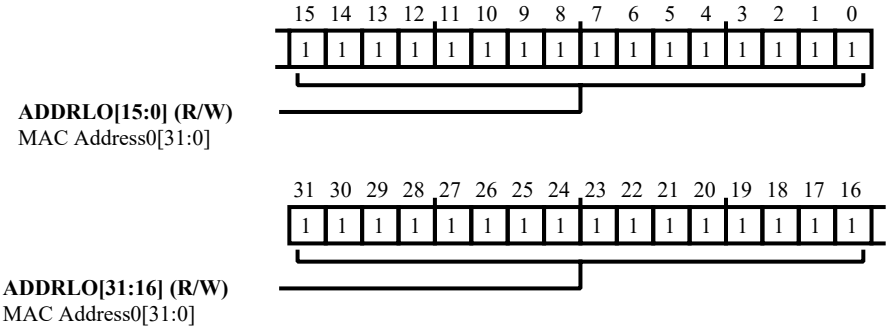


Figure 28-89: EMAC_ADDR0_LO Register Diagram

Table 28-144: EMAC_ADDR0_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRLO	MAC Address0[31:0]. The EMAC_ADDR0_LO.ADDRLO field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address n High Register

The EMAC_ADDR[i]_HI register holds the upper 16 bits of the second 6-byte MAC address of the station.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the EMAC_ADDR0_LO register are written. For proper synchronization updates, the consecutive writes to this EMAC_ADDR0_LO register should be performed after at least four clock cycles in the destination clock domain.

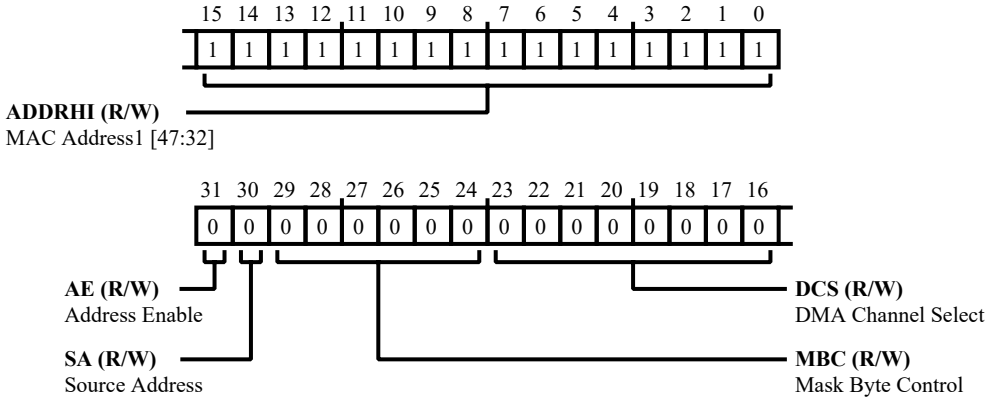


Figure 28-90: EMAC_ADDR[i]_HI Register Diagram

Table 28-145: EMAC_ADDR[i]_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	AE	Address Enable. When the EMAC_ADDR[i]_HI.AE bit is set (=1), the address filter module uses the second MAC address for perfect filtering. When EMAC_ADDR[i]_HI.AE = 0, the address filter module ignores the address for filtering.	
		0	Address is Ignored
		1	Address is Enabled
30 (R/W)	SA	Source Address. When the EMAC_ADDR[i]_HI.SA bit is set (=1), MAC address 1 [47:0] is used to compare with the SA fields of the received packet. When EMAC_ADDR[i]_HI.SA = 0, MAC address 1 [47:0] is used to compare with the DA fields of the received packet.	
		0	Compare with Destination Address
		1	Compare with Source Address

Table 28-145: EMAC_ADDR[i]_HI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	MBC	Mask Byte Control. These bits are mask control bits for comparing each of the MAC address bytes. When set (=1), the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC address 1 registers. Each bit controls the masking of the bytes as follows: Bit 29: `EMAC_ADDR[i]_HI`[15:8] Bit 28: `EMAC_ADDR[i]_HI`[7:0] Bit 27: `EMAC_ADDR[i]_HI`[31:24] .. Bit 24: `EMAC_ADDR[i]_HI`[7:0] Filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.
23:16 (R/W)	DCS	DMA Channel Select. When EMAC_MAC_EXT_CFG.PDC = 0, the `EMAC_ADDR[i]_HI.DCS` field contains the binary representation of the DMA channel number to which an Rx packet whose DA matches the MAC Address(n) content is routed. When EMAC_MAC_EXT_CFG.PDC = 1, the `EMAC_ADDR[i]_HI.DCS` field contains the one-hot representation of one or more DMA channel numbers to which an Rx packet whose DA matches the MAC address(n) content is routed.
15:0 (R/W)	ADDRHI	MAC Address1 [47:32]. The `EMAC_ADDR[i]_HI.ADDRHI` field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MAC Address n Low Register

The EMAC_ADDR[i]_LO register holds the lower 32 bits of the second 6-byte MAC address of the station.

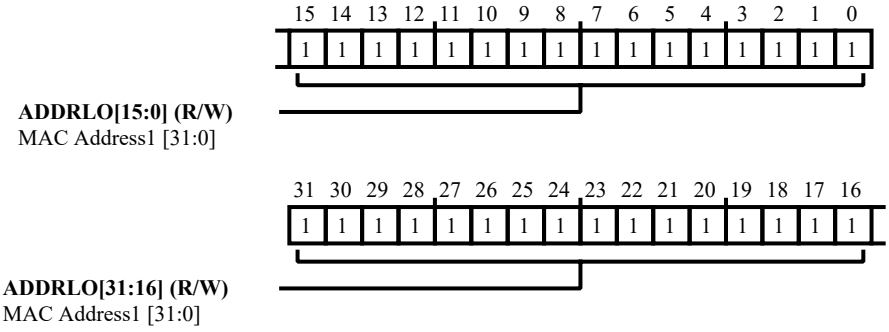


Figure 28-91: EMAC_ADDR[i]_LO Register Diagram

Table 28-146: EMAC_ADDR[i]_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRLO	MAC Address1 [31:0]. The EMAC_ADDR[i]_LO.ADDRLO field contains the lower 32 bits of second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process.

Auxiliary Time Stamp Control Register

The EMAC_TM_AUX_CTL register controls the auxiliary time stamp snapshot.

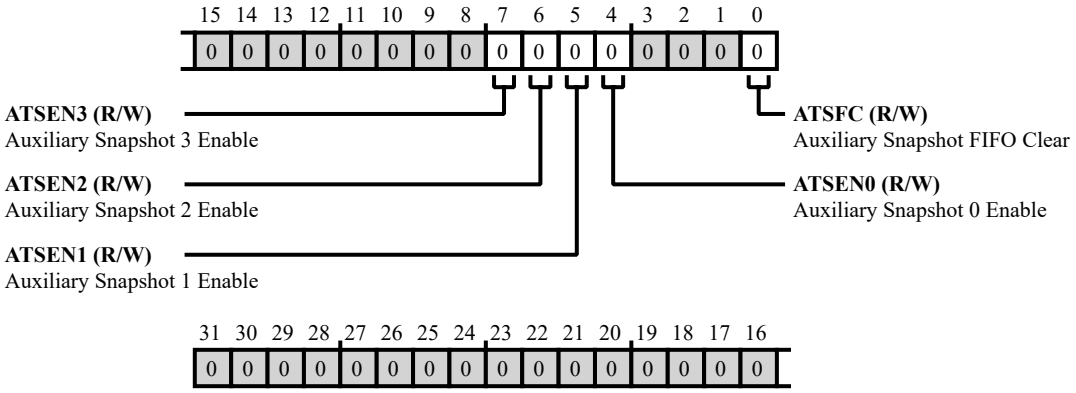


Figure 28-92: EMAC_TM_AUX_CTL Register Diagram

Table 28-147: EMAC_TM_AUX_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	ATSEN3	Auxiliary Snapshot 3 Enable. The EMAC_TM_AUX_CTL.ATSEN3 bit controls the capturing of auxiliary snapshot trigger 3. When the EMAC_TM_AUX_CTL.ATSEN3 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN3 input is enabled. When EMAC_TM_AUX_CTL.ATSEN3 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable
6 (R/W)	ATSEN2	Auxiliary Snapshot 2 Enable. The EMAC_TM_AUX_CTL.ATSEN2 bit controls the capturing of auxiliary snapshot trigger 2. When the EMAC_TM_AUX_CTL.ATSEN2 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN2 input is enabled. When EMAC_TM_AUX_CTL.ATSEN2 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable
5 (R/W)	ATSEN1	Auxiliary Snapshot 1 Enable. The EMAC_TM_AUX_CTL.ATSEN1 bit controls the capturing of auxiliary snapshot trigger 1. When the EMAC_TM_AUX_CTL.ATSEN1 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN1 input is enabled. When EMAC_TM_AUX_CTL.ATSEN1 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable

Table 28-147: EMAC_TM_AUX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	ATSEN0	Auxiliary Snapshot 0 Enable. The EMAC_TM_AUX_CTL.ATSEN0 bit controls the capturing of auxiliary snapshot trigger 0. When the EMAC_TM_AUX_CTL.ATSEN0 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN0 input is enabled. When EMAC_TM_AUX_CTL.ATSEN0 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable
0 (R/W)	ATSFC	Auxiliary Snapshot FIFO Clear. When set, the EMAC_TM_AUX_CTL.ATSFC bit resets the pointers of the Auxiliary Snapshot FIFO. The EMAC_TM_AUX_CTL.ATSFC bit is cleared when the pointers are reset and the FIFO is empty. When the EMAC_TM_AUX_CTL.ATSFC bit is high, the auxiliary snapshots are stored in the FIFO. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable

Auxiliary Time Stamp Nanoseconds Register

The EMAC_TM_AUXTS_NSEC register, along with EMAC_TM_AUXTS_SEC register, gives the 64-bit time stamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4, 8, or 16 as selected while configuring the IP.

Multiple snapshots can be stored in this FIFO. Bits[29:25] in EMAC_TM_STAT register indicate the fill-level of the FIFO. The top of the FIFO is removed only when the last byte of EMAC_TM_AUXTS_SEC register is read. In the Little Endian mode, this means when bits[31:24] are read and in Big Endian mode, bits[7:0] are read.

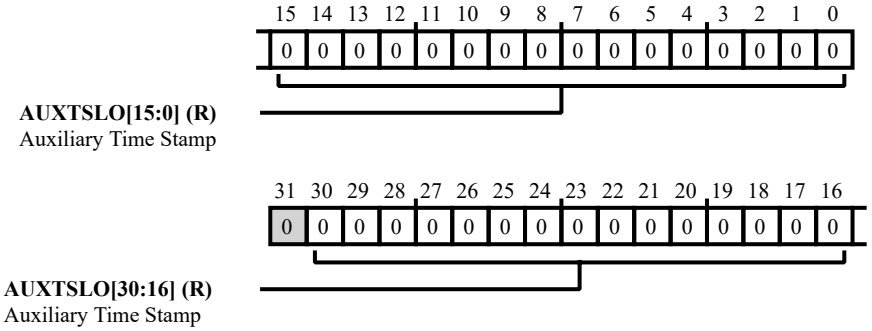


Figure 28-93: EMAC_TM_AUXTS_NSEC Register Diagram

Table 28-148: EMAC_TM_AUXTS_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:0 (R/NW)	AUXTSLO	Auxiliary Time Stamp. The EMAC_TM_AUXTS_NSEC.AUXTSLO field contains the lower 31 bits (nanoseconds field) of the auxiliary time stamp.

Auxiliary Time Stamp Seconds Register

The EMAC_TM_AUXTS_SEC register contains the lower 32 bits of the seconds field of the auxiliary time stamp register.

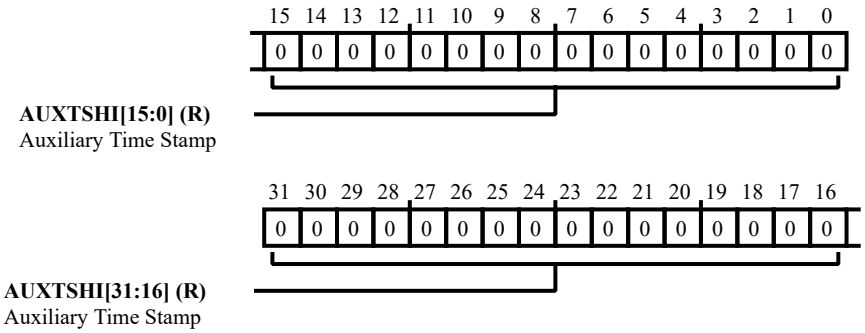


Figure 28-94: EMAC_TM_AUXTS_SEC Register Diagram

Table 28-149: EMAC_TM_AUXTS_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	AUXTSHI	Auxiliary Time Stamp. Contains the lower 32 bits of the seconds field of the auxiliary time stamp.

CSR Software Control Register

The EMAC_CSR_SW_CTL register contains software programmable controls for changing the Control and Status Registers (CSR) access response and status bits clearing.

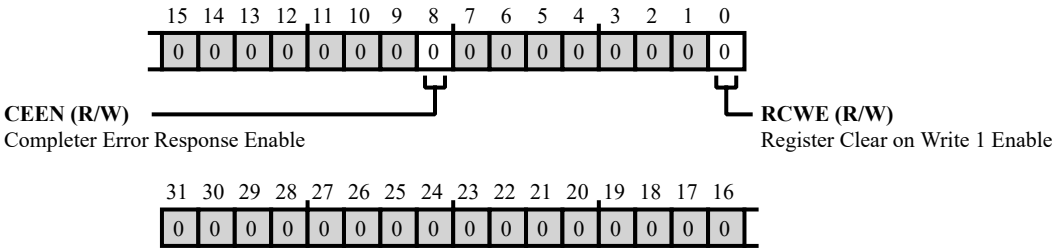


Figure 28-95: EMAC_CSR_SW_CTL Register Diagram

Table 28-150: EMAC_CSR_SW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	CEEN	Completer Error Response Enable. When the EMAC_CSR_SW_CTL.CEEN bit is set, the MAC responds with an error for accesses to reserved registers in CSR space. When EMAC_CSR_SW_CTL.CEEN = 0, the MAC responds with an okay response to any register accessed from CSR space.	
		0	Disable
		1	Enable
0 (R/W)	RCWE	Register Clear on Write 1 Enable. When the EMAC_CSR_SW_CTL.RCWE bit is set, the access mode of some register fields changes to (W1C) write one to clear; the application must set that respective bit to 1 to clear it. When EMAC_CSR_SW_CTL.RCWE = 0, the access mode of these register fields remain as Clear on read.	
		0	Disable
		1	Enable

MAC Configuration Register

The EMAC_MAC_CFG register establishes the operating mode of the MAC.

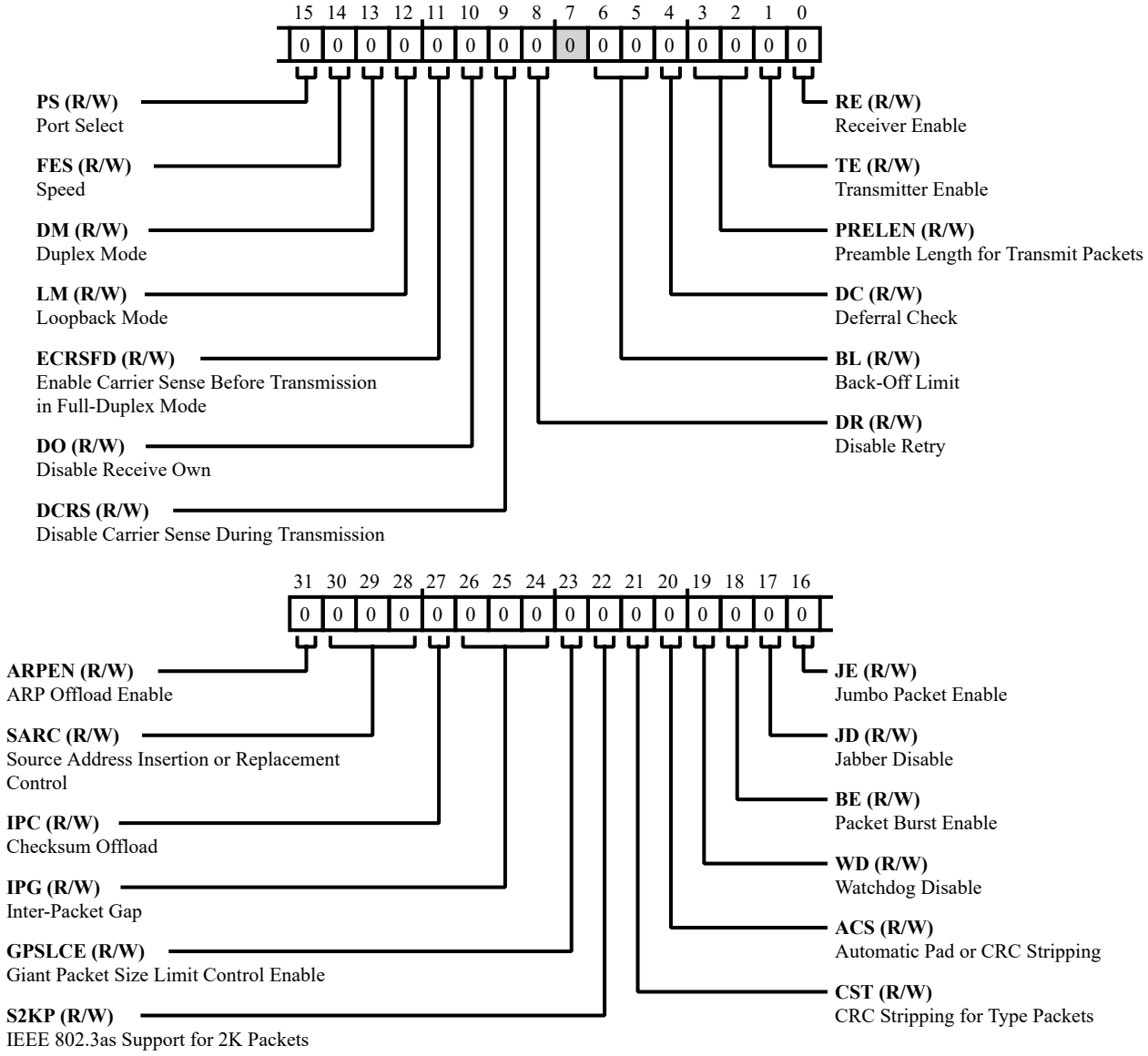


Figure 28-96: EMAC_MAC_CFG Register Diagram

Table 28-151: EMAC_MAC_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ARPEN	ARP Offload Enable. When the EMAC_MAC_CFG.ARPEN bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus. When EMAC_MAC_CFG.ARPEN = 0, the MAC receiver does not recognize any ARP packet and indicates them as type frame in the RxStatus. The EMAC_MAC_CFG.ARPEN bit is available only when enable IPv4 ARP offload is selected.	
		0	ARP Offload is Disabled
		1	ARP Offload is Enabled
30:28 (R/W)	SARC	Source Address Insertion or Replacement Control. The EMAC_MAC_CFG.SARC field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of bits[29:28]: 2'b0x: - Reserved 2'b10: - If Bit 30 = 0, the MAC inserts the content of the MAC Address 0 registers in the SA field of all transmitted packets. - If Bit 30 = 1 and the Enable MAC Address Register 1 option is selected while configuring the IP, the MAC inserts the content of the MAC Address 1 registers in the SA field of all transmitted packets. 2'b11: - If Bit 30 = 0, the MAC replaces the content of the MAC Address 0 registers in the SA field of all transmitted packets. - If Bit 30 = 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers in the SA field of all transmitted packets. Note: Changes to this field take effect only on the start of a packet. A write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.	
		0	Reserved
		2	Contents of MAC Address 0 Inserted in SA Field
		3	Contents of MAC Address 0 Replaces SA Field
		6	Contents of MAC Address 1 Inserted in SA Field
		7	Contents of MAC Address 1 Replaces SA Field

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	IPC	Checksum Offload. When set, the EMAC_MAC_CFG.IPC bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When EMAC_MAC_CFG.IPC = 0, the COE function in the receiver is disabled. The layer 3 and layer 4 packet filter and enable split header features automatically select the IPC full checksum offload engine on the receive side. When any of these features are enabled, the EMAC_MAC_CFG.IPC bit must be set.	
		0	IP Header/payload Checksum Checking is Disabled
		1	IP Header/payload Checksum Checking is Enabled
26:24 (R/W)	IPG	Inter-Packet Gap. The EMAC_MAC_CFG.IPG field control the minimum IPG between packets during transmission. This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of back pressure activation, the MAC does not consider the minimum IPG. This function (IPG less than 96 bit times) is valid only when EMAC_MAC_EXT_CFG.EIPGEN = 0. When EMAC_MAC_EXT_CFG.EIPGEN = 1, the minimum IPG (greater than 96 bit times) is controlled as per the description given in the EMAC_MAC_EXT_CFG.EIPG field register.	
		0	96 Bit Times IPG
		1	88 Bit Times IPG
		2	80 Bit Times IPG
		3	72 Bit Times IPG
		4	64 Bit Times IPG
		5	56 Bit Times IPG
		6	48 Bit Times IPG
		7	40 Bit Times IPG

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	GPSLCE	Giant Packet Size Limit Control Enable. When the EMAC_MAC_CFG.GPSLCE bit is set, the MAC considers the value in EMAC_MAC_EXT_CFG.GPSL to declare a received packet as giant packet. The EMAC_MAC_CFG.GPSLCE field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When EMAC_MAC_CFG.GPSLCE = 0, the MAC considers a received packet as giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet). The watchdog timeout limit, jumbo packet enable and 2K packet enable have higher precedence over the EMAC_MAC_CFG.GPSLCE bit, that is the MAC considers a received packet as giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with jumbo packet enabled and greater than 2,000 bytes with 2K packet enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.	
		0	Giant Packet Size Limit Control is Disabled
		1	Giant Packet Size Limit Control is Enabled
22 (R/W)	S2KP	IEEE 802.3as Support for 2K Packets. When the EMAC_MAC_CFG.S2KP bit is set (=1), the MAC considers all packets with up to 2,000 bytes length as normal packets. When EMAC_MAC_CFG.JE = 0, the MAC considers all received packets of size more than 2K bytes as giant packets. When EMAC_MAC_CFG.S2KP = 0 and EMAC_MAC_CFG.JE = 0, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. Note: When EMAC_MAC_CFG.JE = 1, setting the EMAC_MAC_CFG.S2KP bit has no effect on the giant packet status.	
		0	Support Upto 2K Packet is Disabled
		1	Support Upto 2K Packet is Enabled
21 (R/W)	CST	CRC Stripping for Type Packets. When the EMAC_MAC_CFG.CST bit is set (=1), the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application.	
		0	CRC Stripping for Type Packets is Disabled
		1	CRC Stripping for Type Packets is Enabled

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	ACS	Automatic Pad or CRC Stripping. When the EMAC_MAC_CFG.ACS bit is set, the MAC strips the pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When EMAC_MAC_CFG.ACS = 0, the MAC passes all incoming packets to the application, without any modification.	
		0	Automatic Pad or CRC Stripping is Disabled
		1	Automatic Pad or CRC Stripping is Enabled
19 (R/W)	WD	Watchdog Disable. When the EMAC_MAC_CFG.WD bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes. When EMAC_MAC_CFG.WD = 0, the MAC does not allow more than 2,048 bytes (10,240 if EMAC_MAC_CFG.JE=1) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.	
		0	Watchdog is Enabled
		1	Watchdog is Disabled
18 (R/W)	BE	Packet Burst Enable. When the EMAC_MAC_CFG.BE bit is set, the MAC allows packet bursting during transmission in the RGMII half-duplex mode.	
		0	Packet Burst is Disabled
		1	Packet Burst is Enabled
17 (R/W)	JD	Jabber Disable. When the EMAC_MAC_CFG.JD bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes. When EMAC_MAC_CFG.JD = 0, if the application sends more than 2,048 bytes of data (10,240 if EMAC_MAC_CFG.JE is set (=1)) during transmission, the MAC does not send rest of the bytes in that packet.	
		0	Jabber is Enabled
		1	Jabber is Disabled

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	JE	Jumbo Packet Enable. When the EMAC_MAC_CFG.JE bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.	
		0	Jumbo Packet is Disabled
		1	Jumbo Packet is Enabled
15 (R/W)	PS	Port Select. The EMAC_MAC_CFG.PS bit selects the Ethernet line speed. The EMAC_MAC_CFG.PS bit, along with bit 14, selects the exact line speed. In the 10/100 Mbps only (always 1) or 1000 Mbps only (always 0) configurations, the EMAC_MAC_CFG.PS bit is read-only (RO) with appropriate value. In the default 10/100/1000 Mbps configurations, the EMAC_MAC_CFG.PS bit is read-write (R/W).	
		0	1000 Mbps Operation
		1	10 or 100 Mbps Operation
14 (R/W)	FES	Speed. The EMAC_MAC_CFG.FES bit selects the speed mode.	
		0	10 Mbps when PS = 1; 1 Gbps when PS = 0
		1	100 Mbps when PS = 1; reserved when PS = 0
13 (R/W)	DM	Duplex Mode. When the EMAC_MAC_CFG.DM bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously. The EMAC_MAC_CFG.DM bit is RO with default value of 1'b1 in the full-duplex-only configurations.	
		0	Half-duplex Mode
		1	Full-duplex Mode
12 (R/W)	LM	Loopback Mode. When the EMAC_MAC_CFG.LM bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.	
		0	Loopback is Disabled
		1	Loopback is Enabled

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	ECRSFD	Enable Carrier Sense Before Transmission in Full-Duplex Mode. When the EMAC_MAC_CFG.ECRSFD bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When EMAC_MAC_CFG.ECRSFD = 0, the MAC transmitter ignores the status of the CRS signal.	
		0	ECRSFD is Disabled
		1	ECRSFD is Enabled
10 (R/W)	DO	Disable Receive Own. When the EMAC_MAC_CFG.DO bit is set, the MAC disables the reception of packets when the gmii_txen_o is asserted in the half-duplex mode. When EMAC_MAC_CFG.DO = 0, the MAC receives all packets given by the PHY. The EMAC_MAC_CFG.DO bit is not applicable in the full-duplex mode.	
		0	Enable Receive Own
		1	Disable Receive Own
9 (R/W)	DCRS	Disable Carrier Sense During Transmission. When the EMAC_MAC_CFG.DCRS bit is set, the MAC transmitter ignores the (G)MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of loss of carrier or no carrier during transmission. When EMAC_MAC_CFG.DCRS = 0, the MAC transmitter generates errors because of carrier sense. The MAC can even abort the transmission.	
		0	Enable Carrier Sense During Transmission
		1	Disable Carrier Sense During Transmission
8 (R/W)	DR	Disable Retry. When the EMAC_MAC_CFG.DR bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a packet abort error with excessive collision error in the Tx packet status. When EMAC_MAC_CFG.DR = 0, the MAC retries based on the settings of the BL field. The EMAC_MAC_CFG.DR bit is applicable only in the half-duplex mode.	
		0	Enable Retry
		1	Disable Retry

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	BL	Back-Off Limit. The EMAC_MAC_CFG.BL field determines the random integer number (r) of slot time delays (4,096 bit times for 1000/2500 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. n = retransmission attempt. The random integer r takes the value in the range 0 <= r < 2^k The EMAC_MAC_CFG.BL bit is applicable only in the half-duplex mode.	
		0	k is Minimum of Retransmission Attempt and 10
		1	k is Minimum of Retransmission Attempt and 8
		2	k is Minimum of Retransmission Attempt and 4
		3	k is Minimum of Retransmission Attempt and 1
4 (R/W)	DC	Deferral Check. When the EMAC_MAC_CFG.DC bit is set, the deferral check function is enabled in the MAC. The MAC issues a packet abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode. If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset (=0), and it is restarted. When EMAC_MAC_CFG.DC = 0, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. The EMAC_MAC_CFG.DC bit is applicable only in the half-duplex mode.	
		0	Deferral Check Function is Disabled
		1	Deferral Check Function is Enabled

Table 28-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	PRELEN	Preamble Length for Transmit Packets. The EMAC_MAC_CFG.PRELEN field control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.	
		0	7 Bytes of Preamble
		1	5 Bytes of Preamble
		2	3 Bytes of Preamble
		3	Reserved
1 (R/W)	TE	Transmitter Enable. When the EMAC_MAC_CFG.TE bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When EMAC_MAC_CFG.TE = 0, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.	
		0	Transmitter is Disabled
		1	Transmitter is Enabled
0 (R/W)	RE	Receiver Enable. When the EMAC_MAC_CFG.RE bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When the EMAC_MAC_CFG.RE = 0, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface.	
		0	Receiver is Disabled
		1	Receiver is Enabled

Debug Register

The EMAC_MAC_DBG register provides the debug status of various MAC blocks.

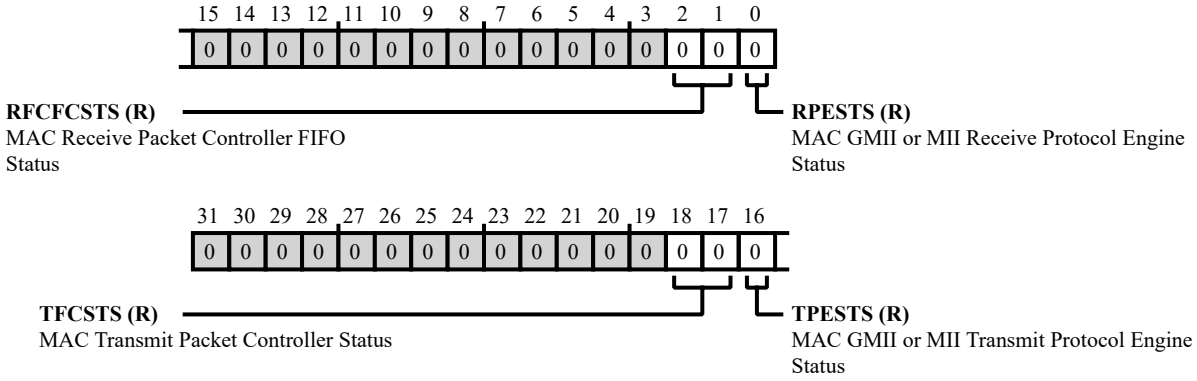


Figure 28-97: EMAC_MAC_DBG Register Diagram

Table 28-152: EMAC_MAC_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:17 (R/NW)	TFCSTS	MAC Transmit Packet Controller Status. The EMAC_MAC_DBG.TFCSTS field indicates the state of the MAC Transmit Packet Controller module.	
		0	Idle State
		1	Waiting for One of the Following: Status of the Previous Packet OR IPG or Back off Period to Be Over
		2	Generating and Transmitting a Pause Control Packet (in Full-Duplex Mode)
		3	Transferring Input Packet for Transmission
16 (R/NW)	TPESTS	MAC GMII or MII Transmit Protocol Engine Status. When the EMAC_MAC_DBG.TPESTS bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.	
		0	MAC GMII or MII Transmit Protocol Engine Status Not Detected
		1	MAC GMII or MII Transmit Protocol Engine Status Detected
2:1 (R/NW)	RFCFCSTS	MAC Receive Packet Controller FIFO Status. The EMAC_MAC_DBG.RFCFCSTS field indicates the active state of the small FIFO read and write controllers of the MAC Receive Packet Controller module.	

Table 28-152: EMAC_MAC_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	RPESTS	MAC GMII or MII Receive Protocol Engine Status. When the `EMAC_MAC_DBG.RPESTS` bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state.	
		0	MAC GMII or MII Receive Protocol Engine Status Not Detected
		1	MAC GMII or MII Receive Protocol Engine Status Detected

MAC Split Mode and Offset Configuration Register

The EMAC_SPLM_OFST_CFG register contains the split mode control field and offset field for the split header feature.

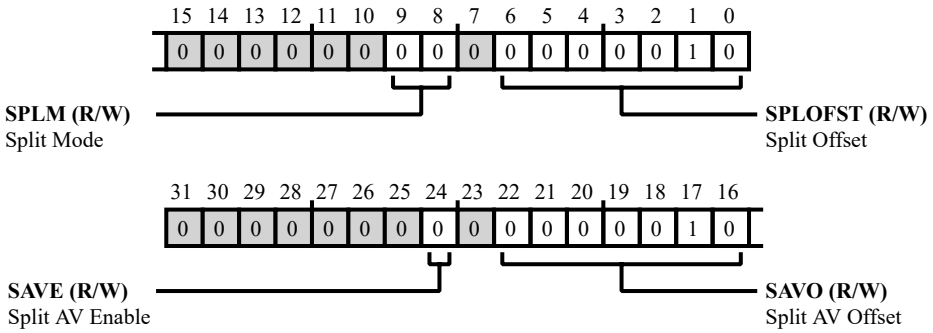


Figure 28-98: EMAC_SPLM_OFST_CFG Register Diagram

Table 28-153: EMAC_SPLM_OFST_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	SAVE	Split AV Enable. When EMAC_SPLM_OFST_CFG.SAVE = 1, and the received packet is an AV Type packet, the header is split at SAVO bytes from the beginning of Length/Type field of the packet, for L2 Split. When EMAC_SPLM_OFST_CFG.SAVE = 0, header is split at SPLOFST bytes from the beginning of Length/Type field of the frame, for L2 Split.
22:16 (R/W)	SAVO	Split AV Offset. When SAVE bit is set (=1), and the received packet is an AV Type packet, the EMAC_SPLM_OFST_CFG.SAVO field indicates the value of the offset from the beginning of Length/Type field at which header should be split, when appropriate SPLM is selected. The reset value of this field is 2 bytes, indicating a split at L2 header. Value is in terms of bytes.

Table 28-153: EMAC_SPLM_OFST_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	SPLM	Split Mode. The `EMAC_SPLM_OFST_CFG.SPLM` field indicates the mode of splitting the incoming Rx packets.	
		0	Split at L3/L4 Header
		1	Split at L2 Header with an Offset. Always Split at SPLOFST Bytes from the Beginning of Length/Type Field of the Frame
		2	Combination Mode: Split Similar to SPLM=00 for IP Packets That are Untagged or Tagged and VLAN Stripped
		3	Reserved
6:0 (R/W)	SPLOFST	Split Offset. The `EMAC_SPLM_OFST_CFG.SPLOFST` field indicates the value of offset from the beginning of Length/Type field at which header split should take place when the appropriate SPLM is selected. The reset value of this field is 2 bytes indicating a split at L2 header. Value is in terms of bytes.	

MAC Extended Configuration Register

The EMAC_MAC_EXT_CFG register establishes the operating mode of the MAC.

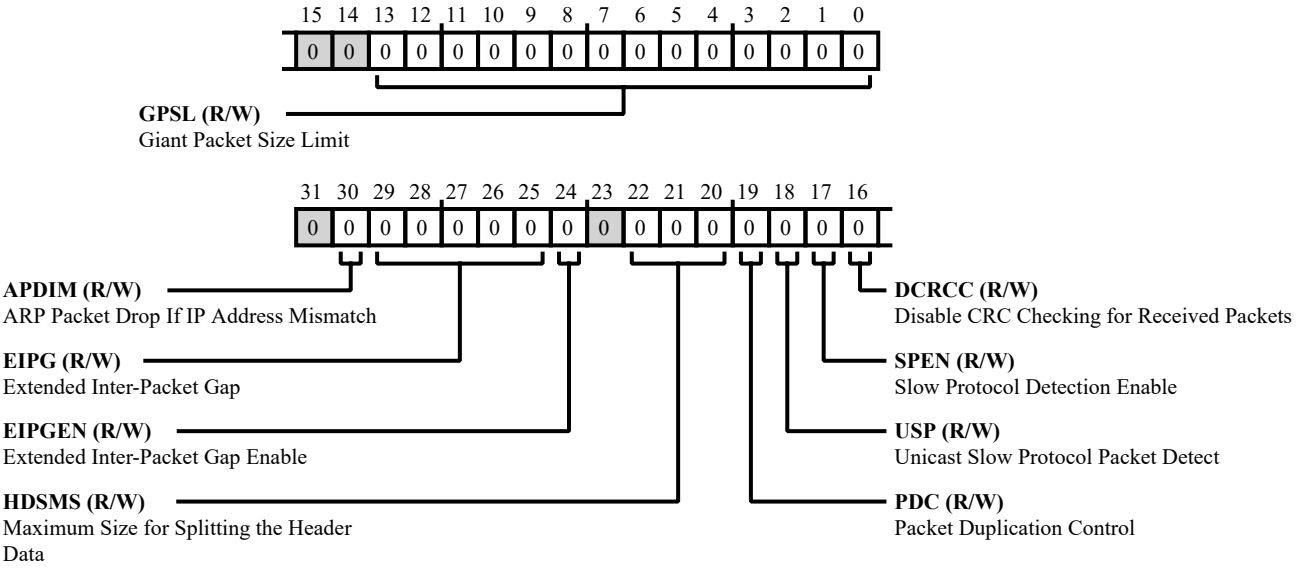


Figure 28-99: EMAC_MAC_EXT_CFG Register Diagram

Table 28-154: EMAC_MAC_EXT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	APDIM	ARP Packet Drop If IP Address Mismatch. When the EMAC_MAC_EXT_CFG.APDIM bit is set, packet for which target protocol address does not match IPv4 address is dropped in the MTL layer. When EMAC_MAC_EXT_CFG.APDIM = 0, when target protocol address does not match, packet is forwarded to MTL maintaining backward compatibility	
		0	Mux Select to Drop the Arp Packet If Target Protocol Address Mismatches IPv4 Address Disabled
		1	Mux Select to Drop the Arp Packet If Target Protocol Address Mismatches IPv4 Address Enabled

Table 28-154: EMAC_MAC_EXT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29:25 (R/W)	EIPG	Extended Inter-Packet Gap. The value in the EMAC_MAC_EXT_CFG.EIPG field is applicable when EMAC_MAC_EXT_CFG.EIPGEN = 1. The EMAC_MAC_EXT_CFG.EIPG field (as most significant bits), along with IPG, gives the minimum IPG greater than 96 bit times in steps of 8 bit times: {EIPG, IPG} 8'h00 - 104 bit times 8'h01 - 112 bit times 8'h02 - 120 bit times ---------- 8'hFF - 2144 bit times	
24 (R/W)	EIPGEN	Extended Inter-Packet Gap Enable. When the EMAC_MAC_EXT_CFG.EIPGEN bit is set, the MAC interprets EMAC_MAC_CFG.IPG and EMAC_MAC_CFG.EIPG together as minimum IPG greater than 96 bit times in steps of 8 bit times. When EMAC_MAC_EXT_CFG.EIPGEN = 0, the MAC ignores EMAC_MAC_CFG.EIPG and interprets EMAC_MAC_CFG.IPGas minimum IPG less than or equal to 96 bit times in steps of 8 bit times. Note: Enable the extended inter-packet gap feature only when operating in full-duplex mode. There might be undesirable effects on back-pressure function and frame transmission if it is enabled in half-duplex mode.	
		0	Extended Inter-Packet Gap is Disabled
		1	Extended Inter-Packet Gap is Enabled
22:20 (R/W)	HDSMS	Maximum Size for Splitting the Header Data. The EMAC_MAC_EXT_CFG.HDSMS field indicates the maximum header size allowed for splitting the header data in the received packet.	
		0	Maximum Size for Splitting the Header Data is 64 Bytes
		1	Maximum Size for Splitting the Header Data is 128 Bytes
		2	Maximum Size for Splitting the Header Data is 256 Bytes
		3	Maximum Size for Splitting the Header Data is 512 Bytes
		4	Maximum Size for Splitting the Header Data is 1024 Bytes
		5	Reserved

Table 28-154: EMAC_MAC_EXT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	PDC	Packet Duplication Control. When the EMAC_MAC_EXT_CFG.PDC bit is set, the received packet with multicast/broadcast destination address is routed to multiple receive DMA channels. The receive DMA channels is identified by the DCS field of EMAC_ADDR[n]_HI register corresponding to the MAC Address register that matches the multicast/broadcast destination address in the received packet. The DCS field is interpreted to be a one-hot value, each bit corresponding to the receive DMA channel. When EMAC_MAC_EXT_CFG.PDC = 0, the received packet is routed to single receive DMA channel. The receive DMA channel is identified by EMAC_ADDR[n]_HI.DCS corresponding to the MAC Address register that matches the Destination address in the received packet. The DCS field is interpreted as a binary value.	
		0	Packet Duplication Control is Disabled
		1	Packet Duplication Control is Enabled
18 (R/W)	USP	Unicast Slow Protocol Packet Detect. When the EMAC_MAC_EXT_CFG.USP bit is set, the MAC detects the slow protocol packets with unicast address of the station specified in the EMAC_ADDR0_HI and EMAC_ADDR0_LO registers. The MAC also detects the slow protocol packets with the slow protocols multicast address (01-80-C2-00-00-02). When EMAC_MAC_EXT_CFG.USP = 0, the MAC detects only slow protocol packets with the slow protocol multicast address specified in the IEEE 802.3-2015, Section 5.	
		0	Unicast Slow Protocol Packet Detection is Disabled
		1	Unicast Slow Protocol Packet Detection is Enabled
17 (R/W)	SPEN	Slow Protocol Detection Enable. When the EMAC_MAC_EXT_CFG.SPEN bit is set, MAC processes the slow protocol packets (Ether Type 0x8809) and provides the slow protocol subtype and code fields in Rx status. When EMAC_MAC_EXT_CFG.SPEN = 0, the MAC forwards all error-free slow protocol packets to the application. The MAC considers such packets as normal type packets.	
		0	Slow Protocol Detection is Disabled
		1	Slow Protocol Detection is Enabled

Table 28-154: EMAC_MAC_EXT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	DCRCC	Disable CRC Checking for Received Packets. When the `EMAC_MAC_EXT_CFG.DCRCC` bit is set, the MAC receiver does not check the CRC field in the received packets. When `EMAC_MAC_EXT_CFG.DCRCC` = 0, the MAC receiver always checks the CRC field in the received packets.	
		0	CRC Checking is Enabled
		1	CRC Checking is Disabled
13:0 (R/W)	GPSL	Giant Packet Size Limit. If the received packet size is greater than the value programmed in the `EMAC_MAC_EXT_CFG.GPSL` field in units of bytes, the MAC declares the received packet as giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. When the enable double VLAN processing option is selected, the MAC adds 8 bytes to the programmed value for double VLAN tagged packets. The value in this field is applicable when EMAC_MAC_CFG.GPSLCE = 1.	

Frame Preemption Control and Status Register

The EMAC_FPE_CTLSTAT register controls the operation of frame preemption.

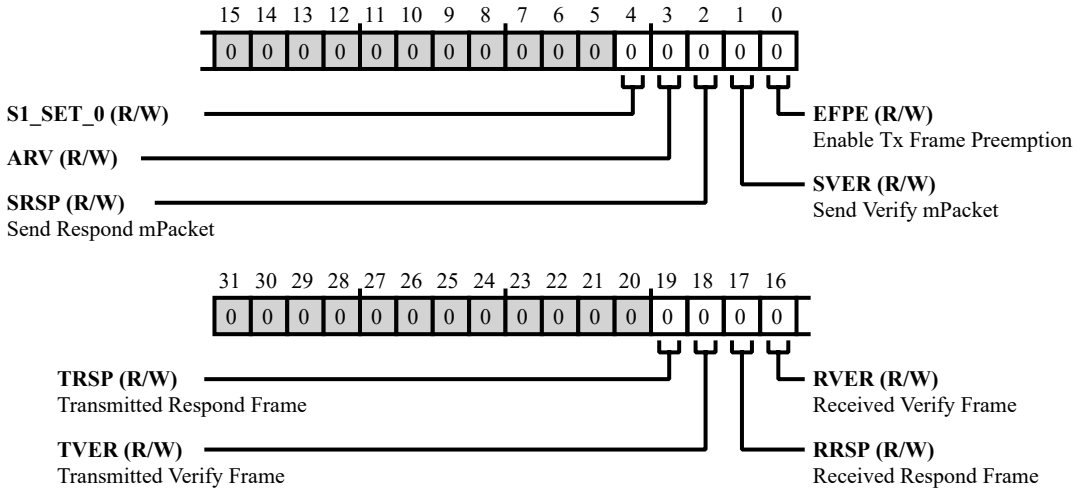


Figure 28-100: EMAC_FPE_CTLSTAT Register Diagram

Table 28-155: EMAC_FPE_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	TRSP	Transmitted Respond Frame. The EMAC_FPE_CTLSTAT.TRSP bit is set when a Respond mPacket is transmitted (triggered by EMAC_FPE_CTLSTAT.SRSP = 1). An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Transmitted Respond Frame
		1	Transmitted Respond Frame
18 (R/W)	TVER	Transmitted Verify Frame. The EMAC_FPE_CTLSTAT.TVER bit is set when a Verify mPacket is transmitted (triggered by EMAC_FPE_CTLSTAT.SVER = 1). An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Transmitted Verify Frame
		1	Transmitted Verify Frame

Table 28-155: EMAC_FPE_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RRSP	Received Respond Frame. The EMAC_FPE_CTLSTAT.RRSP bit is set when a Respond mPacket is received. An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Received Respond Frame
		1	Received Respond Frame
16 (R/W)	RVER	Received Verify Frame. The EMAC_FPE_CTLSTAT.RVER bit is set when a Verify mPacket is received. An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Received Verify Frame
		1	Received Verify Frame
4 (R/W)	S1_SET_0	. Synopsys Reserved, Must be set to "0". This field is reserved for Synopsys Internal use, and must always be set to "0" unless instructed by Synopsys. Setting to "1" might cause unexpected behavior in the IP.	
3 (R/W)	ARV	. Autogenerate Respond mPacket on receiving Verify mPacket When set, enable the function that autogenerate a Respond mPacket on receiving Verify mPacket.	
2 (R/W)	SRSP	Send Respond mPacket. When set (=1), the EMAC_FPE_CTLSTAT.SRSP bit indicates to hardware to send a Respond mPacket. Reset by hardware after sending the Respond mPacket. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Send Respond MPacket is Disabled
		1	Send Respond MPacket is Enabled
1 (R/W)	SVER	Send Verify mPacket. When set (=1), the EMAC_FPE_CTLSTAT.SVER bit indicates to hardware to send a verify mPacket. Reset by hardware after sending the Verify mPacket. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Send Verify MPacket is Disabled
		1	Send Verify MPacket is Enabled

Table 28-155: EMAC_FPE_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EFPE	Enable Tx Frame Preemption. When set (=1), the `EMAC_FPE_CTLSTAT.EFPE` bit enables Tx Frame Preemption.	
		0	Tx Frame Preemption is Disabled
		1	Tx Frame Preemption is Enabled

Hardware Feature Presence Register 0

The EMAC_HW_FTR_PSNT0 register indicates the presence of first set of the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

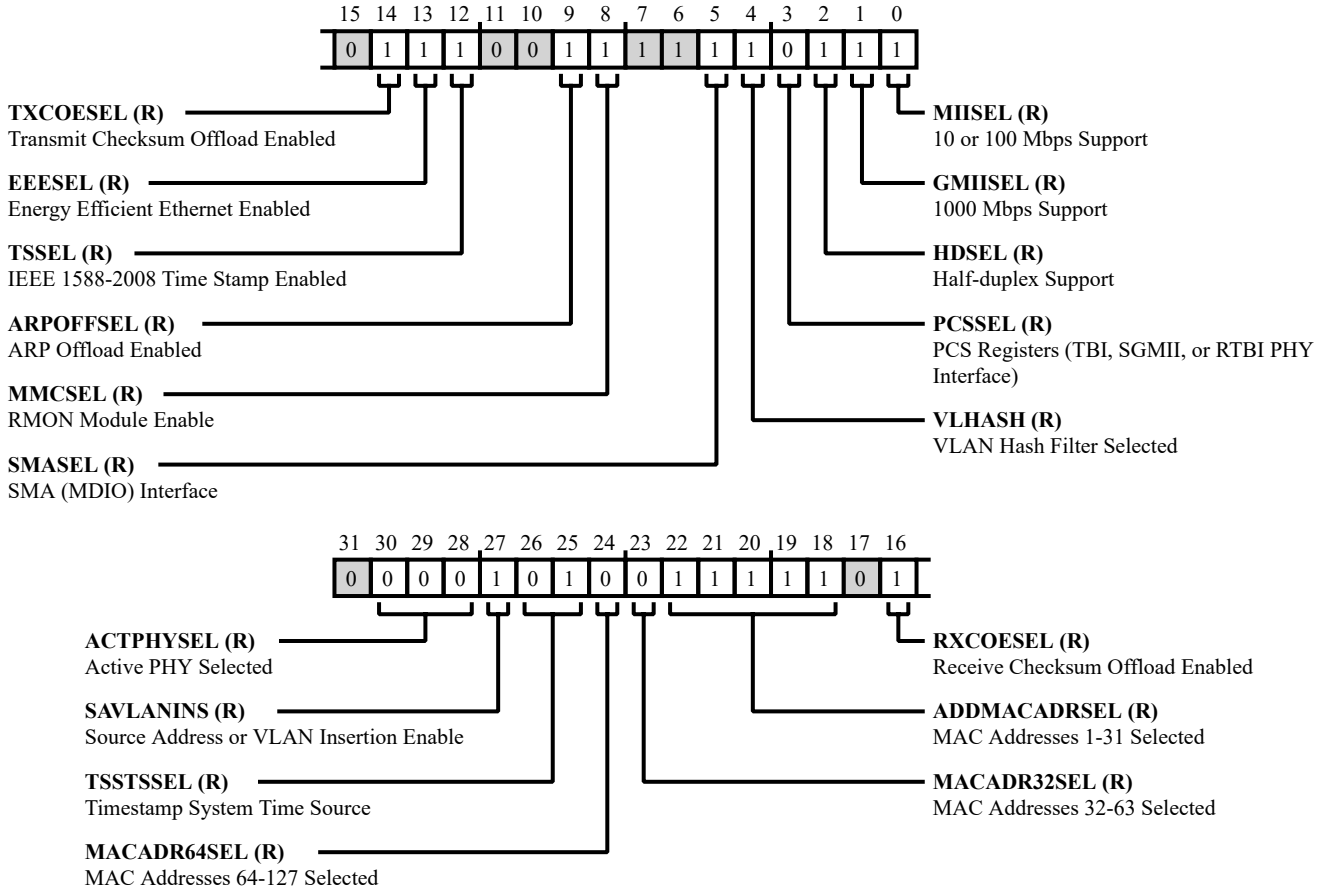


Figure 28-101: EMAC_HW_FTR_PSNT0 Register Diagram

Table 28-156: EMAC_HW_FTR_PSNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:28 (R/NW)	ACTPHYSEL	Active PHY Selected. When there are multiple PHY interfaces in the configuration, the EMAC_HW_FTR_PSNT0.ACTPHYSEL field indicates the sampled value of PADS_PCFG0.EMACPHYISEL during reset deassertion.	
		0	GMII or MII
		1	RGMII
		2	SGMII

Table 28-156: EMAC_HW_FTR_PSNT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	TBI
		4	RMII
		5	RTBI
		6	SMII
		7	RevMII
27 (R/NW)	SAVLANINS	Source Address or VLAN Insertion Enable. The EMAC_HW_FTR_PSNT0.SAVLANINS bit is set (=1) when the Enable SA and VLAN Insertion on Tx option is selected.	
		0	Source Address or VLAN Insertion Enable Option is Not Selected
		1	Source Address or VLAN Insertion Enable Option is Selected
26:25 (R/NW)	TSSTSSEL	Timestamp System Time Source. The EMAC_HW_FTR_PSNT0.TSSTSSEL bit indicates the source of the time stamp system time. The EMAC_HW_FTR_PSNT0.TSSTSSEL bit is set (=1) when the Enable IEEE 1588 time stamp Support option is selected.	
		0	Internal
		1	External
		2	Both
		3	Reserved
24 (R/NW)	MACADR64SEL	MAC Addresses 64-127 Selected. The EMAC_HW_FTR_PSNT0.MACADR64SEL bit is set (=1) when the Enable Additional 64 MAC Address Registers (64-127) option is selected.	
		0	MAC Addresses 64-127 Select Option is Not Selected
		1	MAC Addresses 64-127 Select Option is Selected
23 (R/NW)	MACADR32SEL	MAC Addresses 32-63 Selected. The EMAC_HW_FTR_PSNT0.MACADR32SEL bit is set (=1) when the Enable Additional 32 MAC Address Registers (32-63) option is selected.	
		0	MAC Addresses 32-63 Select Option is Not Selected
		1	MAC Addresses 32-63 Select Option is Selected
22:18 (R/NW)	ADDMACADRSEL	MAC Addresses 1-31 Selected. The EMAC_HW_FTR_PSNT0.ADDMACADRSEL bit is set (=1) when the non-zero value is selected for Enable Additional 1-31 MAC Address Registers option.	

Table 28-156: EMAC_HW_FTR_PSNT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	RXCOESEL	Receive Checksum Offload Enabled. The EMAC_HW_FTR_PSNT0.RXCOESEL bit is set (=1) when the Enable Receive TCP/IP Checksum Check option is selected.	
		0	Receive Checksum Offload Enable Option is Not Selected
		1	Receive Checksum Offload Enable Option is Selected
14 (R/NW)	TXCOESEL	Transmit Checksum Offload Enabled. The EMAC_HW_FTR_PSNT0.TXCOESEL bit is set (=1) when the Enable Transmit TCP/IP Checksum Insertion option is selected.	
		0	Transmit Checksum Offload Enable Option is Not Selected
		1	Transmit Checksum Offload Enable Option is Selected
13 (R/NW)	EEESEL	Energy Efficient Ethernet Enabled. The EMAC_HW_FTR_PSNT0.EEESEL bit is set (=1) when the Enable Energy Efficient Ethernet (EEE) option is selected.	
		0	Energy Efficient Ethernet Enable Option is Not Selected
		1	Energy Efficient Ethernet Enable Option is Selected
12 (R/NW)	TSSEL	IEEE 1588-2008 Time Stamp Enabled. The EMAC_HW_FTR_PSNT0.TSSEL bit is set (=1) when the Enable IEEE 1588 Time Stamp Support Option is selected.	
		0	Option is Not Selected
		1	Option is Selected
9 (R/NW)	ARPOFFSEL	ARP Offload Enabled. The EMAC_HW_FTR_PSNT0.ARPOFFSEL bit is set (=1) when the Enable IPv4 ARP Offload option is selected.	
		0	ARP Offload Enable Option is Not Selected
		1	ARP Offload Enable Option is Selected
8 (R/NW)	MMCSEL	RMON Module Enable. The EMAC_HW_FTR_PSNT0.MMCSEL bit is set (=1) when the Enable MAC Management Counters (MMC) option is selected.	
		0	RMON Module Enable Option is Not Selected
		1	RMON Module Enable Option is Selected

Table 28-156: EMAC_HW_FTR_PSNT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	SMASEL	SMA (MDIO) Interface. The `EMAC_HW_FTR_PSNT0.SMASEL` bit is set (=1) when the Enable Station Management (MDIO Interface) option is selected.	
		0	SMA (MDIO) Interface Not Selected
		1	SMA (MDIO) Interface Selected
4 (R/NW)	VLHASH	VLAN Hash Filter Selected. The `EMAC_HW_FTR_PSNT0.VLHASH` bit is set (=1) when the Enable VLAN Hash Table Based Filtering option is selected.	
		0	VLAN Hash Filter Not Selected
		1	VLAN Hash Filter Selected
3 (R/NW)	PCSSEL	PCS Registers (TBI, SGMII, or RTBI PHY Interface). The `EMAC_HW_FTR_PSNT0.PCSSEL` bit is set (=1) when the TBI, SGMII, or RTBI PHY interface option is selected.	
		0	No PCS Registers (TBI, SGMII, or RTBI PHY Interface)
		1	PCS Registers (TBI, SGMII, or RTBI PHY Interface)
2 (R/NW)	HDSEL	Half-duplex Support. The `EMAC_HW_FTR_PSNT0.HDSEL` bit is set (=1) when the half-duplex mode is selected,	
		0	No Half-duplex Support
		1	Half-duplex Support
1 (R/NW)	GMIISEL	1000 Mbps Support. The `EMAC_HW_FTR_PSNT0.GMIISEL` bit is set (=1) when 1000 Mbps is selected as the mode of operation.	
		0	No 1000 Mbps Support
		1	1000 Mbps Support
0 (R/NW)	MIISEL	10 or 100 Mbps Support. The `EMAC_HW_FTR_PSNT0.MIISEL` bit is set (=1) when 10/100 Mbps is selected as the mode of operation.	
		0	No 10 or 100 Mbps Support
		1	10 or 100 Mbps Support

Hardware Feature Presence Register 1

The EMAC_HW_FTR_PSNT1 register indicates the presence of second set of the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

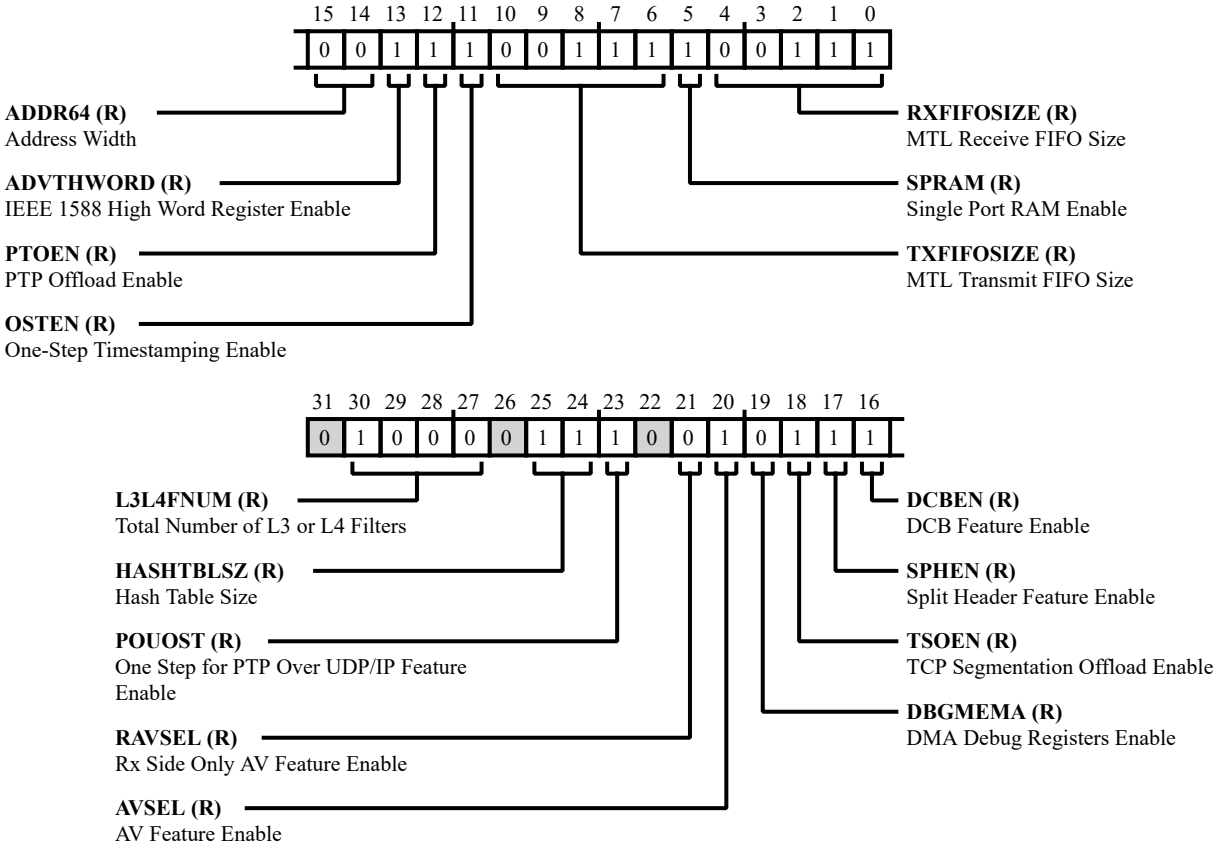


Figure 28-102: EMAC_HW_FTR_PSNT1 Register Diagram

Table 28-157: EMAC_HW_FTR_PSNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:27 (R/NW)	L3L4FNUM	Total Number of L3 or L4 Filters. The EMAC_HW_FTR_PSNT1.L3L4FNUM field indicates the total number of L3 or L4 filters.	
		0	No L3 or L4 Filter
		1	1 L3 or L4 Filter
		2	2 L3 or L4 Filters
		3	3 L3 or L4 Filters

Table 28-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	4 L3 or L4 Filters
		5	5 L3 or L4 Filters
		6	6 L3 or L4 Filters
		7	7 L3 or L4 Filters
		8	8 L3 or L4 Filters
25:24 (R/NW)	HASHTBLSZ	Hash Table Size. The `EMAC_HW_FTR_PSNT1.HASHTBLSZ` field indicates the size of the hash table.	
		0	No Hash Table
		1	64
		2	128
		3	256
23 (R/NW)	POUOST	One Step for PTP Over UDP/IP Feature Enable. The `EMAC_HW_FTR_PSNT1.POUOST` bit is set (=1) when the Enable One Step Time Stamp for PTP over UDP/IP feature is selected.	
		0	One Step for PTP Over UDP/IP Feature is Not Selected
		1	One Step for PTP Over UDP/IP Feature is Selected
21 (R/NW)	RAVSEL	Rx Side Only AV Feature Enable. The `EMAC_HW_FTR_PSNT1.RAVSEL` bit is set (=1) when the Enable Audio Video Bridging option on Rx Side Only is selected.	
		0	Rx Side Only AV Feature is Not Selected
		1	Rx Side Only AV Feature is Selected
20 (R/NW)	AVSEL	AV Feature Enable. The `EMAC_HW_FTR_PSNT1.AVSEL` bit is set (=1) when the Enable Audio Video Bridging option is selected.	
		0	AV Feature is Not Selected
		1	AV Feature is Selected
19 (R/NW)	DBGMEMA	DMA Debug Registers Enable. The `EMAC_HW_FTR_PSNT1.DBGMEMA` bit is set (=1) when the Debug Mode Enable option is selected	
		0	DMA Debug Registers Option is Not Selected
		1	DMA Debug Registers Option is Selected

Table 28-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	TSOEN	TCP Segmentation Offload Enable. The EMAC_HW_FTR_PSNT1.TSOEN bit is set (=1) when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected	
		0	TCP Segmentation Offload Feature is Not Selected
		1	TCP Segmentation Offload Feature is Selected
17 (R/NW)	SPHEN	Split Header Feature Enable. The EMAC_HW_FTR_PSNT1.SPHEN bit is set (=1) when the Enable Split Header Structure option is selected	
		0	Split Header Feature is Not Selected
		1	Split Header Feature is Selected
16 (R/NW)	DCBEN	DCB Feature Enable. The EMAC_HW_FTR_PSNT1.DCBEN bit is set (=1) when the Enable Data Center Bridging option is selected	
		0	DCB Feature is Not Selected
		1	DCB Feature is Selected
15:14 (R/NW)	ADDR64	Address Width. The EMAC_HW_FTR_PSNT1.ADDR64 field indicates the configured address width:	
		0	32
		1	40
		2	48
		3	Reserved
13 (R/NW)	ADVTHWORD	IEEE 1588 High Word Register Enable. The EMAC_HW_FTR_PSNT1.ADVTHWORD bit is set (=1) when the Add IEEE 1588 Higher Word Register option is selected	
		0	IEEE 1588 High Word Register Option is Not Selected
		1	IEEE 1588 High Word Register Option is Selected
12 (R/NW)	PTOEN	PTP Offload Enable. The EMAC_HW_FTR_PSNT1.PTOEN bit is set (=1) when the Enable PTP time stamp Offload Feature is selected.	
		0	PTP Offload Feature is Not Selected
		1	PTP Offload Feature is Selected

Table 28-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	OSTEN	One-Step Timestamping Enable. The `EMAC_HW_FTR_PSNT1.OSTEN` bit is set (=1) when the Enable One-Step time stamp Feature is selected.	
		0	Feature is Not Selected
		1	Feature is Selected
10:6 (R/NW)	TXFIFOSIZE	MTL Transmit FIFO Size. The `EMAC_HW_FTR_PSNT1.TXFIFOSIZE` field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:	
		0	128 Bytes
		1	256 Bytes
		2	512 Bytes
		3	1024 Bytes
		4	2048 Bytes
		5	4096 Bytes
		6	8192 Bytes
		7	16384 Bytes
		8	32 KB
		9	64 KB
		10	128 KB
		11	Reserved
5 (R/NW)	SPRAM	Single Port RAM Enable. The `EMAC_HW_FTR_PSNT1.SPRAM` bit is set (=1) when the Use single port RAM Feature is selected.	
		0	Single Port RAM Feature is Not Selected
		1	Single Port RAM Feature is Selected
4:0 (R/NW)	RXFIFOSIZE	MTL Receive FIFO Size. The `EMAC_HW_FTR_PSNT1.RXFIFOSIZE` field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:	
		0	128 Bytes
		1	256 Bytes
		2	512 Bytes

Table 28-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	1024 Bytes
		4	2048 Bytes
		5	4096 Bytes
		6	8192 Bytes
		7	16384 Bytes
		8	32 KB
		9	64 KB
		10	128 KB
		11	256 KB
		12	Reserved

Hardware Feature Presence Register 2

The EMAC_HW_FTR_PSNT2 register indicates the presence of third set of the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

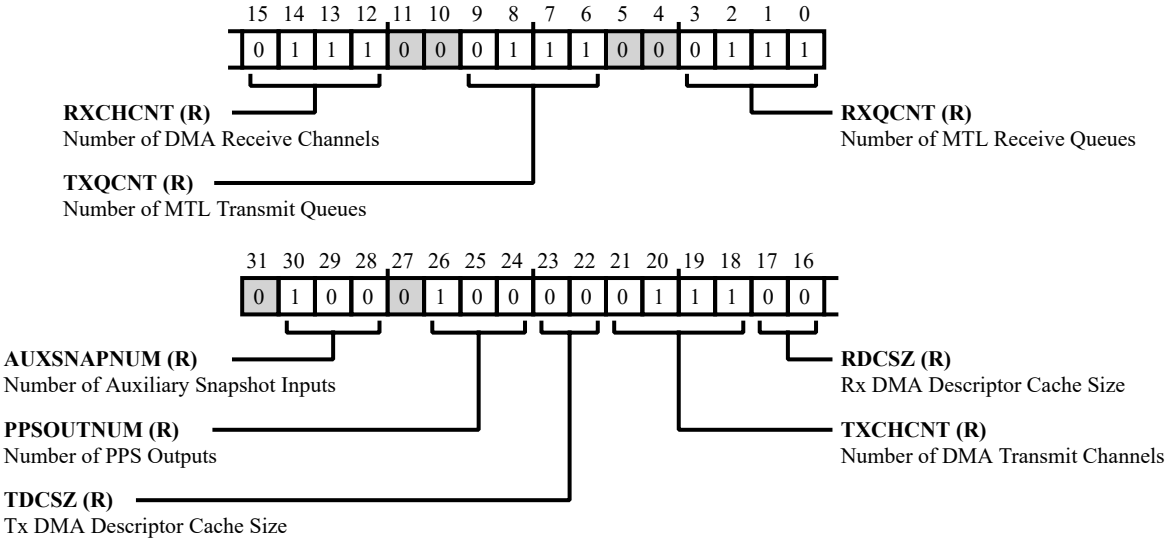


Figure 28-103: EMAC_HW_FTR_PSNT2 Register Diagram

Table 28-158: EMAC_HW_FTR_PSNT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:28 (R/NW)	AUXSNAPNUM	Number of Auxiliary Snapshot Inputs. The EMAC_HW_FTR_PSNT2.AUXSNAPNUM field indicates the number of auxiliary snapshot inputs.	
		0	No Auxiliary Input
		1	1 Auxiliary Input
		2	2 Auxiliary Input
		3	3 Auxiliary Input
		4	4 Auxiliary Input
		5	Reserved
26:24 (R/NW)	PPSOUTNUM	Number of PPS Outputs. The EMAC_HW_FTR_PSNT2.PPSOUTNUM field indicates the number of PPS outputs.	
		0	No PPS Output
		1	1 PPS Output

Table 28-158: EMAC_HW_FTR_PSNT2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	2 PPS Output
		3	3 PPS Output
		4	4 PPS Output
		5	Reserved
23:22 (R/NW)	TDCSZ	Tx DMA Descriptor Cache Size. The `EMAC_HW_FTR_PSNT2.TDCSZ` field indicates the size of the Tx DMA descriptor cache in terms of 16 byte descriptors.	
		0	Cache not configured
		1	Four 16 byte descriptors
		2	Eight 16 byte descriptors
		3	Sixteen 16 byte descriptors
21:18 (R/NW)	TXCHCNT	Number of DMA Transmit Channels. The `EMAC_HW_FTR_PSNT2.TXCHCNT` field indicates the number of DMA transmit channels.	
		0	1 MTL Tx Channel
		1	2 MTL Tx Channels
		2	3 MTL Tx Channels
		3	4 MTL Tx Channels
		4	5 MTL Tx Channels
		5	6 MTL Tx Channels
		6	7 MTL Tx Channels
		7	8 MTL Tx Channels
17:16 (R/NW)	RDCSZ	Rx DMA Descriptor Cache Size. The `EMAC_HW_FTR_PSNT2.RDCSZ` field indicates the size of the Rx DMA descriptor cache in terms of 16 byte descriptors.	
		0	Cache not configured
		1	Four 16 byte descriptors
		2	Eight 16 byte descriptors
		3	Sixteen 16 byte descriptors
15:12 (R/NW)	RXCHCNT	Number of DMA Receive Channels. The `EMAC_HW_FTR_PSNT2.RXCHCNT` field indicates the number of DMA receive channels.	

Table 28-158: EMAC_HW_FTR_PSNT2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	1 MTL Rx Channel
		1	2 MTL Rx Channels
		2	3 MTL Rx Channels
		3	4 MTL Rx Channels
		4	5 MTL Rx Channels
		5	6 MTL Rx Channels
		6	7 MTL Rx Channels
		7	8 MTL Rx Channels
9:6 (R/NW)	TXQCNT	Number of MTL Transmit Queues. The `EMAC_HW_FTR_PSNT2.TXQCNT` field indicates the number of MTL transmit queues.	
		0	1 MTL Tx Queue
		1	2 MTL Tx Queues
		2	3 MTL Tx Queues
		3	4 MTL Tx Queues
		4	5 MTL Tx Queues
		5	6 MTL Tx Queues
		6	7 MTL Tx Queues
		7	8 MTL Tx Queues
3:0 (R/NW)	RXQCNT	Number of MTL Receive Queues. The `EMAC_HW_FTR_PSNT2.RXQCNT` field indicates the number of MTL receive queues.	
		0	1 MTL Rx Queue
		1	2 MTL Rx Queues
		2	3 MTL Rx Queues
		3	4 MTL Rx Queues
		4	5 MTL Rx Queues
		5	6 MTL Rx Queues
		6	7 MTL Rx Queues
		7	8 MTL Rx Queues

Hardware Feature Presence Register 3

The EMAC_HW_FTR_PSNT3 register indicates the presence of fourth set the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

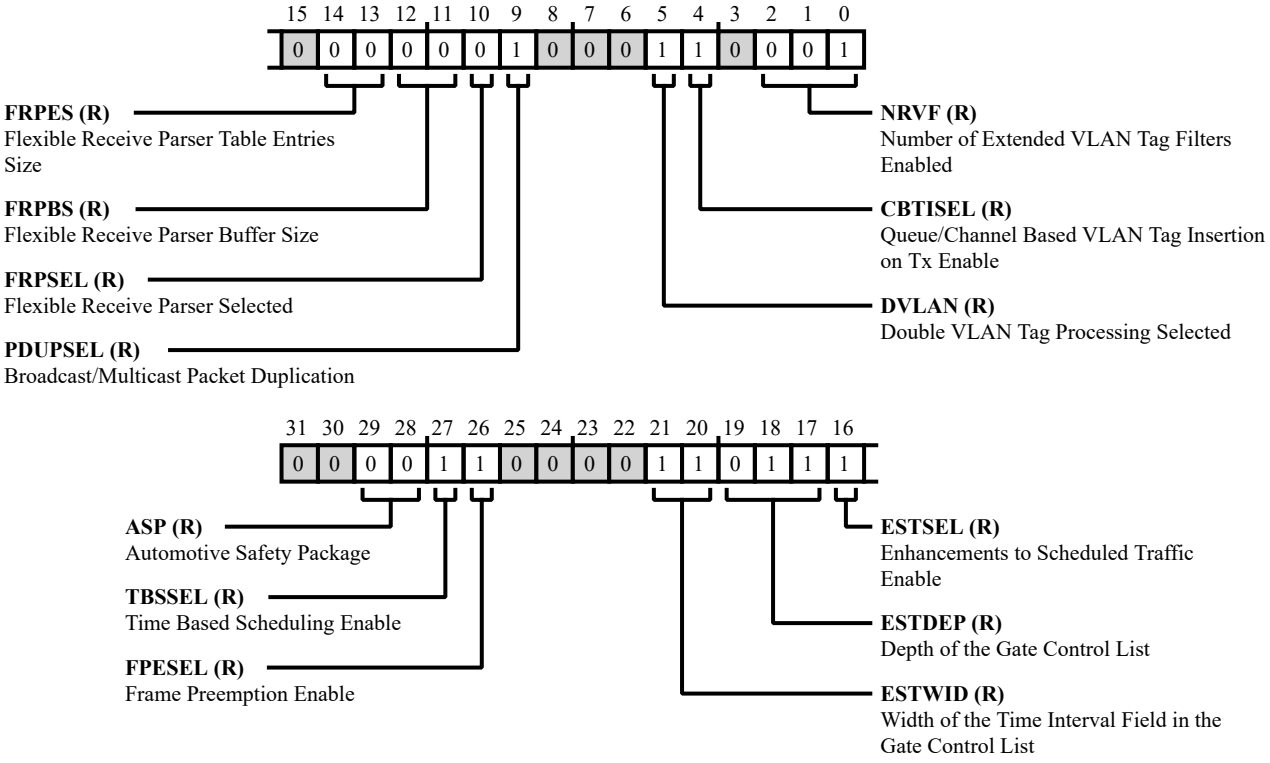


Figure 28-104: EMAC_HW_FTR_PSNT3 Register Diagram

Table 28-159: EMAC_HW_FTR_PSNT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/NW)	ASP	Automotive Safety Package. The EMAC_HW_FTR_PSNT3.ASP field indicates the different safety features.	
		0	No Safety Features Selected
		1	Only "ECC Protection for External Memory" Feature is Selected
		2	All the Automotive Safety Features are Selected Without the "Parity Port Enable for External Interface" Feature
		3	All the Automotive Safety Features are Selected with the "Parity Port Enable for External Interface" Feature

Table 28-159: EMAC_HW_FTR_PSNT3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	TBSSEL	Time Based Scheduling Enable. The `EMAC_HW_FTR_PSNT3.TBSSEL` bit is set (=1) when the Time Based Scheduling feature is selected.	
		0	Time Based Scheduling Enable Feature is Not Selected
		1	Time Based Scheduling Enable Feature is Selected
26 (R/NW)	FPESEL	Frame Preemption Enable. The `EMAC_HW_FTR_PSNT3.FPESEL` bit is set (=1) when the enable frame preemption feature is selected.	
		0	Frame Preemption Enable Feature is Not Selected
		1	Frame Preemption Enable Feature is Selected
21:20 (R/NW)	ESTWID	Width of the Time Interval Field in the Gate Control List. The `EMAC_HW_FTR_PSNT3.ESTWID` field indicates the width of the configured time interval field.	
		0	Width Not Configured
		1	16
		2	20
		3	24
19:17 (R/NW)	ESTDEP	Depth of the Gate Control List. The `EMAC_HW_FTR_PSNT3.ESTDEP` field indicates the depth of the GCL expressed as Log2(DWC_EQOS_EST_DEP)-5	
		0	No Depth Configured
		1	64
		2	128
		3	256
		4	512
		5	1024
		6	Reserved

Table 28-159: EMAC_HW_FTR_PSNT3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	ESTSEL	Enhancements to Scheduled Traffic Enable. The `EMAC_HW_FTR_PSNT3.ESTSEL` bit is set (=1) when the Enable Enhancements to Scheduling Traffic feature is selected.	
		0	Enable Enhancements to Scheduling Traffic Feature is Not Selected
		1	Enable Enhancements to Scheduling Traffic Feature is Selected
14:13 (R/NW)	FRPES	Flexible Receive Parser Table Entries Size. The `EMAC_HW_FTR_PSNT3.FRPES` field indicates the maximum number of parser entries supported the flexible receive parser.	
		0	64 Entries
		1	128 Entries
		2	256 Entries
		3	Reserved
12:11 (R/NW)	FRPBS	Flexible Receive Parser Buffer Size. The `EMAC_HW_FTR_PSNT3.FRPBS` field indicates the supported max number of bytes of the packet data to be parsed by the flexible receive parser.	
		0	64 Bytes
		1	128 Bytes
		2	256 Bytes
		3	Reserved
10 (R/NW)	FRPSEL	Flexible Receive Parser Selected. The `EMAC_HW_FTR_PSNT3.FRPSEL` bit is set (=1) when the enable flexible programmable receive parser option is selected.	
		0	Flexible Receive Parser Feature is Not Selected
		1	Flexible Receive Parser Feature is Selected
9 (R/NW)	PDUPSEL	Broadcast/Multicast Packet Duplication. The `EMAC_HW_FTR_PSNT3.PDUPSEL` bit is set (=1) when the broadcast/multicast packet duplication feature is selected.	
		0	Broadcast/Multicast Packet Duplication Feature is Not Selected
		1	Broadcast/Multicast Packet Duplication Feature is Selected

Table 28-159: EMAC_HW_FTR_PSNT3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	DVLAN	Double VLAN Tag Processing Selected. The `EMAC_HW_FTR_PSNT3.DVLAN` bit is set (=1) when the enable double VLAN processing feature is selected.	
		0	Double VLAN Option is Not Selected
		1	Double VLAN Option is Selected
4 (R/NW)	CBTISEL	Queue/Channel Based VLAN Tag Insertion on Tx Enable. The `EMAC_HW_FTR_PSNT3.CBTISEL` bit is set (=1) when the enable queue/channel-based VLAN tag insertion on Tx feature is selected.	
		0	Enable Queue/Channel Based VLAN Tag Insertion on Tx Feature is Not Selected
		1	Enable Queue/Channel Based VLAN Tag Insertion on Tx Feature is Selected
2:0 (R/NW)	NRVF	Number of Extended VLAN Tag Filters Enabled. The `EMAC_HW_FTR_PSNT3.NRVF` field indicates the number of extended VLAN tag filters selected.	
		0	No Extended Rx VLAN Filters
		1	4 Extended Rx VLAN Filters
		2	8 Extended Rx VLAN Filters
		3	16 Extended Rx VLAN Filters
		4	24 Extended Rx VLAN Filters
		5	32 Extended Rx VLAN Filters
		6	Reserved

MAC Hardware Feature Presence Register 4

The EMAC_MAC_HW_FEATURE4 register indicates the presence of the fifth set of optional features or functions of the EMAC modules. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

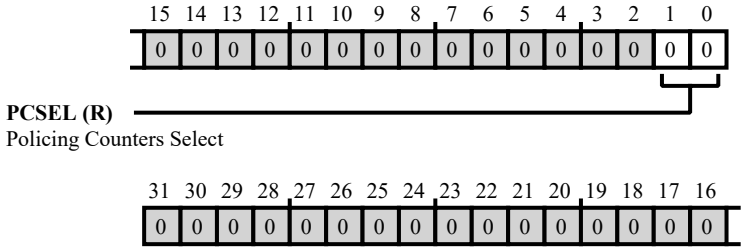


Figure 28-105: EMAC_MAC_HW_FEATURE4 Register Diagram

Table 28-160: EMAC_MAC_HW_FEATURE4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PCSEL	Policing Counters Select. The EMAC_MAC_HW_FEATURE4.PCSEL field indicates the number of policing counters selected. This refers to the TBI, SGMII, or RTBI PHY interface.	
		0	0 Policing Counters
		1	1 Policing Counter
		2	2 Policing Counters
		3	3 Policing Counters

Hash Table Register 0

The EMAC_HASHTBL_REG0 register contains the first 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

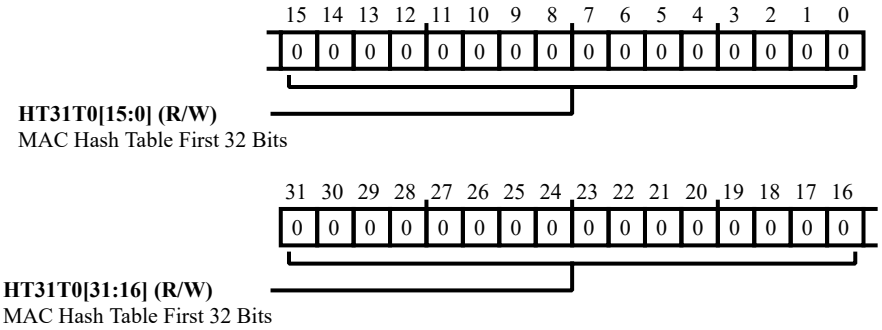


Figure 28-106: EMAC_HASHTBL_REG0 Register Diagram

Table 28-161: EMAC_HASHTBL_REG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT31T0	MAC Hash Table First 32 Bits. The EMAC_HASHTBL_REG0.HT31T0 field contains the first 32 bits [31:0] of the hash table.

Hash Table Register 1

The EMAC_HASHTBL_REG1 register contains the second 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

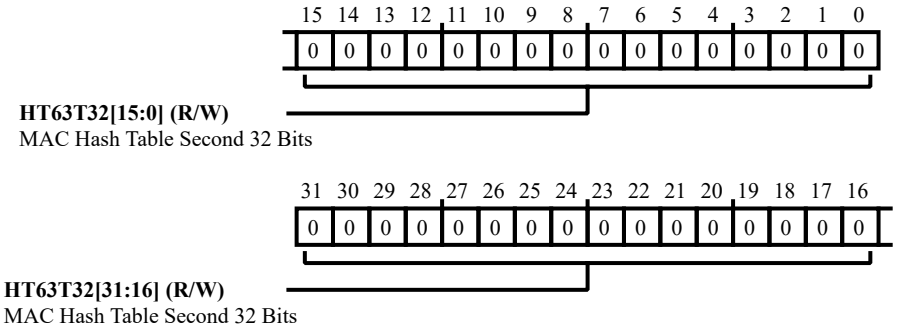


Figure 28-107: EMAC_HASHTBL_REG1 Register Diagram

Table 28-162: EMAC_HASHTBL_REG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT63T32	MAC Hash Table Second 32 Bits. The EMAC_HASHTBL_REG1.HT63T32 field contains the second 32 bits [63:32] of the hash table.

Hash Table Register 2

The EMAC_HASHTBL_REG2 register contains the third 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

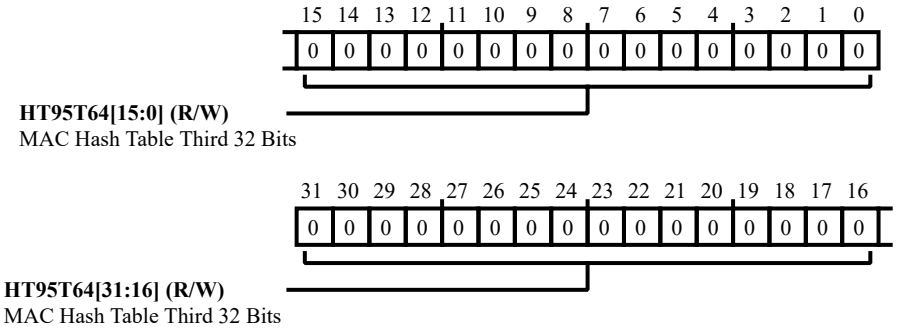


Figure 28-108: EMAC_HASHTBL_REG2 Register Diagram

Table 28-163: EMAC_HASHTBL_REG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT95T64	MAC Hash Table Third 32 Bits. The EMAC_HASHTBL_REG2.HT95T64 field contains the third 32 bits [95:64] of the hash table.

Hash Table Register 3

The EMAC_HASHTBL_REG3 register contains the fourth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

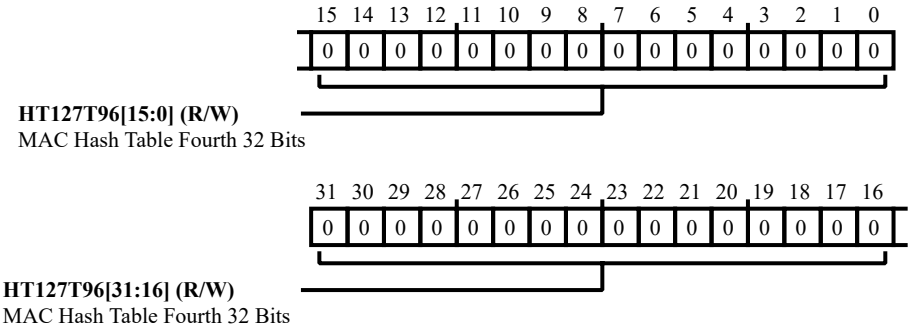


Figure 28-109: EMAC_HASHTBL_REG3 Register Diagram

Table 28-164: EMAC_HASHTBL_REG3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT127T96	MAC Hash Table Fourth 32 Bits. The EMAC_HASHTBL_REG3.HT127T96 field contains the fourth 32 bits [127:96] of the hash table.

Hash Table Register 4

The EMAC_HASHTBL_REG4 register contains the fifth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

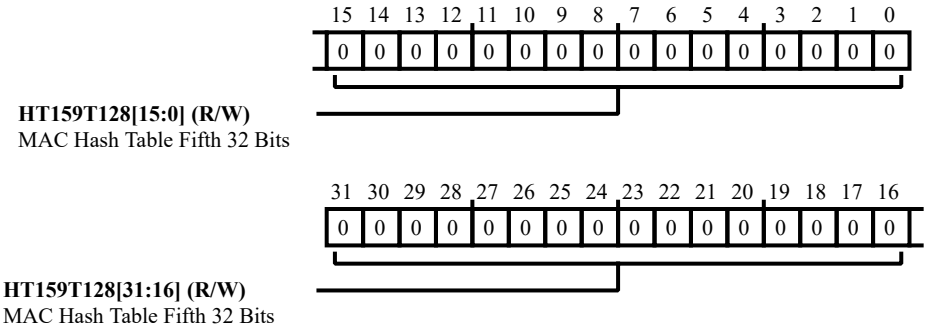


Figure 28-110: EMAC_HASHTBL_REG4 Register Diagram

Table 28-165: EMAC_HASHTBL_REG4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT159T128	MAC Hash Table Fifth 32 Bits. The EMAC_HASHTBL_REG4.HT159T128 field contains the fifth 32 bits [159:128] of the hash table.

Hash Table Register 5

The EMAC_HASHTBL_REG5 register contains the sixth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

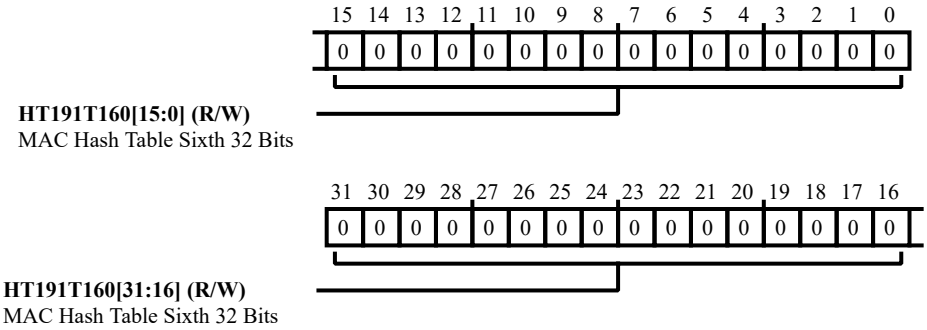


Figure 28-111: EMAC_HASHTBL_REG5 Register Diagram

Table 28-166: EMAC_HASHTBL_REG5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT191T160	MAC Hash Table Sixth 32 Bits. The EMAC_HASHTBL_REG5.HT191T160 field contains the sixth 32 bits [191:160] of the hash table.

Hash Table Register 6

The EMAC_HASHTBL_REG6 register contains the seventh 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

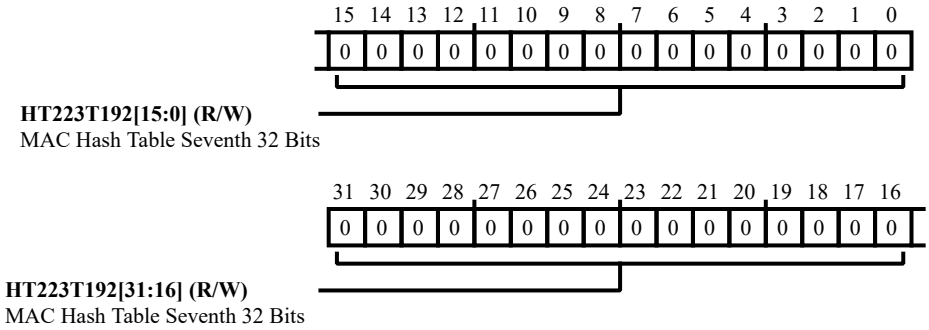


Figure 28-112: EMAC_HASHTBL_REG6 Register Diagram

Table 28-167: EMAC_HASHTBL_REG6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT223T192	MAC Hash Table Seventh 32 Bits. The EMAC_HASHTBL_REG6.HT223T192 field contains the seventh 32 bits [223:192] of the hash table.

Hash Table Register 7

The EMAC_HASHTBL_REG7 register contains the eighth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

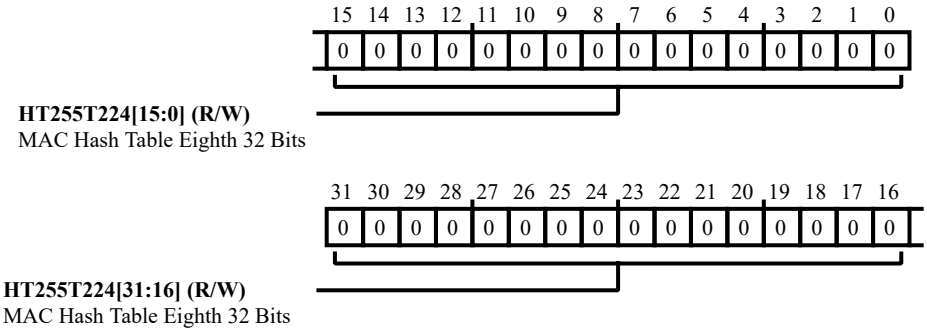


Figure 28-113: EMAC_HASHTBL_REG7 Register Diagram

Table 28-168: EMAC_HASHTBL_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT255T224	MAC Hash Table Eighth 32 Bits. The EMAC_HASHTBL_REG7.HT255T224 field contains the eighth 32 bits [255:224] of the hash table.

Indirect Access Control Register

The EMAC_INDR_ACC_CTL register provides the indirect access control and status for MAC_<MSEl>_<AOFF> registers.

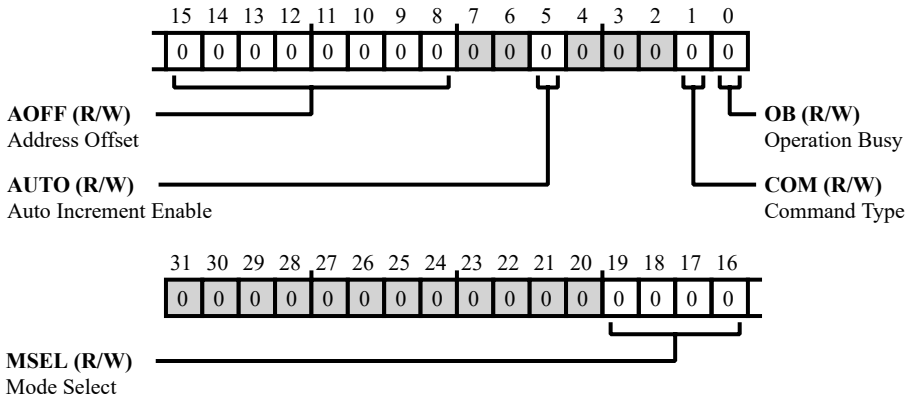


Figure 28-114: EMAC_INDR_ACC_CTL Register Diagram

Table 28-169: EMAC_INDR_ACC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	MSEL	Mode Select. The EMAC_INDR_ACC_CTL.MSEL field is used in indirect access of MAC_<MSEL>_<AOFF>. The EMAC_INDR_ACC_CTL.MSEL field must be set along with initiation of read/write to MAC_<MSEL>_<AOFF> and should not be changed until the OB is cleared. Values: - 0000:Typ_RXQ_ (Type based RXQ mapping) - 0001-1111: Reserved
15:8 (R/W)	AOFF	Address Offset. The EMAC_INDR_ACC_CTL.AOFF field is used in indirect access of MAC_<MSEL>_<AOFF>. The EMAC_INDR_ACC_CTL.AOFF field must be configured along with initiation of read/write to MAC_<MSEL>_<AOFF> and should not be changed until the OB is cleared. Values: - 00000000: IndReg0(Indirect register 0) - 00000001: IndReg1(Indirect register 1) - ... - 00000111: IndReg7(Indirect register 7) - 00001000-11111111: Reserved

Table 28-169: EMAC_INDR_ACC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	AUTO	Auto Increment Enable. The EMAC_INDR_ACC_CTL.AUTO bit enables the address offset auto increment feature. When set (=1), AOFF is incremented by 1. Software should ensure not to cause a wrap condition. Byte wise read/write is not supported when auto increment is enabled. When the EMAC_INDR_ACC_CTL.AUTO bit is cleared (=0), AOFF is not incremented automatically. Software should program the correct Address Offset for each access.	
1 (R/W)	COM	Command Type. The EMAC_INDR_ACC_CTL.COM bit indicates the register access type. - 1: Indicates a read operation. - 0: Indicates a write operation.	
		0	Write Operation
		1	Read Operation
0 (R/W)	OB	Operation Busy. The EMAC_INDR_ACC_CTL.OB bit is set along with a read or write command for initiating the indirect access to MAC_<MSEL>_<AOFF> register. The EMAC_INDR_ACC_CTL.OB bit is cleared (=0) when the read or write command to MAC_<MSEL>_<AOFF> register is complete. The next indirect register access can be initiated only after EMAC_INDR_ACC_CTL.OB = 0. During a write operation, the bit is cleared (=0) only after the data has been written into MAC_<MSEL>_<AOFF> register. During a read operation, the data should be read from the EMAC_INDR_ACC_DAT register only after EMAC_INDR_ACC_CTL.OB = 0.	

Indirect Access Data Register

The EMAC_INDR_ACC_DAT register holds the read/write data for indirect access of MAC_<MSEL_<AOFF> registers.

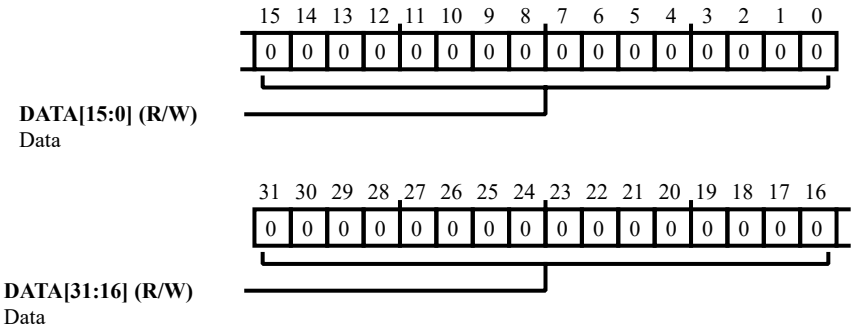


Figure 28-115: EMAC_INDR_ACC_DAT Register Diagram

Table 28-170: EMAC_INDR_ACC_DAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Data. The EMAC_INDR_ACC_DAT.DATA field contains data to read/write for indirect address access associated with EMAC_INDR_ACC_DAT register.

Inner VLAN Tag Inclusion or Replacement Register

The EMAC_INVLAN_INCL register contains the inner VLAN tag to be inserted or replaced in the transmit packet. It also contains the inner VLAN tag insertion controls.

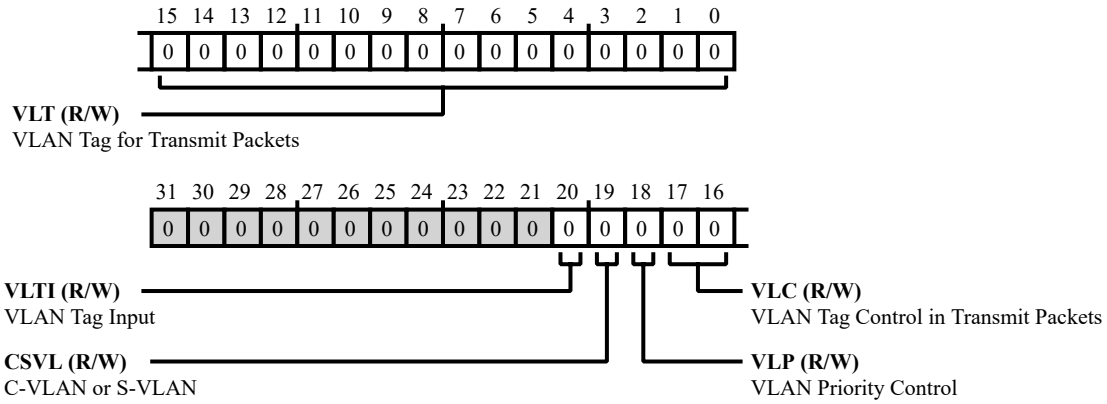


Figure 28-116: EMAC_INVLAN_INCL Register Diagram

Table 28-171: EMAC_INVLAN_INCL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	VLTI	VLAN Tag Input. When the EMAC_INVLAN_INCL.VLTI bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.	
		0	VLAN Tag Input is Disabled
		1	VLAN Tag Input is Enabled
19 (R/W)	CSVL	C-VLAN or S-VLAN. When the EMAC_INVLAN_INCL.CSVL bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 17th and 18th bytes of transmitted packets. When EMAC_INVLAN_INCL.CSVL = 0, C-VLAN type (0x8100) is inserted or replaced in the 17th and 18th bytes of transmitted packets.	
		0	C-VLAN Type (0x8100) is Inserted
		1	S-VLAN Type (0x88A8) is Inserted
18 (R/W)	VLP	VLAN Priority Control. When the EMAC_INVLAN_INCL.VLP bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When EMAC_INVLAN_INCL.VLP = 0, the TDES2.VTIR descriptor control is used and the VLC field is ignored.	
		0	VLAN Priority Control is Disabled
		1	VLAN Priority Control is Enabled

Table 28-171: EMAC_INVLAN_INCL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	VLC	VLAN Tag Control in Transmit Packets. The EMAC_INVLAN_INCL.VLC field indicates the VLAN tag control in transmit packets. 2'b00: No VLAN tag deletion, insertion, or replacement 2'b01: VLAN tag deletion. The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags. 2'b10: VLAN tag insertion. The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 2'b11: VLAN tag replacement. The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8). Note: Changes to this field take effect only on the start of a packet. If this register field is written to when a packet is being transmitted, only the subsequent packet can use the updated value; the current packet does not use the updated value.	
		0	No VLAN Tag Deletion, Insertion, or Replacement
		1	VLAN Tag Deletion
		2	VLAN Tag Insertion
		3	VLAN Tag Replacement
15:0 (R/W)	VLT	VLAN Tag for Transmit Packets. The EMAC_INVLAN_INCL.VLT field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: - Bits[15:13]: User Priority - Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) - Bits[11:0]: VLAN Identifier (VID) field of VLAN tag	

MAC Interrupt Enable Register

The EMAC_MAC_IEN register contains the masks for generating the interrupts.

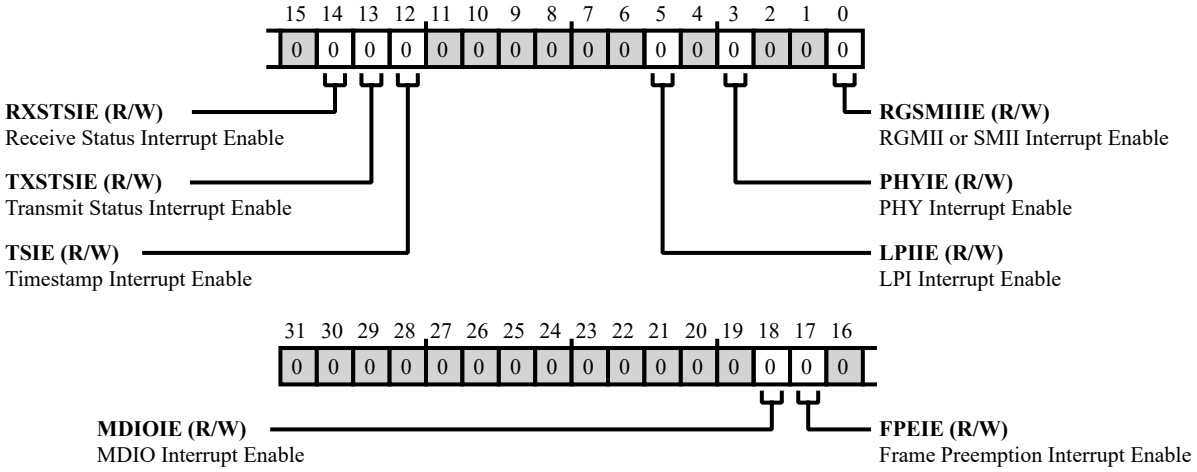


Figure 28-117: EMAC_MAC_IEN Register Diagram

Table 28-172: EMAC_MAC_IEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	MDIOIE	MDIO Interrupt Enable. When the EMAC_MAC_IEN.MDIOIE bit is set, it enables the assertion of the interrupt when EMAC_MAC_ISTAT.MDIOIS = 1.	
		0	MDIO Interrupt is Disabled
		1	MDIO Interrupt is Enabled
17 (R/W)	FPEIE	Frame Preemption Interrupt Enable. When the EMAC_MAC_IEN.FPEIE bit is set, it enables the assertion of the interrupt when EMAC_MAC_ISTAT.FPEIS = 1.	
		0	Frame Preemption Interrupt is Disabled
		1	Frame Preemption Interrupt is Enabled
14 (R/W)	RXSTSIE	Receive Status Interrupt Enable. When the EMAC_MAC_IEN.RXSTSIE bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.RXSTSIS bit.	
		0	Receive Status Interrupt is Disabled
		1	Receive Status Interrupt is Enabled

Table 28-172: EMAC_MAC_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TXSTSIE	Transmit Status Interrupt Enable. When the `EMAC_MAC_IEN.TXSTSIE` bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.TXSTSIS bit.	
		0	Disable
		1	Enable
12 (R/W)	TSIE	Timestamp Interrupt Enable. When the `EMAC_MAC_IEN.TSIE` bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.TSIS bit.	
		0	Disable
		1	Enable
5 (R/W)	LPIIE	LPI Interrupt Enable. When the `EMAC_MAC_IEN.LPIIE` bit is set, it enables the assertion of the interrupt signal because of EMAC_MAC_ISTAT.LPIIS.	
		0	LPI Interrupt is Disabled
		1	LPI Interrupt is Enabled
3 (R/W)	PHYIE	PHY Interrupt Enable. When the `EMAC_MAC_IEN.PHYIE` bit is set, it enables the assertion of the interrupt signal because of the EMAC_MAC_ISTAT.PHYIS bit.	
		0	PHY Interrupt is Disabled
		1	PHY Interrupt is Enabled
0 (R/W)	RGSMIIIE	RGMII or SMII Interrupt Enable. When the `EMAC_MAC_IEN.RGSMIIIE` bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.RGSMIIIS bit.	
		0	RGMII or SMII Interrupt is Disabled
		1	RGMII or SMII Interrupt is Enabled

MAC Interrupt Status Register

The EMAC_MAC_ISTAT register contains the status of interrupts.

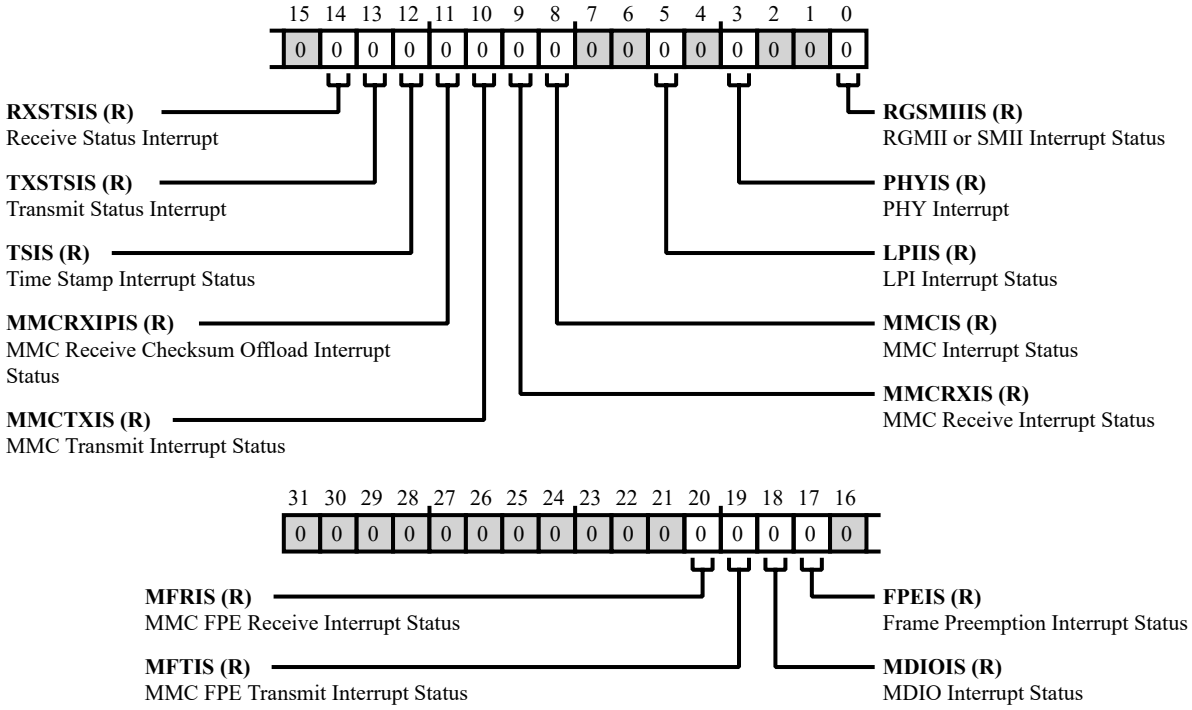


Figure 28-118: EMAC_MAC_ISTAT Register Diagram

Table 28-173: EMAC_MAC_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	MFRIS	MMC FPE Receive Interrupt Status. The EMAC_MAC_ISTAT.MFRIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_FPE_RXINT register. The EMAC_MAC_ISTAT.MFRIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MFRIS bit is valid only when MMC and FPE support are enabled.	
		0	MMC FPE Receive Interrupt Status Not Active
		1	MMC FPE Receive Interrupt Status Active

Table 28-173: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	MFTIS	MMC FPE Transmit Interrupt Status. The EMAC_MAC_ISTAT.MFTIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_FPE_TXINT register. The EMAC_MAC_ISTAT.MFTIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MFTIS bit is valid only when MMC and FPE support are enabled.	
		0	MMC FPE Transmit Interrupt Status Not Active
		1	MMC FPE Transmit Interrupt Status Active
18 (R/NW)	MDIOIS	MDIO Interrupt Status. The EMAC_MAC_ISTAT.MDIOIS bit indicates an interrupt event after the completion of MDIO operation. To reset the EMAC_MAC_ISTAT.MDIOIS bit, the application has to read the EMAC_MAC_ISTAT.MDIOIS bit/write 1 to the EMAC_MAC_ISTAT.MDIOIS bit when EMAC_CSR_SW_CTL.RCWE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	MDIO Interrupt Status Not Active
		1	MDIO Interrupt Status Active
17 (R/NW)	FPEIS	Frame Preemption Interrupt Status. The EMAC_MAC_ISTAT.FPEIS bit indicates an interrupt event during the operation of frame preemption (Bits[19:16] of EMAC_FPE_CTLSTAT register are set). To reset the EMAC_MAC_ISTAT.FPEIS bit, the application must clear the event in EMAC_FPE_CTLSTAT that caused the Interrupt.	
		0	Frame Preemption Interrupt Status Not Active
		1	Frame Preemption Interrupt Status Active
14 (R/NW)	RXSTSIS	Receive Status Interrupt. The EMAC_MAC_ISTAT.RXSTSIS bit indicates the status of received packets. The EMAC_MAC_ISTAT.RXSTSIS bit is set when the EMAC_RXTX_STAT.RWT = 1. The EMAC_MAC_ISTAT.RXSTSIS bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1) in the EMAC_RXTX_STAT register.	
		0	Receive Interrupt Status Not Active
		1	Receive Interrupt Status Active

Table 28-173: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/NW)	TXSTSIS	Transmit Status Interrupt. The EMAC_MAC_ISTAT.TXSTSIS bit indicates the status of transmitted packets. The EMAC_MAC_ISTAT.TXSTSIS bit is set when any of the following bits is set in the EMAC_RXTX_STAT register: Excessive Collision (EXCOL) Late Collision (LCOL) Excessive Deferral (EXDEF) Loss of Carrier (LCARR) No Carrier (NCARR) Jabber Timeout (TJT) The EMAC_MAC_ISTAT.TXSTSIS bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1) in the EMAC_RXTX_STAT register).	
		0	Status Not Active
		1	Status Active
12 (R/NW)	TSIS	Time Stamp Interrupt Status. If the time stamp feature is enabled, the EMAC_MAC_ISTAT.TSIS bit is set when any of the following conditions are true: The system time value is equal to or exceeds the value specified in the Target Time High and Low registers. There is an overflow in the seconds register. The Target Time Error occurred, that is, programmed target time already elapsed. If the Auxiliary Snapshot feature is enabled, the EMAC_MAC_ISTAT.TSIS bit is set when the auxiliary snapshot trigger is asserted. In configurations other than EQOS_CORE, when drop transmit status is enabled in MTL, the EMAC_MAC_ISTAT.TSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers. When PTP offload feature is enabled, the EMAC_MAC_ISTAT.TSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers, for PTO generated Delay Request and Pdelay request packets. The EMAC_MAC_ISTAT.TSIS bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1) in the EMAC_TM_STAT register.	
		0	Status Not Active
		1	Status Active

Table 28-173: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	MMCRXIPIS	MMC Receive Checksum Offload Interrupt Status. The EMAC_MAC_ISTAT.MMCRXIPIS bit is set (=1) when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. The EMAC_MAC_ISTAT.MMCRXIPIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MMCRXIPIS bit is valid only when the Enable MAC Management Counters (MMC) and Enable Receive TCP/IP Checksum Check options are selected.	
		0	MMC Receive Checksum Offload Interrupt Status Not Active
		1	MMC Receive Checksum Offload Interrupt Status Active
10 (R/NW)	MMCTXIS	MMC Transmit Interrupt Status. The EMAC_MAC_ISTAT.MMCTXIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_TXINT register. The EMAC_MAC_ISTAT.MMCTXIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MMCTXIS bit is valid only when the Enable MAC Management Counters (MMC) option is selected.	
		0	MMC Transmit Interrupt Status Not Active
		1	MMC Transmit Interrupt Status Active
9 (R/NW)	MMCRXIS	MMC Receive Interrupt Status. The EMAC_MAC_ISTAT.MMCRXIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_RXINT register. The EMAC_MAC_ISTAT.MMCRXIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MMCRXIS bit is valid only when the Enable MAC Management Counters (MMC) option is selected.	
		0	MMC Receive Interrupt Status Not Active
		1	MMC Receive Interrupt Status Active
8 (R/NW)	MMCIS	MMC Interrupt Status. The EMAC_MAC_ISTAT.MMCIS bit is set (=1) when Bit 11, Bit 10, or Bit 9 is set (=1). The EMAC_MAC_ISTAT.MMCIS bit is cleared only when all these bits are low. The EMAC_MAC_ISTAT.MMCIS bit is valid only when the Enable MAC Management Counters (MMC) option is selected.	
		0	MMC Interrupt Status Not Active
		1	MMC Interrupt Status Active

Table 28-173: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	LPIIS	LPI Interrupt Status. When the Energy Efficient Ethernet feature is enabled, the EMAC_MAC_ISTAT.LPIIS bit is set for any LPI state entry or exit in the MAC transmitter or receiver. The EMAC_MAC_ISTAT.LPIIS bit is cleared when the corresponding interrupt source bit of the EMAC_LPI_CTLSTAT register is read (or corresponding interrupt source bit of EMAC_LPI_CTLSTAT register is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	LPI Interrupt Status Not Active
		1	LPI Interrupt Status Active
3 (R/NW)	PHYIS	PHY Interrupt. The EMAC_MAC_ISTAT.PHYIS bit is set when rising edge is detected on the phy_intr_i input. The EMAC_MAC_ISTAT.PHYIS bit is cleared when this register is read (or the EMAC_MAC_ISTAT.PHYIS bit is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1)).	
		0	PHY Interrupt Not Detected
		1	PHY Interrupt Detected
0 (R/NW)	RGSMIIIS	RGMII or SMII Interrupt Status. The EMAC_MAC_ISTAT.RGSMIIIS bit is set because of any change in value of the link status of RGMII or SMII interface (EMAC_PHYIF_CTLSTAT.LNKSTS). The EMAC_MAC_ISTAT.RGSMIIIS bit is cleared when the EMAC_PHYIF_CTLSTAT register is read (or EMAC_PHYIF_CTLSTAT.LNKSTS is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). The EMAC_MAC_ISTAT.RGSMIIIS bit is valid only when the optional RGMII or SMII PHY interface is selected.	
		0	RGMII or SMII Interrupt Status is Not Active
		1	RGMII or SMII Interrupt Status is Active

Layer 3 and Layer 4 Control Register

The EMAC_L3L4_CTL7 register controls the operations of filter 0 of layer 3 and layer 4.

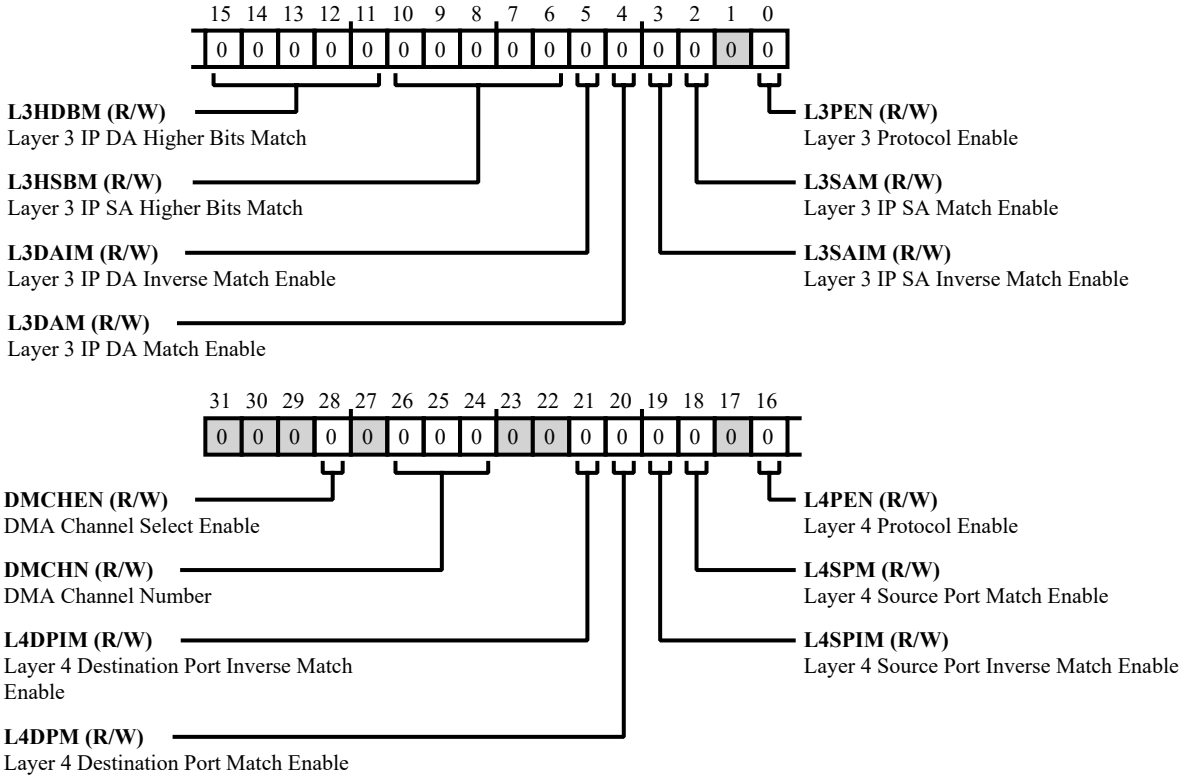


Figure 28-119: EMAC_L3L4_CTL7 Register Diagram

Table 28-174: EMAC_L3L4_CTL7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	DMCHEN	DMA Channel Select Enable. When set, the EMAC_L3L4_CTL7.DMCHEN bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When EMAC_L3L4_CTL7.DMCHEN = 0, the DMA channel is not decided by this filter.	
		0	DMA Channel Select is Disabled
		1	DMA Channel Select is Enabled
26:24 (R/W)	DMCHN	DMA Channel Number. When DMCHEN is set (=1), the EMAC_L3L4_CTL7.DMCHN field selects the DMA channel number to which the packet passed by this filter is routed. The width of the EMAC_L3L4_CTL7.DMCHN field depends on the number of the DMA channels present in the configuration.	

Table 28-174: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	L4DPIM	Layer 4 Destination Port Inverse Match Enable. When the `EMAC_L3L4_CTL7.L4DPIM` bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When `EMAC_L3L4_CTL7.L4DPIM` = 0, the Layer 4 Destination Port number field is enabled for perfect matching. The `EMAC_L3L4_CTL7.L4DPIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Destination Port Inverse Match is Disabled
		1	Layer 4 Destination Port Inverse Match is Enabled
20 (R/W)	L4DPM	Layer 4 Destination Port Match Enable. When the `EMAC_L3L4_CTL7.L4DPM` bit is set, the Layer 4 Destination Port number field is enabled for matching. When `EMAC_L3L4_CTL7.L4DPM` = 0, the MAC ignores the Layer 4 Destination Port number field for matching.	
		0	Layer 4 Destination Port Match is Disabled
		1	Layer 4 Destination Port Match is Enabled
19 (R/W)	L4SPIM	Layer 4 Source Port Inverse Match Enable. When the `EMAC_L3L4_CTL7.L4SPIM` bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When `EMAC_L3L4_CTL7.L4SPIM` = 0, the Layer 4 Source Port number field is enabled for perfect matching. The `EMAC_L3L4_CTL7.L4SPIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4SPM= 1.	
		0	Layer 4 Source Port Inverse Match is Disabled
		1	Layer 4 Source Port Inverse Match is Enabled
18 (R/W)	L4SPM	Layer 4 Source Port Match Enable. When the `EMAC_L3L4_CTL7.L4SPM` bit is set, the Layer 4 Source Port number field is enabled for matching. When `EMAC_L3L4_CTL7.L4SPM` = 0, the MAC ignores the Layer 4 Source Port number field for matching.	
		0	Layer 4 Source Port Match is Disabled
		1	Layer 4 Source Port Match is Enabled

Table 28-174: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	L4PEN	Layer 4 Protocol Enable. When the EMAC_L3L4_CTL7.L4PEN bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When EMAC_L3L4_CTL7.L4PEN = 0, the Source and Destination Port number fields of TCP packets are used for matching. The layer 4 matching is done only when the EMAC_L3L4_CTL[n].L4SPM or EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Protocol is Disabled
		1	Layer 4 Protocol is Enabled
15:11 (R/W)	L3HDBM	Layer 3 IP DA Higher Bits Match. For IPv4 packets, the EMAC_L3L4_CTL7.L3HDBM field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the bits[12:11] of this field correspond to bits[6:5] of L3HSBM which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM[1:0] and L3HSBM bits: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 127: All bits except MSb are masked The EMAC_L3L4_CTL7.L3HDBM field is valid and applicable only when the L3DAM or L3SAM bit is set.	

Table 28-174: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:6 (R/W)	L3HSBM	Layer 3 IP SA Higher Bits Match. For IPv4 packets, the EMAC_L3L4_CTL7.L3HSBM field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the EMAC_L3L4_CTL7.L3HSBM field contains bits[4:0] of L3HSBM. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. The EMAC_L3L4_CTL7.L3HSBM field is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM or L3SAM bit is set (=1).	
5 (R/W)	L3DAIM	Layer 3 IP DA Inverse Match Enable. When the EMAC_L3L4_CTL7.L3DAIM bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When EMAC_L3L4_CTL7.L3DAIM = 0, the Layer 3 IP Destination Address field is enabled for perfect matching. The EMAC_L3L4_CTL7.L3DAIM bit is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP DA Inverse Match is Disabled
		1	Layer 3 IP DA Inverse Match is Enabled
4 (R/W)	L3DAM	Layer 3 IP DA Match Enable. When the EMAC_L3L4_CTL7.L3DAM bit is set, the Layer 3 IP Destination Address field is enabled for matching. When EMAC_L3L4_CTL7.L3DAM = 0, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When EMAC_L3L4_CTL[n].L3PEN = 1, set either the EMAC_L3L4_CTL7.L3DAM bit or the L3SAM bit because either IPv6 DA or SA can be checked for filtering.	
		0	Layer 3 IP DA Match is Disabled
		1	Layer 3 IP DA Match is Enabled

Table 28-174: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	L3SAIM	Layer 3 IP SA Inverse Match Enable. When the EMAC_L3L4_CTL7.L3SAIM bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When the EMAC_L3L4_CTL7.L3SAIM bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. The EMAC_L3L4_CTL7.L3SAIM bit is valid and applicable only when EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP SA Inverse Match is Disabled
		1	Layer 3 IP SA Inverse Match is Enabled
2 (R/W)	L3SAM	Layer 3 IP SA Match Enable. When the EMAC_L3L4_CTL7.L3SAM bit is set, the Layer 3 IP Source Address field is enabled for matching. When EMAC_L3L4_CTL7.L3SAM = 0, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When the EMAC_L3L4_CTL[n].L3PEN bit is set, set either the EMAC_L3L4_CTL7.L3SAM bit or the EMAC_L3L4_CTL[n].L3DAM bit because either IPv6 SA or DA can be checked for filtering.	
		0	Layer 3 IP SA Match is Disabled
		1	Layer 3 IP SA Match is Enabled
0 (R/W)	L3PEN	Layer 3 Protocol Enable. When the EMAC_L3L4_CTL7.L3PEN bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When EMAC_L3L4_CTL7.L3PEN = 0, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets. The layer 3 matching is done only when the EMAC_L3L4_CTL[n].L3SAM or EMAC_L3L4_CTL[n].L3DAM bit is set (=1).	
		0	Layer 3 Protocol is Disabled
		1	Layer 3 Protocol is Enabled

Layer 3 and Layer 4 Control Register

The EMAC_L3L4_CTL[i] register controls the operations of filter 0 of layer 3 and layer 4.

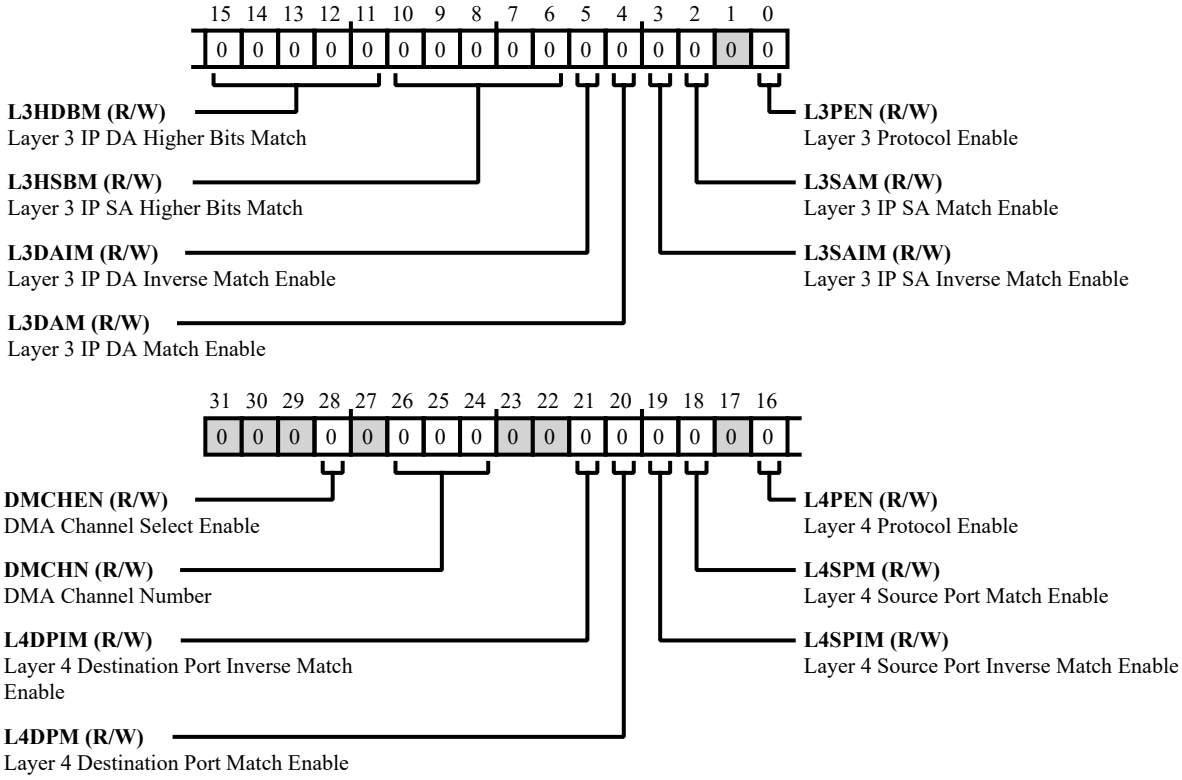


Figure 28-120: EMAC_L3L4_CTL[i] Register Diagram

Table 28-175: EMAC_L3L4_CTL[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	DMCHEN	DMA Channel Select Enable. When set, the EMAC_L3L4_CTL[i].DMCHEN bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When EMAC_L3L4_CTL[i].DMCHEN = 0, the DMA channel is not decided by this filter.	
		0	DMA Channel Select is Disabled
		1	DMA Channel Select is Enabled
26:24 (R/W)	DMCHN	DMA Channel Number. When DMCHEN is set (=1), the EMAC_L3L4_CTL[i].DMCHN field selects the DMA channel number to which the packet passed by this filter is routed. The width of the EMAC_L3L4_CTL[i].DMCHN field depends on the number of the DMA channels present in the configuration.	

Table 28-175: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	L4DPIM	Layer 4 Destination Port Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L4DPIM` bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When `EMAC_L3L4_CTL[i].L4DPIM` = 0, the Layer 4 Destination Port number field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L4DPIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Destination Port Inverse Match is Disabled
		1	Layer 4 Destination Port Inverse Match is Enabled
20 (R/W)	L4DPM	Layer 4 Destination Port Match Enable. When the `EMAC_L3L4_CTL[i].L4DPM` bit is set, the Layer 4 Destination Port number field is enabled for matching. When `EMAC_L3L4_CTL[i].L4DPM` = 0, the MAC ignores the Layer 4 Destination Port number field for matching.	
		0	Layer 4 Destination Port Match is Disabled
		1	Layer 4 Destination Port Match is Enabled
19 (R/W)	L4SPIM	Layer 4 Source Port Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L4SPIM` bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When `EMAC_L3L4_CTL[i].L4SPIM` = 0, the Layer 4 Source Port number field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L4SPIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4SPM= 1.	
		0	Layer 4 Source Port Inverse Match is Disabled
		1	Layer 4 Source Port Inverse Match is Enabled
18 (R/W)	L4SPM	Layer 4 Source Port Match Enable. When the `EMAC_L3L4_CTL[i].L4SPM` bit is set, the Layer 4 Source Port number field is enabled for matching. When `EMAC_L3L4_CTL[i].L4SPM` = 0, the MAC ignores the Layer 4 Source Port number field for matching.	
		0	Layer 4 Source Port Match is Disabled
		1	Layer 4 Source Port Match is Enabled

Table 28-175: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	L4PEN	Layer 4 Protocol Enable. When the `EMAC_L3L4_CTL[i].L4PEN` bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When `EMAC_L3L4_CTL[i].L4PEN` = 0, the Source and Destination Port number fields of TCP packets are used for matching. The layer 4 matching is done only when the EMAC_L3L4_CTL[n].L4SPM or EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Protocol is Disabled
		1	Layer 4 Protocol is Enabled
15:11 (R/W)	L3HDBM	Layer 3 IP DA Higher Bits Match. For IPv4 packets, the `EMAC_L3L4_CTL[i].L3HDBM` field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the bits[12:11] of this field correspond to bits[6:5] of L3HSBM which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM[1:0] and L3HSBM bits: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 127: All bits except MSb are masked The `EMAC_L3L4_CTL[i].L3HDBM` field is valid and applicable only when the L3DAM or L3SAM bit is set.	

Table 28-175: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:6 (R/W)	L3HSBM	Layer 3 IP SA Higher Bits Match. For IPv4 packets, the `EMAC_L3L4_CTL[i].L3HSBM` field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the `EMAC_L3L4_CTL[i].L3HSBM` field contains bits[4:0] of L3HSBM. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. The `EMAC_L3L4_CTL[i].L3HSBM` field is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM or L3SAM bit is set (=1).	
5 (R/W)	L3DAIM	Layer 3 IP DA Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L3DAIM` bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When `EMAC_L3L4_CTL[i].L3DAIM` = 0, the Layer 3 IP Destination Address field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L3DAIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP DA Inverse Match is Disabled
		1	Layer 3 IP DA Inverse Match is Enabled
4 (R/W)	L3DAM	Layer 3 IP DA Match Enable. When the `EMAC_L3L4_CTL[i].L3DAM` bit is set, the Layer 3 IP Destination Address field is enabled for matching. When `EMAC_L3L4_CTL[i].L3DAM` = 0, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When EMAC_L3L4_CTL[n].L3PEN = 1, set either the `EMAC_L3L4_CTL[i].L3DAM` bit or the L3SAM bit because either IPv6 DA or SA can be checked for filtering.	
		0	Layer 3 IP DA Match is Disabled
		1	Layer 3 IP DA Match is Enabled

Table 28-175: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	L3SAIM	Layer 3 IP SA Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L3SAIM` bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When the `EMAC_L3L4_CTL[i].L3SAIM` bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L3SAIM` bit is valid and applicable only when EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP SA Inverse Match is Disabled
		1	Layer 3 IP SA Inverse Match is Enabled
2 (R/W)	L3SAM	Layer 3 IP SA Match Enable. When the `EMAC_L3L4_CTL[i].L3SAM` bit is set, the Layer 3 IP Source Address field is enabled for matching. When `EMAC_L3L4_CTL[i].L3SAM` = 0, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When the EMAC_L3L4_CTL[n].L3PEN bit is set, set either the `EMAC_L3L4_CTL[i].L3SAM` bit or the EMAC_L3L4_CTL[n].L3DAM bit because either IPv6 SA or DA can be checked for filtering.	
		0	Layer 3 IP SA Match is Disabled
		1	Layer 3 IP SA Match is Enabled
0 (R/W)	L3PEN	Layer 3 Protocol Enable. When the `EMAC_L3L4_CTL[i].L3PEN` bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When `EMAC_L3L4_CTL[i].L3PEN` = 0, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets. The layer 3 matching is done only when the EMAC_L3L4_CTL[n].L3SAM or EMAC_L3L4_CTL[n].L3DAM bit is set (=1).	
		0	Layer 3 Protocol is Disabled
		1	Layer 3 Protocol is Enabled

LPI Control and Status Register

The EMAC_LPI_CTLSTAT register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

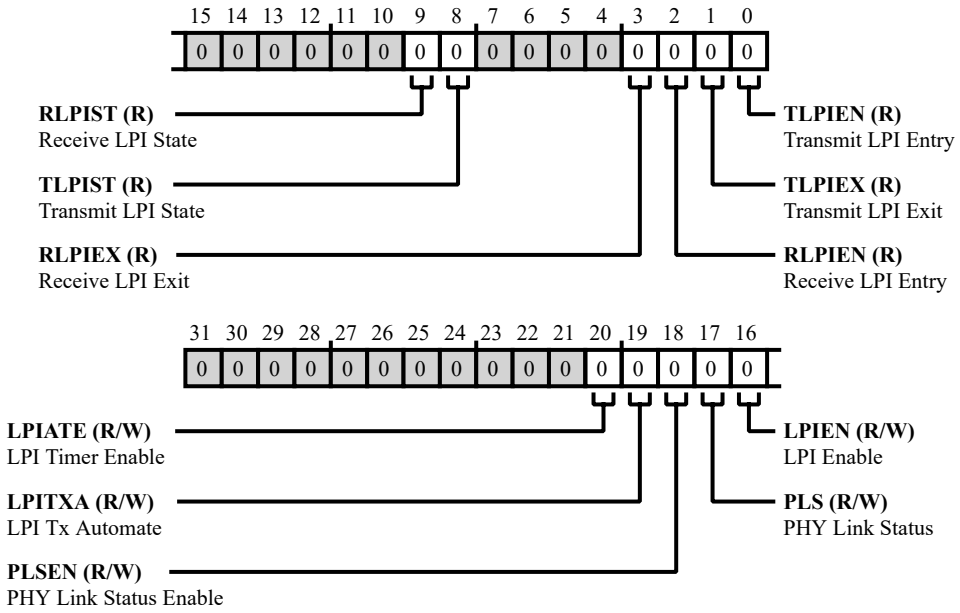


Figure 28-121: EMAC_LPI_CTLSTAT Register Diagram

Table 28-176: EMAC_LPI_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	LPIATE	LPI Timer Enable. The EMAC_LPI_CTLSTAT.LPIATE bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPIATE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the EMAC_LPI_ENTR_TMR register. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again. When EMAC_LPI_CTLSTAT.LPIATE = 0, the LPI auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.	
		0	LPI Timer is Disabled
		1	LPI Timer is Enabled

Table 28-176: EMAC_LPI_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	LPITXA	LPI Tx Automate. The EMAC_LPI_CTLSTAT.LPITXA bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the transmit side. If the EMAC_LPI_CTLSTAT.LPITXA and LPIEN bits are set (=1), the MAC enters the LPI mode only after all outstanding packets (in the IP) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO flush is set in EMAC_TQ0_OPMODE.FTQ, when the MAC is in the LPI mode, it exits the LPI mode. When the EMAC_LPI_CTLSTAT.LPITXA bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.	
		0	LPI Tx Automate is Disabled
		1	LPI Tx Automate is Enabled
18 (R/W)	PLSEN	PHY Link Status Enable. The EMAC_LPI_CTLSTAT.PLSEN bit enables the link status received on the RGMII receive paths to be used for activating the LPI LS TIMER. When the EMAC_LPI_CTLSTAT.PLSEN bit is set, the MAC uses the link-status bits of the EMAC_PHYIF_CTLSTAT register and the PLS bit for the LPI LS timer trigger. When EMAC_LPI_CTLSTAT.PLSEN = 0, the MAC ignores the link-status bits of the EMAC_PHYIF_CTLSTAT register and takes only the PLS bit.	
		0	PHY Link Status is Disabled
		1	PHY Link Status is Enabled
17 (R/W)	PLS	PHY Link Status. The EMAC_LPI_CTLSTAT.PLS bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When the EMAC_LPI_CTLSTAT.PLS bit is set, the link is considered to be okay (UP) and when EMAC_LPI_CTLSTAT.PLS = 0, the link is considered to be down.	
		0	Link is Down
		1	Link is Okay (UP)

Table 28-176: EMAC_LPI_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	LPIEN	LPI Enable. When the EMAC_LPI_CTLSTAT.LPIEN bit is set, it instructs the MAC transmitter to enter the LPI state. When EMAC_LPI_CTLSTAT.LPIEN = 0, it instructs the MAC to exit the LPI state and resume normal transmission. The EMAC_LPI_CTLSTAT.LPIEN bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.	
		0	LPI State is Disabled
		1	LPI State is Enabled
9 (R/NW)	RLPIST	Receive LPI State. When the EMAC_LPI_CTLSTAT.RLPIST bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface.	
		0	Receive LPI State Not Detected
		1	Receive LPI State Detected
8 (R/NW)	TLPIST	Transmit LPI State. When the EMAC_LPI_CTLSTAT.TLPIST bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface.	
		0	Transmit LPI State Not Detected
		1	Transmit LPI State Detected
3 (R/NW)	RLPIEX	Receive LPI Exit. When the EMAC_LPI_CTLSTAT.RLPIEX bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. The EMAC_LPI_CTLSTAT.RLPIEX bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.RLPIEX bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Note: The EMAC_LPI_CTLSTAT.RLPIEX bit might not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of SCLK0.	
		0	Receive LPI Exit Not Detected
		1	Receive LPI Exit Detected

Table 28-176: EMAC_LPI_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	RLPIEN	Receive LPI Entry. When the EMAC_LPI_CTLSTAT.RLPIEN bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. The EMAC_LPI_CTLSTAT.RLPIEN bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.RLPIEN bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Note: The EMAC_LPI_CTLSTAT.RLPIEN bit might not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of SCLK0.	
		0	Receive LPI Entry Not Detected
		1	Receive LPI Entry Detected
1 (R/NW)	TLPIEX	Transmit LPI Exit. When the EMAC_LPI_CTLSTAT.TLPIEX bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW timer has expired. The EMAC_LPI_CTLSTAT.TLPIEX bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.TLPIEX bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	Transmit LPI Exit Not Detected
		1	Transmit LPI Exit Detected
0 (R/NW)	TLPIEN	Transmit LPI Entry. When the EMAC_LPI_CTLSTAT.TLPIEN bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. The EMAC_LPI_CTLSTAT.TLPIEN bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.TLPIEN bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	Transmit LPI Entry Not Detected
		1	Transmit LPI Entry Detected

Transmit LPI Entry Timer Register

The EMAC_LPI_ENTR_TMR register controls the Tx LPI entry timer. This counter is enabled only when EMAC_LPI_CTLSTAT.LPITE = 1.

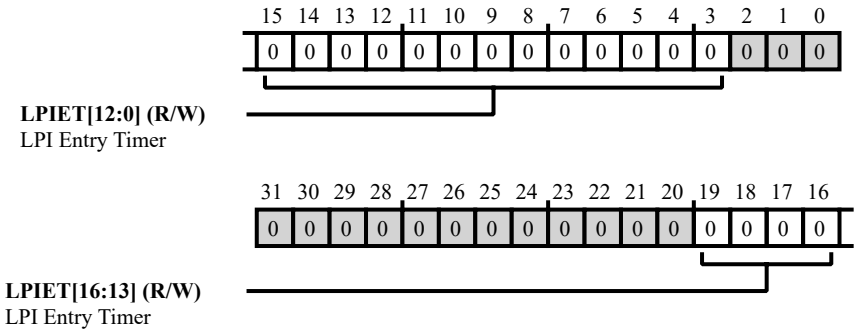


Figure 28-122: EMAC_LPI_ENTR_TMR Register Diagram

Table 28-177: EMAC_LPI_ENTR_TMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:3 (R/W)	LPIET	LPI Entry Timer. The EMAC_LPI_ENTR_TMR.LPIET field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. The EMAC_LPI_ENTR_TMR.LPIET field is valid and used only when LPITE and LPITXA are set (=1). Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 microseconds.

LPI Timers Control Register

The EMAC_LPI_TMRSCTL register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

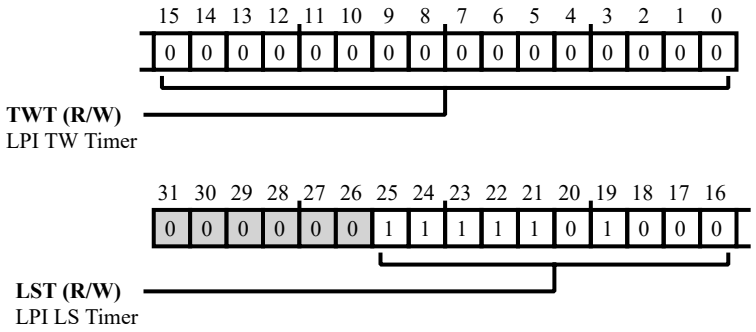


Figure 28-123: EMAC_LPI_TMRSCTL Register Diagram

Table 28-178: EMAC_LPI_TMRSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/W)	LST	LPI LS Timer. The EMAC_LPI_TMRSCTL.LST field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.
15:0 (R/W)	TWT	LPI TW Timer. The EMAC_LPI_TMRSCTL.TWT field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

Layer 3 Address 0 Register 7

For IPv4 packets, the EMAC_L3_ADDR0_REG7 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains bits[31:0] of the 128-bit IP Source Address or Destination Address field.

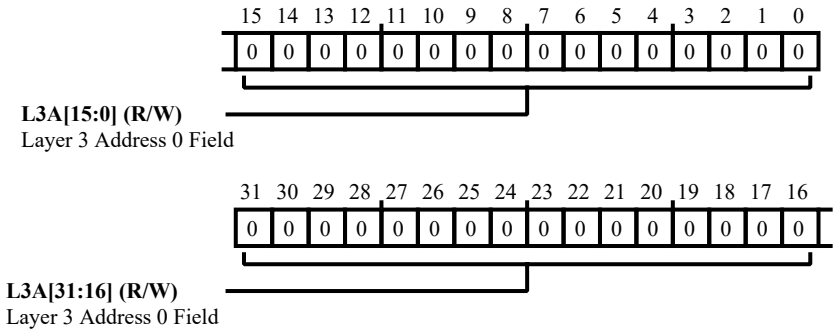


Figure 28-124: EMAC_L3_ADDR0_REG7 Register Diagram

Table 28-179: EMAC_L3_ADDR0_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A	Layer 3 Address 0 Field. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3SAM = 1, the EMAC_L3_ADDR0_REG7.L3A field contains the value to be matched with bits[31:0] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3DAM = 1, the EMAC_L3_ADDR0_REG7.L3A field contains the value to be matched with bits[31:0] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM = 1, the EMAC_L3_ADDR0_REG7.L3A field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer 3 Address 0 Register [i]

For IPv4 packets, the `EMAC_L3_ADDR0_REG[i]` register contains the 32-bit IP Source Address field. For IPv6 packets, it contains bits[31:0] of the 128-bit IP Source Address or Destination Address field.

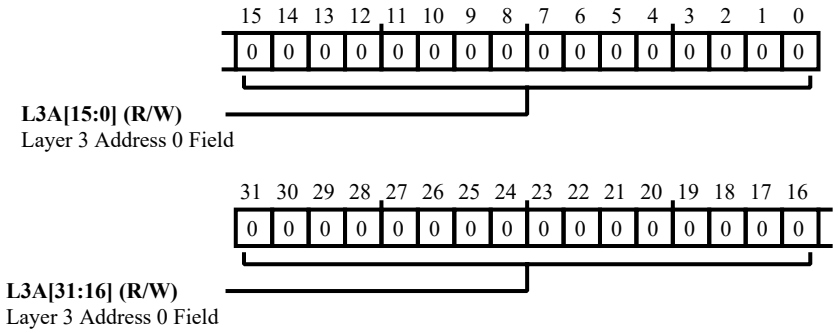


Figure 28-125: EMAC_L3_ADDR0_REG[i] Register Diagram

Table 28-180: EMAC_L3_ADDR0_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A	Layer 3 Address 0 Field. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3SAM = 1, the `EMAC_L3_ADDR0_REG[i].L3A` field contains the value to be matched with bits[31:0] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3DAM = 1, the `EMAC_L3_ADDR0_REG[i].L3A` field contains the value to be matched with bits[31:0] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM = 1, the `EMAC_L3_ADDR0_REG[i].L3A` field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer 3 Address 1 Register 7

For IPv4 packets, the EMAC_L3_ADDR1_REG7 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains bits[63:32] of the 128-bit IP Source Address or Destination Address field.

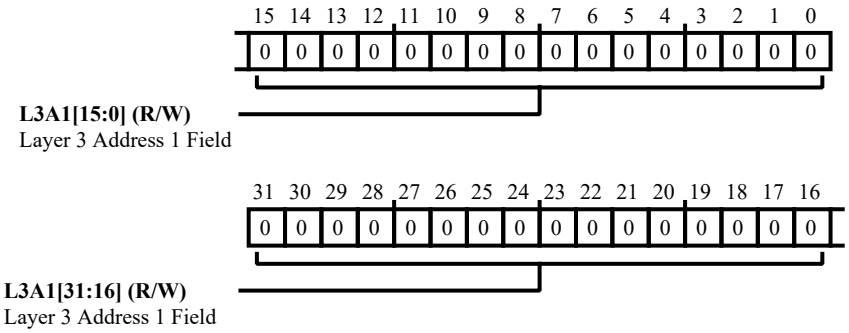


Figure 28-126: EMAC_L3_ADDR1_REG7 Register Diagram

Table 28-181: EMAC_L3_ADDR1_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A1	Layer 3 Address 1 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR1_REG7.L3A1 field contains the value to be matched with bits[63:32] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR1_REG7.L3A1 field contains the value to be matched with bits[63:32] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR1_REG7.L3A1 field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer 3 Address 1 Register [i]

For IPv4 packets, the EMAC_L3_ADDR1_REG[i] register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains bits[63:32] of the 128-bit IP Source Address or Destination Address field.

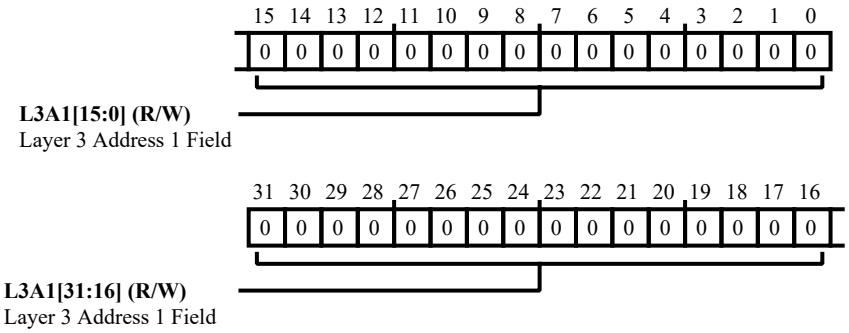


Figure 28-127: EMAC_L3_ADDR1_REG[i] Register Diagram

Table 28-182: EMAC_L3_ADDR1_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A1	Layer 3 Address 1 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR1_REG[i].L3A1 field contains the value to be matched with bits[63:32] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR1_REG[i].L3A1 field contains the value to be matched with bits[63:32] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR1_REG[i].L3A1 field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer 3 Address 2 Register 7

The EMAC_L3_ADDR2_REG7 register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR2_REG7 register contains bits[95:64] of 128-bit IP Source Address or Destination Address field.

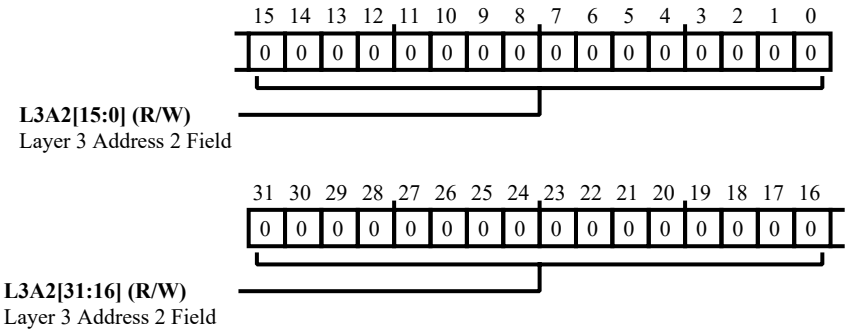


Figure 28-128: EMAC_L3_ADDR2_REG7 Register Diagram

Table 28-183: EMAC_L3_ADDR2_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A2	Layer 3 Address 2 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR2_REG7.L3A2 field contains the value to be matched with bits[95:64] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR2_REG7.L3A2 field contains the value to be matched with bits[95:64] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0, the EMAC_L3_ADDR2_REG7.L3A2 field is not used.

Layer 3 Address 2 Register [i]

The EMAC_L3_ADDR2_REG[i] register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR2_REG[i] register contains bits[95:64] of 128-bit IP Source Address or Destination Address field.

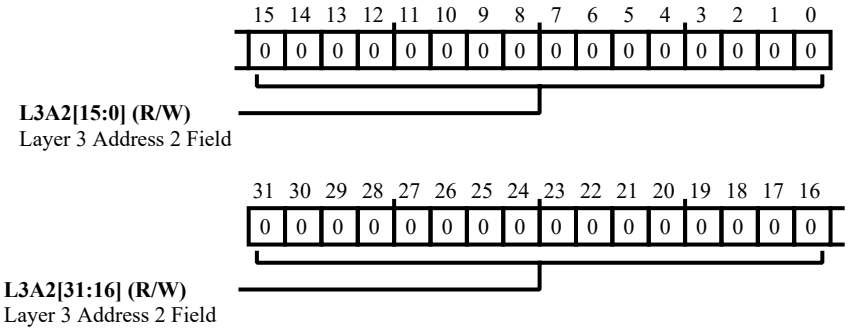


Figure 28-129: EMAC_L3_ADDR2_REG[i] Register Diagram

Table 28-184: EMAC_L3_ADDR2_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A2	Layer 3 Address 2 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR2_REG[i].L3A2 field contains the value to be matched with bits[95:64] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR2_REG[i].L3A2 field contains the value to be matched with bits[95:64] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0, the EMAC_L3_ADDR2_REG[i].L3A2 field is not used.

Layer 3 Address 3 Register 7

The EMAC_L3_ADDR3_REG7 register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR3_REG7 register contains bits[127:96] of 128-bit IP Source Address or Destination Address field.

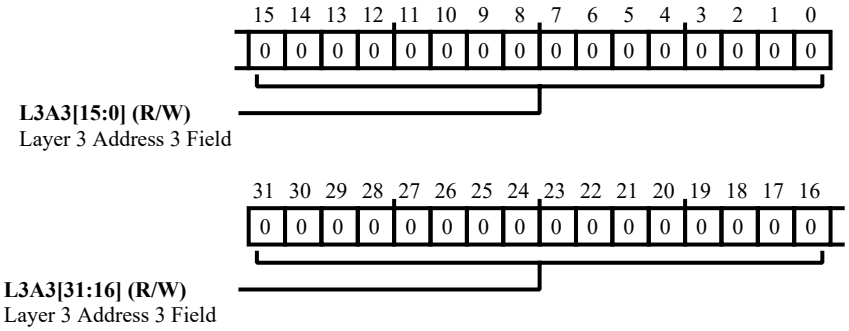


Figure 28-130: EMAC_L3_ADDR3_REG7 Register Diagram

Table 28-185: EMAC_L3_ADDR3_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A3	Layer 3 Address 3 Field. When MAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR3_REG7.L3A3 field contains the value to be matched with bits[127:96] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, this field contains the value to be matched with bits[127:96] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, this field is not used.

Layer 3 Address 3 Register [i]

The EMAC_L3_ADDR3_REG[i] register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR3_REG[i] register contains bits[127:96] of 128-bit IP Source Address or Destination Address field.

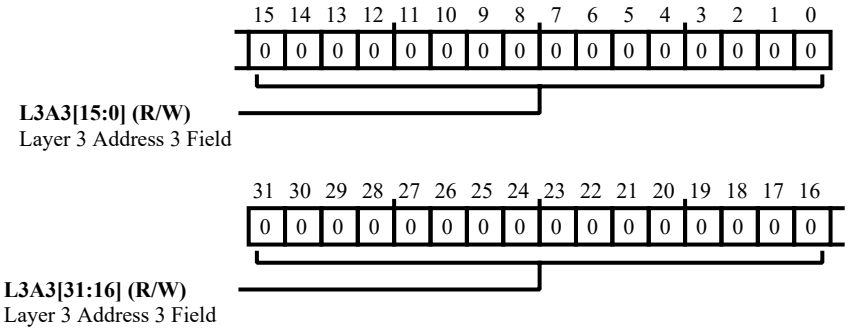


Figure 28-131: EMAC_L3_ADDR3_REG[i] Register Diagram

Table 28-186: EMAC_L3_ADDR3_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A3	Layer 3 Address 3 Field. When MAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR3_REG[i].L3A3 field contains the value to be matched with bits[127:96] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, this field contains the value to be matched with bits[127:96] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, this field is not used.

Layer 4 Address Register [i]

The EMAC_L4_ADDR7 register is double synchronized to the Rx clock domain. Thus, for proper synchronization updates, the program should perform consecutive writes to the same layer 3/4 address registers only after at least four clock cycles delay of the (G) MII clock domain.

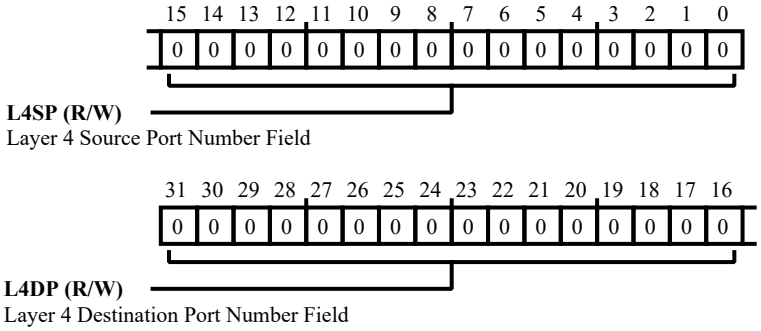


Figure 28-132: EMAC_L4_ADDR7 Register Diagram

Table 28-187: EMAC_L4_ADDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	L4DP	Layer 4 Destination Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.
15:0 (R/W)	L4SP	Layer 4 Source Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4SPM =1, the EMAC_L4_ADDR7.L4SP field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 1 and EMAC_L3L4_CTL[n].L4SPM =1, the EMAC_L4_ADDR7.L4SP field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer 4 Address Register [i]

The `EMAC_L4_ADDR[i]` register is double synchronized to the Rx clock domain. Thus, for proper synchronization updates, the program should perform consecutive writes to the same layer 3/4 address registers only after at least four clock cycles delay of the (G) MII clock domain.

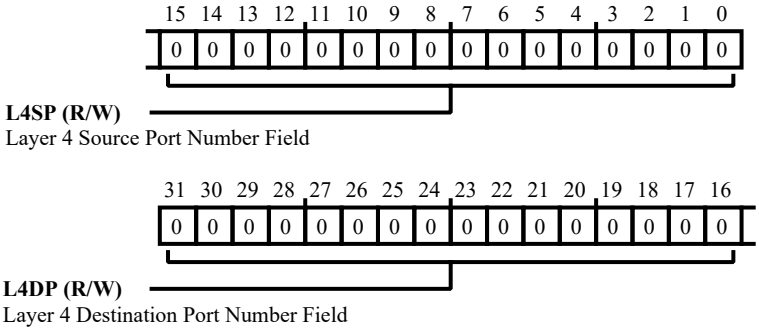


Figure 28-133: EMAC_L4_ADDR[i] Register Diagram

Table 28-188: EMAC_L4_ADDR[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	L4DP	Layer 4 Destination Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.
15:0 (R/W)	L4SP	Layer 4 Source Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4SPM =1, the `EMAC_L4_ADDR[i].L4SP` field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 1 and EMAC_L3L4_CTL[n].L4SPM =1, the `EMAC_L4_ADDR[i].L4SP` field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

PTP Log Message Interval Register

The EMAC_LOG_MSG_INTVL register contains the periodic intervals for automatic PTP packet generation.

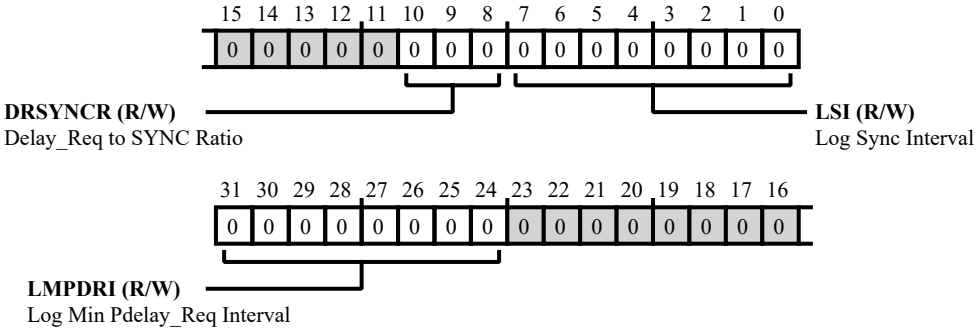


Figure 28-134: EMAC_LOG_MSG_INTVL Register Diagram

Table 28-189: EMAC_LOG_MSG_INTVL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	LMPDRI	Log Min Pdelay_Req Interval. The EMAC_LOG_MSG_INTVL.LMPDRI field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.	
10:8 (R/W)	DRSYNCR	Delay_Req to SYNC Ratio. In completer mode, the EMAC_LOG_MSG_INTVL.DRSYNCR field is used for controlling the frequency of Delay_Req messages transmitted. The requester sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the completer. The EMAC module receiver processes this value from the received DelayResp messages and updates this field accordingly. In completer mode, the host must not write/update this register unless it has to override the received value. In requester mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message. Access restriction applies. Updated based on the event. Write 1 to set. Write 0 to clear.	
		0	DelayReq Generated for Every Received SYNC
		1	DelayReq Generated Every Alternate Reception of SYNC
		2	For Every 4 SYNC Messages
		3	For Every 8 SYNC Messages
		4	For Every 16 SYNC Messages
		5	For Every 32 SYNC Messages

Table 28-189: EMAC_LOG_MSG_INTVL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Reserved
7:0 (R/W)	LSI	Log Sync Interval. The `EMAC_LOG_MSG_INTVL.LSI` field indicates the periodicity of the automatically generated SYNC message when the PTP node is the requester. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.	

MDIO Address Register

The EMAC_MDIO_ADDR register controls the management cycles to external PHY through a management interface.

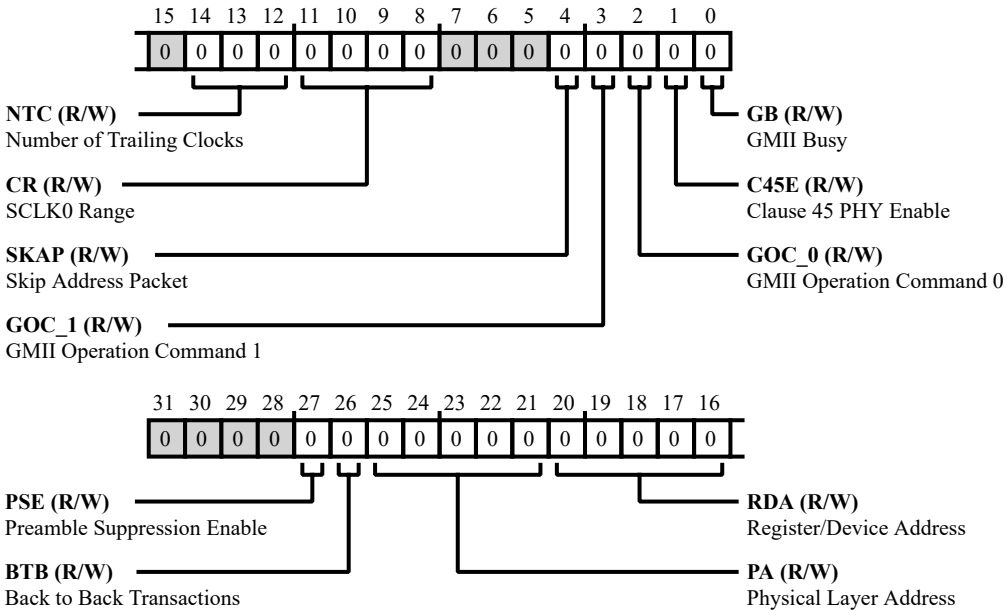


Figure 28-135: EMAC_MDIO_ADDR Register Diagram

Table 28-190: EMAC_MDIO_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	PSE	Preamble Suppression Enable. When the EMAC_MDIO_ADDR.PSE bit is set, the SMA suppresses the 32-bit preamble and transmits MDIO frames with only 1 preamble bit. When the EMAC_MDIO_ADDR.PSE bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.	
		0	Preamble Suppression Disabled
		1	Preamble Suppression Enabled

Table 28-190: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	BTB	Back to Back Transactions. When the EMAC_MDIO_ADDR.BTB bit is set and the NTC has value greater than 0, the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame. When EMAC_MDIO_ADDR.BTB = 0, then the read/write command completion (GB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. The EMAC_MDIO_ADDR.BTB bit must not be set when NTC=0.	
		0	Back to Back Transactions Disabled
		1	Back to Back Transactions Enabled
25:21 (R/W)	PA	Physical Layer Address. The EMAC_MDIO_ADDR.PA field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing. The EMAC_MDIO_ADDR.PA field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.	
20:16 (R/W)	RDA	Register/Device Address. The EMAC_MDIO_ADDR.RDA field selects the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.	
14:12 (R/W)	NTC	Number of Trailing Clocks. The EMAC_MDIO_ADDR.NTC field controls the number of trailing clock cycles generated on ETHx_MDC (MDC) after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.	

Table 28-190: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/W)	CR	SCLK0 Range. The EMAC_MDIO_ADDR.CR field indicates the frequency of the MDC clock according to the SCLK0 frequency used: 0000: SCLK0 = 60-100 MHz; MDC clock = SCLK0/42 0001: SCLK0 = 100-150 MHz; MDC clock = SCLK0/62 0010: SCLK0 = 20-35 MHz; MDC clock = SCLK0/16 0011: SCLK0 = 35-60 MHz; MDC clock = SCLK0/26 0100: SCLK0 = 150-250 MHz; MDC clock = SCLK0/102 0101: Reserved 0110: Reserved 0111: Reserved The suggested range of SCLK0 clock frequency applicable for each value (when bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When bit 11 is set, a higher frequency of the MDC clock can be achieved than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when SCLK0 is of 100 MHz frequency and program these bits as 1010; the resultant MDC clock is 12.5 MHz which is beyond the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks: 1000: SCLK0 /4 1001: SCLK0 /6 1010: SCLK0 /8 1011: SCLK0 /10 1100: SCLK0 /12 1101: SCLK0 /14 1110: SCLK0 /16 1111: SCLK0 /18	
4 (R/W)	SKAP	Skip Address Packet. When the EMAC_MDIO_ADDR.SKAP bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. The EMAC_MDIO_ADDR.SKAP bit is valid only when EMAC_MDIO_ADDR.C45E = 1.	
		0	Skip Address Packet is Disabled
		1	Skip Address Packet is Enabled

Table 28-190: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	GOC_1	GMII Operation Command 1. The EMAC_MDIO_ADDR.GOC_1 bit is higher bit of the operation command to the PHY, GOC_1 and GOC_O is encoded as follows: 00: Reserved 01: Write 10: Post read increment address for clause 45 PHY 11: Read When clause 22 PHY is enabled, only write and read commands are valid.	
		0	GMII Operation Command 1 is Disabled
		1	GMII Operation Command 1 is Enabled
2 (R/W)	GOC_0	GMII Operation Command 0. EMAC_MDIO_ADDR.GOC_0 is the lower bit of the operation command to the PHY. When in SMA mode (MDIO requester) the EMAC_MDIO_ADDR.GOC_0 bit along with GOC_1 determines the operation to be performed to the PHY.	
		0	GMII Operation Command 0 is Disabled
		1	GMII Operation Command 0 is Enabled
1 (R/W)	C45E	Clause 45 PHY Enable. When the EMAC_MDIO_ADDR.C45E bit is set, Clause 45 capable PHY is connected to MDIO. When EMAC_MDIO_ADDR.C45E = 0, Clause 22 capable PHY is connected to MDIO.	
		0	Clause 45 PHY is Disabled
		1	Clause 45 PHY is Enabled

Table 28-190: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	GB	GMII Busy. The application sets the EMAC_MDIO_ADDR.GB bit to instruct the SMA to initiate a read or write access to the MDIO completer. The MAC clears the EMAC_MDIO_ADDR.GB bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in EMAC_MDIO_ADDR and EMAC_MDIO_DATA registers as long as this bit is set. For write transfers, the application must first write 16-bit data in the GDl field (and also RA field when EMAC_MDIO_ADDR.C45E = 1) in EMAC_MDIO_DATA register before setting the EMAC_MDIO_ADDR.GB bit. When EMAC_MDIO_ADDR.C45E = 1, it should also write to EMAC_MDIO_DATA.RA before initiating a read transfer. When a read transfer is completed (GB=0), the data read from the PHY register is valid in EMAC_MDIO_DATA.GD. Note: Even if the addressed PHY is not present, there is no change in the functionality of the EMAC_MDIO_ADDR.GB bit. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	GMII Busy is Disabled
		1	GMII Busy is Enabled

MDIO Data Register

The EMAC_MDIO_DATA register stores the write data to be written to the PHY register located at the address specified in EMAC_MDIO_ADDR. The EMAC_MDIO_DATA register also stores the read data from the PHY register located at the address specified by MDIO Address register.

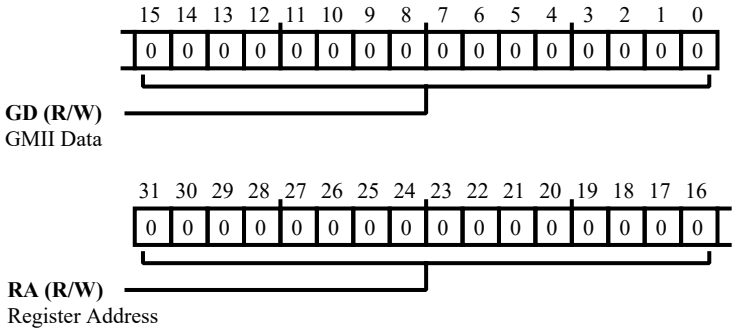


Figure 28-136: EMAC_MDIO_DATA Register Diagram

Table 28-191: EMAC_MDIO_DATA Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RA	Register Address. The EMAC_MDIO_DATA.RA field is valid only when EMAC_MDIO_ADDR.C45E = 1. It contains the register address in the PHY for the intended MDIO frame.
15:0 (R/W)	GD	GMII Data. The EMAC_MDIO_DATA.GD field contains the 16-bit data value read from the PHY after a management read operation or the 16-bit data value to be written to the PHY before a management write operation.

PHY Interface Control and Status Register

The EMAC_PHYIF_CTLSTAT register indicates the status signals received by the RGMII interface (selected at reset) from the PHY. The EMAC_PHYIF_CTLSTAT register is optional.

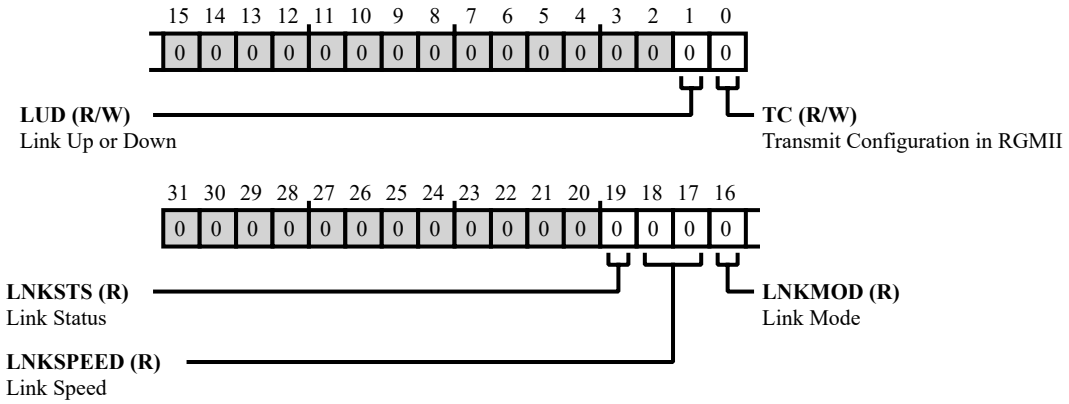


Figure 28-137: EMAC_PHYIF_CTLSTAT Register Diagram

Table 28-192: EMAC_PHYIF_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	LNKSTS	Link Status. The EMAC_PHYIF_CTLSTAT.LNKSTS bit indicates whether the link is up (1'b1) or down (1'b0).	
		0	Link Down
		1	Link up
18:17 (R/NW)	LNKSPEED	Link Speed. The EMAC_PHYIF_CTLSTAT.LNKSPEED bit indicates the current speed of the link.	
		0	2.5 MHz
		1	25 MHz
		2	125 MHz
		3	Reserved
16 (R/NW)	LNKMOD	Link Mode. The EMAC_PHYIF_CTLSTAT.LNKMOD bit indicates the current mode of operation of the link.	
		0	Half-duplex Mode
		1	Full-duplex Mode

Table 28-192: EMAC_PHYIF_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	LUD	Link Up or Down. The EMAC_PHYIF_CTLSTAT.LUD bit indicates whether the link is up or down during transmission of configuration in the RGMII, SGMII, or SMII interface.	
		0	Link Down
		1	Link Up
0 (R/W)	TC	Transmit Configuration in RGMII. When set, the EMAC_PHYIF_CTLSTAT.TC bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII port. When EMAC_PHYIF_CTLSTAT.TC = 0, no such information is driven to the PHY.	
		0	Disable Transmit Configuration in RGMII
		1	Enable Transmit Configuration in RGMII

PPS n Interval Register

The EMAC_PPS[n]_INTVL register contains the number of units of sub-second increment value between the rising edges of ETH0_PTPPPSn signal output.

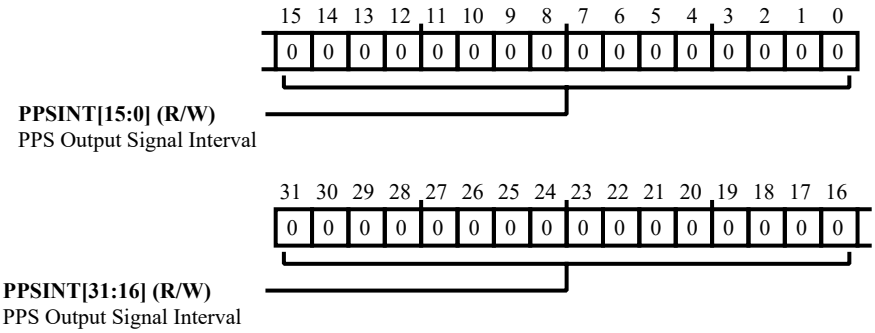


Figure 28-138: EMAC_PPS[n]_INTVL Register Diagram

Table 28-193: EMAC_PPS[n]_INTVL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PPSINT	PPS Output Signal Interval. The EMAC_PPS[n]_INTVL.PPSINT field stores the interval between the rising edges of the ETH0_PTPPPSn signal output. The interval is stored in terms of number of units of sub-second increment value. Program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of the ETH0_PTPPPSn signal output is 100 ns (that is, 5 units of sub-second increment value), program value 4 (5-1) in this register.

PPS n Target Time Nanoseconds Register

PPS0 Target Time Nanoseconds register.

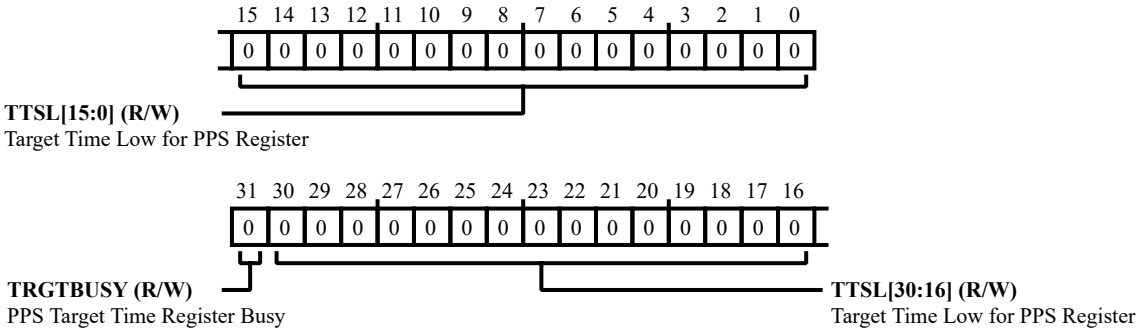


Figure 28-139: EMAC_PPS[n]_TGTM_NSEC Register Diagram

Table 28-194: EMAC_PPS[n]_TGTM_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TRGTBUSY	PPS Target Time Register Busy. The MAC sets the `EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY` bit when the EMAC_PPS_CTL.PPSCMD0 field is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time registers to the PTP clock domain. The MAC clears the `EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY` bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when the `EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY` bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.	
		0	PPS Target Time Register Busy Status is Not Detected
		1	PPS Target Time Register Busy is Detected
30:0 (R/W)	TTSL	Target Time Low for PPS Register. The `EMAC_PPS[n]_TGTM_NSEC` field stores the time in (signed) nanoseconds. When the value of the time stamp matches the value in both target time stamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in EMAC_PPS_CTL. When EMAC_TM_CTL.TSCTRLSSR = 0, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of sub-second increment value. When EMAC_TM_CTL.TSCTRLSSR = 1, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output might have an error margin up to one unit of sub-second increment value. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	

PPS n Target Time Seconds Register

The EMAC_PPS[n]_TGTM_SEC register, along with the EMAC_PPS[n]_TGTM_NSEC register, is used to schedule an interrupt event (bit 1 of EMAC_TM_STAT) when the system time exceeds the value programmed in these registers.

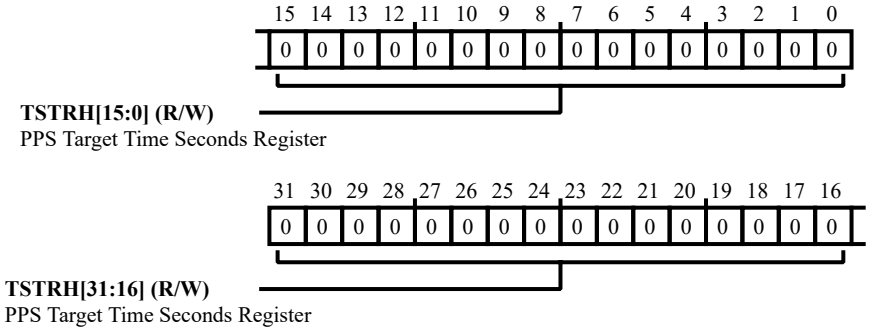


Figure 28-140: EMAC_PPS[n]_TGTM_SEC Register Diagram

Table 28-195: EMAC_PPS[n]_TGTM_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSTRH	PPS Target Time Seconds Register. The EMAC_PPS[n]_TGTM_SEC.TSTRH field stores the time in seconds. When the time stamp value matches or exceeds both target time stamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the target time mode selected for the corresponding PPS output in the EMAC_PPS_CTL register. If EMAC_TM_CTL.PTGE = 1 with the presentation time control set in recovery mode, then these bits indicate the TPT being programmed by the application and in generation mode it indicates the CPT generated at the sampled trigger.

PPS n Width Register

The EMAC_PPS[n]_WID register contains the number of units of sub-second increment value between the rising and corresponding falling edges of ETH0_PTPPPSn signal output.

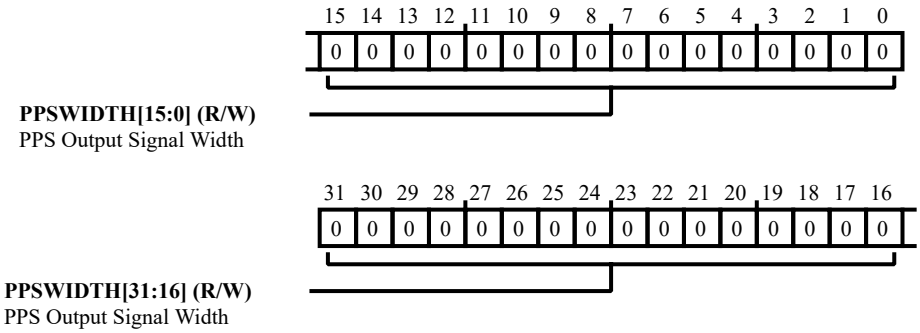


Figure 28-141: EMAC_PPS[n]_WID Register Diagram

Table 28-196: EMAC_PPS[n]_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PPSWIDTH	PPS Output Signal Width. The EMAC_PPS[n]_WID.PPSWIDTH field store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of sub-second increment value. Program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of sub-second increment value), program value 3 (4-1) in this register. Note: The value programmed in this register must be lesser than the value programmed in the EMAC_PPS[n]_INTVL register.

PPS Control Register

Bits[30:24] of the EMAC_PPS_CTL register are valid only when four flexible PPS outputs are selected. Bits[22:16] are valid only when three or more flexible PPS outputs are selected. Bits[14:8] are valid only when two or more flexible PPS outputs are selected. Bits[6:4] are valid only when flexible PPS feature is selected.

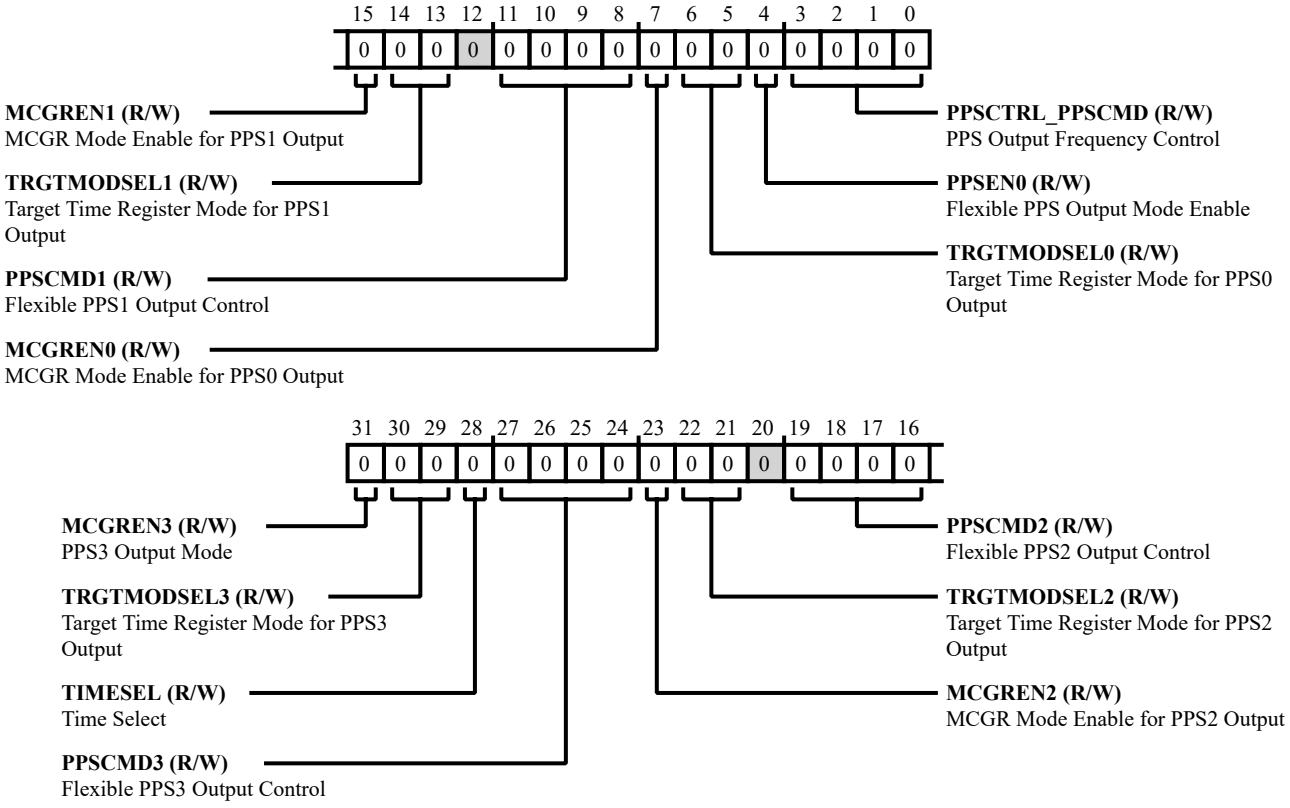


Figure 28-142: EMAC_PPS_CTL Register Diagram

Table 28-197: EMAC_PPS_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	MCGREN3	PPS3 Output Mode. The EMAC_PPS_CTL.MCGREN3 bit indicates whether the 3rd PPS instance operates in PPS or MCGR mode.	
		0	PPS Mode
		1	MCGR Mode

Table 28-197: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/W)	TRGTMODSEL3	Target Time Register Mode for PPS3 Output. The EMAC_PPS_CTL.TRGTMODSEL3 field indicates the target time registers (EMAC_PPS[3]_TGTM_SEC and EMAC_PPS[3]_TGTM_NSEC) mode for PPS3 output signal.	
		0	Generate the Interrupt Only. Target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may me observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The interrupt status bit is indicated by EMAC_TM_STAT.TSTARGT3.
		2	Generate Interrupt and Start/Stop the PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
28 (R/W)	TIMESEL	Time Select. When the EMAC_PPS_CTL.TIMESEL bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input. When EMAC_PPS_CTL.TIMESEL = 0, presentation time is used to capture time at trigger input, maintaining backward compatibility.	
27:24 (R/W)	PPSCMD3	Flexible PPS3 Output Control. The EMAC_PPS_CTL.PPSCMD3 field controls the flexible PPS3 output (ETH0_PTPPS3) signal. The EMAC_PPS_CTL.PPSCMD3 field is similar to the PPSCMD0[2:0] field. When EMAC_PPS_CTL.MCGREN3= 1, the EMAC_PPS_CTL.PPSCMD3 field indicated by the four bits [27:24] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 3. The EMAC_PPS_CTL.PPSCMD3 field is similar to the PPSCMD0 Presentation Time Control bits. When EMAC_PPS_CTL.MCGREN3= 0, only 3 bits from [26:24] are used as PPSCMD3 and the fourth bit must be cleared (=0). Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
23 (R/W)	MCGREN2	MCGR Mode Enable for PPS2 Output. The EMAC_PPS_CTL.MCGREN2 field enables the 2nd PPS instance to operate in PPS or MCGR mode.	
		0	PPS Mode
		1	MCGR Mode

Table 28-197: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22:21 (R/W)	TRGTMODSEL2	Target Time Register Mode for PPS2 Output. The EMAC_PPS_CTL.TRGTMODSEL2 field indicates the target time registers (EMAC_PPS[2]_TGTM_SEC and EMAC_PPS[2]_TGTM_NSEC) mode for PPS2 output signal.	
		0	Generate the Interrupt Event. Target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may me observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The interrupt status bit is indicated by TSTARGT3.
		2	Generate Interrupt and Start/Stop the PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
19:16 (R/W)	PPSCMD2	Flexible PPS2 Output Control. The EMAC_PPS_CTL.PPSCMD2 field controls the flexible PPS2 output (ETH0_PTPPS2) signal. The EMAC_PPS_CTL.PPSCMD2 field is similar to the PPSCMD0 field. When EMAC_PPS_CTL.MCGREN2= 1, the EMAC_PPS_CTL.PPSCMD2 field indicated by the four bits, [19:16] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 2. The EMAC_PPS_CTL.PPSCMD2 field is similar to the PPSCMD0 Presentation Time Control bits. When EMAC_PPS_CTL.MCGREN2= 0, only 3 bits, [18:16] are used as PPSCMD2 and the fourth bit must be configured as 0. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
15 (R/W)	MCGREN1	MCGR Mode Enable for PPS1 Output. The EMAC_PPS_CTL.MCGREN1 field enables the 1st PPS instance to operate in PPS or MCGR mode.	
		0	PPS Mode
		1	MCGR Mode

Table 28-197: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14:13 (R/W)	TRGTMODSEL1	Target Time Register Mode for PPS1 Output. The EMAC_PPS_CTL.TRGTMODSEL1 field indicates the target time registers (EMAC_PPS[1]_TGTM_SEC and EMAC_PPS[1]_TGTM_NSEC) mode for PPS1 output signal.	
		0	Generate the Interrupt Event. Target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may me observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The interrupt status bit is indicated by TSTARGT3.
		2	Generate Interrupt and Start/Stop the PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
11:8 (R/W)	PPSCMD1	Flexible PPS1 Output Control. The EMAC_PPS_CTL.PPSCMD1 field controls the flexible PPS1 output (ETH0_PTPPS1) signal. The EMAC_PPS_CTL.PPSCMD1 field is similar to the PPSCMD0 field. When EMAC_PPS_CTL.MCGREN1= 1, PPSCMD1 indicated by the four bits [11:8] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 1. The EMAC_PPS_CTL.PPSCMD1 field is similar to the PPSCMD0 Presentation Time Control bits. When EMAC_PPS_CTL.MCGREN1= 0, only 3 bits, [10:8] are used as PPSCMD1 and the fourth bit must be cleared (=0). Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
7 (R/W)	MCGREN0	MCGR Mode Enable for PPS0 Output. The EMAC_PPS_CTL.MCGREN0 field enables the 0th PPS instance to operate in PPS or MCGR mode. When set, it operates in MCGR mode and when cleared, it operates in PPS mode.	
		0	PPS Mode
		1	MCGR Mode

Table 28-197: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	TRGTMODSEL0	Target Time Register Mode for PPS0 Output. The `EMAC_PPS_CTL.TRGTMODSEL0` field indicates the target time registers (EMAC_PPS[0]_TGTM_SEC and EMAC_PPS[0]_TGTM_NSEC) mode for PPS0 output signal.	
		0	Generate the Interrupt Event. The target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may be observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The status bit is indicated by TSTARGT.
		2	Generate the Interrupt Event and PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
4 (R/W)	PPSEN0	Flexible PPS Output Mode Enable. When the `EMAC_PPS_CTL.PPSEN0` bit is set (=1), bits[3:0] function as PPSCMD. When the `EMAC_PPS_CTL.PPSEN0` bit is cleared (=0), bits[3:0] function as PPSCTRL (fixed PPS mode).	
		0	Flexible PPS Output Mode is Disabled
		1	Flexible PPS Output Mode is Enabled

Table 28-197: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	PPSCTRL_PPSCMD	PPS Output Frequency Control. The EMAC_PPS_CTL.PPSCTRL_PPSCMD field controls the frequency of the ETH0_PTPPPS0 output (PTP clock) signal. The default value is 0000, and the PPS output is 1 pulse (of width PTP clock) every second. For other values of EMAC_PPS_CTL.PPSCTRL_PPSCMD, the PPS output becomes a generated clock of following frequencies: - 0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz. - 0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz. - 0011: The binary rollover is 8 Hz; digital rollover is 4 Hz. - 0100: The binary rollover is 16 Hz; digital rollover is 8 Hz. ... - 1111: The binary rollover is 32.768 KHz; digital rollover is 16.384 KHz. See the Fixed Pulse-Per-Second Output topic for details. When EMAC_PPS_CTL.PPSCTRL_PPSCMD = 0011, the PPS (4 Hz) is a sequence of: Three clocks of 50 percent duty cycle and 268 ms period. A fourth clock of 195 ms period (134 ms low and 61 ms high). This behavior is due to the non-linear toggling of bits in the digital rollover mode in the EMAC_TM_NSEC register. For Flexible PPS Output (ETH0_PTPPPS0) control, programming the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, the bits are cleared automatically. Software should ensure that these bits are programmed only when they are 'all-zero'. When EMAC_PPS_CTL.MCGREN0 = 1, the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits are treated as presentation time control bits: - 0000: MCGR operation is not carried out. If configured to this value in the middle of clock recovery or generation, all processing inputs are flushed. - 0001: Capture the presentation time at the rising edge of the EMAC0_MCG_PST trigger request into the EMAC_PPS[0]_TGTM_SEC register. - 0010: Capture the presentation time at the falling edge of the EMAC0_MCG_PST trigger request into the EMAC_PPS[0]_TGTM_SEC register. - 0011: Capture the presentation time at both edges of the EMAC0_MCG_PST trigger request into the EMAC_PPS[0]_TGTM_SEC register. - 0100-1000: Reserved - 1001: Toggle output on compare - 1010: Pulse output low on compare for one PTP clock cycle. - 1011: Pulse output high on compare for one PTP clock cycle. - 1100-1111: Reserved

PTP Time Stamp Offload Control Register

The EMAC_PTO_CTL register controls the PTP offload engine operation.

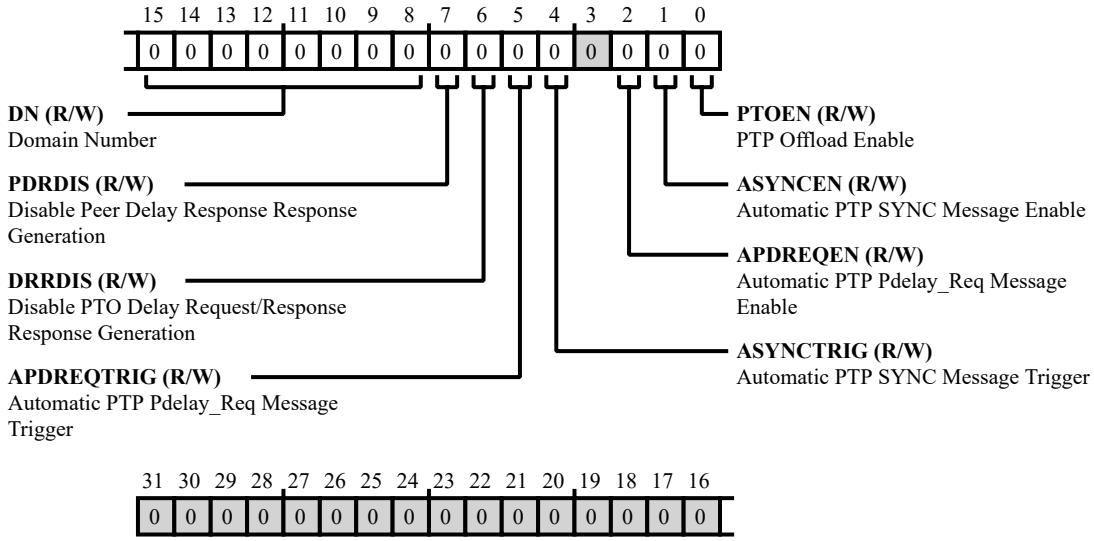


Figure 28-143: EMAC_PTO_CTL Register Diagram

Table 28-198: EMAC_PTO_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/W)	DN	Domain Number. The EMAC_PTO_CTL.DN field indicates the domain number in which the PTP node is operating.	
7 (R/W)	PDRDIS	Disable Peer Delay Response Response Generation. When the EMAC_PTO_CTL.PDRDIS bit is set, the peer delay response (Pdelay_Resp) response is not be generated for received peer delay request (Pdelay_Req) request packet, as required by the programmed mode. Note: Setting the EMAC_PTO_CTL.PDRDIS bit to 1 affects the normal PTP Offload operation and the time synchronization. So, the EMAC_PTO_CTL.PDRDIS bit must be set only if there is problem with Pdelay_Resp generation in hardware and/or Pdelay_Resp generation is handled by software.	
		0	Peer Delay Response Response Generation is Enabled
		1	Peer Delay Response Response Generation is Disabled

Table 28-198: EMAC_PTO_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DRRDIS	Disable PTO Delay Request/Response Response Generation. When the EMAC_PTO_CTL.DRRDIS bit is set, the delay request and delay response is not generated for received SYNC and delay request packet respectively, as required by the programmed mode.	
		0	PTO Delay Request/Response Response Generation is Enabled
		1	PTO Delay Request/Response Response Generation is Disabled
5 (R/W)	APDREQTRIG	Automatic PTP Pdelay_Req Message Trigger. When the EMAC_PTO_CTL.APDREQTRIG bit is set, one PTP Pdelay_Req message is transmitted. The EMAC_PTO_CTL.APDREQTRIG bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the EMAC_PTO_CTL.APDREQEN bit for this operation. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Automatic PTP Pdelay_Req Message Trigger is Disabled
		1	Automatic PTP Pdelay_Req Message Trigger is Enabled
4 (R/W)	ASYNCTRIG	Automatic PTP SYNC Message Trigger. When the EMAC_PTO_CTL.ASYNCTRIG bit is set, one PTP SYNC message is transmitted. The EMAC_PTO_CTL.ASYNCTRIG bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the EMAC_PTO_CTL.ASYNCEN bit for this operation. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Automatic PTP SYNC Message Trigger is Disabled
		1	Automatic PTP SYNC Message Trigger is Enabled
2 (R/W)	APDREQEN	Automatic PTP Pdelay_Req Message Enable. When the EMAC_PTO_CTL.APDREQEN bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in peer-to-peer transparent mode.	
		0	Automatic PTP Pdelay_Req Message is Disabled
		1	Automatic PTP Pdelay_Req Message is Enabled
1 (R/W)	ASYNCEN	Automatic PTP SYNC Message Enable. When the EMAC_PTO_CTL.ASYNCEN bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in clock requester mode.	
		0	Automatic PTP SYNC Message is Disabled
		1	Automatic PTP SYNC Message is Enabled

Table 28-198: EMAC_PTO_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PTOEN	PTP Offload Enable. When the `EMAC_PTO_CTL.PTOEN` bit is set, the PTP offload feature is enabled.	
		0	PTP Offload Feature is Disabled
		1	PTP Offload Feature is Enabled

MAC Packet Filter Register

The EMAC_MACPKT_FILT register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

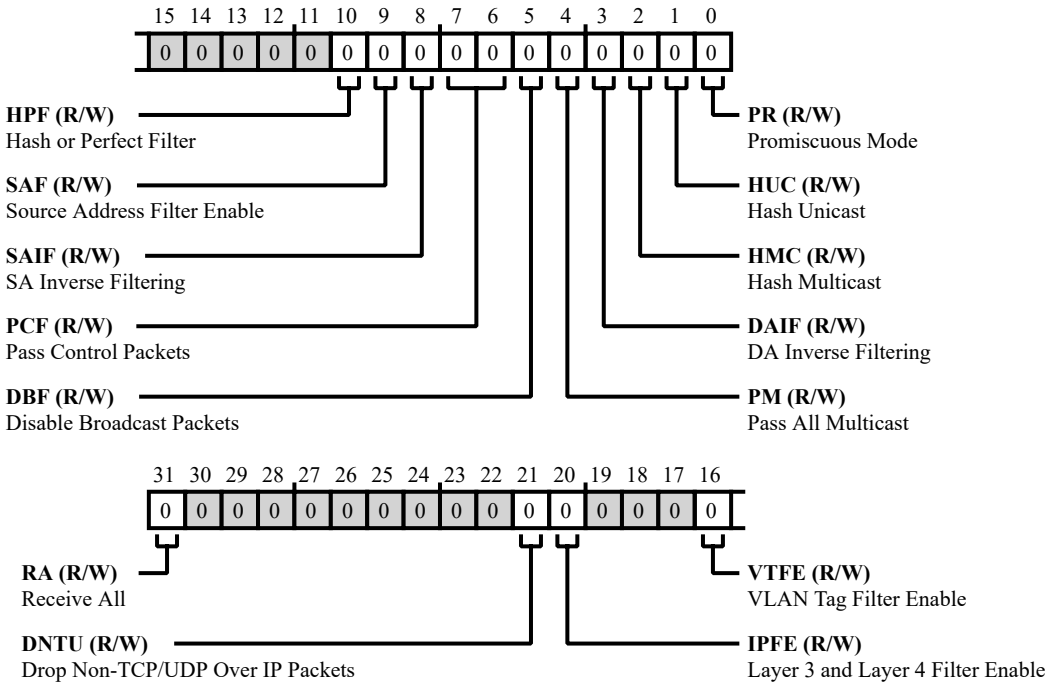


Figure 28-144: EMAC_MACPKT_FILT Register Diagram

Table 28-199: EMAC_MACPKT_FILT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	RA	Receive All. When the EMAC_MACPKT_FILT.RA bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx status word. When EMAC_MACPKT_FILT.RA = 0, the receiver module passes only those packets to the application that pass the SA or DA address filter.	
		0	Receive All is Disabled
		1	Receive All is Enabled

Table 28-199: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	DNTU	Drop Non-TCP/UDP Over IP Packets. When the EMAC_MACPKT_FILT.DNTU bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the layer 4 filter. When EMAC_MACPKT_FILT.DNTU = 0, the MAC forwards all non-TCP or UDP over IP packets.	
		0	Forward Non-TCP/UDP Over IP Packets
		1	Drop Non-TCP/UDP Over IP Packets
20 (R/W)	IPFE	Layer 3 and Layer 4 Filter Enable. When the EMAC_MACPKT_FILT.IPFE bit is set, the MAC drops packets that do not match the enabled layer 3 and layer 4 filters. If layer 3 or layer 4 filters are not enabled for matching, the EMAC_MACPKT_FILT.IPFE bit does not have any effect. When EMAC_MACPKT_FILT.IPFE = 0, the MAC forwards all packets irrespective of the match status of the layer 3 and layer 4 fields.	
		0	Layer 3 and Layer 4 Filters are Disabled
		1	Layer 3 and Layer 4 Filters are Enabled
16 (R/W)	VTFE	VLAN Tag Filter Enable. When the EMAC_MACPKT_FILT.VTFE bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN tag. When EMAC_MACPKT_FILT.VTFE = 0, the MAC forwards all packets irrespective of the match status of the VLAN tag.	
		0	VLAN Tag Filter is Disabled
		1	VLAN Tag Filter is Enabled
10 (R/W)	HPF	Hash or Perfect Filter. When the EMAC_MACPKT_FILT.HPF bit is set, the address filter passes a packet if it matches either the perfect filtering or hash filtering as configured by the EMAC_MACPKT_FILT.HUC or EMAC_MACPKT_FILT.HMC bit. When EMAC_MACPKT_FILT.HPF = 0 and the EMAC_MACPKT_FILT.HUC or EMAC_MACPKT_FILT.HMC bit is set, the packet is passed only if it matches the hash filter.	
		0	Hash or Perfect Filter is Disabled
		1	Hash or Perfect Filter is Enabled

Table 28-199: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SAF	Source Address Filter Enable. When the `EMAC_MACPKT_FILT.SAF` bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet. When `EMAC_MACPKT_FILT.SAF` = 0, the MAC forwards the received packet to the application with updated SAF bit of the Rx status depending on the SA address comparison. Note: According to the IEEE specification, bit 47 of the SA is reserved. However, in the EMAC module, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.	
		0	SA Filtering is Disabled
		1	SA Filtering is Enabled
8 (R/W)	SAIF	SA Inverse Filtering. When the `EMAC_MACPKT_FILT.SAIF` bit is set, the address check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA address filter. When `EMAC_MACPKT_FILT.SAIF` = 0, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA address filter.	
		0	SA Inverse Filtering is Disabled
		1	SA Inverse Filtering is Enabled
7:6 (R/W)	PCF	Pass Control Packets. The `EMAC_MACPKT_FILT.PCF` field control the forwarding of all control packets (including unicast and multicast pause packets).	
		0	MAC filters all control packets from reaching the application
		1	MAC forwards all control packets except pause packets to the application even if they fail the address filter
		2	MAC forwards all control packets to the application even if they fail the address filter
		3	MAC forwards the control packets that pass the address filter

Table 28-199: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DBF	Disable Broadcast Packets. When the EMAC_MACPKT_FILT.DBF bit is set, the AFM module blocks all the incoming broadcast packets. In addition, it overrides all other filter settings. When EMAC_MACPKT_FILT.DBF = 0, the AFM module passes all received broadcast packets.	
		0	Enable Broadcast Packets
		1	Disable Broadcast Packets
4 (R/W)	PM	Pass All Multicast. When the EMAC_MACPKT_FILT.PM bit is set, it indicates that all the received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When EMAC_MACPKT_FILT.PM = 0, filtering of multicast packet depends on the EMAC_MACPKT_FILT.HMC bit.	
		0	Pass All Multicast is Disabled
		1	Pass All Multicast is Enabled
3 (R/W)	DAIF	DA Inverse Filtering. When the EMAC_MACPKT_FILT.DAIF bit is set, the address check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When EMAC_MACPKT_FILT.DAIF = 0, normal filtering of packets is performed.	
		0	DA Inverse Filtering is Disabled
		1	DA Inverse Filtering is Enabled
2 (R/W)	HMC	Hash Multicast. When the EMAC_MACPKT_FILT.HMC bit is set, the MAC performs the destination address filtering of received multicast packets according to the hash table. When EMAC_MACPKT_FILT.HMC = 0, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.	
		0	Hash Multicast is Disabled
		1	Hash Multicast is Enabled

Table 28-199: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	HUC	Hash Unicast. When the EMAC_MACPKT_FILT.HUC bit is set, the MAC performs the destination address filtering of unicast packets according to the hash table. When EMAC_MACPKT_FILT.HUC = 0, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.	
		0	Hash Unicast is Disabled
		1	Hash Unicast is Enabled
0 (R/W)	PR	Promiscuous Mode. When the EMAC_MACPKT_FILT.PR bit is set, the address filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA filter fails status bits of the Rx status word are always cleared when EMAC_MACPKT_FILT.PR is set.	
		0	Promiscuous Mode is Disabled
		1	Promiscuous Mode is Enabled

Presentation Time Update Register

The EMAC_PRSNTM_UPDT register holds the 32-bit value of MAC 1722 Presentation Time in ns, that should be added to the Current Presentation Time Counter value. Init happens when EMAC_TM_CTL.TSINIT = 1, and update happens when EMAC_TM_CTL.TSUPDT = 1.

The EMAC_PRSNTM_UPDT register exists when DWC_EQOS_FLEXI_PPS_OUT_EN is configured.

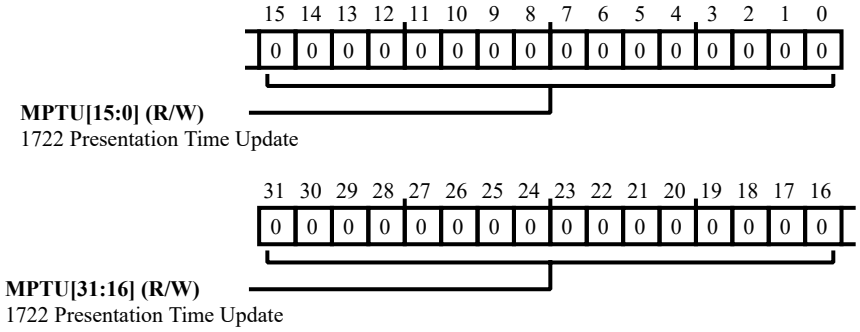


Figure 28-145: EMAC_PRSNTM_UPDT Register Diagram

Table 28-200: EMAC_PRSNTM_UPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MPTU	1722 Presentation Time Update. The EMAC_PRSNTM_UPDT.MPTU field holds the init value or the update value for the presentation time. When used for update, this field holds the 32-bit value in ns, that should be added to the Current Presentation Time Counter value. When EMAC_TM_NSECUPDT.ADDSUB = 1, this value is directly used for subtraction.

Presentation Time Nanosecond Register

The EMAC_PRSNTM_NSEC register contains the 32-bit binary rollover equivalent time of the PTP system time in ns.

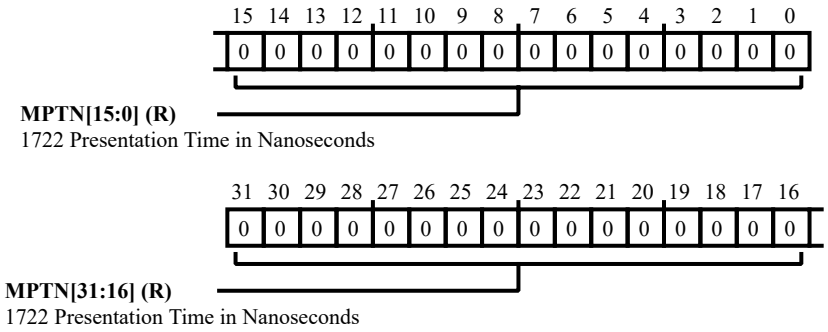


Figure 28-146: EMAC_PRSNTM_NSEC Register Diagram

Table 28-201: EMAC_PRSNTM_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPTN	1722 Presentation Time in Nanoseconds. The EMAC_PRSNTM_NSEC.MPTN field indicates the value of the 32-bit binary rollover equivalent time of the PTP system time in nanoseconds.

Queue 0 Transmit Flow Control Register

The EMAC_Q0_TXFLOW_CTL register controls the generation and reception of the Control (Pause Command) packets by the flow control module of the MAC. A write to a register with the Busy bit set (=1) triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

When the PFCE bit in the MAC_Rx_Flow_Ctrl register is enabled, this register controls the generation of Priority Flow Control (PFC) frames with priorities mapped according to PSRQ0 in the MAC_RxQ_Ctrl2 register.

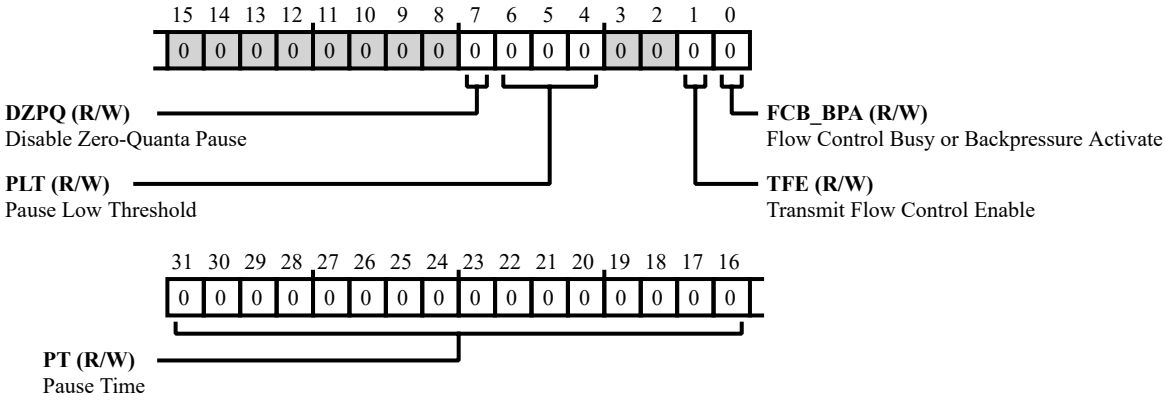


Figure 28-147: EMAC_Q0_TXFLOW_CTL Register Diagram

Table 28-202: EMAC_Q0_TXFLOW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	PT	Pause Time. The EMAC_Q0_TXFLOW_CTL.PT field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the (G)MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.

Table 28-202: EMAC_Q0_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	DZPQ	Disable Zero-Quanta Pause. When the EMAC_Q0_TXFLOW_CTL.DZPQ bit is set, it disables the automatic generation of the zero-quanta Pause packets on deassertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When EMAC_Q0_TXFLOW_CTL.DZPQ = 0, normal operation with automatic zero-quanta Pause packet generation is enabled.	
		0	Zero-Quanta Pause Packet Generation is Enabled
		1	Zero-Quanta Pause Packet Generation is Disabled
6:4 (R/W)	PLT	Pause Low Threshold. The EMAC_Q0_TXFLOW_CTL.PLT field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values. The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.	
		0	Pause Time Minus 4 Slot Times
		1	Pause Time Minus 28 Slot Times
		2	Pause Time Minus 36 Slot Times
		3	Pause Time Minus 144 Slot Times
		4	Pause Time Minus 256 Slot Times
		5	Pause Time Minus 512 Slot Times
		6	Reserved

Table 28-202: EMAC_Q0_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	TFE	Transmit Flow Control Enable. In the full-duplex mode, when the EMAC_Q0_TXFLOW_CTL.TFE bit is set, the MAC enables the flow control operation to Tx Pause packets. When EMAC_Q0_TXFLOW_CTL.TFE = 0, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets. In the half-duplex mode, when the EMAC_Q0_TXFLOW_CTL.TFE bit is set, the MAC enables the back pressure operation. When EMAC_Q0_TXFLOW_CTL.TFE = 0, the back pressure feature is disabled.	
		0	Transmit Flow Control is Disabled
		1	Transmit Flow Control is Enabled
0 (R/W)	FCB_BPA	Flow Control Busy or Backpressure Activate. The EMAC_Q0_TXFLOW_CTL.FCB_BPA bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. In full-duplex mode, the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit should be read as 1'b0 before writing to this register. To initiate a Pause packet, the application must set the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit to 1'b1. During Control packet transfer, the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit to 1'b0. Do not write to this register until the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit is cleared. When the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the back pressure. During back pressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the mti_flowctrl_i input signal for the back pressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flow Control Busy or Backpressure Activate is Disabled
		1	Flow Control Busy or Backpressure Activate is Enabled

Queue [i] Transmit Flow Control Register

The EMAC_Q[i]_TXFLOW_CTL register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A write to a register with the Busy bit set (= 1) triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register. (EMAC_RXFLOW_CTL.PFCE = 1.)

The register controls the generation of PFC control packets of priorities mapped per the PSRQn field in the EMAC_RXQ_CTL2.EMAC_RXQ_CTL3 registers.

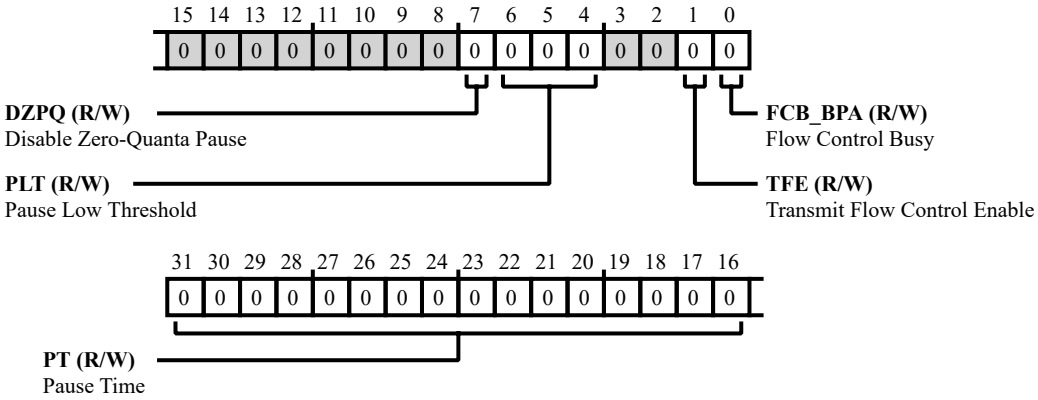


Figure 28-148: EMAC_Q[i]_TXFLOW_CTL Register Diagram

Table 28-203: EMAC_Q[i]_TXFLOW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/W)	PT	Pause Time. The EMAC_Q[i]_TXFLOW_CTL.PT field holds the value to be used in the Pause Time field in the Tx control packet. The bits are configured to be double-synchronized to the (G)MII clock domain. Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.	
7 (R/W)	DZPQ	Disable Zero-Quanta Pause. When the EMAC_Q[i]_TXFLOW_CTL.DZPQ bit is set, it disables the automatic generation of the zero-quanta Pause packets on deassertion of the flow-control signal from the FIFO layer. When EMAC_Q[i]_TXFLOW_CTL.DZPQ = 0, normal operation with automatic zero-quanta Pause packet generation is enabled.	
		0	Zero-Quanta Pause Packet Generation is Enabled
		1	Zero-Quanta Pause Packet Generation is Disabled

Table 28-203: EMAC_Q[i]_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	PLT	Pause Low Threshold. The `EMAC_Q[i]_TXFLOW_CTL.PLT` field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i is checked for automatic retransmission of the Pause packet. The threshold values must be less than the Pause Time (PT) configured in bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first pause packet is transmitted. The following list provides the threshold values for different values. The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.	
		0	PT - 4 Slot Times
		1	PT - 28 Slot Times
		2	PT - 36 Slot Times
		3	PT - 144 Slot Times
		4	PT - 256 Slot Times
		5	PT - 512 Slot Times
		6	Reserved
1 (R/W)	TFE	Transmit Flow Control Enable. When the `EMAC_Q[i]_TXFLOW_CTL.TFE` bit is set in full-duplex mode, the MAC enables the flow control operation to Tx pause packets. When `EMAC_Q[i]_TXFLOW_CTL.TFE` = 0, the flow control operation in the MAC is disabled, and the MAC does not transmit any pause packets.	
		0	Transmit Flow Control is Disabled
		1	Transmit Flow Control is Enabled

Table 28-203: EMAC_Q[i]_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	FCB_BPA	Flow Control Busy. The EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit initiates a PFC packet if EMAC_Q[i]_TXFLOW_CTL.TFE = 1. To initiate a PFC packet, the application must set the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit to 1'b1. During the control packet transfer, the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit continues to be set to indicate that a packet transmission is in progress. When PFC packet transmission is complete, the MAC resets the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit to 1'b0. Do not write to this register until the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit is cleared. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flow Control Busy or Backpressure Activate is Disabled
		1	Flow Control Busy or Backpressure Activate is Enabled

Receive Queue Control 0 Register

The EMAC_RXQ_CTL0 register controls the queue management in the MAC receiver.

Note: In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

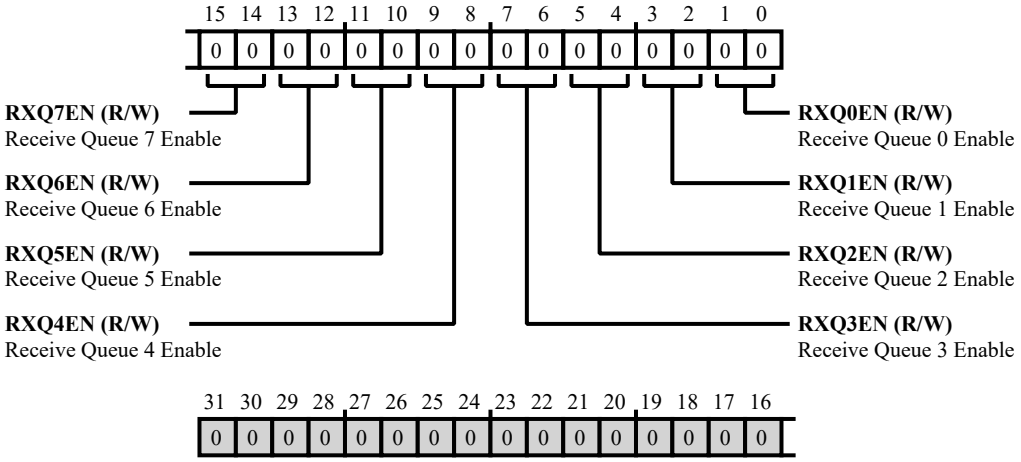


Figure 28-149: EMAC_RXQ_CTL0 Register Diagram

Table 28-204: EMAC_RXQ_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/W)	RXQ7EN	Receive Queue 7 Enable. The EMAC_RXQ_CTL0.RXQ7EN field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
13:12 (R/W)	RXQ6EN	Receive Queue 6 Enable. The EMAC_RXQ_CTL0.RXQ6EN field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved

Table 28-204: EMAC_RXQ_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	RXQ5EN	Receive Queue 5 Enable. The `EMAC_RXQ_CTL0.RXQ5EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
9:8 (R/W)	RXQ4EN	Receive Queue 4 Enable. The `EMAC_RXQ_CTL0.RXQ4EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
7:6 (R/W)	RXQ3EN	Receive Queue 3 Enable. The `EMAC_RXQ_CTL0.RXQ3EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
5:4 (R/W)	RXQ2EN	Receive Queue 2 Enable. The `EMAC_RXQ_CTL0.RXQ2EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
3:2 (R/W)	RXQ1EN	Receive Queue 1 Enable. The `EMAC_RXQ_CTL0.RXQ1EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved

Table 28-204: EMAC_RXQ_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	RXQ0EN	Receive Queue 0 Enable. The `EMAC_RXQ_CTL0.RXQ0EN` field indicates whether Rx queue 0 is enabled for AV or DCB.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved

Receive Queue Control 1 Register

The EMAC_RXQ_CTL1 register controls the routing of multicast, broadcast, AV, DCB, and untagged packets to the Rx queues.

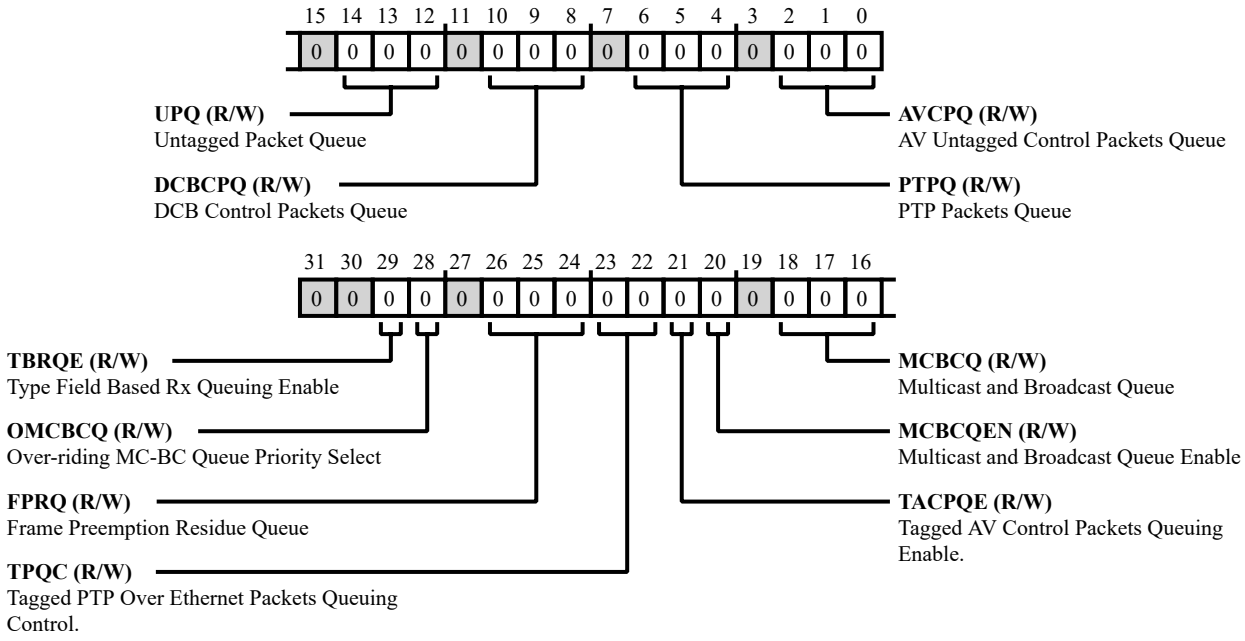


Figure 28-150: EMAC_RXQ_CTL1 Register Diagram

Table 28-205: EMAC_RXQ_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	TBRQE	Type Field Based Rx Queuing Enable. When the EMAC_RXQ_CTL1.TBRQE bit is set, it enables the Type field based Rx queuing where the Type field of received packet is compared with programmed TYP field in MAC_TMRQ_Regs(n) and if a match occurs the packet is routed to the corresponding TMRQ field.	
28 (R/W)	OMCBCQ	Over-riding MC-BC Queue Priority Select. When set (=1), the EMAC_RXQ_CTL1.OMCBCQ bit indicates that the priority of MCBCQ is reduced and the received packet is first routed to PTPQ, AVCPQ, DCBCPQ depending on packet type. When cleared (=0), the EMAC_RXQ_CTL1.OMCBCQ bit indicates that the received multicast/broadcast packet is routed to MCBCQ.	
		0	Overriding MCBCQ Priority Disabled
		1	Overriding MCBCQ Priority Enabled

Table 28-205: EMAC_RXQ_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/W)	FPRQ	Frame Preemption Residue Queue. The EMAC_RXQ_CTL1.FPRQ field holds the Rx queue number to which the residual preemption frames must be forwarded. Preemption frames that are tagged and pass the SA/DA/VLAN filtering are routed based on PSRQ and all other frames are treated as residual frames and is routed to the queue number mentioned in this field. The queue 0 is used as a default queue for express frames, so this field cannot be programmed to a value 0.	
23:22 (R/W)	TPQC	Tagged PTP Over Ethernet Packets Queuing Control.. The EMAC_RXQ_CTL1.TPQC field controls the routing of the VLAN Tagged PTPoE packets. The following programmable options are allowed: - 2'b00: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for only non-AV enabled Rx queue). - 2'b01: VLAN Tagged PTPoE packets are routed to Rx queue specified by PTPQ field (Rx queue can be enabled for AV or non-AV traffic). - 2'b10: VLAN Tagged PTPoE packets are routed to only AV enabled Rx queues based on PSRQ. - 2'b11: Reserved If DWC_EQOS_AV_ENABLE is not selected in the configuration, the following programmable options are allowed. - 1'b0: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for DCB/Generic enabled Rx queues). - 1'b1: VLAN Tagged PTPoE packets are routed to Rx queue specified by PTPQ field.	
21 (R/W)	TACPQE	Tagged AV Control Packets Queuing Enable.. When the EMAC_RXQ_CTL1.TACPQE bit is set, the MAC routes the received Tagged AV Control packets to the Rx queue specified by AVCPQ field. When EMAC_RXQ_CTL1.TACPQE = 0, the MAC routes the received Tagged AV Control packets based on the tag priority matching the PSRQn fields in EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers.	
		0	Tagged AV Control Packets Queuing is Disabled
		1	Tagged AV Control Packets Queuing is Enabled
20 (R/W)	MCBCQEN	Multicast and Broadcast Queue Enable. The EMAC_RXQ_CTL1.MCBCQEN bit specifies that multicast or broadcast packets routing to the Rx queue is enabled and the multicast or broadcast packets must be routed to Rx queue specified in the EMAC_RXQ_CTL1.MCBCQ field.	
		0	Multicast and Broadcast Queue is Disabled
		1	Multicast and Broadcast Queue is Enabled

Table 28-205: EMAC_RXQ_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	MCBCQ	Multicast and Broadcast Queue. The `EMAC_RXQ_CTL1.MCBCQ` field specifies the Rx queue onto which Multicast or Broadcast Packets are routed. Any Rx queue enabled for Generic/DCB/AV traffic can be used to route the Multicast or Broadcast Packets.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
14:12 (R/W)	UPQ	Untagged Packet Queue. The `EMAC_RXQ_CTL1.UPQ` field indicates the Rx queue to which untagged packets are to be routed. Any Rx queue enabled for generic/DCB/AV traffic can be used to route the untagged packets.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
10:8 (R/W)	DCBCPQ	DCB Control Packets Queue. The `EMAC_RXQ_CTL1.DCBCPQ` field specifies the Rx queue on which the received DCB control packets are routed. The DCB data packets are routed based on the EMAC_Q[n]_TXFLOW_CTL.PRSQ field of corresponding queue.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4

Table 28-205: EMAC_RXQ_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
6:4 (R/W)	PTPQ	PTP Packets Queue. The EMAC_RXQ_CTL1.PTPQ field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. When EMAC_TM_CTL.AV8021ASMEN = 1, only untagged PTP over Ethernet packets are routed on an Rx queue. If the bit is not set, then based on programming of the EMAC_RXQ_CTL1.TPQC field, both tagged and untagged PTPoE packets can be routed to this Rx queue.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
2:0 (R/W)	AVCPQ	AV Untagged Control Packets Queue. The EMAC_RXQ_CTL1.AVCPQ field specifies the receive queue on which the received AV tagged and untagged control packets are routed. The AV tagged (when EMAC_RXQ_CTL1.TACPQE= 1) and untagged control packets are routed to receive queue specified by this field.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7

Receive Queue Control 2 Register

The EMAC_RXQ_CTL2 register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 0 to 3.

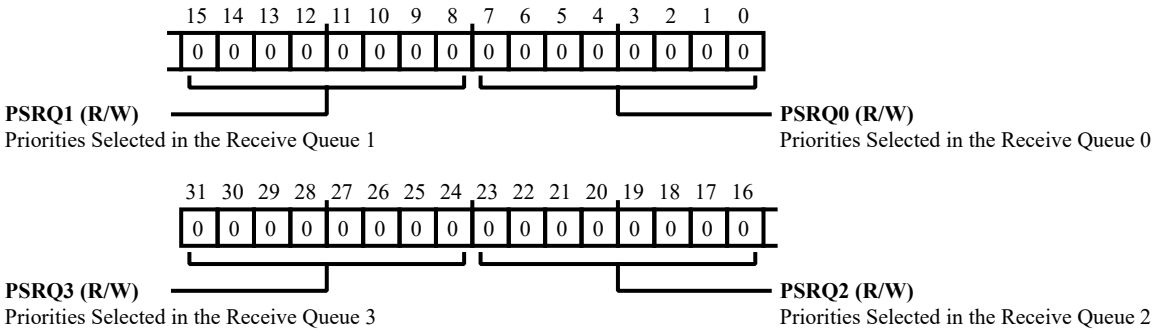


Figure 28-151: EMAC_RXQ_CTL2 Register Diagram

Table 28-206: EMAC_RXQ_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSRQ3	Priorities Selected in the Receive Queue 3. The EMAC_RXQ_CTL2.PSRQ3 field selects the priorities assigned to Rx queue 3. All packets with priorities that match the values set in this field are routed to Rx queue 3. For example, if PSRQ3[6, 3] are set, packets with USP field equal to 3 or 6 are routed to Rx queue 3. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 3 crosses the flow control threshold settings.
23:16 (R/W)	PSRQ2	Priorities Selected in the Receive Queue 2. The EMAC_RXQ_CTL2.PSRQ2 field selects the priorities assigned to Rx queue 2. All packets with priorities that match the values set in this field are routed to Rx queue 2. For example, if PSRQ2[1, 0] are set, packets with USP field equal to 1 or 0 are routed to Rx queue 2. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 2 crosses the flow control threshold settings.

Table 28-206: EMAC_RXQ_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	PSRQ1	Priorities Selected in the Receive Queue 1. The `EMAC_RXQ_CTL2.PSRQ1` field selects the priorities assigned to Rx queue 1. All packets with priorities that match the values set in this field are routed to Rx queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 1 crosses the flow control threshold settings.
7:0 (R/W)	PSRQ0	Priorities Selected in the Receive Queue 0. The `EMAC_RXQ_CTL2.PSRQ0` field selects the priorities assigned to Rx queue 0. All packets with priorities that match the values configured in this field are routed to Rx queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 0 crosses the flow control threshold settings.

Receive Queue Control 3 Register

The EMAC_RXQ_CTL3 register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 4 to 7.

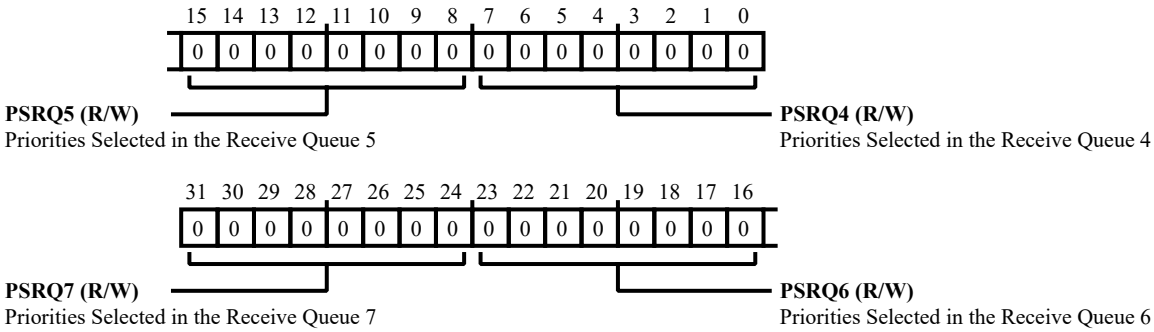


Figure 28-152: EMAC_RXQ_CTL3 Register Diagram

Table 28-207: EMAC_RXQ_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSRQ7	Priorities Selected in the Receive Queue 7. The EMAC_RXQ_CTL3.PSRQ7 field selects the priorities assigned to Rx queue 7. All packets with priorities that match the values set in this field are routed to Rx queue 7. For example, if PSRQ7[7, 4] are set, packets with USP field equal to 7 or 4 are routed to Rx queue 7. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 7 crosses the flow control threshold settings.
23:16 (R/W)	PSRQ6	Priorities Selected in the Receive Queue 6. The EMAC_RXQ_CTL3.PSRQ6 field selects the priorities assigned to Rx queue 6. All packets with priorities that match the values set in this field are routed to Rx queue 6. For example, if PSRQ6[5] are set, packets with USP field equal to 5 are routed to Rx queue 6. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 6 crosses the flow control threshold settings.

Table 28-207: EMAC_RXQ_CTL3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	PSRQ5	Priorities Selected in the Receive Queue 5. The `EMAC_RXQ_CTL3.PSRQ5` field selects the priorities assigned to Rx queue 5. All packets with priorities that match the values set in this field are routed to Rx queue 5. For example, if PSRQ5[6] is set, packets with USP field equal to 6 are routed to Rx queue 5. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 5 crosses the flow control threshold settings.
7:0 (R/W)	PSRQ4	Priorities Selected in the Receive Queue 4. The `EMAC_RXQ_CTL3.PSRQ4` field selects the priorities assigned to Rx queue 4. All packets with priorities that match the values set in this field are routed to Rx queue 4. For example, if PSRQ4[7:4] is set, packets with USP field equal to 7, 6, 5, or 4 are routed to Rx queue 4. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 4 crosses the flow control threshold settings.

Receive Queue Control 4 Register

The EMAC_RXQ_CTL4 register controls the routing of unicast and multicast packets that fail the destination or source address filter to the Rx queues.

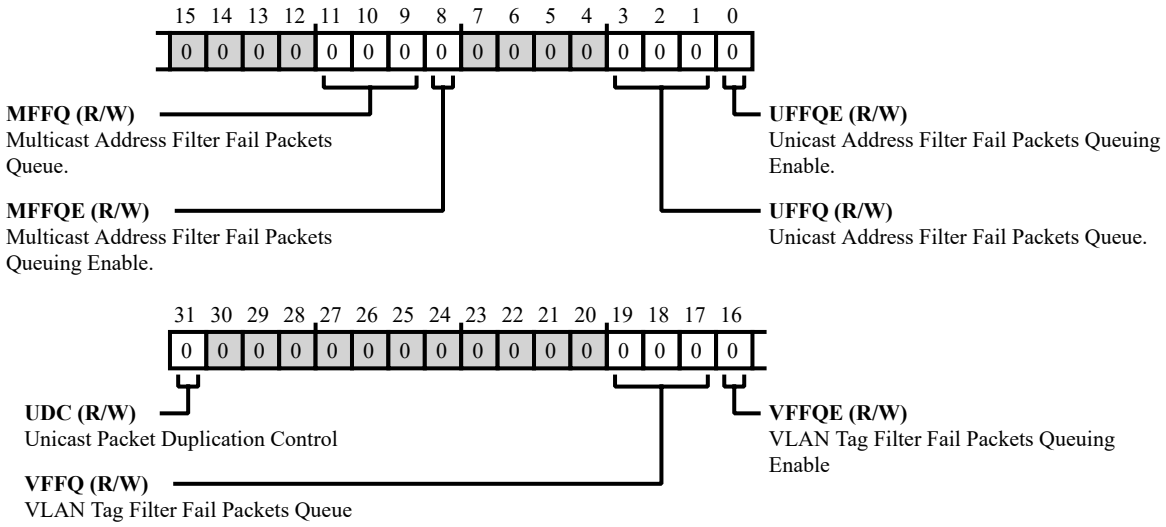


Figure 28-153: EMAC_RXQ_CTL4 Register Diagram

Table 28-208: EMAC_RXQ_CTL4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	UDC	Unicast Packet Duplication Control. When theEMAC_RXQ_CTL4.UDC bit is set along with PDC bit of the MAC_Ext_Configuration register, the MAC provides DMA Channel number specified as one-hot value in DCS field of MAC_Address(#i)_High register that matched the Unicast Destination Address of the received packet. Note: When the above condition is satisfied for unicast packet but the highest Receive Queue selected in configuration is not enabled or when DA/SA/VLAN filter fails and corresponding filter fail Receive Queue is enabled, it is routed to Receive Queue determined by other queuing methods. When the EMAC_RXQ_CTL4.UDC bit is reset, the MAC routes the received packet with Unicast Destination Address based on other queuing methods, the DMA Channel number selected is binary value from DCS field of MAC_Address(#i)_High register (if PDC bit is reset) or lowest bit set in one-hot value (if PDC is set) in DCS field of MAC_Address(#i)_High register that matched the Unicast Destination Address of the received packet.

Table 28-208: EMAC_RXQ_CTL4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:17 (R/W)	VFFQ	VLAN Tag Filter Fail Packets Queue. The EMAC_RXQ_CTL4.VFFQ field holds the Rx queue number to which the tagged packets failing the Destination or Source Address filter (and EMAC_RXQ_CTL4.UFFQE/ EMAC_RXQ_CTL4.MFFQE = 0) or failing the VLAN tag filter must be routed to. The EMAC_RXQ_CTL4.VFFQ field is valid only when EMAC_RXQ_CTL4.VFFQE = 1.	
16 (R/W)	VFFQE	VLAN Tag Filter Fail Packets Queuing Enable. When the EMAC_RXQ_CTL4.VFFQE bit is set, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter, are routed to the Rx queue number programmed in the EMAC_RXQ_CTL4.VFFQ field. When EMAC_RXQ_CTL4.VFFQE = 0, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter are routed based on other routing options. The EMAC_RXQ_CTL4.VFFQE bit is valid only when the EMAC_MACPKT_FILT.RA = 1.	
		0	VLAN Tag Filter Fail Packets Queuing is Disabled
		1	VLAN Tag Filter Fail Packets Queuing is Enabled
11:9 (R/W)	MFFQ	Multicast Address Filter Fail Packets Queue.. The EMAC_RXQ_CTL4.MFFQ field holds the Rx queue number to which the Multicast packets failing the Destination or Source Address filter are routed to. The EMAC_RXQ_CTL4.MFFQ field is valid only when EMAC_RXQ_CTL4.MFFQE = 1.	
8 (R/W)	MFFQE	Multicast Address Filter Fail Packets Queuing Enable.. When the EMAC_RXQ_CTL4.MFFQE bit is set, the Multicast packets which fail the Destination or Source address filter is routed to the Rx queue Number programmed in the EMAC_RXQ_CTL4.MFFQ. When EMAC_RXQ_CTL4.MFFQE = 0, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options. The EMAC_RXQ_CTL4.MFFQE bit is valid only when EMAC_MACPKT_FILT.RA = 1.	
		0	Multicast Address Filter Fail Packets Queuing is Disabled
		1	Multicast Address Filter Fail Packets Queuing is Enabled

Table 28-208: EMAC_RXQ_CTL4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:1 (R/W)	UFFQ	Unicast Address Filter Fail Packets Queue.. The EMAC_RXQ_CTL4.UFFQ field holds the Rx queue number to which the Unicast packets failing the Destination or Source Address filter are routed to. The EMAC_RXQ_CTL4.UFFQ field is valid only when EMAC_RXQ_CTL4.UFFQE = 1.	
0 (R/W)	UFFQE	Unicast Address Filter Fail Packets Queuing Enable.. When the EMAC_RXQ_CTL4.UFFQE bit is set, the Unicast packets which fail the Destination or Source address filter is routed to the Rx queue number programmed in the EMAC_RXQ_CTL4.UFFQ field. When EMAC_RXQ_CTL4.UFFQE = 0, the Unicast packets which fail the Destination or Source address filter is routed based on other routing options. The EMAC_RXQ_CTL4.UFFQE bit is valid only when EMAC_MACPKT_FILT.RA = 1.	
		0	Unicast Address Filter Fail Packets Queuing is Disabled
		1	Unicast Address Filter Fail Packets Queuing is Enabled

MAC Receive Time Domain Increment Register

The EMAC_MAC_RX_DOMAIN_TIME_INCR register indicates the increment value of the PTP time maintained in receive clock domain that is used for time stamping.

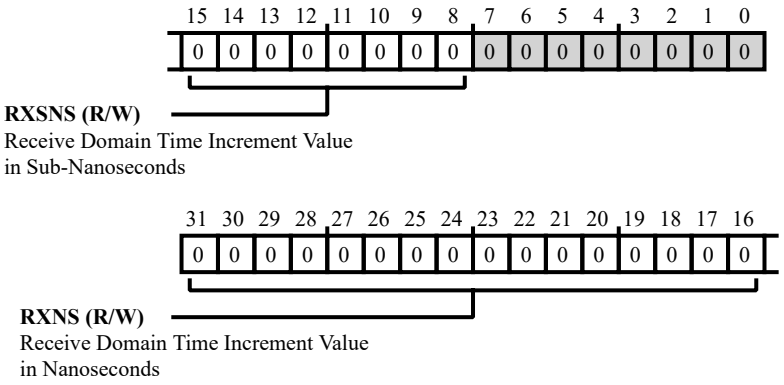


Figure 28-154: EMAC_MAC_RX_DOMAIN_TIME_INCR Register Diagram

Table 28-209: EMAC_MAC_RX_DOMAIN_TIME_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RXNS	Receive Domain Time Increment Value in Nanoseconds. The EMAC_MAC_RX_DOMAIN_TIME_INCR.RXNS bit field indicates the nanosecond part of the increment value of the PTP time maintained in receive clock domain that is used for time stamping. For example, when operating in 2.5G speed the clock period is 3.2ns, so for digital rollover mode the RXNS and RXSNS fields must be programmed with 3 and (0.2*256)=51 respectively. In binary rollover mode, 3/0.466=6.44 and 0.2+0.44=0.64, so RXNS and RXSNS fields must be programmed with 6 and (0.64*256)=164 respectively. When both RXSNS and RXNS fields are all-zeros, the fixed value based on operating speed is used for PTP time increment in receive clock domain.
15:8 (R/W)	RXSNS	Receive Domain Time Increment Value in Sub-Nanoseconds. The EMAC_MAC_RX_DOMAIN_TIME_INCR.RXSNS bit field indicates the sub-nanosecond part of the increment value of the PTP time maintained in receive clock domain that is used for time stamping. This field has granularity of 1/256 ns. When both RXSNS and RXNS fields are all-zeros, the fixed value based on operating speed is used for PTP time increment in receive clock domain.

Receive Flow Control Register

The EMAC_RXFLOW_CTL register controls the pausing of MAC transmit based on the received pause packet.

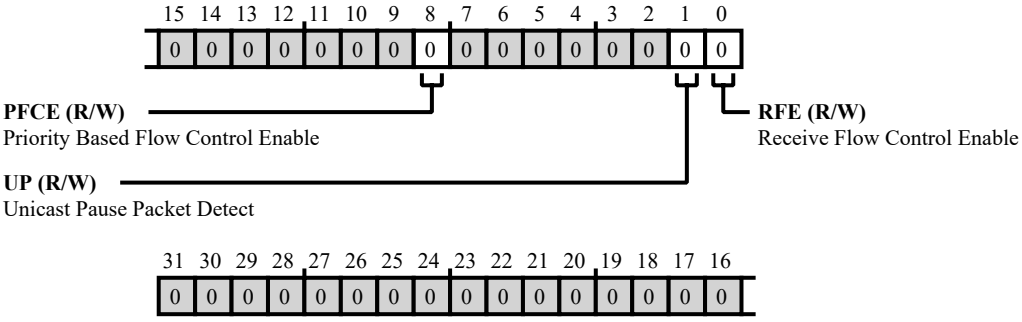


Figure 28-155: EMAC_RXFLOW_CTL Register Diagram

Table 28-210: EMAC_RXFLOW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	PFCE	Priority Based Flow Control Enable. When the EMAC_RXFLOW_CTL.PFCE bit is set, it enables generation and reception of priority-based flow control (PFC) packets. When EMAC_RXFLOW_CTL.PFCE = 0, it enables generation and reception of 802.3x Pause control packets.	
		0	Priority Based Flow Control is Disabled
		1	Priority Based Flow Control is Enabled
1 (R/W)	UP	Unicast Pause Packet Detect. A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When the EMAC_RXFLOW_CTL.UP bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in EMAC_ADDR0_HI and EMAC_ADDR0_LO. When EMAC_RXFLOW_CTL.UP = 0, the MAC only detects Pause packets with unique multicast address. Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.	
		0	Unicast Pause Packet Detect Disabled
		1	Unicast Pause Packet Detect Enabled

Table 28-210: EMAC_RXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	RFE	Receive Flow Control Enable. When the EMAC_RXFLOW_CTL.RFE bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When EMAC_RXFLOW_CTL.RFE = 0 or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.	
		0	Receive Flow Control is Disabled
		1	Receive Flow Control is Enabled

Receive Transmit Status Register

The EMAC_RXTX_STAT register contains the receive and transmit error status.

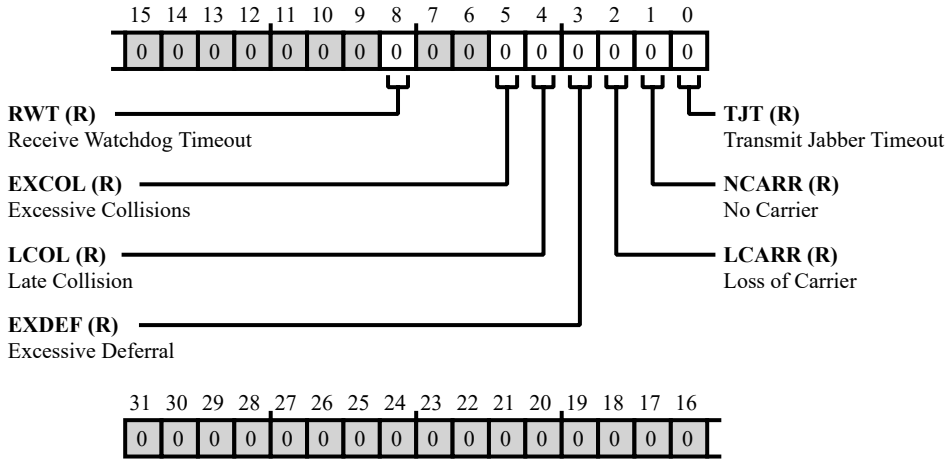


Figure 28-156: EMAC_RXTX_STAT Register Diagram

Table 28-211: EMAC_RXTX_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	RWT	Receive Watchdog Timeout. The EMAC_RXTX_STAT.RWT bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1) and EMAC_MAC_CFG.WD = 0. The EMAC_RXTX_STAT.RWT bit is set when a packet with length greater than 16,383 bytes is received and EMAC_MAC_CFG.WD = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Receive Watchdog Timeout
		1	Receive Watchdog Timed Out
5 (R/NW)	EXCOL	Excessive Collisions. When EMAC_MTL_OPMODE.DTXSTS = 1, the EMAC_RXTX_STAT.EXCOL bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. When EMAC_MAC_CFG.DR = 1, the EMAC_RXTX_STAT.EXCOL bit is set after the first collision and the packet transmission is aborted. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Collision
		1	Excessive Collision is Sensed

Table 28-211: EMAC_RXTX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	LCOL	Late Collision. When EMAC_MTL_OPMODE.DTXSTS = 1, the `EMAC_RXTX_STAT.LCOL` bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode). The `EMAC_RXTX_STAT.LCOL` bit is not valid if the Underflow error occurs. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Collision
		1	Late Collision is Sensed
3 (R/NW)	EXDEF	Excessive Deferral. When EMAC_MTL_OPMODE.DTXSTS = 1 and EMAC_MAC_CFG.DC = 1, the `EMAC_RXTX_STAT.EXDEF` bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000/2500 Mbps mode or when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1)). Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Excessive Deferral
		1	Excessive Deferral
2 (R/NW)	LCARR	Loss of Carrier. When EMAC_MTL_OPMODE.DTXSTS = 1, the `EMAC_RXTX_STAT.LCARR` bit indicates that the loss of carrier occurred during packet transmission, that is, the ETHx_CRS signal was inactive for one or more transmission clock periods during packet transmission. The `EMAC_RXTX_STAT.LCARR` bit is valid only for packets transmitted without collision. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Carrier is Present
		1	Loss of Carrier
1 (R/NW)	NCARR	No Carrier. When EMAC_MTL_OPMODE.DTXSTS = 1, the `EMAC_RXTX_STAT.NCARR` bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Carrier is Present
		1	No Carrier

Table 28-211: EMAC_RXTX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	TJT	Transmit Jabber Timeout. The `EMAC_RXTX_STAT.TJT` bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1) and EMAC_MAC_CFG.JD = 0. The `EMAC_RXTX_STAT.TJT` bit is set when the packet size exceeds 16,383 bytes and the EMAC_MAC_CFG.JD = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Transmit Jabber Timeout
		1	Transmit Jabber Timeout Occurred

Source Port Identity 0 Register

The EMAC_SRCPRT_IDNTY0 register contains bits[31:0] of the 80-bit source port identity of the PTP node.

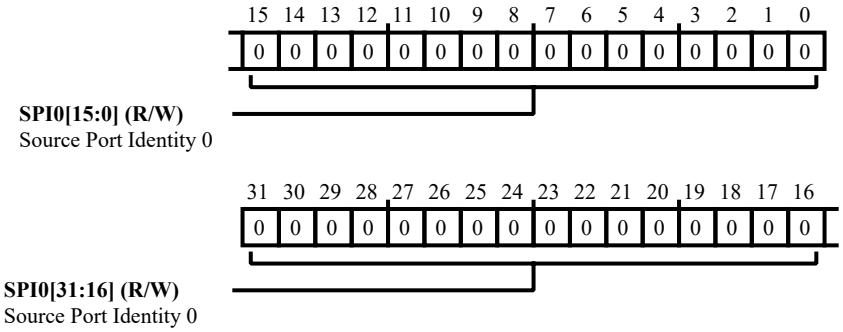


Figure 28-157: EMAC_SRCPRT_IDNTY0 Register Diagram

Table 28-212: EMAC_SRCPRT_IDNTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SPI0	Source Port Identity 0. The EMAC_SRCPRT_IDNTY0.SPI0 field indicates bits [31:0] of sourcePortIdentity of PTP node.

Source Port Identity 1 Register

The EMAC_SRCPRT_IDNTY1 register contains bits[63:32] of the 80-bit Source Port Identity of the PTP node.

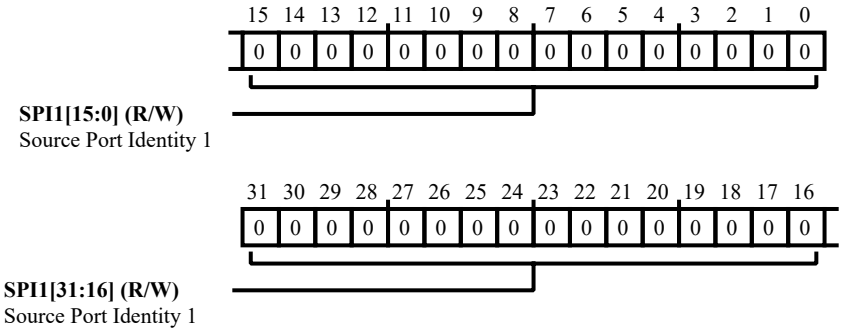


Figure 28-158: EMAC_SRCPRT_IDNTY1 Register Diagram

Table 28-213: EMAC_SRCPRT_IDNTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SPI1	Source Port Identity 1. The EMAC_SRCPRT_IDNTY1.SPI1 field indicates bits [63:32] of sourcePortIdentity of PTP node.

Source Port Identity 2 Register

The EMAC_SRCPRT_IDNTY2 register contains bits[79:64] of the 80-bit Source Port Identity of the PTP node.

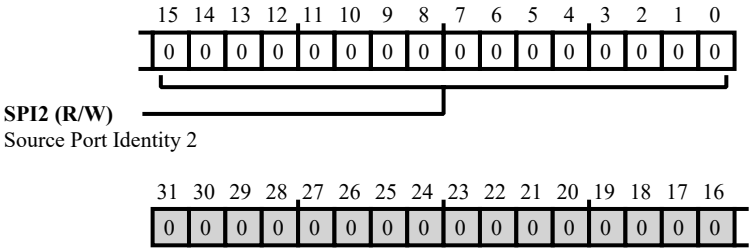


Figure 28-159: EMAC_SRCPRT_IDNTY2 Register Diagram

Table 28-214: EMAC_SRCPRT_IDNTY2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	SPI2	Source Port Identity 2. The EMAC_SRCPRT_IDNTY2.SPI2 field indicates bits [79:64] of sourcePortIdentity of PTP node.

Subsecond Increment Register

The EMAC_TM_SUBSEC register specifies the value to be added to the internal system time register every cycle of PTP clock.

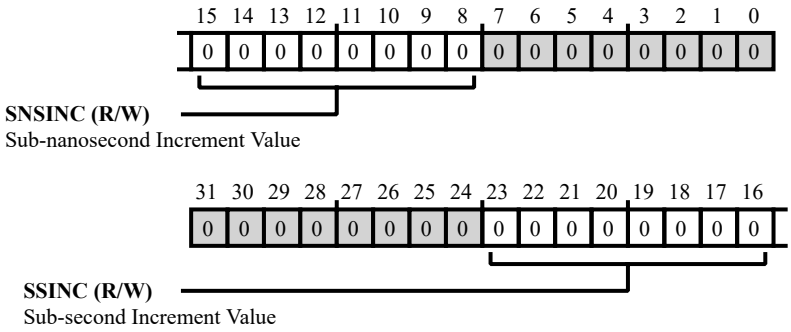


Figure 28-160: EMAC_TM_SUBSEC Register Diagram

Table 28-215: EMAC_TM_SUBSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	SSINC	Sub-second Increment Value. The value programmed in the EMAC_TM_SUBSEC.SSINC field is accumulated every PTP clock cycle with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), program 20 (0x14) when the EMAC_TM_NSEC register has an accuracy of 1 ns (EMAC_TM_CTL.TSCTRLSSR = 1). When EMAC_TM_CTL.TSCTRLSSR = 0, the EMAC_TM_NSEC register has a resolution of ~0.465 ns. In this case, program a value of 43 (0x2B) which is derived by 20 ns/0.465.
15:8 (R/W)	SNSINC	Sub-nanosecond Increment Value. The EMAC_TM_SUBSEC.SNSINC field contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2^8. This value is accumulated with the sub-nanoseconds field of the subsecond register. For example, when EMAC_TM_CTL.TSCTRLSSR = 1. and if the required increment is 5.3ns, then EMAC_TM_SUBSEC.SSINC should be 0x05 and EMAC_TM_SUBSEC.SNSINC should be 0x4C.

System Time Higher Word Seconds Register

System Time - Higher Word Seconds register.

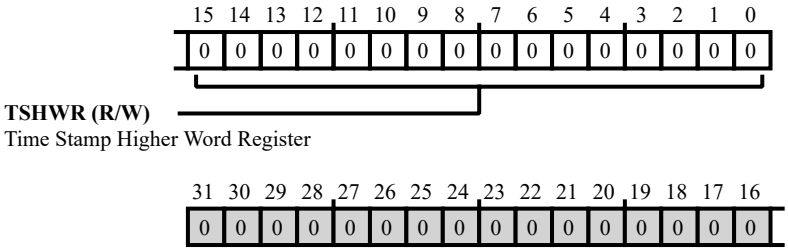


Figure 28-161: EMAC_TM_HISEC Register Diagram

Table 28-216: EMAC_TM_HISEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TSHWR	Time Stamp Higher Word Register. The `EMAC_TM_HISEC.TSHWR` field contains the most-significant 16-bits of time stamp seconds value. The `EMAC_TM_HISEC` register is optional. Add this register by selecting the Add IEEE 1588 Higher Word Register option. The `EMAC_TM_HISEC` register is directly written to initialize the value and it is incremented when there is an overflow from 32 bits of the EMAC_TM_SEC register. Access restriction applies. Updated based on the event. Write 1 to set. Write 0 to clear.

System Time Nanoseconds Register

The EMAC_TM_NSEC register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

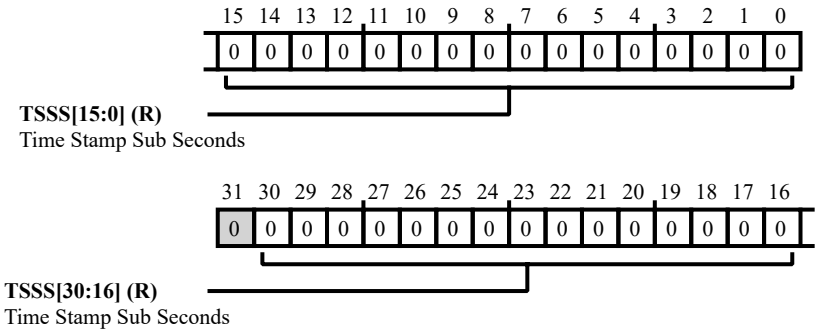


Figure 28-162: EMAC_TM_NSEC Register Diagram

Table 28-217: EMAC_TM_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:0 (R/NW)	TSSS	Time Stamp Sub Seconds. The value in the EMAC_TM_NSEC.TSSS field has the sub-second representation of time, with an accuracy of 0.46 ns. (EMAC_TM_CTL.TSCTRLSSR = 1), each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls over to zero.

System Time Nanoseconds Update Register

MAC System Time Nanoseconds Update register.

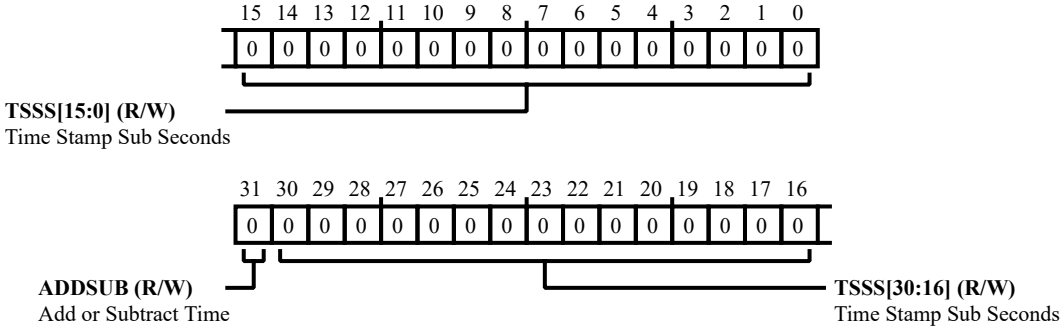


Figure 28-163: EMAC_TM_NSECUPDT Register Diagram

Table 28-218: EMAC_TM_NSECUPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ADDSUB	Add or Subtract Time. When the `EMAC_TM_NSECUPDT.ADDSUB` bit is set, the time value is subtracted with the contents of the update register. When `EMAC_TM_NSECUPDT.ADDSUB` = 0, the time value is added with the contents of the update register.	
		0	Add Time
		1	Subtract Time

Table 28-218: EMAC_TM_NSECUPDT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30:0 (R/W)	TSSS	Time Stamp Sub Seconds. The `EMAC_TM_NSECUPDT.TSSS` field is the sub-seconds part of the update. When EMAC_TM_NSECUPDT.ADDSUB = 1, the `EMAC_TM_NSECUPDT.TSSS` field must be programmed with the complement of the sub-seconds part of the update value as described. When EMAC_TM_NSECUPDT.ADDSUB = 0, the `EMAC_TM_NSECUPDT.TSSS` field must be programmed with the sub-seconds part of the update value, with an accuracy based on the EMAC_TM_CTL.TSCTRLSSR bit. For EMAC_TM_CTL.TSCTRLSSR = 1: -- The programmed value must be 10^9 - <sub-second value>. -- Each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF. For EMAC_TM_CTL.TSCTRLSSR = 0: -- The programmed value must be 2^{31} - <sub-second_value>. -- Each bit represents an accuracy of 0.46 ns. For example, to subtract 2.000000001 seconds from the system time, then the EMAC_TM_NSECUPDT.TSSS field must be 0x7FFF_FFFF (that is, 2^{31} - 1), when EMAC_TM_CTL.TSCTRLSSR = 0 and 0x3B9A_C9FF (that is, 10^9 - 1), when EMAC_TM_CTL.TSCTRLSSR = 1.

System Time Seconds Register

The EMAC_TM_SEC register, along with the EMAC_TM_NSEC register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from PTP clock to SCLK0).

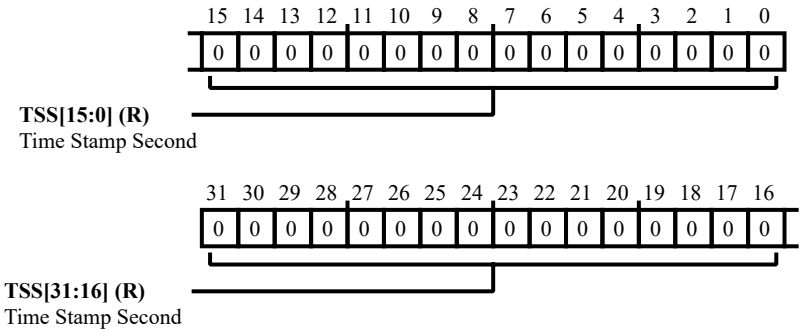


Figure 28-164: EMAC_TM_SEC Register Diagram

Table 28-219: EMAC_TM_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TSS	Time Stamp Second. The value in the EMAC_TM_SEC.TSS field indicates the current value in seconds of the System Time maintained by the MAC.

System Time Seconds Update Register

The EMAC_TM_SECUPDT register, along with the EMAC_TM_NSECUPDT register, initializes or updates the system time maintained by the MAC. Write both registers before setting the EMAC_TM_CTL.TSINIT or EMAC_TM_CTL.TSUPDT bits.

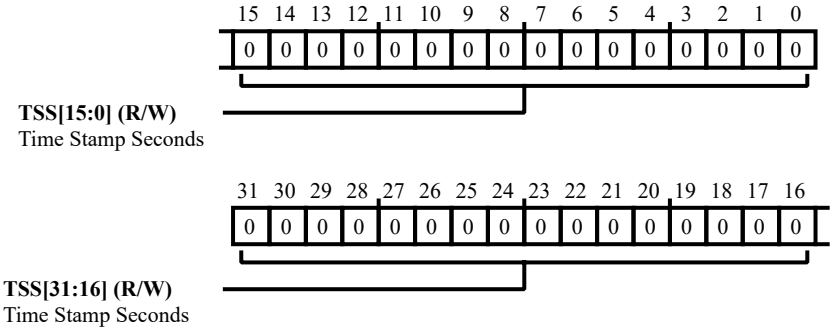


Figure 28-165: EMAC_TM_SECUPDT Register Diagram

Table 28-220: EMAC_TM_SECUPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSS	Time Stamp Seconds. The value in the EMAC_TM_SECUPDT.TSS field is the seconds part of the update. When EMAC_TM_NSECUPDT.ADDSUB = 0, the EMAC_TM_SECUPDT.TSS field must be programmed with the seconds part of the update value. When EMAC_TM_NSECUPDT.ADDSUB = 1, the EMAC_TM_SECUPDT.TSS field must be programmed with the complement of the seconds part of the update value. For example, to subtract 2.000000001 seconds from the system time, the EMAC_TM_SECUPDT.TSS field must be 0xFFFF_FFFE (that is, 2^{32} - 2).

Time Stamp Addend Register

The EMAC_TM_ADDEND register value is used only when the system time is configured for Fine Update mode (EMAC_TM_CTL.TSCFUPDT). The content of this register is added to a 32-bit accumulator in every clock cycle (PTP clock) and the system time is updated when the accumulator overflows.

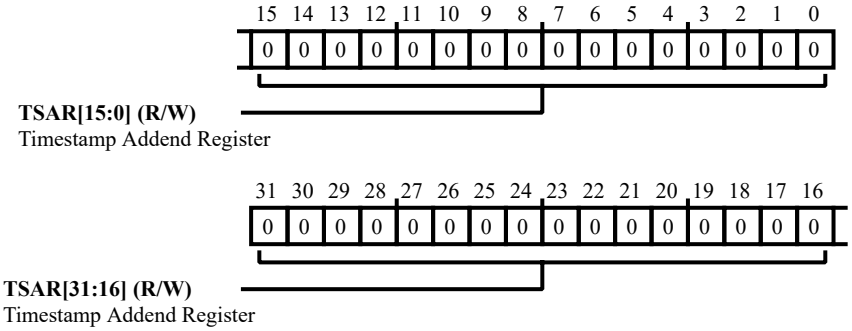


Figure 28-166: EMAC_TM_ADDEND Register Diagram

Table 28-221: EMAC_TM_ADDEND Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSAR	Timestamp Addend Register. The EMAC_TM_ADDEND.TSAR field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Time Stamp Control Register

The EMAC_TM_CTL register controls the operation of the System Time generator and processing of PTP packets for time stamping in the receiver.

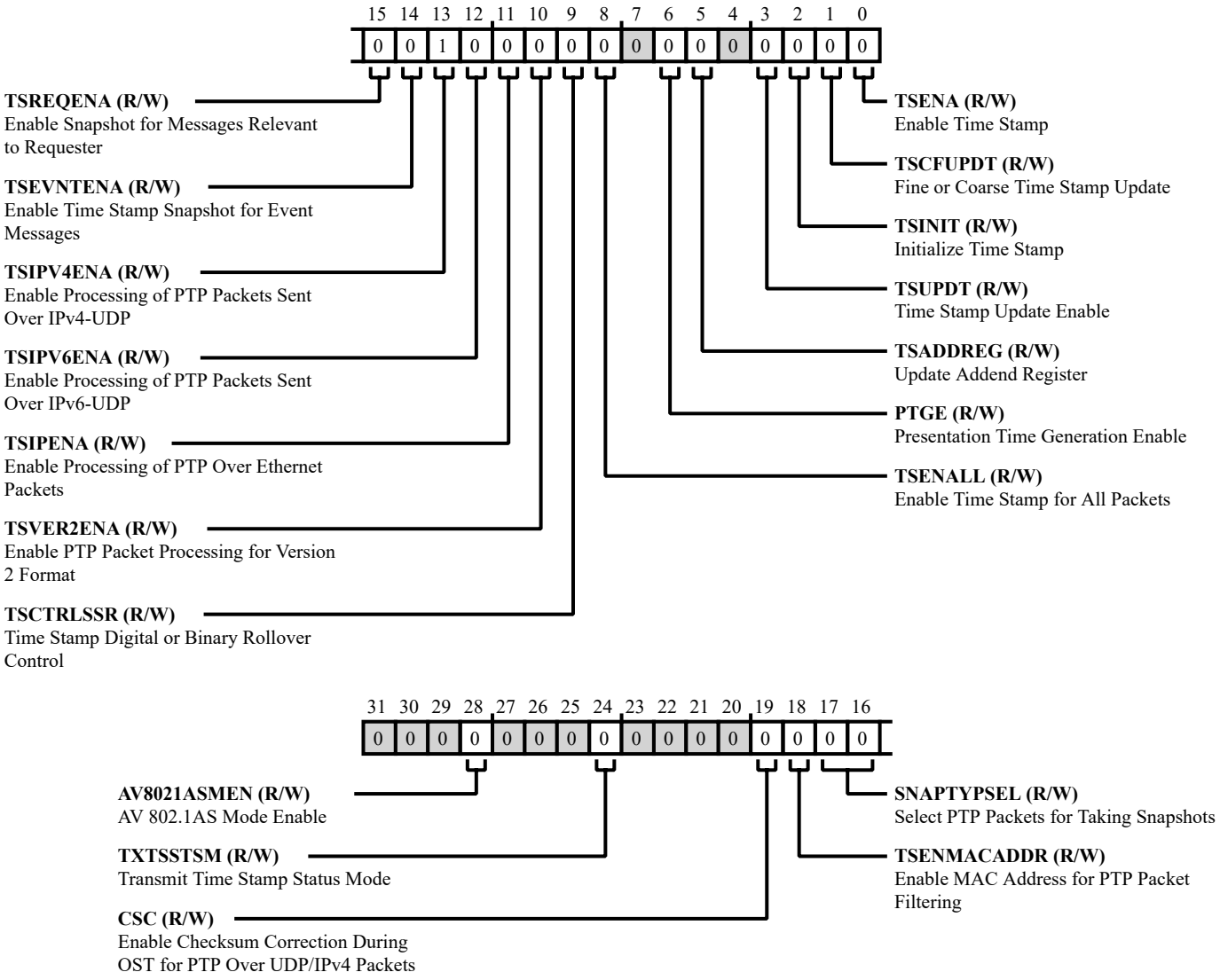


Figure 28-167: EMAC_TM_CTL Register Diagram

Table 28-222: EMAC_TM_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	AV8021ASMEN	AV 802.1AS Mode Enable. When the EMAC_TM_CTL.AV8021ASMEN bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing time stamp snapshots, that is, IEEE 802.1AS mode of operation. When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of the EMAC_TM_CTL.AV8021ASMEN bit.	
		0	AV 802.1AS Mode is Disabled
		1	AV 802.1AS Mode is Enabled
24 (R/W)	TXTSSTSM	Transmit Time Stamp Status Mode. When the EMAC_TM_CTL.TXTSSTSM bit is set, the MAC overwrites the earlier transmit time stamp status even if it is not read by the software. The MAC indicates this by setting the EMAC_TXTMSTAT_NSEC.TXTSSMIS bit. When EMAC_TM_CTL.TXTSSTSM = 0, the MAC ignores the time stamp status of current packet if the time stamp status of previous packet is not read by the software. The MAC indicates this by setting the EMAC_TXTMSTAT_NSEC.TXTSSMIS bit.	
		0	Disable
		1	Enable
19 (R/W)	CSC	Enable Checksum Correction During OST for PTP Over UDP/IPv4 Packets. When the EMAC_TM_CTL.CSC bit is set, the last two bytes of PTP message sent over UDP/IPv4 is updated to keep the UDP checksum correct, for changes made to origin time stamp and/or correction field as part of one step time stamp operation. The application forms the packet with these two dummy bytes. When cleared (=0), no updates are done to keep the UDP checksum correct. The application forms the packet with UDP checksum configured to 0.	
		0	Checksum Correction During OST for PTP Over UDP/IPv4 Packets is Disabled
		1	Checksum Correction During OST for PTP Over UDP/IPv4 Packets is Enabled

Table 28-222: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	TSENMACADDR	Enable MAC Address for PTP Packet Filtering. When the EMAC_TM_CTL.TSENMACADDR bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet. When the EMAC_TM_CTL.TSENMACADDR bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated, when PTP is directly sent over Ethernet. For normal time stamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching. For PTP offload, only MAC address register 0 is considered for unicast destination address matching.	
		0	MAC Address for PTP Packet Filtering is Disabled
		1	MAC Address for PTP Packet Filtering is Enabled
17:16 (R/W)	SNAPTYPSEL	Select PTP Packets for Taking Snapshots. The EMAC_TM_CTL.SNAPTYPSEL field, along with bits 15 and 14, decide the set of PTP packet types for which snapshot needs to be taken.	
15 (R/W)	TSREQENA	Enable Snapshot for Messages Relevant to Requester. When the EMAC_TM_CTL.TSREQENA bit is set, the snapshot is taken only for the messages that are relevant to the requester. Otherwise, the snapshot is taken for the messages relevant to the target node.	
		0	Disable
		1	Enable
14 (R/W)	TSEVNTENA	Enable Time Stamp Snapshot for Event Messages. When the EMAC_TM_CTL.TSEVNTENA bit is set, the time stamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When EMAC_TM_CTL.TSEVNTENA = 0, the snapshot is taken for all messages except Announce, Management, and Signaling.	
		0	Timestamp Snapshot for Event Messages is Disabled
		1	Timestamp Snapshot for Event Messages is Enabled

Table 28-222: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TSIPV4ENA	Enable Processing of PTP Packets Sent Over IPv4-UDP. When the EMAC_TM_CTL.TSIPV4ENA bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When EMAC_TM_CTL.TSIPV4ENA = 0, the MAC ignores the PTP transported over IPv4 UDP packets. The EMAC_TM_CTL.TSIPV4ENA bit is set by default.	
		0	Processing of PTP Packets Sent Over IPv4-UDP is Disabled
		1	Processing of PTP Packets Sent Over IPv4-UDP is Enabled
12 (R/W)	TSIPV6ENA	Enable Processing of PTP Packets Sent Over IPv6-UDP. When the EMAC_TM_CTL.TSIPV6ENA bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When the EMAC_TM_CTL.TSIPV6ENA bit is clear, the MAC ignores the PTP transported over IPv6 UDP packets.	
		0	Processing of PTP Packets Sent Over IPv6-UDP is Disabled
		1	Processing of PTP Packets Sent Over IPv6-UDP is Enabled
11 (R/W)	TSIPENA	Enable Processing of PTP Over Ethernet Packets. When the EMAC_TM_CTL.TSIPENA bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When EMAC_TM_CTL.TSIPENA = 0, the MAC ignores the PTP over Ethernet packets.	
		0	Processing of PTP Over Ethernet Packets is Disabled
		1	Processing of PTP Over Ethernet Packets is Enabled
10 (R/W)	TSVER2ENA	Enable PTP Packet Processing for Version 2 Format. When the EMAC_TM_CTL.TSVER2ENA bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When EMAC_TM_CTL.TSVER2ENA = 0, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.	
		0	PTP Packet Processing for Version 2 Format is Disabled
		1	PTP Packet Processing for Version 2 Format is Enabled

Table 28-222: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	TSCTRLSSR	Time Stamp Digital or Binary Rollover Control. When the EMAC_TM_CTL.TSCTRLSSR bit is set, the time stamp low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the time stamp (high) seconds. When EMAC_TM_CTL.TSCTRLSSR = 0, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of the EMAC_TM_CTL.TSCTRLSSR bit.	
		0	Disable
		1	Enable
8 (R/W)	TSENALL	Enable Time Stamp for All Packets. When the EMAC_TM_CTL.TSENALL bit is set, the time stamp snapshot is enabled for all packets received by the MAC.	
		0	Disable
		1	Enable
6 (R/W)	PTGE	Presentation Time Generation Enable. When the EMAC_TM_CTL.PTGE bit is set, presentation time generation is enabled.	
		0	Presentation Time Generation is Disabled
		1	Presentation Time Generation is Enabled
5 (R/W)	TSADDREG	Update Addend Register. When the EMAC_TM_CTL.TSADDREG bit is set, the content of the EMAC_TM_ADDEND register is updated in the PTP block for fine correction. The EMAC_TM_CTL.TSADDREG bit is cleared when the update is complete. The EMAC_TM_CTL.TSADDREG bit should be zero before it is set. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Addend Register is Not Updated
		1	Addend Register is Updated

Table 28-222: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TSUPDT	Time Stamp Update Enable. When EMAC_TM_CTL.TSUPDT = 1, the system time is updated (added or subtracted) with the value specified in the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers. The EMAC_TM_CTL.TSUPDT bit should be zero before updating it. The EMAC_TM_CTL.TSUPDT bit is reset when the update is complete in hardware. The EMAC_TM_HISEC register (if enabled while configuring the IP) is not updated. When Media Clock Generation and Recovery is enabled, the EMAC_PRSNTM_UPDT register should also be updated before setting this field. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable
2 (R/W)	TSINIT	Initialize Time Stamp. When the EMAC_TM_CTL.TSINIT bit is set, the system time is initialized (overwritten) with the value specified in the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers. The EMAC_TM_CTL.TSINIT bit should be zero before it is updated. The EMAC_TM_CTL.TSINIT bit is cleared when the initialization is complete. The EMAC_TM_HISEC register (if enabled while configuration the IP) can only be initialized. When media clock generation and recovery is enabled the EMAC_PRSNTM_UPDT register must also be updated before setting this field. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable
1 (R/W)	TSCFUPDT	Fine or Coarse Time Stamp Update. When the EMAC_TM_CTL.TSCFUPDT bit is set, the fine method is used to update system time stamp. When EMAC_TM_CTL.TSCFUPDT = 0, coarse method is used to update the system time stamp.	
		0	Coarse Method is Used to Update System Timestamp
		1	Fine Method is Used to Update System Timestamp

Table 28-222: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	TSENA	Enable Time Stamp. When the `EMAC_TM_CTL.TSENA` bit is set, the time stamp is added for transmit and receive packets. When disabled, time stamp is not added for transmit and receive packets and the time stamp generator is also suspended. Initialize the time stamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 packets only if the `EMAC_TM_CTL.TSENA` bit is set.	
		0	Disable
		1	Enable

Time Stamp Egress Asymmetry Correction Register

The EMAC_TM_EGASYM_CORR register contains the egress asymmetry correction value to be used while updating the correction field in PDelay_Req PTP messages.

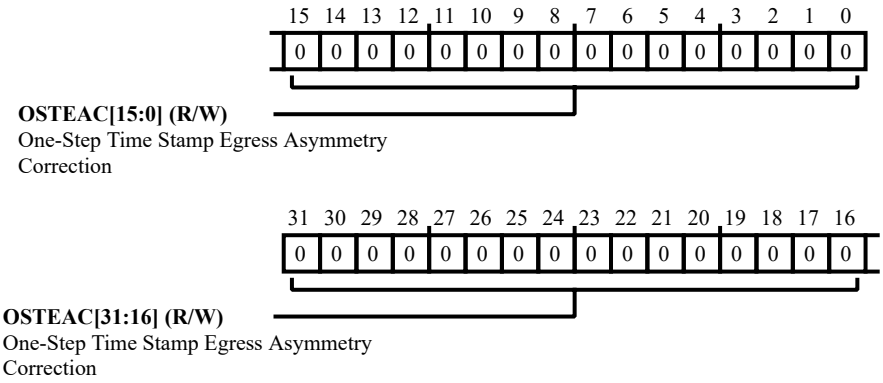


Figure 28-168: EMAC_TM_EGASYM_CORR Register Diagram

Table 28-223: EMAC_TM_EGASYM_CORR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	OSTEAC	One-Step Time Stamp Egress Asymmetry Correction. The EMAC_TM_EGASYM_CORR.OSTEAC field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}. For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD_8000, which is the 2's complement of 0x0002_8000(2.5 * 216). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003_4CCC (3.3 * 216).

Time Stamp Egress Correction Nanosecond Register

The EMAC_TM_EGCORR_NSEC register contains the correction value in nanoseconds to be used with the captured time stamp value in the egress path.

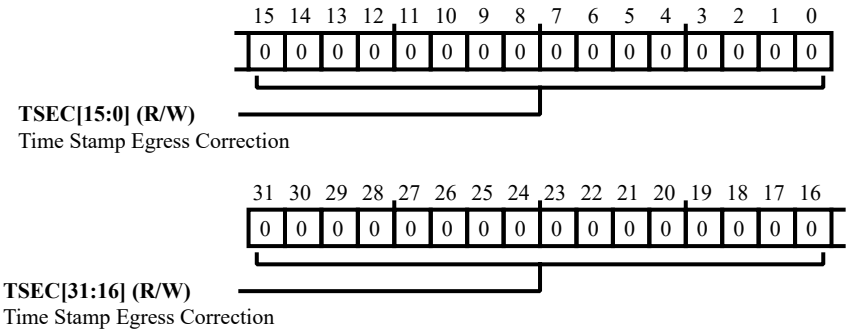


Figure 28-169: EMAC_TM_EGCORR_NSEC Register Diagram

Table 28-224: EMAC_TM_EGCORR_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSEC	Time Stamp Egress Correction. The EMAC_TM_EGCORR_NSEC.TSEC field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Time Stamp Egress Correction Sub Nanosecond Register

The EMAC_TM_EGCORR_SNSEC register contains the sub-nanosecond part of the correction value to be used with the captured time stamp value, for egress direction.

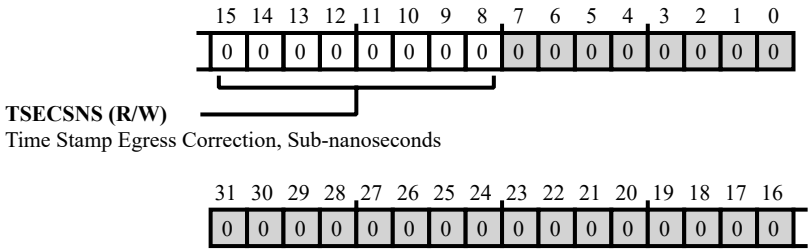


Figure 28-170: EMAC_TM_EGCORR_SNSEC Register Diagram

Table 28-225: EMAC_TM_EGCORR_SNSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	TSECSNS	Time Stamp Egress Correction, Sub-nanoseconds. The EMAC_TM_EGCORR_SNSEC.TSECSNS field contains the sub-nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Time Stamp Egress Latency Register

The EMAC_TM_EG_LTNCY register holds the egress MAC latency.

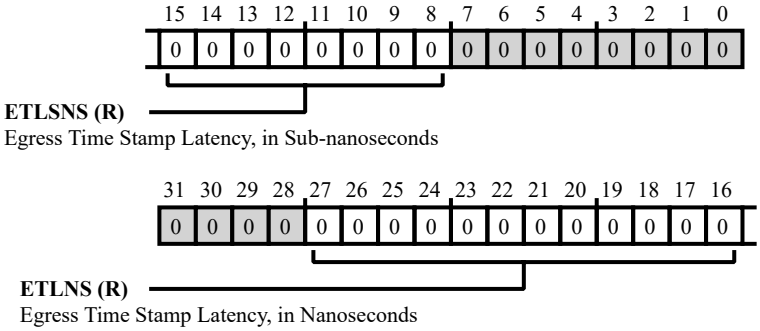


Figure 28-171: EMAC_TM_EG_LTNCY Register Diagram

Table 28-226: EMAC_TM_EG_LTNCY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/NW)	ETLNS	Egress Time Stamp Latency, in Nanoseconds. The EMAC_TM_EG_LTNCY register holds the average latency in nanoseconds between the actual point (GMII/MII) where the egress time stamp is taken and the output ports (ETHx_TXDx) of the MAC.
15:8 (R/NW)	ETLSNS	Egress Time Stamp Latency, in Sub-nanoseconds. The EMAC_TM_EG_LTNCY register holds the average latency in sub-nanoseconds between the actual point (GMII/MII) where the egress time stamp is taken and the output ports (phy_txd_o) of the MAC.

Time Stamp Ingress Asymmetry Correction Register

The EMAC_TM_INGASYM_CORR register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

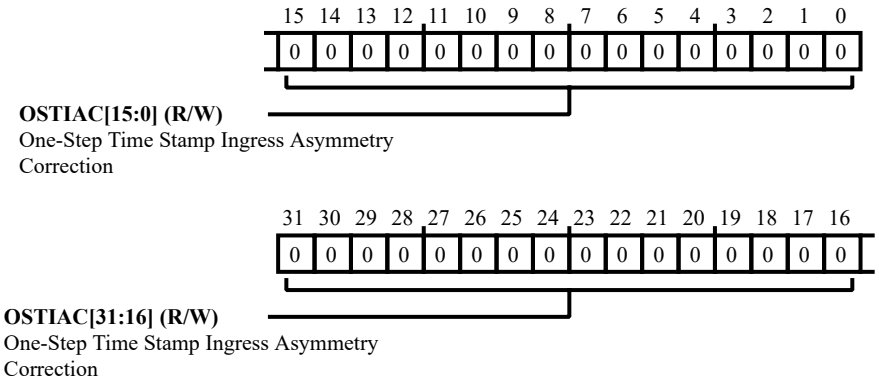


Figure 28-172: EMAC_TM_INGASYM_CORR Register Diagram

Table 28-227: EMAC_TM_INGASYM_CORR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	OSTIAC	One-Step Time Stamp Ingress Asymmetry Correction. The EMAC_TM_INGASYM_CORR.OSTIAC field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Time Stamp Ingress Correction Nanosecond Register

The EMAC_TM_INGCORR_NSEC register contains the correction value in nanoseconds to be used with the captured time stamp value in the ingress path.

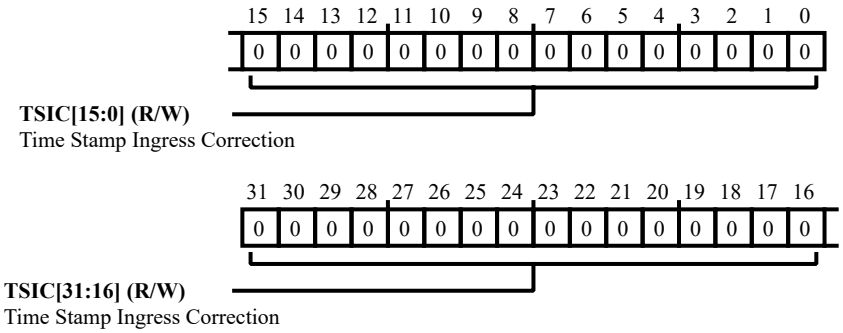


Figure 28-173: EMAC_TM_INGCORR_NSEC Register Diagram

Table 28-228: EMAC_TM_INGCORR_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSIC	Time Stamp Ingress Correction. The EMAC_TM_INGCORR_NSEC.TSIC field contains the ingress path correction value as defined by the Ingress Correction expression.

Time Stamp Ingress Correction Sub Nanosecond Register

The EMAC_TM_INGCORR_SNSEC register contains the sub-nanosecond part of the correction value to be used with the captured time stamp value, for ingress direction.

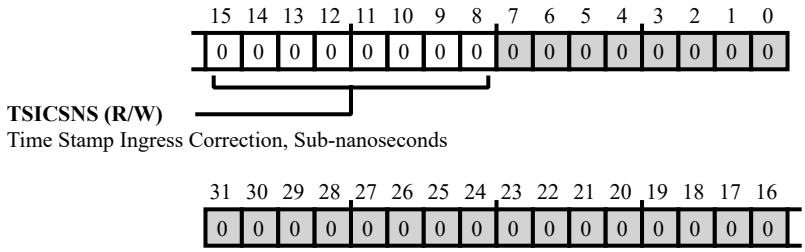


Figure 28-174: EMAC_TM_INGCORR_SNSEC Register Diagram

Table 28-229: EMAC_TM_INGCORR_SNSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	TSICSNS	Time Stamp Ingress Correction, Sub-nanoseconds. The EMAC_TM_INGCORR_SNSEC.TSICSNS field contains the sub-nanoseconds part of the ingress path correction value as defined by the Ingress Correction expression.

Time Stamp Ingress Latency Register

The EMAC_TM_ING_LTNCY register holds the ingress MAC latency.

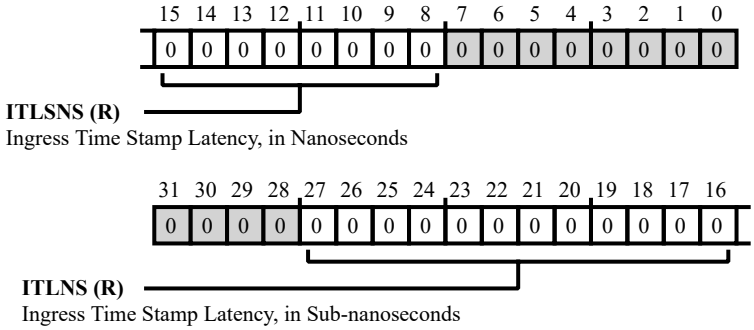


Figure 28-175: EMAC_TM_ING_LTNCY Register Diagram

Table 28-230: EMAC_TM_ING_LTNCY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/NW)	ITLNS	Ingress Time Stamp Latency, in Sub-nanoseconds. The EMAC_TM_ING_LTNCY register holds the average latency in sub-nanoseconds between the input ports (ETHx_RXDx) of MAC and the actual point (GMII/MII) where the ingress time stamp is taken. Ingress correction value is computed as described in the section, "Ingress Correction".
15:8 (R/NW)	ITLSNS	Ingress Time Stamp Latency, in Nanoseconds. The EMAC_TM_ING_LTNCY register holds the average latency in nanoseconds between the input ports (phy_rxd_i) of MAC and the actual point (GMII/MII) where the ingress time stamp is taken. Ingress correction value is computed as described in the section, "Ingress Correction".

Time Stamp Status Register

All bits except bits[27:25] are cleared when the application reads the EMAC_TM_STAT register.

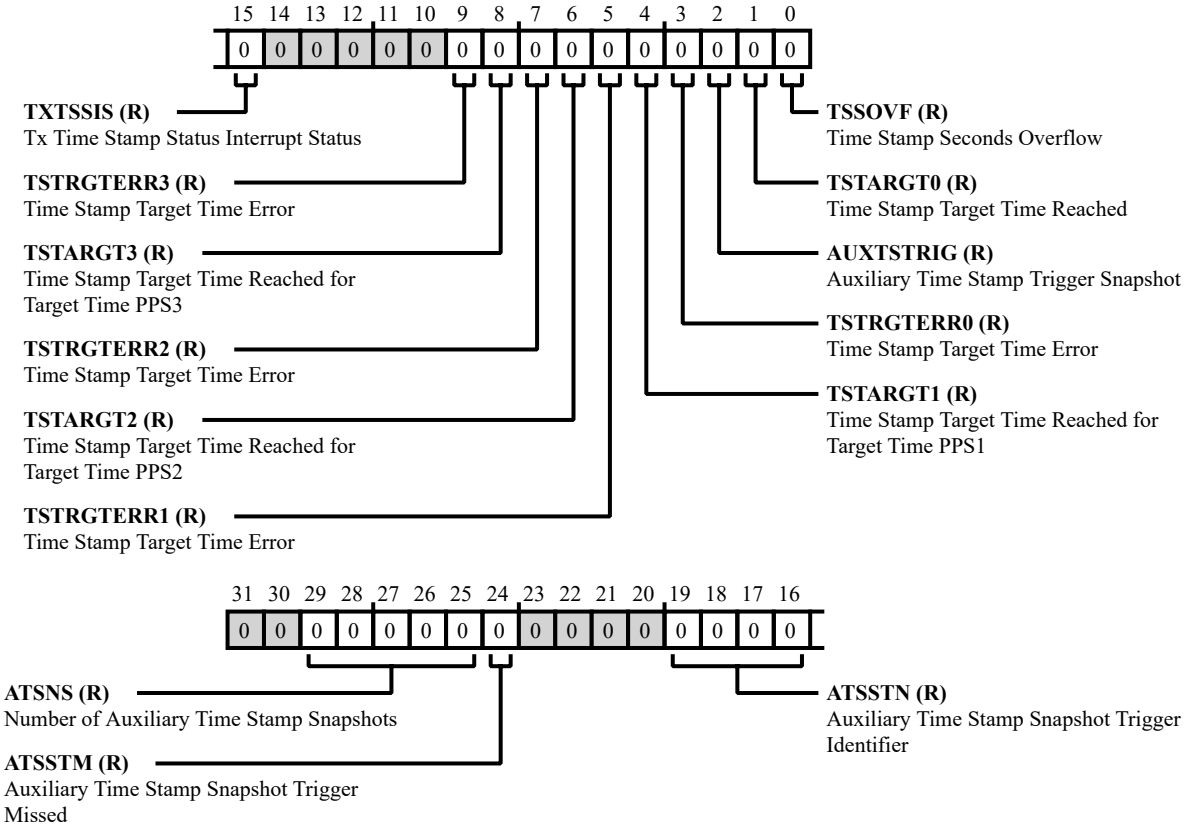


Figure 28-176: EMAC_TM_STAT Register Diagram

Table 28-231: EMAC_TM_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/NW)	ATSNS	Number of Auxiliary Time Stamp Snapshots. The EMAC_TM_STAT.ATSNS field indicates the number of snapshots available in the FIFO. A value equal to the selected depth of FIFO (4, 8, or 16) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set. The EMAC_TM_STAT.ATSNS bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.

Table 28-231: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	ATSSTM	Auxiliary Time Stamp Snapshot Trigger Missed. The EMAC_TM_STAT.ATSSTM bit is set when the auxiliary time stamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO. The EMAC_TM_STAT.ATSSTM bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.	
		0	Status Not Detected
		1	Status Detected
19:16 (R/NW)	ATSSTN	Auxiliary Time Stamp Snapshot Trigger Identifier. The EMAC_TM_STAT.ATSSTN field identifies the auxiliary trigger inputs for which the time stamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list: - Bit 16: Auxiliary trigger 0 - Bit 17: Auxiliary trigger 1 - Bit 18: Auxiliary trigger 2 - Bit 19: Auxiliary trigger 3 The software can read this register to find the triggers that are set when the time stamp is taken.	
15 (R/NW)	TXTSSIS	Tx Time Stamp Status Interrupt Status. When the drop transmit status is enabled in MTL, the EMAC_TM_STAT.TXTSSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers. When PTP offload feature is enabled, the EMAC_TM_STAT.TXTSSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers, for PTO generated Delay Request and Pdelay request packets. The EMAC_TM_STAT.TXTSSIS bit is cleared when the EMAC_TXTMSTAT_SEC register is read (or write to EMAC_TXTMSTAT_SEC register when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	Status Not Detected
		1	Status Detected

Table 28-231: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	TSTRGTERR3	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR3 bit is set when the latest target time programmed in the EMAC_PPS[3]_TGTM_SEC and EMAC_PPS[3]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR3 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR3 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR3 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
8 (R/NW)	TSTARGT3	Time Stamp Target Time Reached for Target Time PPS3. When the EMAC_TM_STAT.TSTARGT3 bit is set and EMAC_PPS_CTL.MCGREN3 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[3]_TGTM_SEC and EMAC_PPS[3]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT3 bit is set and EMAC_PPS_CTL.MCGREN3 = 1, it indicates that EMAC0_MCGR3 trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT3 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
7 (R/NW)	TSTRGTERR2	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR2 bit is set when the latest target time programmed in the EMAC_PPS[2]_TGTM_SEC and EMAC_PPS[2]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR2 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR2 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR2 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Table 28-231: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	TSTARGT2	Time Stamp Target Time Reached for Target Time PPS2. When the EMAC_TM_STAT.TSTARGT2 bit is set and EMAC_PPS_CTL.MCGREN2 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[2]_TGTM_SEC and EMAC_PPS[2]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT2 bit is set and EMAC_PPS_CTL.MCGREN2 = 1, it indicates that EMAC0_MCGR2 trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT2 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
5 (R/NW)	TSTRGTERR1	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR1 bit is set when the latest target time programmed in the EMAC_PPS[1]_TGTM_SEC and EMAC_PPS[1]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR1 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR1 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR1 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
4 (R/NW)	TSTARGT1	Time Stamp Target Time Reached for Target Time PPS1. When the EMAC_TM_STAT.TSTARGT1 bit is set and EMAC_PPS_CTL.MCGREN1 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[1]_TGTM_SEC and EMAC_PPS[1]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT1 bit is set and EMAC_PPS_CTL.MCGREN1 = 1, it indicates that EMAC0_MCGR1 trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT1 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Table 28-231: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	TSTRGTERR0	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR0 bit is set when the latest target time programmed in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR0 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR0 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR0 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
2 (R/NW)	AUXTSTRIG	Auxiliary Time Stamp Trigger Snapshot. The EMAC_TM_STAT.AUXTSTRIG bit is set (=1) when the auxiliary snapshot is written to the FIFO. The EMAC_TM_STAT.AUXTSTRIG bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected. Access restriction applies. Clears on read (or the EMAC_TM_STAT.AUXTSTRIG bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Auxiliary Timestamp Trigger Snapshot Status Not Detected
		1	Auxiliary Timestamp Trigger Snapshot Status Detected
1 (R/NW)	TSTARGT0	Time Stamp Target Time Reached. When the EMAC_TM_STAT.TSTARGT0 bit is set and EMAC_PPS_CTL.MCGREN0 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT0 bit is set and EMAC_PPS_CTL.MCGREN0 = 1, it indicates that EMAC0_MCGR trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT0 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Table 28-231: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	TSSOVF	Time Stamp Seconds Overflow. When the `EMAC_TM_STAT.TSSOVF` bit is set, it indicates that the seconds value of the time stamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. Access restriction applies. Clears on read (or the `EMAC_TM_STAT.TSSOVF` bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Transmit Queue Priority Mapping 0 Register

The EMAC_TXQ_PRTY_MAP0 register contains the priority values assigned to Tx queue 0 through Tx queue 3.

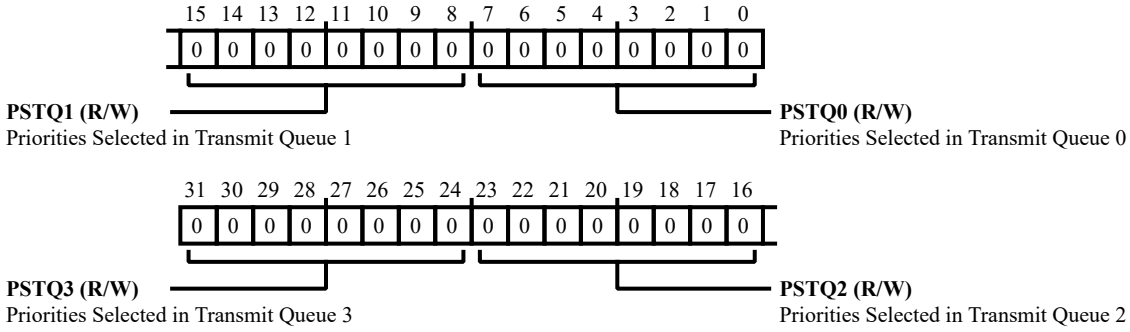


Figure 28-177: EMAC_TXQ_PRTY_MAP0 Register Diagram

Table 28-232: EMAC_TXQ_PRTY_MAP0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSTQ3	Priorities Selected in Transmit Queue 3. The EMAC_TXQ_PRTY_MAP0.PSTQ3 bit is similar to the PSTQ0 bit.
23:16 (R/W)	PSTQ2	Priorities Selected in Transmit Queue 2. The EMAC_TXQ_PRTY_MAP0.PSTQ2 bit is similar to the PSTQ0 bit.
15:8 (R/W)	PSTQ1	Priorities Selected in Transmit Queue 1. The EMAC_TXQ_PRTY_MAP0.PSTQ1 bit is similar to the PSTQ0 bit.
7:0 (R/W)	PSTQ0	Priorities Selected in Transmit Queue 0. The EMAC_TXQ_PRTY_MAP0.PSTQ0 field holds the priorities assigned to Tx queue 0 by the software. The EMAC_TXQ_PRTY_MAP0.PSTQ0 field determines if Tx queue 0 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programmed in this field. If the content of this field is not mutually exclusive to the corresponding fields of other Transmit queues, that is, same priority is mapped to multiple Tx queues, the MAC blocks all queues with matching priority, for the specified time.

Transmit Queue Priority Mapping 1 Register

The EMAC_TXQ_PRTY_MAP1 register contains the priority values assigned to Tx queue 4 through Tx queue 7.

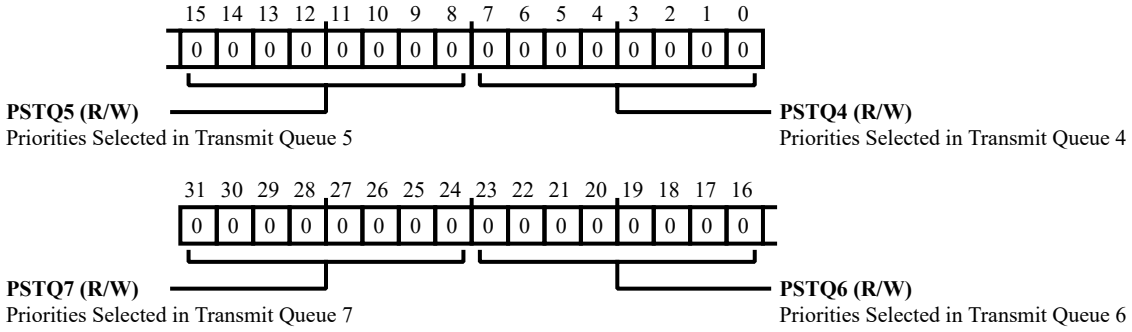


Figure 28-178: EMAC_TXQ_PRTY_MAP1 Register Diagram

Table 28-233: EMAC_TXQ_PRTY_MAP1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSTQ7	Priorities Selected in Transmit Queue 7. The EMAC_TXQ_PRTY_MAP1.PSTQ7 bit is similar to the PSTQ4 bit.
23:16 (R/W)	PSTQ6	Priorities Selected in Transmit Queue 6. The EMAC_TXQ_PRTY_MAP1.PSTQ6 bit is similar to the PSTQ4 bit.
15:8 (R/W)	PSTQ5	Priorities Selected in Transmit Queue 5. The EMAC_TXQ_PRTY_MAP1.PSTQ5 bit is similar to the PSTQ4 bit.
7:0 (R/W)	PSTQ4	Priorities Selected in Transmit Queue 4. The EMAC_TXQ_PRTY_MAP1.PSTQ4 field holds the priorities assigned to Tx queue 4 by the software. The EMAC_TXQ_PRTY_MAP1.PSTQ4 field determines if Tx queue 4 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programmed in this field. If the content of this field is not mutually exclusive to corresponding fields of other Transmit queues, that is, same priority is mapped to multiple Tx queues, the MAC blocks all queues with matching priority for specified time.

MAC Transmit Time Domain Increment Register

The EMAC_MAC_TX_DOMAIN_TIME_INCR register indicates the increment value of the PTP time maintained in transmit clock domain that is used for time stamping.

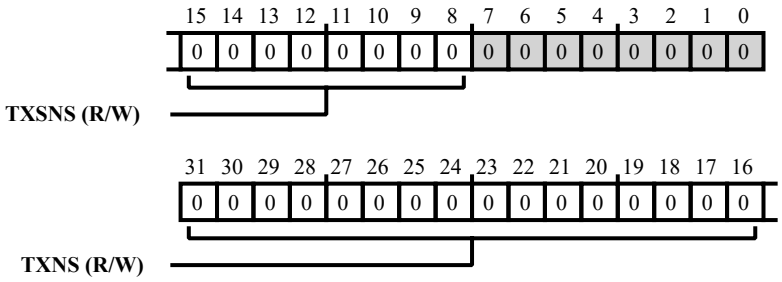


Figure 28-179: EMAC_MAC_TX_DOMAIN_TIME_INCR Register Diagram

Table 28-234: EMAC_MAC_TX_DOMAIN_TIME_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	TXNS	. Transmit Domain Time Increment Value in Nanoseconds. This field indicates the Nanosecond part of the increment value of the PTP time maintained in Transmit clock domain that is used for timestamping. For example, when operating in 2.5G speed the clock period is 3.2ns, so for Digital Rollover mode the TXNS and RXSNS fields must be programmed with 3 and (0.2*256)=51 respectively. In Binary Rollover mode, 3/0.466=6.44 and 0.2+0.44=0.64, so TXNS and TXSNS fields must be programmed with 6 and (0.64*256)=164 respectively. When both TXSNS and TXNS fields are all-zeros, the fixed value based on operating speed is used for PTP time increment in Transmit clock domain.
15:8 (R/W)	TXSNS	. Transmit Domain Time Increment Value in Sub-Nanoseconds. This field indicates the Sub-Nanosecond part of the increment value of the PTP time maintained in Transmit clock domain that is used for timestamping. This field has granularity of 1/256 ns. When both TXSNS and TXNS fields are all-zeros, the fixed value based on operating speed is used for PTP time increment in Transmit clock domain.

Transmit Time Stamp Status Nanoseconds Register

The EMAC_TXTMSTAT_NSEC register contains the nanosecond part of time stamp captured for transmit packets when Tx status is disabled. The EMAC_TXTMSTAT_NSEC register, along with EMAC_TXTMSTAT_SEC, gives the 64-bit time stamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of the EMAC_TXTMSTAT_NSEC register is read. In the little-endian mode, this means when bits[31:24] are read; in big-endian mode, bits[7:0] are read. If the application does not read these registers and time stamp of another packet is captured, then either the current time stamp is lost (overwritten) or the new time stamp is lost (dropped), depending on the configuration of the EMAC_TM_CTL.TXTSSTSM bit. The status bit EMAC_TM_STAT.TXTSSIS is set when the MAC transmitter captures the time stamp.

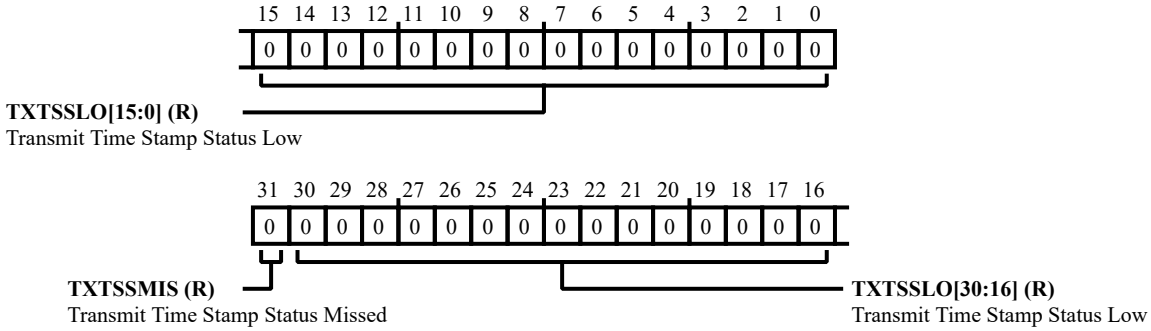


Figure 28-180: EMAC_TXTMSTAT_NSEC Register Diagram

Table 28-235: EMAC_TXTMSTAT_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	TXTSSMIS	Transmit Time Stamp Status Missed. When the EMAC_TXTMSTAT_NSEC.TXTSSMIS bit is set, it indicates one of the following: The time stamp of the current packet is ignored if EMAC_TM_CTL.TXTSSTSM = 0. The time stamp of the previous packet is overwritten with time stamp of the current packet if EMAC_TM_CTL.TXTSSTSM = 1. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
30:0 (R/NW)	TXTSSLO	Transmit Time Stamp Status Low. The EMAC_TXTMSTAT_NSEC.TXTSSLO field contains the 31 bits of the nanoseconds field of the transmit packet's captured time stamp.	

Transmit Time Stamp Status Seconds Register

The EMAC_TXTMSTAT_SEC register contains the higher 32 bits of the time stamp (in seconds) captured when a PTP packet is transmitted.

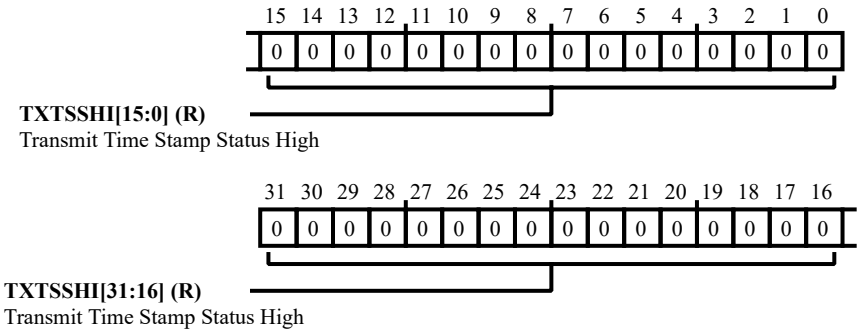


Figure 28-181: EMAC_TXTMSTAT_SEC Register Diagram

Table 28-236: EMAC_TXTMSTAT_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXTSSHI	Transmit Time Stamp Status High. The EMAC_TXTMSTAT_SEC.TXTSSHI field contains the lower 32 bits of the seconds field of transmit packet's captured time stamp.

VLAN Hash Table Register

When EMAC_VLANTAG_CTL.VTHM = 1, the 16-bit EMAC_VLAN_HASHTBL register is used for group address filtering based on the VLAN tag. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on EMAC_VLANTAG_CTL.ETV) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated hash value are used to index the contents of the VLAN hash table. For example, hash value of 4b'1000 selects bit 8 of the VLAN hash table.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).

2. Perform bitwise reversal for the value obtained in step 1.

3. Take the upper four bits from the value obtained in step 2. The VLAN hash table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when bits[15:8] (in little-endian mode) or bits[7:0] (in big-endian mode) of this register are written.

Consecutive writes to this register should be performed after at least four clock cycles in the (G)MII clock domain.

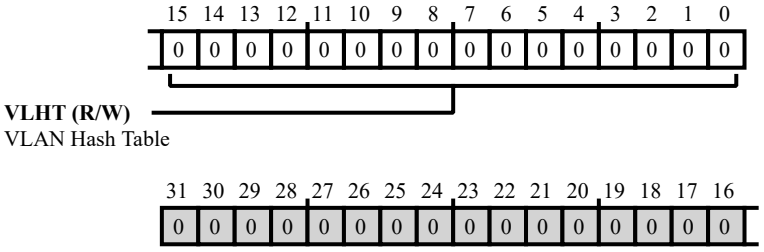


Figure 28-182: EMAC_VLAN_HASHTBL Register Diagram

Table 28-237: EMAC_VLAN_HASHTBL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VLHT	VLAN Hash Table. The EMAC_VLAN_HASHTBL.VLHT field contains the 16-bit VLAN Hash Table.

VLAN Tag Inclusion or Replacement Register

The EMAC_VLAN_INCL register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

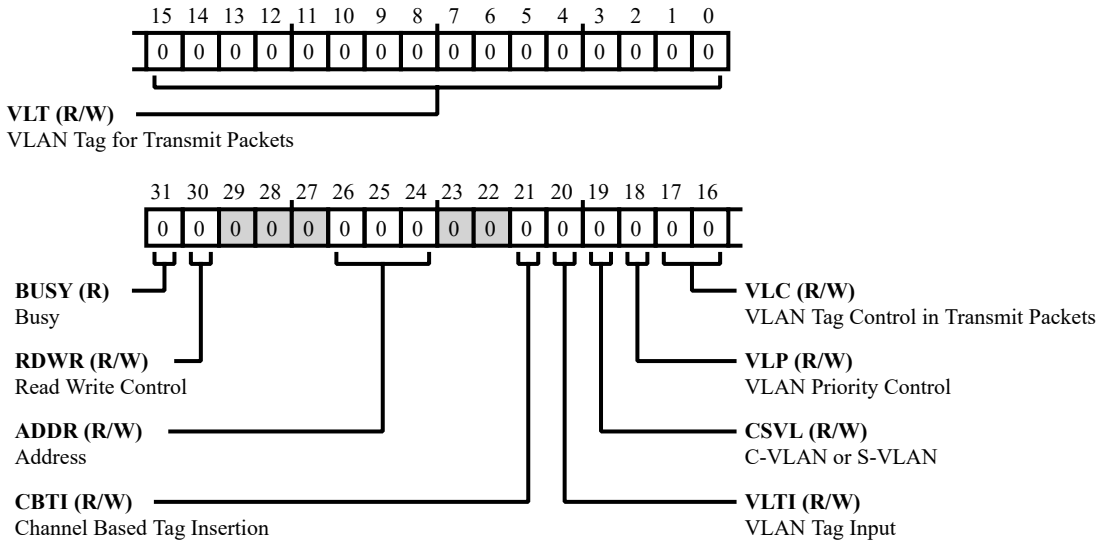


Figure 28-183: EMAC_VLAN_INCL Register Diagram

Table 28-238: EMAC_VLAN_INCL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	BUSY	Busy. The EMAC_VLAN_INCL.BUSY bit indicates the status of the read/write operation of indirect access to the queue/channel specific VLAN inclusion register. For write operation write to a register is complete when EMAC_VLAN_INCL.BUSY = 0. For read operation the read data is valid when the bit is reset. The application must make sure that EMAC_VLAN_INCL.BUSY = 0 before attempting subsequent access to this register.	
		0	Busy Status Not Detected
		1	Busy Status Detected
30 (R/W)	RDWR	Read Write Control. The EMAC_VLAN_INCL.RDWR bit controls the read or write operation for indirectly accessing the queue/channel specific VLAN Inclusion register. When set indicates write operation and when cleared (=0) indicates read operation. This does not have any effect when CBTI is reset.	
		0	Read Operation of Indirect Access
		1	Write Operation of Indirect Access

Table 28-238: EMAC_VLAN_INCL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/W)	ADDR	Address. The EMAC_VLAN_INCL.ADDR field selects one of the queue/channel specific VLAN Inclusion register for read/write access. This does not have any effect when CBTI is reset.	
21 (R/W)	CBTI	Channel Based Tag Insertion. When the EMAC_VLAN_INCL.CBTI bit is set, outer VLAN tag is inserted for every packets transmitted by the MAC. The tag value is taken from the queue/channel specific VLAN tag register. The VLTI, VLP, VLC, and VLT fields of this register are ignored when the EMAC_VLAN_INCL.CBTI bit is set. When the EMAC_VLAN_INCL.CBTI bit is set, a write operation to byte 3 of this register initiates the read/write access to the indirect register. When cleared (=0), outer VLAN operation is based on the setting of VLTI, VLP, VLC and VLT fields of this register.	
		0	Channel Based Tag Insertion is Disabled
		1	Channel Based Tag Insertion is Enabled
20 (R/W)	VLTI	VLAN Tag Input. When the EMAC_VLAN_INCL.VLTI bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from: - The Tx descriptor	
		0	VLAN Tag Input is Disabled
		1	VLAN Tag Input is Enabled
19 (R/W)	CSVL	C-VLAN or S-VLAN. When the EMAC_VLAN_INCL.CSVL bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When EMAC_VLAN_INCL.CSVL = 0, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.	
		0	C-VLAN Type (0x8100) is Inserted or Replaced
		1	S-VLAN Type (0x88A8) is Inserted or Replaced
18 (R/W)	VLP	VLAN Priority Control. When the EMAC_VLAN_INCL.VLP bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When EMAC_VLAN_INCL.VLP = 0, the mti_vlan_ctrl_i control input is used and bits[17:16] are ignored.	
		0	VLAN Priority Control is Disabled
		1	VLAN Priority Control is Enabled

Table 28-238: EMAC_VLAN_INCL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	VLC	VLAN Tag Control in Transmit Packets. The `EMAC_VLAN_INCL.VLC` field indicates the VLAN tag control in transmit packets. 2'b00: No VLAN tag deletion, insertion, or replacement 2'b01: VLAN tag deletion The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags. 2'b10: VLAN tag insertion The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 2'b11: VLAN tag replacement The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8). Note: Changes to this field take effect only on the start of a packet. If this field is written when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.	
		0	No VLAN Tag Deletion, Insertion, or Replacement
		1	VLAN Tag Deletion
		2	VLAN Tag Insertion
		3	VLAN Tag Replacement
15:0 (R/W)	VLT	VLAN Tag for Transmit Packets. The `EMAC_VLAN_INCL.VLT` field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: Bits[15:13]: User Priority Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) Bits[11:0]: VLAN Identifier (VID) field of VLAN tag	

VLAN Tag Control Register

The EMAC_VLANTAG_CTL register contains control and status bits for VLAN filtering operation. It is also used for indirect addressing. It contains the address offset, command type and busy bit for CSR access of the per VLAN tag registers.

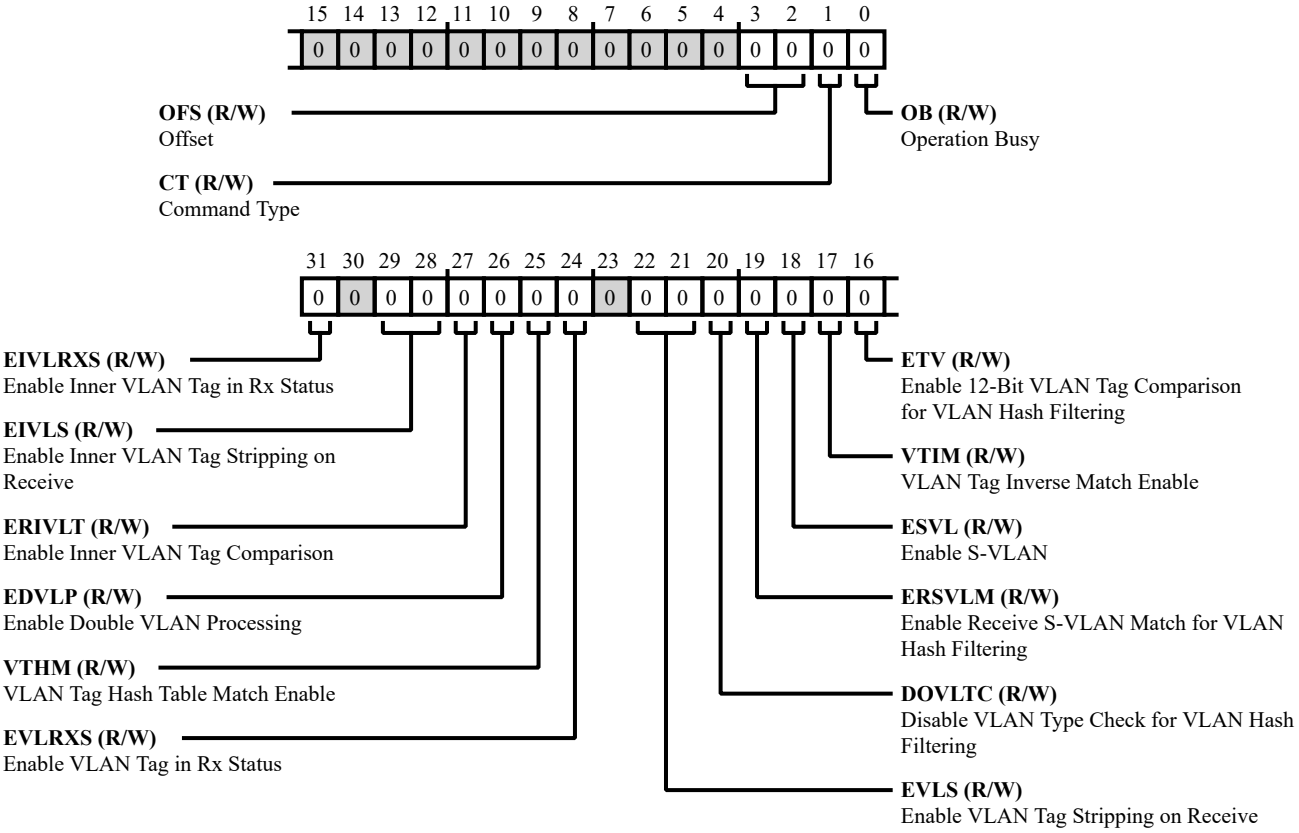


Figure 28-184: EMAC_VLANTAG_CTL Register Diagram

Table 28-239: EMAC_VLANTAG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EIVLRXS	Enable Inner VLAN Tag in Rx Status. When the EMAC_VLANTAG_CTL.EIVLRXS bit is set, the MAC provides the inner VLAN tag in the Rx status. When EMAC_VLANTAG_CTL.EIVLRXS = 0, the MAC does not provide the inner VLAN tag in Rx status.	
		0	Inner VLAN Tag in Rx Status is Disabled
		1	Inner VLAN Tag in Rx Status is Enabled

Table 28-239: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/W)	EIVLS	Enable Inner VLAN Tag Stripping on Receive. The EMAC_VLANTAG_CTL.EIVLS field indicates the stripping operation on inner VLAN tag in received packet.	
		0	Do Not Strip
		1	Strip If VLAN Filter Passes
		2	Strip If VLAN Filter Fails
		3	Always Strip
27 (R/W)	ERIVLT	Enable Inner VLAN Tag Comparison. When the EMAC_VLANTAG_CTL.ERIVLT bit, EMAC_VLANTAG_CTL.VTHM bit, and the EMAC_VLANTAG_CTL.EDVLP field are set, the MAC receiver enables VLAN hash filtering operation on the inner VLAN tag (if present). When EMAC_VLANTAG_CTL.ERIVLT = 0 and EMAC_VLANTAG_CTL.VTHM = 1, the MAC receiver enables VLAN hash filtering operation on the outer VLAN tag (if present). The EMAC_VLANTAG_CTL.ERSVLM bit and EMAC_VLANTAG_CTL.DOVLTC bit determines which VLAN type is enabled for filtering.	
		0	Inner VLAN Tag is Disabled
		1	Inner VLAN Tag is Enabled
26 (R/W)	EDVLP	Enable Double VLAN Processing. When the EMAC_VLANTAG_CTL.EDVLP bit is set, the MAC enables processing of up to two VLAN tags on Tx and Rx (if present). When EMAC_VLANTAG_CTL.EDVLP = 0, the MAC enables processing of up to one VLAN tag on Tx and Rx (if present).	
		0	Double VLAN Processing is Disabled
		1	Double VLAN Processing is Enabled

Table 28-239: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	VTHM	VLAN Tag Hash Table Match Enable. When the EMAC_VLANTAG_CTL.VTHM bit is set, the most significant four bits of CRC of VLAN tag are used to index the content of the EMAC_VLAN_HASHTBL_REGn. A value of 1 in the VLAN hash table register, corresponding to the index, indicates that the packet matched the VLAN hash table. When EMAC_VLANTAG_CTL.ETV= 1, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When EMAC_VLANTAG_CTL.ETV= 0, the CRC of the 16-bit VLAN tag is used for comparison. When EMAC_VLANTAG_CTL.VTHM = 0, the VLAN Hash match operation is not performed.	
		0	VLAN Tag Hash Table Match is Disabled
		1	VLAN Tag Hash Table Match is Enabled
24 (R/W)	EVLRXS	Enable VLAN Tag in Rx Status. When the EMAC_VLANTAG_CTL.EVLRXS bit is set, MAC provides the outer VLAN Tag in the Rx status. When EMAC_VLANTAG_CTL.EVLRXS = 0, the MAC does not provide the outer VLAN tag in Rx status.	
		0	VLAN Tag in Rx Status is Disabled
		1	VLAN Tag in Rx Status is Enabled
22:21 (R/W)	EVLS	Enable VLAN Tag Stripping on Receive. The EMAC_VLANTAG_CTL.EVLS field indicates the stripping operation on the outer VLAN tag in received packet.	
		0	Do Not Strip
		1	Strip If VLAN Filter Passes
		2	Strip If VLAN Filter Fails
		3	Always Strip
20 (R/W)	DOVLTC	Disable VLAN Type Check for VLAN Hash Filtering. When the EMAC_VLANTAG_CTL.DOVLTC bit is set, the MAC VLAN hash filter does not check whether the VLAN tag specified by the EMAC_VLANTAG_CTL.ERIVLT bit is of type S-VLAN or C-VLAN. When EMAC_VLANTAG_CTL.DOVLTC = 0, the MAC VLAN hash filter filters or matches the VLAN tag specified by the EMAC_VLANTAG_CTL.ERIVLT bit only when VLAN tag type is similar to the one specified by the EMAC_VLANTAG_CTL.ERSVLM bit.	
		0	VLAN Type Check is Enabled
		1	VLAN Type Check is Disabled

Table 28-239: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	ERSVLM	Enable Receive S-VLAN Match for VLAN Hash Filtering. When the EMAC_VLANTAG_CTL.ERSVLM bit is set, the MAC receiver enables VLAN hash filtering or matching for S-VLAN (type = 0x88A8) packets. When EMAC_VLANTAG_CTL.ERSVLM = 0, the MAC receiver enables VLAN hash filtering or matching for C-VLAN (type = 0x8100) packets. The EMAC_VLANTAG_CTL.ERIVLT bit determines the VLAN tag position considered for VLAN hash filtering or matching.	
		0	Receive S-VLAN Match is Disabled
		1	Receive S-VLAN Match is Enabled
18 (R/W)	ESVL	Enable S-VLAN. When the EMAC_VLANTAG_CTL.ESVL bit is set, the MAC transmitter and receiver consider the S-VLAN packets (type = 0x88A8) as valid VLAN tagged packets.	
		0	S-VLAN is Disabled
		1	S-VLAN is Enabled
17 (R/W)	VTIM	VLAN Tag Inverse Match Enable. When the EMAC_VLANTAG_CTL.VTIM bit is set, the EMAC_VLANTAG_CTL.VTIM bit enables the VLAN tag inverse matching. The packets without matching VLAN tag are marked as matched. When cleared (=0), the EMAC_VLANTAG_CTL.VTIM bit enables the VLAN tag perfect matching. The packets with matched VLAN tag are marked as matched.	
		0	VLAN Tag Inverse Match is Disabled
		1	VLAN Tag Inverse Match is Enabled
16 (R/W)	ETV	Enable 12-Bit VLAN Tag Comparison for VLAN Hash Filtering. When the EMAC_VLANTAG_CTL.ETV bit is set, a 12-bit VLAN identifier is used for VLAN hash filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag in the received VLAN-tagged packet are used for hash-based VLAN filtering. When EMAC_VLANTAG_CTL.ETV = 0, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for VLAN hash filtering.	
		0	12-Bit VLAN Tag Comparison is Disabled
		1	12-Bit VLAN Tag Comparison is Enabled
3:2 (R/W)	OFS	Offset. The EMAC_VLANTAG_CTL.OFS field holds the address offset of the MAC VLAN Tag Filter Register which the application is trying to access. The width of the field depends on the number of MAC VLAN tag registers enabled.	

Table 28-239: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	CT	Command Type. The EMAC_VLANTAG_CTL.CT bit indicates if the current register access is a read or a write. When set, the EMAC_VLANTAG_CTL.CT bit indicates a read operation. When cleared (=0), it indicates a write operation.	
		0	Write Operation
		1	Read Operation
0 (R/W)	OB	Operation Busy. The EMAC_VLANTAG_CTL.OB bit is set along with a read or write command for initiating the indirect access to per VLAN Tag Filter register. The EMAC_VLANTAG_CTL.OB bit is reset when the read or write command to per VLAN Tag Filter indirect access register is complete. The next indirect register access can be initiated only after EMAC_VLANTAG_CTL.OB = 0. During a write operation, the bit is reset only after the data has been written into the Per VLAN Tag register. During a read operation, the data should be read from the EMAC_VLANTAG_DAT register only after EMAC_VLANTAG_CTL.OB = 0.	
		0	Operation Busy is Disabled
		1	Operation Busy is Enabled

VLAN Tag Indirect Access Data Register

The EMAC_VLANTAG_DAT register holds the read/write data for Indirect Access of the Per VLAN Tag registers. During the read access, this field contains valid read data only after the OB bit is reset. During the write access, this field should be valid prior to EMAC_VLANTAG_CTL.OB = 1

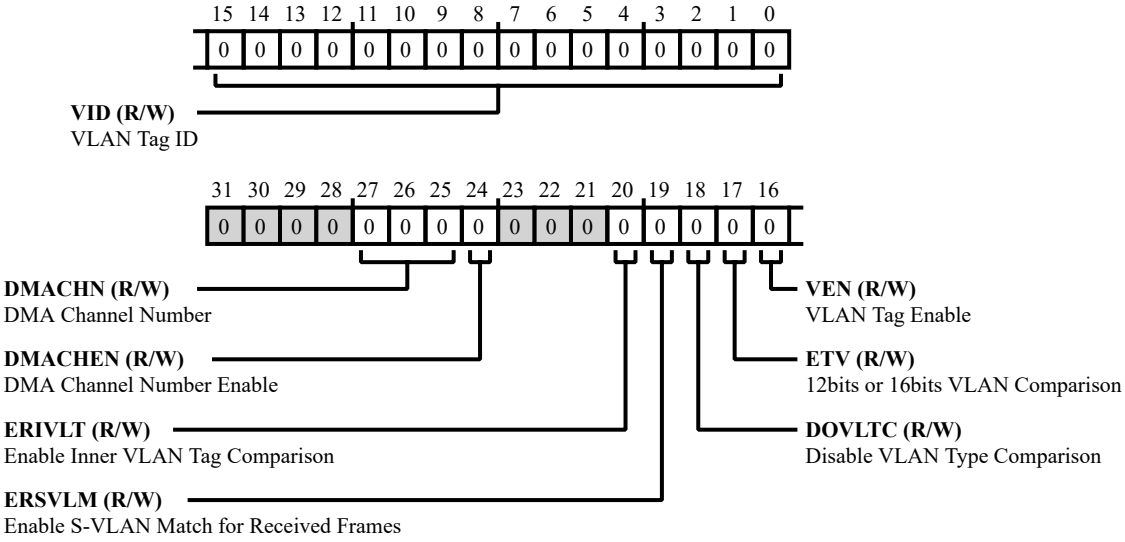


Figure 28-185: EMAC_VLANTAG_DAT Register Diagram

Table 28-240: EMAC_VLANTAG_DAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27:25 (R/W)	DMACHN	DMA Channel Number. The EMAC_VLANTAG_DAT.DMACHN field indicates the DMA channel number to which the VLAN tagged frame is to be routed, when the routing is based on the VLAN tag filter. If the routing is not based on the VLAN tag filter, this field does not need to be programmed.	
24 (R/W)	DMACHEN	DMA Channel Number Enable. The EMAC_VLANTAG_DAT.DMACHEN bit is the enable for the DMA channel number value programmed in the field DMACH. When EMAC_VLANTAG_DAT.DMACHEN = 0, the routing does not occur based on VLAN filter result. The frame is routed based on DA-based DMA channel routing.	
		0	DMA Channel Number is Disabled
		1	DMA Channel Number is Enabled

Table 28-240: EMAC_VLANTAG_DAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	ERIVLT	Enable Inner VLAN Tag Comparison. The EMAC_VLANTAG_DAT.ERIVLT bit is valid only when the Double VLAN Tag Enable of the filter is set. When the EMAC_VLANTAG_DAT.ERIVLT bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN tag (if present). When EMAC_VLANTAG_DAT.ERIVLT = 0, the MAC receiver enables operation on the outer VLAN tag (if present).	
		0	Inner VLAN Tag Comparison is Disabled
		1	Inner VLAN Tag Comparison is Enabled
19 (R/W)	ERSVLM	Enable S-VLAN Match for Received Frames. The EMAC_VLANTAG_DAT.ERSVLM bit is valid only when VLAN Tag enable of the filter is set. When the EMAC_VLANTAG_DAT.ERSVLM bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When EMAC_VLANTAG_DAT.ERSVLM = 0, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.	
		0	Receive S-VLAN Match is Disabled
		1	Receive S-VLAN Match is Enabled
18 (R/W)	DOVLTC	Disable VLAN Type Comparison. The EMAC_VLANTAG_DAT.DOVLTC bit is valid only when VLAN tag enable of the filter is set. When the EMAC_VLANTAG_DAT.DOVLTC bit is set, the MAC does not check whether the VLAN tag specified by the Enable Inner VLAN Tag Comparison bit is of type S-VLAN or C-VLAN. When EMAC_VLANTAG_DAT.DOVLTC = 0, the MAC filters or matches the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit only when VLAN Tag type is similar to the one specified by the Enable S-VLAN match for received frames bit.	
		0	VLAN Type Comparison is Enabled
		1	VLAN Type Comparison is Disabled

Table 28-240: EMAC_VLANTAG_DAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	ETV	12bits or 16bits VLAN Comparison. The EMAC_VLANTAG_DAT.ETV bit is valid only when VEN of the filter is set. When the EMAC_VLANTAG_DAT.ETV bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet.	
		0	16 Bit VLAN Comparison
		1	12 Bit VLAN Comparison
16 (R/W)	VEN	VLAN Tag Enable. The EMAC_VLANTAG_DAT.VEN bit is used to enable or disable the VLAN tag. When the EMAC_VLANTAG_DAT.VEN bit is set, the MAC compares the VLAN tag of received packet with the VLAN Tag ID. When EMAC_VLANTAG_DAT.VEN = 0, no comparison is performed irrespective of the programming of the other fields.	
		0	VLAN Tag is Disabled
		1	VLAN Tag is Enabled
15:0 (R/W)	VID	VLAN Tag ID. The EMAC_VLANTAG_DAT.VID field holds the VLAN tag value which is used by the MAC for perfect comparison. It is valid when VLAN Tag Enable is set.	

Watchdog and Jabber Timeout Register

The Watchdog and Jabber Timeout register controls the watchdog timeout limit for received packets and jabber timeout limit for transmitted packets.

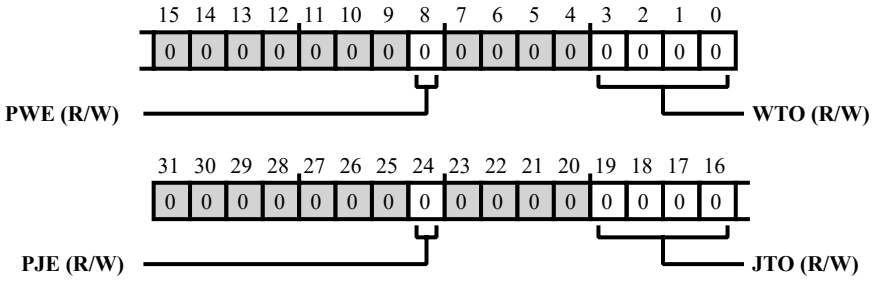


Figure 28-186: EMAC_WDT_JB_TMOUT Register Diagram

Table 28-241: EMAC_WDT_JB_TMOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	PJE	. Programmable Jabber Enable When this bit is set and the JD bit of the MAC_Configuration register is reset, the JTO field is used as jabber timeout limit for a transmitted packet. When this bit is cleared, the jabber timeout for a received packet is controlled by setting of JD and JE bits in MAC_Configuration register.	
		0	Programmable Watchdog is disabled.
		1	Programmable Watchdog is enabled.
19:16 (R/W)	JTO	. Jabber Timeout When the PJE bit is set and the JD bit of the MAC_Configuration register is reset, this field is used as jabber timeout limit for a transmitted packet. If the length of a transmitted packet exceeds the value of this field, such packet is terminated and declared as an error packet. Note: When the PJE bit is set, the value in this field must be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and terminated.	
		0	2 KB.
		1	3 KB.
		2	4 KB.
		3	5 KB.
		4	6 KB.
		5	7 KB.
		6	8 KB.
		7	9 KB.

Table 28-241: EMAC_WDT_JB_TMOUT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		8	10 KB.
		9	11 KB.
		10	12 KB.
		11	13 KB.
		12	14 KB.
		13	15 KB.
		14	16383 Bytes.
		15	Reserved
8 (R/W)	PWE	. Programmable Watchdog Enable When this bit is set and the WD bit of the MAC_Configuration register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in MAC_Configuration register.	
		0	Programmable Watchdog is disabled.
		1	Programmable Watchdog is enabled.
3:0 (R/W)	WTO	. Watchdog Timeout When the PWE bit is set and the WD bit of the MAC_Configuration register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.	
		0	2 KB.
		1	3 KB.
		2	4 KB.
		3	5 KB.
		4	6 KB.
		5	7 KB.
		6	8 KB.
		7	9 KB.
		8	10 KB.
		9	11 KB.
		10	12 KB.

Table 28-241: EMAC_WDT_JB_TMOUT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		11	13 KB.
		12	14 KB.
		13	15 KB.
		14	16383 Bytes.
		15	Reserved

MMC Control Register

The EMAC_MMC_CTRL register establishes the operating mode of MMC.

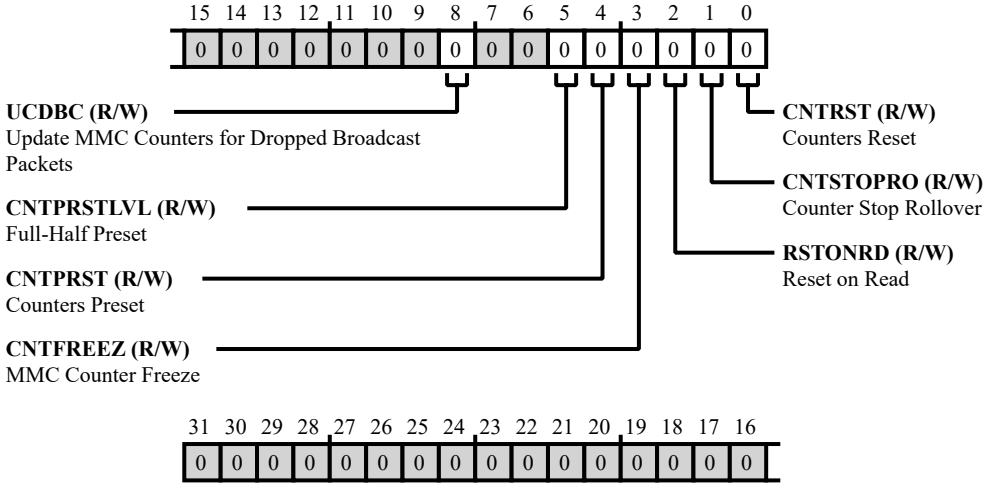


Figure 28-187: EMAC_MMC_CTRL Register Diagram

Table 28-242: EMAC_MMC_CTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	UCDBC	Update MMC Counters for Dropped Broadcast Packets. Note: The EMAC_MMC_CTRL.CNTRST bit has a higher priority than the EMAC_MMC_CTRL.CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the EMAC_MMC_CTRL.CNTPRST bit is not set. When EMAC_MMC_CTRL.UCDBC is set (=1), the MAC updates all related MMC counters for broadcast packets that are dropped because of the EMAC_MACPKT_FILT.DBF bit. When EMAC_MMC_CTRL.UCDBC is cleared (=0), the MMC counters are not updated for dropped broadcast packets.	
		0	Update MMC Counters for Dropped Broadcast Packets is Disabled
		1	Update MMC Counters for Dropped Broadcast Packets is Enabled

Table 28-242: EMAC_MMC_CTRL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	CNTPRSTLVL	Full-Half Preset. When EMAC_MMC_CTRL.CNTPRSTLVL = 0 and EMAC_MMC_CTRL.CNTPRST = 1, all MMC counters are preset to almost-half value. All octet counters are preset to 0x7FFF_F800 (Half 2KB) and all packet-counters gets preset to 0x7FFF_FFF0 (Half 16). When EMAC_MMC_CTRL.CNTPRSTLVL= 1 and EMAC_MMC_CTRL.CNTPRST = 1, all MMC counters are preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (Full 2KB) and all packet-counters gets preset to 0xFFFF_FFF0 (Full 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.	
		0	Full-Half Preset is Disabled
		1	Full-Half Preset is Enabled
4 (R/W)	CNTPRST	Counters Preset. When the EMAC_MMC_CTRL.CNTPRST bit is set, all counters are initialized or preset to almost full or almost half according to the EMAC_MMC_CTRL.CNTPRSTLVL bit. The EMAC_MMC_CTRL.CNTPRST bit is cleared automatically after 1 clock cycle. The EMAC_MMC_CTRL.CNTPRST bit, along with the EMAC_MMC_CTRL.CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. Access restriction applies. Self-cleared. Write 0 to clear. Write 1 to set.	
		0	Counters Preset is Disabled
		1	Counters Preset is Enabled
3 (R/W)	CNTFREEZ	MMC Counter Freeze. When the EMAC_MMC_CTRL.CNTFREEZ bit is set (=1), it freezes all MMC counters to their current value. Until EMAC_MMC_CTRL.CNTFREEZ = 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the reset on read bit set, then that counter is also cleared in this mode.	
		0	MMC Counter Freeze is Disabled
		1	MMC Counter Freeze is Enabled

Table 28-242: EMAC_MMC_CTRL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	RSTONRD	Reset on Read. When the `EMAC_MMC_CTRL.RSTONRD` bit is set (=1), the MMC counters are reset to zero after read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.	
		0	Reset on Read is Disabled
		1	Reset on Read is Enabled
1 (R/W)	CNTSTOPRO	Counter Stop Rollover. When the `EMAC_MMC_CTRL.CNTSTOPRO` bit is set (=1), the counter does not roll over to zero after reaching the maximum value.	
		0	Counter Stop Rollover is Disabled
		1	Counter Stop Rollover is Enabled
0 (R/W)	CNTRST	Counters Reset. When the `EMAC_MMC_CTRL.CNTRST` bit is set (=1), all counters are reset. The `EMAC_MMC_CTRL.CNTRST` bit is cleared automatically after 1 clock cycle. Access restriction applies. Self-cleared. Write 0 to clear. Write 1 to set.	
		0	Counters are Not Reset
		1	All Counters are Reset

MMC Receive FPE Interrupt Status Register

The EMAC_MMC_FPE_RXINT register maintains the interrupts generated from all FPE-related receive statistics counters. The EMAC_MMC_FPE_RXINT register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for the 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The EMAC_MMC_FPE_RXINT register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

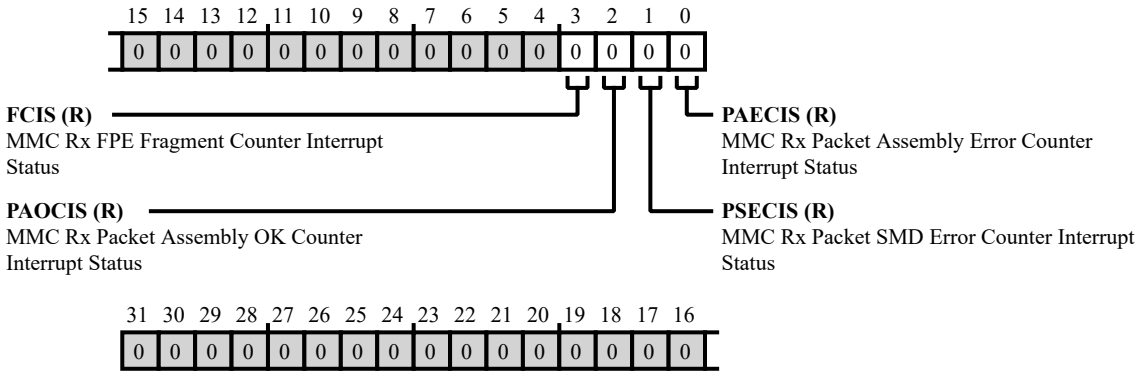


Figure 28-188: EMAC_MMC_FPE_RXINT Register Diagram

Table 28-243: EMAC_MMC_FPE_RXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	FCIS	MMC Rx FPE Fragment Counter Interrupt Status. The EMAC_MMC_FPE_RXINT.FCIS bit is set when the EMAC_RXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx FPE Fragment Counter Interrupt Status Not Detected
		1	MMC Rx FPE Fragment Counter Interrupt Status Detected

Table 28-243: EMAC_MMC_FPE_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	PAOCIS	MMC Rx Packet Assembly OK Counter Interrupt Status. The `EMAC_MMC_FPE_RXINT.PAOCIS` bit is set when the EMAC_RXPKT_ASOK_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet Assembly OK Counter Interrupt Status Not Detected
		1	MMC Rx Packet Assembly OK Counter Interrupt Status Detected
1 (R/NW)	PSECIS	MMC Rx Packet SMD Error Counter Interrupt Status. The `EMAC_MMC_FPE_RXINT.PSECIS` bit is set when the EMAC_RXPKT_SMDERR_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet SMD Error Counter Interrupt Status Not Detected
		1	MMC Rx Packet SMD Error Counter Interrupt Status Detected
0 (R/NW)	PAECIS	MMC Rx Packet Assembly Error Counter Interrupt Status. The `EMAC_MMC_FPE_RXINT.PAECIS` bit is set when the EMAC_RXPKT_ASERR_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet Assembly Error Counter Interrupt Status Not Detected
		1	MMC Rx Packet Assembly Error Counter Interrupt Status Detected

MMC Receive FPE Interrupt Mask Register

The EMAC_MMC_FPE_RXIMSK register maintains the masks for interrupts generated from all FPE-related receive statistics counters. The EMAC_MMC_RXIMSK register maintains the masks for the interrupts generated when FPE-related receive statistic counters reach half of their maximum value or the maximum values. The EMAC_MMC_FPE_RXIMSK register is 32 bit wide.

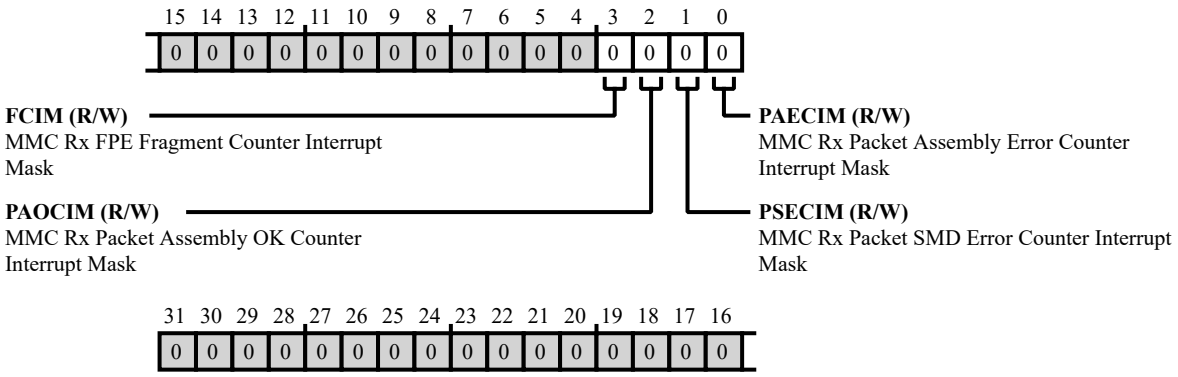


Figure 28-189: EMAC_MMC_FPE_RXIMSK Register Diagram

Table 28-244: EMAC_MMC_FPE_RXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FCIM	MMC Rx FPE Fragment Counter Interrupt Mask. Setting the EMAC_MMC_FPE_RXIMSK.FCIM bit masks the interrupt when the EMAC_TXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx FPE Fragment Counter Interrupt Mask is Disabled
		1	MMC Rx FPE Fragment Counter Interrupt Mask is Enabled

Table 28-244: EMAC_MMC_FPE_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	PAOCIM	MMC Rx Packet Assembly OK Counter Interrupt Mask. Setting the `EMAC_MMC_FPE_RXIMSK.PAOCIM` bit masks the interrupt when the EMAC_RXPKT_ASOK_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.	
		0	MMC Rx Packet Assembly OK Counter Interrupt Mask is Disabled
		1	MMC Rx Packet Assembly OK Counter Interrupt Mask is Enabled
1 (R/W)	PSECIM	MMC Rx Packet SMD Error Counter Interrupt Mask. Setting the `EMAC_MMC_FPE_RXIMSK.PSECIM` bit masks the interrupt when the EMAC_RXPKT_SMDERR_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet SMD Error Counter Interrupt Mask is Disabled
		1	MMC Rx Packet SMD Error Counter Interrupt Mask is Enabled
0 (R/W)	PAECIM	MMC Rx Packet Assembly Error Counter Interrupt Mask. Setting the `EMAC_MMC_FPE_RXIMSK.PAECIM` bit masks the interrupt when the EMAC_RXPKT_ASERR_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet Assembly Error Counter Interrupt Mask is Disabled
		1	MMC Rx Packet Assembly Error Counter Interrupt Mask is Enabled

MMC Transmit FPE Interrupt Status Register

The EMAC_MMC_FPE_TXINT register maintains the interrupts generated from all FPE related Transmit statistics counters. The MMC FPE Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones. The MMC FPE Transmit Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

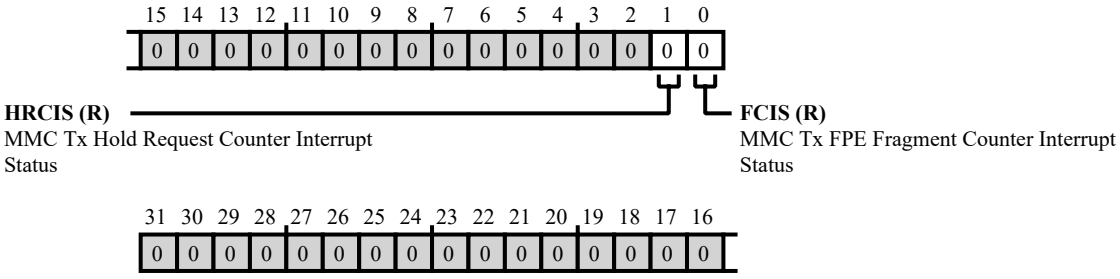


Figure 28-190: EMAC_MMC_FPE_TXINT Register Diagram

Table 28-245: EMAC_MMC_FPE_TXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	HRCIS	MMC Tx Hold Request Counter Interrupt Status. The EMAC_MMC_FPE_TXINT.HRCIS bit is set when the EMAC_TXHLDRQ_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE with AV_EST Enabled configuration.	
		0	MMC Tx Hold Request Counter Interrupt Status Not Detected
		1	MMC Tx Hold Request Counter Interrupt Status Detected

Table 28-245: EMAC_MMC_FPE_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	FCIS	MMC Tx FPE Fragment Counter Interrupt Status. The `EMAC_MMC_FPE_TXINT.FCIS` bit is set when the EMAC_TXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Tx FPE Fragment Counter Interrupt Status Not Detected
		1	MMC Tx FPE Fragment Counter Interrupt Status Detected

MMC Transmit FPE Interrupt Mask Register

The EMAC_MMC_FPE_TXIMSK register maintains the masks for interrupts generated from all FPE related Transmit statistics counters. The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when FPE related receive statistic counters reach half of their maximum value or the maximum values. The EMAC_MMC_FPE_TXIMSK register is 32 bit wide.

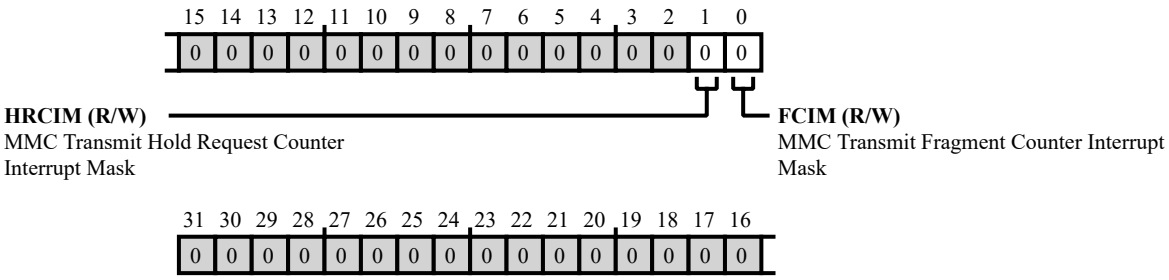


Figure 28-191: EMAC_MMC_FPE_TXIMSK Register Diagram

Table 28-246: EMAC_MMC_FPE_TXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	HRCIM	MMC Transmit Hold Request Counter Interrupt Mask. Setting the EMAC_MMC_FPE_TXIMSK.HRCIM bit masks the interrupt when the EMAC_TXHLDRQ_CNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Hold Request Counter Interrupt Mask is Disabled
		1	MMC Transmit Hold Request Counter Interrupt Mask is Enabled
0 (R/W)	FCIM	MMC Transmit Fragment Counter Interrupt Mask. Setting the EMAC_MMC_FPE_TXIMSK.FCIM bit masks the interrupt when the EMAC_TXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Transmit Fragment Counter Interrupt Mask is Disabled
		1	MMC Transmit Fragment Counter Interrupt Mask is Enabled

MMC Receive IPC Interrupt Status Register

The EMAC_MMC_IPC_RXINT register maintains the interrupt that the receive IPC statistic counters generate.

The EMAC_MMC_IPC_RXINT register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32 bit counter and 0xFFFF for 16 bit counter). When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The EMAC_MMC_IPC_RXINT register is 32 bit wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (Bits[7:0]) must be read to clear the interrupt bit.

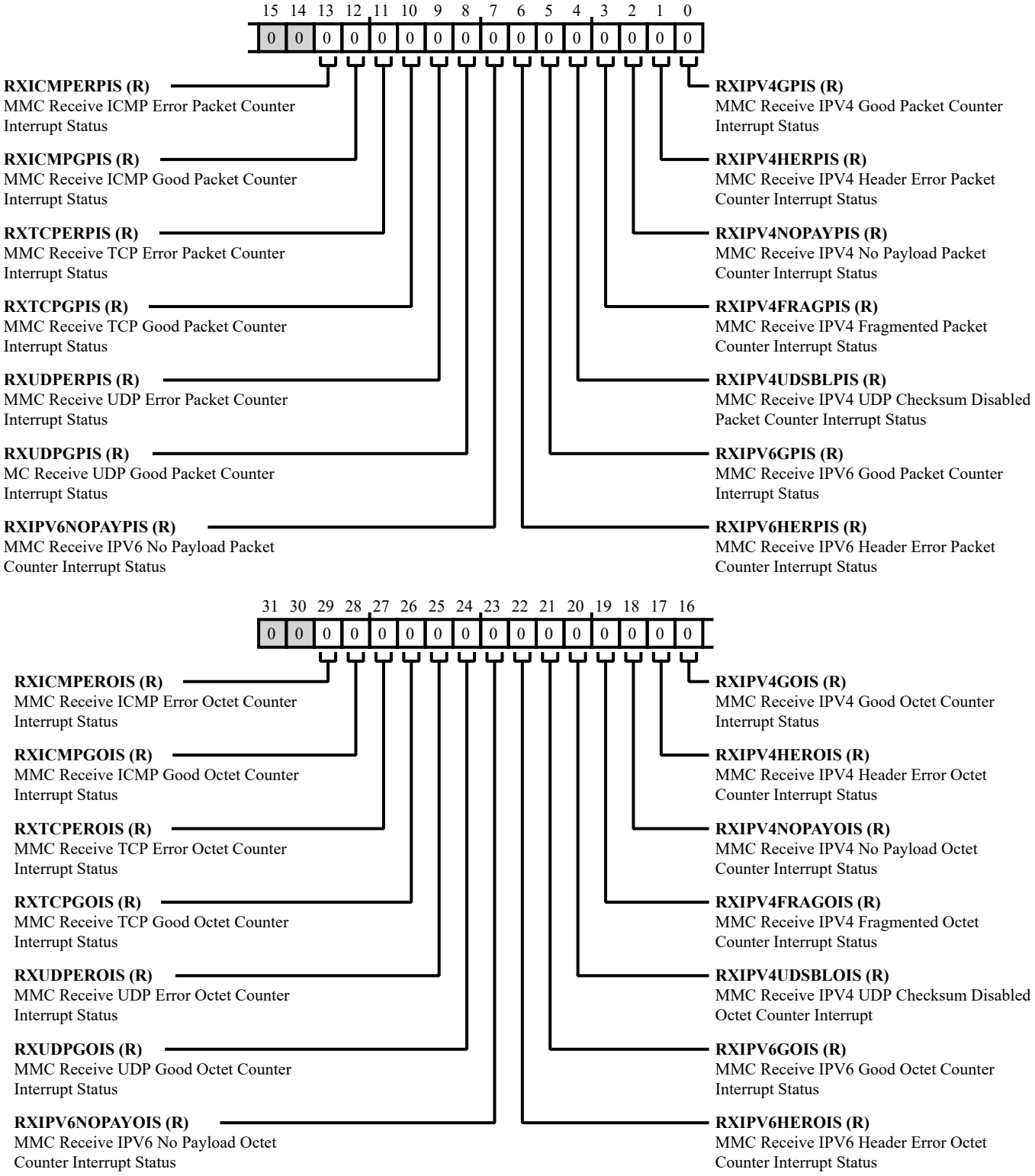


Figure 28-192: EMAC_MMC_IPC_RXINT Register Diagram

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/NW)	RXICMPEROIS	MMC Receive ICMP Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPEROIS` bit is set when the EMAC_RXICMP_ERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Error Octet Counter Interrupt Status Detected
28 (R/NW)	RXICMPGOIS	MMC Receive ICMP Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPGOIS` bit is set when the EMAC_RXICMP_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Good Octet Counter Interrupt Status Detected
27 (R/NW)	RXTCPEROIS	MMC Receive TCP Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPEROIS` bit is set when the EMAC_RXTCP_ERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Error Octet Counter Interrupt Status Detected
26 (R/NW)	RXTCPGOIS	MMC Receive TCP Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPGOIS` bit is set when the EMAC_RXTCP_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Good Octet Counter Interrupt Status Detected

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/NW)	RXUDPEROIS	MMC Receive UDP Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPEROIS` bit is set when the EMAC_RXUDP_ERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Error Octet Counter Interrupt Status Detected
24 (R/NW)	RXUDPGOIS	MMC Receive UDP Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPGOIS` bit is set when the EMAC_RXUDP_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Good Octet Counter Interrupt Status Detected
23 (R/NW)	RXIPV6NOPAYOIS	MMC Receive IPV6 No Payload Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6NOPAYOIS` bit is set when the EMAC_RXIPV6_PYLD_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 No Payload Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 No Payload Octet Counter Interrupt Status Detected
22 (R/NW)	RXIPV6HEROIS	MMC Receive IPV6 Header Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6HEROIS` bit is set when the EMAC_RXIPV6_HDRERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Header Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Header Error Octet Counter Interrupt Status Detected

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/NW)	RXIPV6GOIS	MMC Receive IPV6 Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6GOIS` bit is set when the EMAC_RXIPV6_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Good Octet Counter Interrupt Status Detected
20 (R/NW)	RXIPV4UDSBLOIS	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt. The `EMAC_MMC_IPC_RXINT.RXIPV4UDSBLOIS` bit is set when the EMAC_RXIPV4_UDPCSD_OCNT reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status Detected
19 (R/NW)	RXIPV4FRAGOIS	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4FRAGOIS` bit is set when the EMAC_RXIPV4_FRGMT_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status Detected
18 (R/NW)	RXIPV4NOPAYOIS	MMC Receive IPV4 No Payload Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4NOPAYOIS` bit is set when the EMAC_RXIPV4_PYLD_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 No Payload Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 No Payload Octet Counter Interrupt Status Detected

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	RXIPV4HEROIS	MMC Receive IPV4 Header Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4HEROIS` bit is set when the EMAC_RXIPV4_HDRERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Header Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Header Error Octet Counter Interrupt Status Detected
16 (R/NW)	RXIPV4GOIS	MMC Receive IPV4 Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4GOIS` bit is set when the EMAC_RXIPV4_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Good Octet Counter Interrupt Status Detected
13 (R/NW)	RXICMPERPIS	MMC Receive ICMP Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPERPIS` bit is set when the EMAC_RXICMP_ERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Error Packet Counter Interrupt Status Detected
12 (R/NW)	RXICMPGPIS	MMC Receive ICMP Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPGPIS` bit is set when the EMAC_RXICMP_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Good Packet Counter Interrupt Status Detected

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	RXTCPERPIS	MMC Receive TCP Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPERPIS` bit is set when the EMAC_RXTCP_ERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Error Packet Counter Interrupt Status Detected
10 (R/NW)	RXTCPGPIS	MMC Receive TCP Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPGPIS` bit is set when the EMAC_RXTCP_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Good Packet Counter Interrupt Status Detected
9 (R/NW)	RXUDPERPIS	MMC Receive UDP Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPERPIS` bit is set when the EMAC_RXUDP_ERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Error Packet Counter Interrupt Status Detected
8 (R/NW)	RXUDPGPIS	MC Receive UDP Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPGPIS` bit is set when the EMAC_RXUDP_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Good Packet Counter Interrupt Status Detected

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	RXIPV6NOPAYPIS	MMC Receive IPV6 No Payload Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6NOPAYPIS` bit is set when the EMAC_RXIPV6_PYLD_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 No Payload Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 No Payload Packet Counter Interrupt Status Detected
6 (R/NW)	RXIPV6HERPIS	MMC Receive IPV6 Header Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6HERPIS` bit is set when the EMAC_RXIPV6_HDRERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Header Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Header Error Packet Counter Interrupt Status Detected
5 (R/NW)	RXIPV6GPIS	MMC Receive IPV6 Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6GPIS` bit is set when the EMAC_RXIPV6_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Good Packet Counter Interrupt Status Detected
4 (R/NW)	RXIPV4UDSBLPIS	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4UDSBLPIS` bit is set when the EMAC_RXIPV4_UDPCSD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status Detected

Table 28-247: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	RXIPV4FRAGPIS	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4FRAGPIS` bit is set when the EMAC_RXIPV4_FRGMT_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status Detected
2 (R/NW)	RXIPV4NOPAYPIS	MMC Receive IPV4 No Payload Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4NOPAYPIS` bit is set when the EMAC_RXIPV4_PYLD_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 No Payload Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 No Payload Packet Counter Interrupt Status Detected
1 (R/NW)	RXIPV4HERPIS	MMC Receive IPV4 Header Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4HERPIS` bit is set when the EMAC_RXIPV4_HDRERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Header Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Header Error Packet Counter Interrupt Status Detected
0 (R/NW)	RXIPV4GPIS	MMC Receive IPV4 Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4GPIS` bit is set when the EMAC_RXIPV4_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Good Packet Counter Interrupt Status Detected

MMC Receive IPC Interrupt Mask Register

The EMAC_MMC_IPC_RXIMSK register maintains the mask for the interrupt generated from the receive IPC statistic counters.

The MMC Receive Checksum Off load Interrupt Mask register maintains the masks for the interrupts generated when the receive IPC (Checksum Off load) statistic counters reach half their maximum value, and when they reach their maximum values. The EMAC_MMC_IPC_RXIMSK register is 32 bits wide.

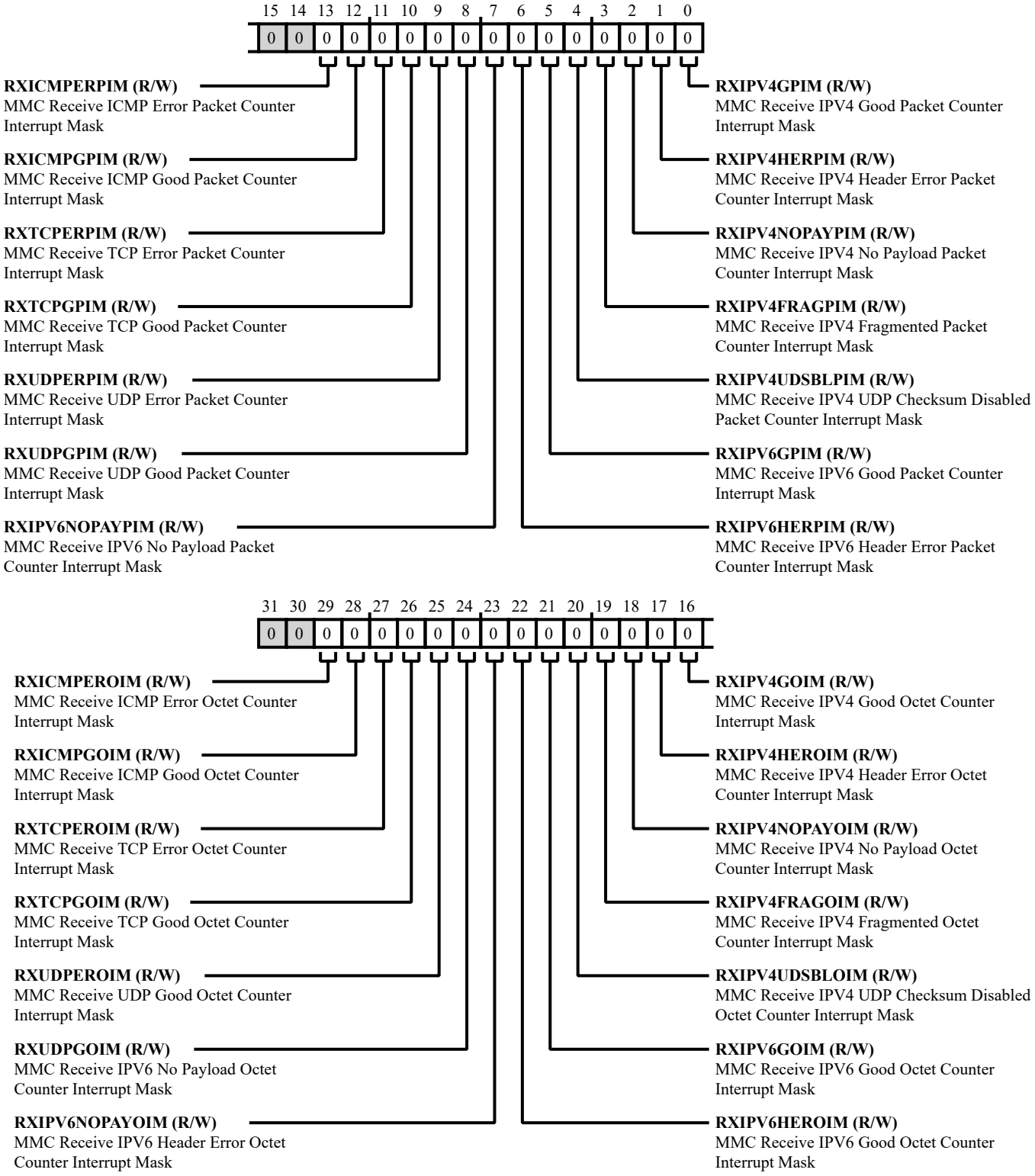


Figure 28-193: EMAC_MMC_IPC_RXIMSK Register Diagram

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	RXICMPEROIM	MMC Receive ICMP Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPEROIM` bit masks the interrupt when the EMAC_RXICMP_ERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Error Octet Counter Interrupt Mask is Enabled
28 (R/W)	RXICMPGOIM	MMC Receive ICMP Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPGOIM` bit masks the interrupt when the EMAC_RXICMP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Good Octet Counter Interrupt Mask is Enabled
27 (R/W)	RXTCPEROIM	MMC Receive TCP Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPEROIM` bit masks the interrupt when the EMAC_RXTCP_ERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Error Octet Counter Interrupt Mask is Enabled
26 (R/W)	RXTCPGOIM	MMC Receive TCP Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPGOIM` bit masks the interrupt when the EMAC_RXTCP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Good Octet Counter Interrupt Mask is Enabled

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	RXUDPEROIM	MMC Receive UDP Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPEROIM` bit masks the interrupt when the EMAC_RXUDP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive UDP Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive UDP Good Octet Counter Interrupt Mask is Enabled
24 (R/W)	RXUDPGOIM	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPGOIM` bit masks the interrupt when the EMAC_RXUDP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask is Enabled
23 (R/W)	RXIPV6NOPAYOIM	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6NOPAYOIM` bit masks the interrupt when the EMAC_RXIPV6_PYLD_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask is Enabled
22 (R/W)	RXIPV6HEROIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6HEROIM` bit masks the interrupt when the EMAC_RXIPV6_HDRERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Enabled

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	RXIPV6GOIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6GOIM` bit masks the interrupt when the EMAC_RXIPV6_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Enabled
20 (R/W)	RXIPV4UDSBLOIM	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4UDSBLOIM` bit masks the interrupt when the EMAC_RXIPV4_UDPCSD_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask is Enabled
19 (R/W)	RXIPV4FRAGOIM	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4FRAGOIM` bit masks the interrupt when the EMAC_RXIPV4_FRGMT_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask is Enabled
18 (R/W)	RXIPV4NOPAYOIM	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4NOPAYOIM` bit masks the interrupt when the EMAC_RXIPV4_PYLD_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask is Enabled

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RXIPV4HEROIM	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4HEROIM` bit masks the interrupt when the EMAC_RXIPV4_HDRERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask is Enabled
16 (R/W)	RXIPV4GOIM	MMC Receive IPV4 Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4GOIM` bit masks the interrupt when the EMAC_RXIPV4_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Good Octet Counter Interrupt Mask is Enabled
13 (R/W)	RXICMPERPIM	MMC Receive ICMP Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPERPIM` bit masks the interrupt when the EMAC_RXICMP_ERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Error Packet Counter Interrupt Mask is Enabled
12 (R/W)	RXICMPGPIM	MMC Receive ICMP Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPGPIM` bit masks the interrupt when the EMAC_RXICMP_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Good Packet Counter Interrupt Mask is Enabled

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	RXTCPERPIM	MMC Receive TCP Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPERPIM` bit masks the interrupt when the EMAC_RXTCP_ERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Error Packet Counter Interrupt Mask is Enabled
10 (R/W)	RXTCPGPIM	MMC Receive TCP Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPGPIM` bit masks the interrupt when the EMAC_RXTCP_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Good Packet Counter Interrupt Mask is Enabled
9 (R/W)	RXUDPERPIM	MMC Receive UDP Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPERPIM` bit masks the interrupt when the EMAC_RXUDP_ERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive UDP Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive UDP Error Packet Counter Interrupt Mask is Enabled
8 (R/W)	RXUDPGPIM	MMC Receive UDP Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPGPIM` bit masks the interrupt when the EMAC_RXUDP_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive UDP Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive UDP Good Packet Counter Interrupt Mask is Enabled

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXIPV6NOPAYPIM	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6NOPAYPIM` bit masks the interrupt when the EMAC_RXIPV4_PYLD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask is Enabled
6 (R/W)	RXIPV6HERPIM	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6HERPIM` bit masks the interrupt when the EMAC_RXIPV6_HDRERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask is Enabled
5 (R/W)	RXIPV6GPIM	MMC Receive IPV6 Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6GPIM` bit masks the interrupt when the EMAC_RXIPV6_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Good Packet Counter Interrupt Mask is Enabled
4 (R/W)	RXIPV4UDSBLPIM	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4UDSBLPIM` bit masks the interrupt when the EMAC_RXIPV4_UDPCSD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask is Enabled

Table 28-248: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXIPV4FRAGPIM	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4FRAGPIM` bit masks the interrupt when the EMAC_RXIPV4_FRGMT_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask is Enabled
2 (R/W)	RXIPV4NOPAYPIM	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4NOPAYPIM` bit masks the interrupt when the EMAC_RXIPV4_PYLD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask is Enabled
1 (R/W)	RXIPV4HERPIM	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4HERPIM` bit masks the interrupt when the EMAC_RXIPV4_HDRERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask is Enabled
0 (R/W)	RXIPV4GPIM	MMC Receive IPV4 Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4GPIM` bit masks the interrupt when the EMAC_RXIPV4_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Good Packet Counter Interrupt Mask is Enabled

MMC Receive FPE Fragment Counter Register

The EMAC_RXFPE_FRGMNT_CNT register provides the number of additional mPackets received due to preemption.

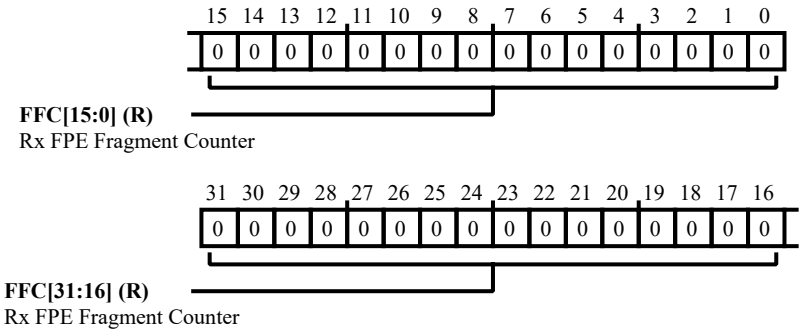


Figure 28-194: EMAC_RXFPE_FRGMNT_CNT Register Diagram

Table 28-249: EMAC_RXFPE_FRGMNT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	FFC	Rx FPE Fragment Counter. The EMAC_RXFPE_FRGMNT_CNT.FFC field indicates the number of additional mPackets received due to preemption. Exists when at least one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Receive Interrupt Status Register

The EMAC_MMC_RXINT register maintains the interrupts generated from all receive statistics counters when the following occurs:

Receive statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter).

Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the Counter Stop Rollover is set, interrupts are set but the counter remains at all-ones. The EMAC_MMC_RXINT register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

Note: R_SS_RC means that this register bit is set internally, and it is cleared when the counter register is read.

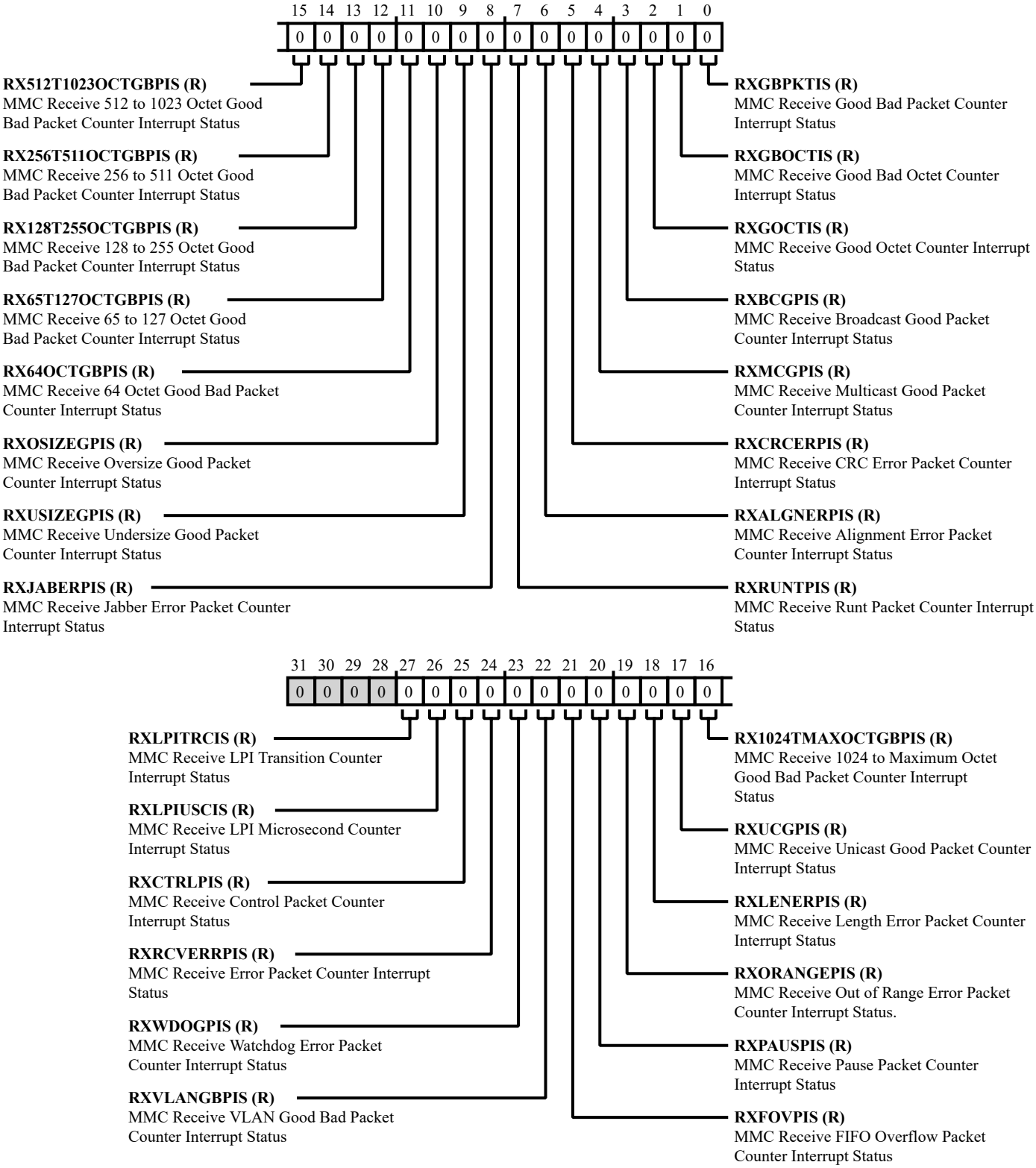


Figure 28-195: EMAC_MMC_RXINT Register Diagram

Table 28-250: EMAC_MMC_RXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	RXLPITRCIS	MMC Receive LPI Transition Counter Interrupt Status. The `EMAC_MMC_RXINT.RXLPITRCIS` bit is set when the EMAC_RX_LPITRAN_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive LPI Transition Counter Interrupt Status Not Detected
		1	MMC Receive LPI Transition Counter Interrupt Status Detected
26 (R/NW)	RXLPIUSCIS	MMC Receive LPI Microsecond Counter Interrupt Status. The `EMAC_MMC_RXINT.RXLPIUSCIS` bit is set when the EMAC_RX_LPIUSEC_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive LPI Microsecond Counter Interrupt Status Not Detected
		1	MMC Receive LPI Microsecond Counter Interrupt Status Detected
25 (R/NW)	RXCTRLPIS	MMC Receive Control Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXCTRLPIS` bit is set when the EMAC_RX_CTL_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Control Packet Counter Interrupt Status Not Detected
		1	MMC Receive Control Packet Counter Interrupt Status Detected
24 (R/NW)	RXRCVERRPIS	MMC Receive Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXRCVERRPIS` bit is set when the EMAC_RX_ERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Error Packet Counter Interrupt Status Detected

Table 28-250: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/NW)	RXWDOGPIS	MMC Receive Watchdog Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXWDOGPIS` bit is set when the EMAC_RX_WDTERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Watchdog Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Watchdog Error Packet Counter Interrupt Status Detected
22 (R/NW)	RXVLANGBPIS	MMC Receive VLAN Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXVLANGBPIS` bit is set when the EMAC_RX_VLAN_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive VLAN Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive VLAN Good Bad Packet Counter Interrupt Status Detected
21 (R/NW)	RXFOVPIS	MMC Receive FIFO Overflow Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXFOVPIS` bit is set when the EMAC_RX_FIFOOVFL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive FIFO Overflow Packet Counter Interrupt Status Not Detected
		1	MMC Receive FIFO Overflow Packet Counter Interrupt Status Detected
20 (R/NW)	RXPAUSPIS	MMC Receive Pause Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXPAUSPIS` bit is set when the EMAC_RX_PAUSE_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Pause Packet Counter Interrupt Status Not Detected
		1	MMC Receive Pause Packet Counter Interrupt Status Detected

Table 28-250: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	RXORANGEPIS	MMC Receive Out of Range Error Packet Counter Interrupt Status.. The EMAC_MMC_RXINT.RXORANGEPIS bit is set when the EMAC_RX_OR-TYPE_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Out of Range Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Out of Range Error Packet Counter Interrupt Status Detected
18 (R/NW)	RXLENERPIS	MMC Receive Length Error Packet Counter Interrupt Status. The EMAC_MMC_RXINT.RXLENERPIS bit is set when the EMAC_RX_LE-NERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Length Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Length Error Packet Counter Interrupt Status Detected
17 (R/NW)	RXUCGPIS	MMC Receive Unicast Good Packet Counter Interrupt Status. The EMAC_MMC_RXINT.RXUCGPIS bit is set when the EMAC_RX_UCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Unicast Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Unicast Good Packet Counter Interrupt Status Detected
16 (R/NW)	RX1024TMAXOCTGBPIS	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status. The EMAC_MMC_RXINT.RX1024TMAXOCTGBPIS bit is set when the EMAC_RX1024TOMAX_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Detected

Table 28-250: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	RX512T1023OCTGBPIS	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX512T1023OCTGBPIS` bit is set when the EMAC_RX512TO1023_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Detected
14 (R/NW)	RX256T511OCTGBPIS	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX256T511OCTGBPIS` bit is set when the EMAC_RX256TO511_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status Detected
13 (R/NW)	RX128T255OCTGBPIS	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX128T255OCTGBPIS` bit is set when the EMAC_RX128TO255_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status Detected
12 (R/NW)	RX65T127OCTGBPIS	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX65T127OCTGBPIS` bit is set when the EMAC_RX65TO127_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status Detected

Table 28-250: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	RX64OCTGBPIS	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX64OCTGBPIS` bit is set when the EMAC_RX64_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status Detected
10 (R/NW)	RXOSIZEGPIS	MMC Receive Oversize Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXOSIZEGPIS` bit is set when the EMAC_RX_OVRSZ_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Oversize Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Oversize Good Packet Counter Interrupt Status Detected
9 (R/NW)	RXUSIZEGPIS	MMC Receive Undersize Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXUSIZEGPIS` bit is set when the EMAC_RX_UNDRSZ_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Undersize Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Undersize Good Packet Counter Interrupt Status Detected
8 (R/NW)	RXJABERPIS	MMC Receive Jabber Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXJABERPIS` bit is set when the EMAC_RX_JBERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Jabber Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Jabber Error Packet Counter Interrupt Status Detected

Table 28-250: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	RXRUNTPIS	MMC Receive Runt Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXRUNTPIS` bit is set when the EMAC_RX_RNTERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Runt Packet Counter Interrupt Status Not Detected
		1	MMC Receive Runt Packet Counter Interrupt Status Detected
6 (R/NW)	RXALGNERPIS	MMC Receive Alignment Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXALGNERPIS` bit is set when the EMAC_RX_ALGNERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Alignment Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Alignment Error Packet Counter Interrupt Status Detected
5 (R/NW)	RXCRCERPIS	MMC Receive CRC Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXCRCERPIS` bit is set when the EMAC_RX_CRCERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive CRC Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive CRC Error Packet Counter Interrupt Status Detected
4 (R/NW)	RXMCGPIS	MMC Receive Multicast Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXMCGPIS` bit is set when the EMAC_RX_MCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Multicast Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Multicast Good Packet Counter Interrupt Status Detected

Table 28-250: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	RXBCGPIS	MMC Receive Broadcast Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXBCGPIS` bit is set when the EMAC_RX_BCERR_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Broadcast Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Broadcast Good Packet Counter Interrupt Status Detected
2 (R/NW)	RXGOCTIS	MMC Receive Good Octet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXGOCTIS` bit is set when the EMAC_RX_OCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive Good Octet Counter Interrupt Status Detected
1 (R/NW)	RXGBOCTIS	MMC Receive Good Bad Octet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXGBOCTIS` bit is set when the EMAC_RX_OCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Good Bad Octet Counter Interrupt Status Not Detected
		1	MMC Receive Good Bad Octet Counter Interrupt Status Detected
0 (R/NW)	RXGBPKTIS	MMC Receive Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXGBPKTIS` bit is set when the EMAC_RX_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive Good Bad Packet Counter Interrupt Status Detected

MMC Receive Interrupt Mask Register

The EMAC_MMC_RXIMSK register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

The EMAC_MMC_RXIMSK register is 32 bit wide.

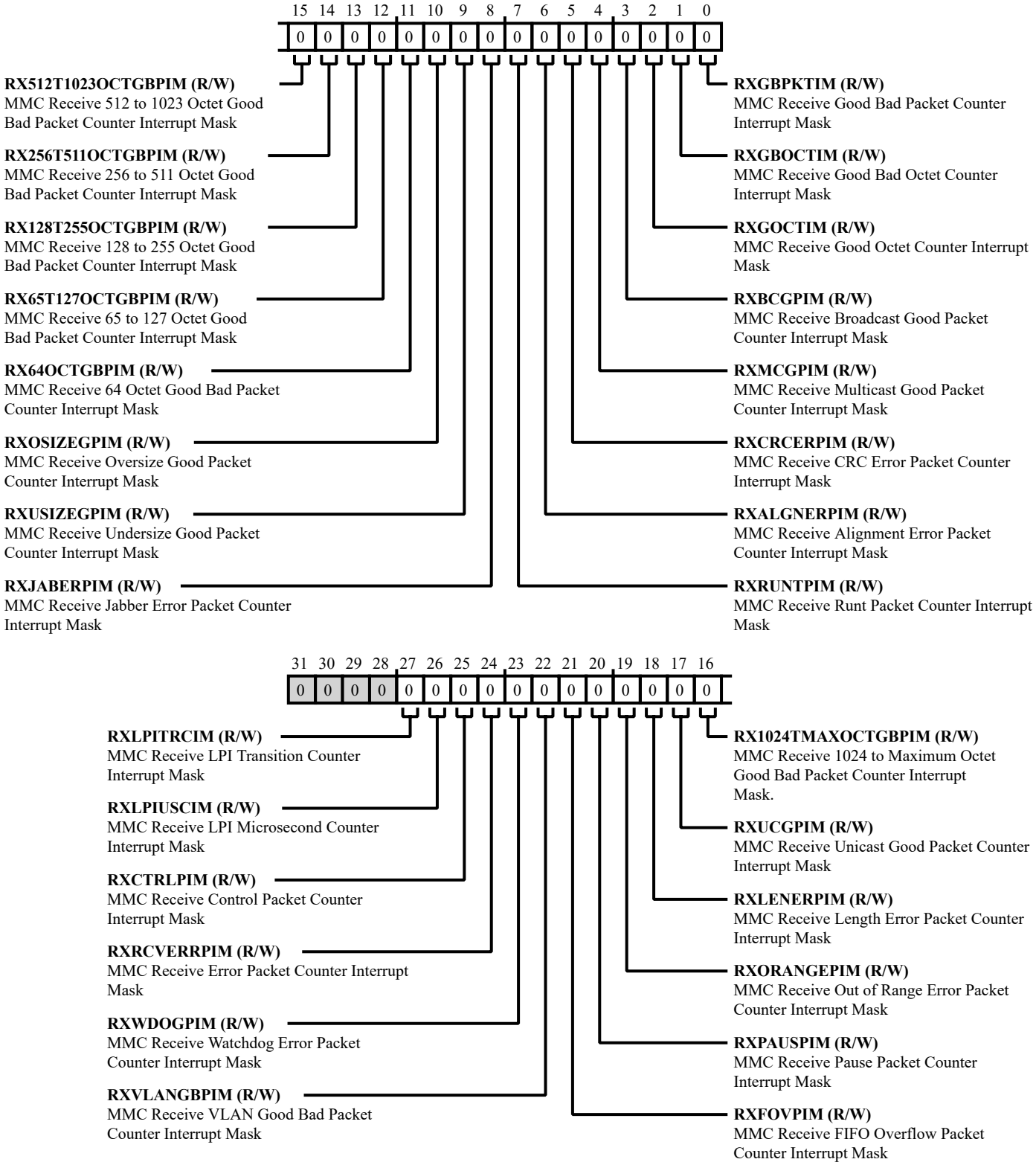


Figure 28-196: EMAC_MMC_RXIMSK Register Diagram

Table 28-251: EMAC_MMC_RXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	RXLPITRCIM	MMC Receive LPI Transition Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXLPITRCIM` bit masks the interrupt when the EMAC_RX_LPITRAN_CNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive LPI Transition Counter Interrupt Mask is Disabled
		1	MMC Receive LPI Transition Counter Interrupt Mask is Enabled
26 (R/W)	RXLPIUSCIM	MMC Receive LPI Microsecond Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXLPIUSCIM` bit masks the interrupt when the EMAC_RX_LPIUSEC_CNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive LPI Microsecond Counter Interrupt Mask is Disabled
		1	MMC Receive LPI Microsecond Counter Interrupt Mask is Enabled
25 (R/W)	RXCTRLPIM	MMC Receive Control Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXCTRLPIM` bit masks the interrupt when the EMAC_RXICMP_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Control Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Control Packet Counter Interrupt Mask is Enabled
24 (R/W)	RXRCVERRPIM	MMC Receive Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXRCVERRPIM` bit masks the interrupt when the EMAC_RX_ERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Error Packet Counter Interrupt Mask is Enabled

Table 28-251: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	RXWDOGPIM	MMC Receive Watchdog Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXWDOGPIM` bit masks the interrupt when the EMAC_RX_WDTERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Watchdog Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Watchdog Error Packet Counter Interrupt Mask is Enabled
22 (R/W)	RXVLANGBPIM	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXVLANGBPIM` bit masks the interrupt when the EMAC_RX_VLAN_PCNT_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask is Enabled
21 (R/W)	RXFOVPIM	MMC Receive FIFO Overflow Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXFOVPIM` bit masks the interrupt when the EMAC_RX_FIFOOVFL_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive FIFO Overflow Packet Counter Interrupt Mask is Disabled
		1	MMC Receive FIFO Overflow Packet Counter Interrupt Mask is Enabled
20 (R/W)	RXPAUSPIM	MMC Receive Pause Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXPAUSPIM` bit masks the interrupt when the EMAC_RX_PAUSE_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Pause Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Pause Packet Counter Interrupt Mask is Enabled

Table 28-251: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	RXORANGEPIM	MMC Receive Out of Range Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXORANGEPIM` bit masks the interrupt when the EMAC_RX_ORTYPE_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Out of Range Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Out of Range Error Packet Counter Interrupt Mask is Enabled
18 (R/W)	RXLENERPIM	MMC Receive Length Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXLENERPIM` bit masks the interrupt when the EMAC_RX_LENERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Length Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Length Error Packet Counter Interrupt Mask is Enabled
17 (R/W)	RXUCGPIM	MMC Receive Unicast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXUCGPIM` bit masks the interrupt when the EMAC_RX_UCST_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Unicast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Unicast Good Packet Counter Interrupt Mask is Enabled
16 (R/W)	RX1024TMAXOCTGB-PIM	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask.. Setting the `EMAC_MMC_RXIMSK.RX1024TMAXOCTGBPIM` bit masks the interrupt when the EMAC_RX1024TOMAX_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 28-251: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	RX512T1023OCTGBPIM	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX512T1023OCTGBPIM` bit masks the interrupt when the EMAC_RX512TO1023_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Enabled
14 (R/W)	RX256T511OCTGBPIM	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX256T511OCTGBPIM` bit masks the interrupt when the EMAC_RX256TO511_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Enabled
13 (R/W)	RX128T255OCTGBPIM	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX128T255OCTGBPIM` bit masks the interrupt when the EMAC_RX128TO255_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Enabled
12 (R/W)	RX65T127OCTGBPIM	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX65T127OCTGBPIM` bit masks the interrupt when the EMAC_RX65TO127_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 28-251: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	RX64OCTGBPIM	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX64OCTGBPIM` bit masks the interrupt when the EMAC_RX64_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask is Enabled
10 (R/W)	RXOSIZEGPIM	MMC Receive Oversize Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXOSIZEGPIM` bit masks the interrupt when the EMAC_RX_OVRSZ_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Oversize Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Oversize Good Packet Counter Interrupt Mask is Enabled
9 (R/W)	RXUSIZEGPIM	MMC Receive Undersize Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXUSIZEGPIM` bit masks the interrupt when the EMAC_RX_UNDRSZ_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Undersize Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Undersize Good Packet Counter Interrupt Mask is Enabled
8 (R/W)	RXJABERPIM	MMC Receive Jabber Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXJABERPIM` bit masks the interrupt when the EMAC_RX_JBERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Jabber Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Jabber Error Packet Counter Interrupt Mask is Enabled

Table 28-251: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXRUNTPIM	MMC Receive Runt Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXRUNTPIM` bit masks the interrupt when the EMAC_RX_RNTERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Runt Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Runt Packet Counter Interrupt Mask is Enabled
6 (R/W)	RXALGNERPIM	MMC Receive Alignment Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXALGNERPIM` bit masks the interrupt when the EMAC_RX_ALGNERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Alignment Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Alignment Error Packet Counter Interrupt Mask is Enabled
5 (R/W)	RXCRCERPIM	MMC Receive CRC Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXCRCERPIM` bit masks the interrupt when the EMAC_RX_CRCERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive CRC Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive CRC Error Packet Counter Interrupt Mask is Enabled
4 (R/W)	RXMCGPIM	MMC Receive Multicast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXMCGPIM` bit masks the interrupt when the EMAC_RX_MCST_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Multicast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Multicast Good Packet Counter Interrupt Mask is Enabled

Table 28-251: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXBCGPIM	MMC Receive Broadcast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXBCGPIM` bit masks the interrupt when the EMAC_RX_BCERR_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Broadcast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Broadcast Good Packet Counter Interrupt Mask is Enabled
2 (R/W)	RXGOCTIM	MMC Receive Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXGOCTIM` bit masks the interrupt when the EMAC_RX_OCNT_G register counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive Good Octet Counter Interrupt Mask is Enabled
1 (R/W)	RXGBOCTIM	MMC Receive Good Bad Octet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXGBOCTIM` bit masks the interrupt when the EMAC_RX_OCNT_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Good Bad Octet Counter Interrupt Mask is Disabled
		1	MMC Receive Good Bad Octet Counter Interrupt Mask is Enabled
0 (R/W)	RXGBPKTIM	MMC Receive Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXGBPKTIM` bit masks the interrupt when the EMAC_RX_PCNT_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Good Bad Packet Counter Interrupt Mask is Enabled

MMC Receive FPE Packet Assembly Error Counter Register

The EMAC_RXPKT_ASERR_CNT register provides the number of MAC frames with reassembly errors on the receiver, due to mismatch in the fragment count value.

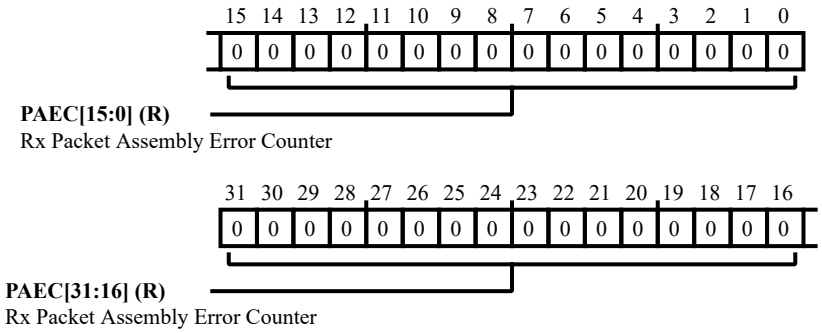


Figure 28-197: EMAC_RXPKT_ASERR_CNT Register Diagram

Table 28-252: EMAC_RXPKT_ASERR_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PAEC	Rx Packet Assembly Error Counter. The EMAC_RXPKT_ASERR_CNT.PAEC field indicates the number of MAC frames with reassembly errors on the receiver, due to mismatch in the Fragment Count value. Exists when any one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Receive FPE Packet Assembly Okay Counter Register

The EMAC_RXPKT_ASOK_CNT register provides the number of MAC frames that were successfully reassembled and delivered to MAC.

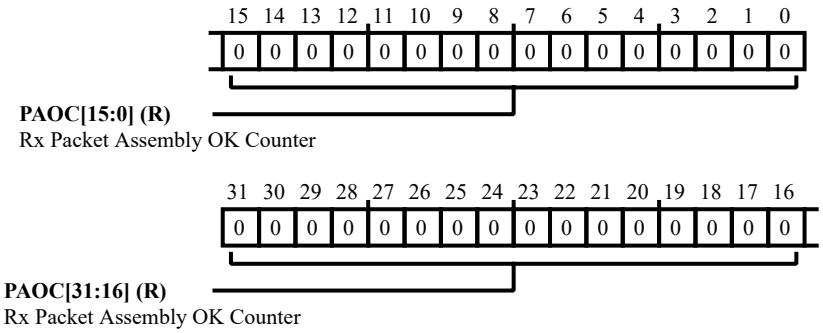


Figure 28-198: EMAC_RXPKT_ASOK_CNT Register Diagram

Table 28-253: EMAC_RXPKT_ASOK_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PAOC	Rx Packet Assembly OK Counter. The EMAC_RXPKT_ASOK_CNT.PAOC field indicates the number of MAC frames that were successfully reassembled and delivered to MAC. Exists when at least one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Receive FPE Packet SMD Error Counter Register

The EMAC_RXPKT_SMDERR_CNT register provides the number of received MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame.

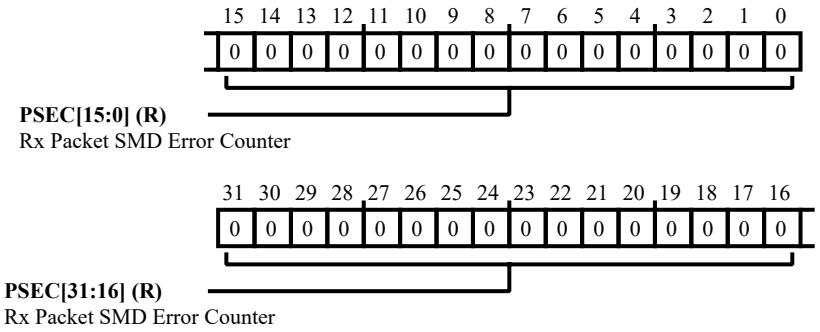


Figure 28-199: EMAC_RXPKT_SMDERR_CNT Register Diagram

Table 28-254: EMAC_RXPKT_SMDERR_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PSEC	Rx Packet SMD Error Counter. The EMAC_RXPKT_SMDERR_CNT.PSEC field indicates the number of MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame. Exists when at least one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Transmit FPE Packet Fragment Counter Register

The EMAC_TXFPE_FRGMNT_CNT register provides the number of additional mPackets transmitted due to preemption.

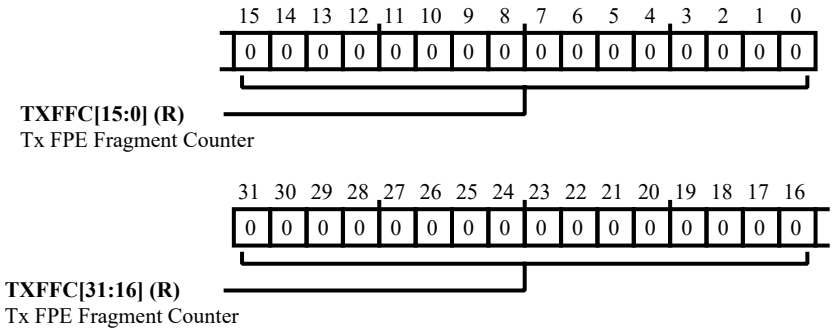


Figure 28-200: EMAC_TXFPE_FRGMNT_CNT Register Diagram

Table 28-255: EMAC_TXFPE_FRGMNT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXFFC	Tx FPE Fragment Counter. The EMAC_TXFPE_FRGMNT_CNT.TXFFC field indicates the number of additional mPackets that has been transmitted due to preemption Exists when any one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Transmit FPE Hold Request Counter Register

The EMAC_TXHLDRQ_CNT register provides the count of number of times a hold request is given to MAC.

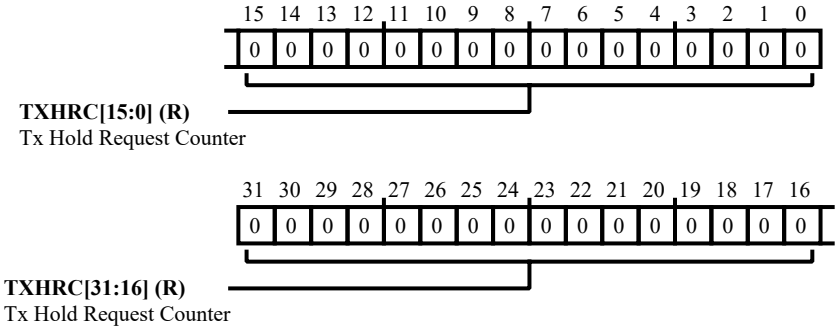


Figure 28-201: EMAC_TXHLDRQ_CNT Register Diagram

Table 28-256: EMAC_TXHLDRQ_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXHRC	Tx Hold Request Counter. The EMAC_TXHLDRQ_CNT.TXHRC field indicates count of number of a hold request is given to MAC.

MMC Transmit Interrupt Status Register

The EMAC_MMC_TXINT register maintains the interrupts generated from all transmit statistics counters when transmit statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The EMAC_MMC_TXINT register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (bits[7:0]) of the respective counter must be read to clear the interrupt bit.

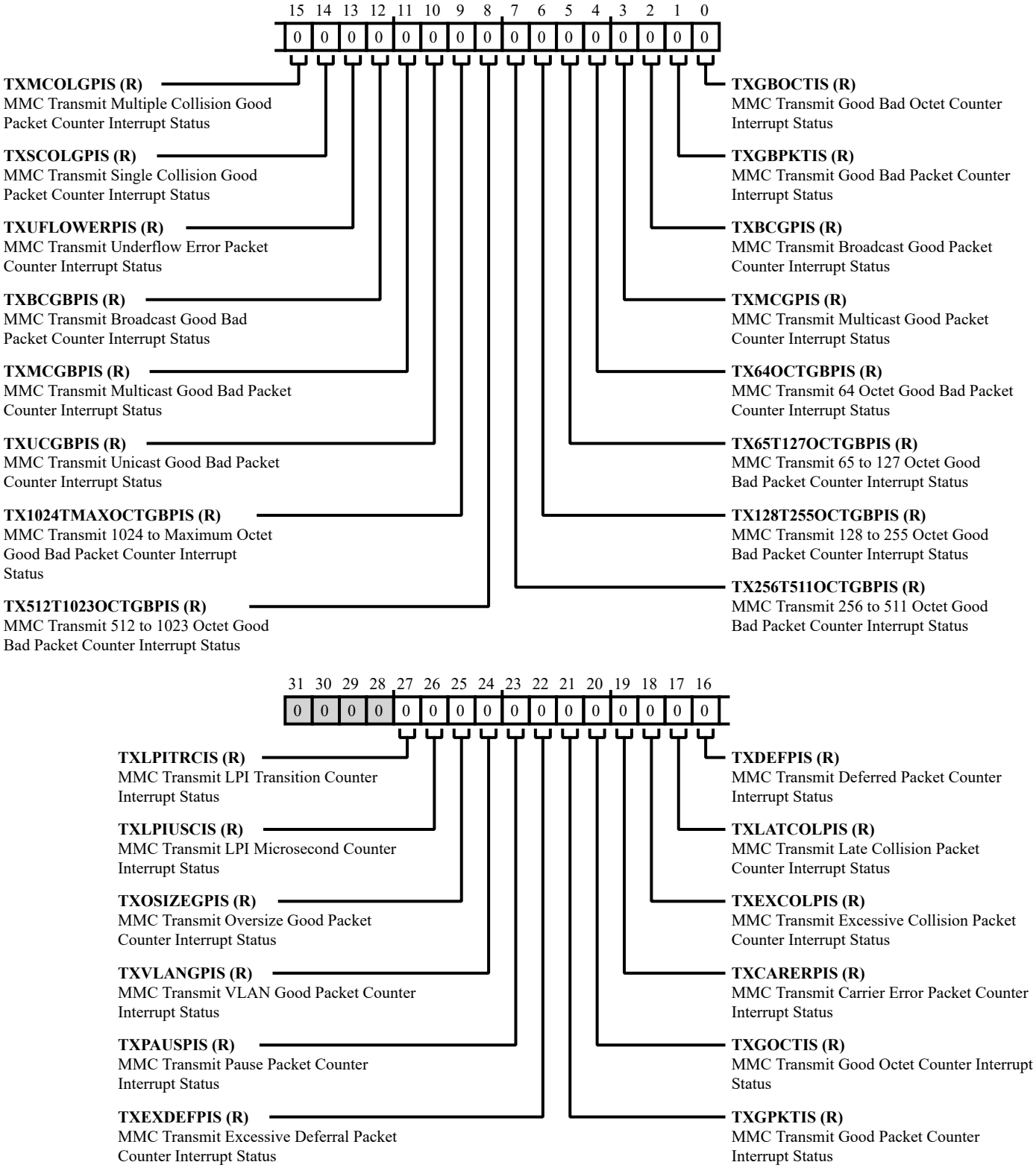


Figure 28-202: EMAC_MMC_TXINT Register Diagram

Table 28-257: EMAC_MMC_TXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	TXLPITRCIS	MMC Transmit LPI Transition Counter Interrupt Status. The `EMAC_MMC_TXINT.TXLPITRCIS` bit is set when the EMAC_TX_LPITRAN_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit LPI Transition Counter Interrupt Status Not Detected
		1	MMC Transmit LPI Transition Counter Interrupt Status Detected
26 (R/NW)	TXLPIUSCIS	MMC Transmit LPI Microsecond Counter Interrupt Status. The `EMAC_MMC_TXINT.TXLPIUSCIS` bit is set when the EMAC_TX_LPIUSEC_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit LPI Microsecond Counter Interrupt Status Not Detected
		1	MMC Transmit LPI Microsecond Counter Interrupt Status Detected
25 (R/NW)	TXOSIZEGPIS	MMC Transmit Oversize Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXOSIZEGPIS` bit is set when the EMAC_TX_OSIZE_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Oversize Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Oversize Good Packet Counter Interrupt Status Detected
24 (R/NW)	TXVLANGPIS	MMC Transmit VLAN Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXVLANGPIS` bit is set when the EMAC_TX_VLAN_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit VLAN Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit VLAN Good Packet Counter Interrupt Status Detected

Table 28-257: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/NW)	TXPAUSPIS	MMC Transmit Pause Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXPAUSPIS` bit is set when the EMAC_TX_PAUSE_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Pause Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Pause Packet Counter Interrupt Status Detected
22 (R/NW)	TXEXDEFPIS	MMC Transmit Excessive Deferral Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXEXDEFPIS` bit is set when the EMAC_TX_EXDFRL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Excessive Deferral Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Excessive Deferral Packet Counter Interrupt Status Detected
21 (R/NW)	TXGPKTIS	MMC Transmit Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGPKTIS` bit is set when the EMAC_TX_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Packet Counter Interrupt Status Detected
20 (R/NW)	TXGOCTIS	MMC Transmit Good Octet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGOCTIS` bit is set when the EMAC_TX_OCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Octet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Octet Counter Interrupt Status Detected

Table 28-257: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	TXCARERPIS	MMC Transmit Carrier Error Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXCARERPIS` bit is set when the EMAC_TX_CRERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Carrier Error Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Carrier Error Packet Counter Interrupt Status Detected
18 (R/NW)	TXEXCOLPIS	MMC Transmit Excessive Collision Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXEXCOLPIS` bit is set when the EMAC_TX_EXCOL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Excessive Collision Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Excessive Collision Packet Counter Interrupt Status Detected
17 (R/NW)	TXLATCOLPIS	MMC Transmit Late Collision Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXLATCOLPIS` bit is set when the EMAC_TX_LTCOL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Late Collision Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Late Collision Packet Counter Interrupt Status Detected
16 (R/NW)	TXDEFPIS	MMC Transmit Deferred Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXDEFPIS` bit is set when the EMAC_TX_DFRD_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Deferred Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Deferred Packet Counter Interrupt Status Detected

Table 28-257: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	TXMCOLGPIS	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXMCOLGPIS` bit is set when the EMAC_TX_MCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status Detected
14 (R/NW)	TXSCOLGPIS	MMC Transmit Single Collision Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXSCOLGPIS` bit is set when the EMAC_TX_SNGLCOL_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Single Collision Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Single Collision Good Packet Counter Interrupt Status Detected
13 (R/NW)	TXUFLOWERPIS	MMC Transmit Underflow Error Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXUFLOWERPIS` bit is set when the EMAC_TX_URFL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Underflow Error Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Underflow Error Packet Counter Interrupt Status Detected
12 (R/NW)	TXBCGBPIS	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXBCGBPIS` bit is set when the EMAC_TX_BCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status Detected

Table 28-257: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	TXMCGBPIS	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status. The bit is set when the EMAC_TX_MCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status Detected
10 (R/NW)	TXUCGBPIS	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXUCGBPIS` bit is set when the EMAC_TX_UCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status Detected
9 (R/NW)	TX1024TMAXOCTGBPIS	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX1024TMAXOCTGBPIS` bit is set when the EMAC_TX1024TOMAX_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Detected
8 (R/NW)	TX512T1023OCTGBPIS	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX512T1023OCTGBPIS` bit is set when the EMAC_TX512TO1023_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Detected

Table 28-257: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	TX256T511OCTGBPIS	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX256T511OCTGBPIS` bit is set when the EMAC_TX256TO511_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status Detected
6 (R/NW)	TX128T255OCTGBPIS	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX128T255OCTGBPIS` bit is set when the EMAC_TX128TO255_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status Detected
5 (R/NW)	TX65T127OCTGBPIS	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX65T127OCTGBPIS` bit is set when the EMAC_TX65TO127_GB register reaches half the maximum value, and also when it reaches the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status Detected
4 (R/NW)	TX64OCTGBPIS	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX64OCTGBPIS` bit is set when the EMAC_TX64_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status Detected

Table 28-257: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	TXMCGPIS	MMC Transmit Multicast Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXMCGPIS` bit is set when the EMAC_TX_MCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Multicast Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Multicast Good Packet Counter Interrupt Status Detected
2 (R/NW)	TXBCGPIS	MMC Transmit Broadcast Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXBCGPIS` bit is set when the EMAC_TX_BCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Broadcast Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Broadcast Good Packet Counter Interrupt Status Detected
1 (R/NW)	TXGBPKTIS	MMC Transmit Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGBPKTIS` bit is set when the EMAC_TX_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Bad Packet Counter Interrupt Status Detected
0 (R/NW)	TXGBOCTIS	MMC Transmit Good Bad Octet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGBOCTIS` bit is set when the EMAC_TX_OCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Bad Octet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Bad Octet Counter Interrupt Status Detected

MMC Transmit Interrupt Mask Register

The EMAC_MMC_TXIMSK register maintains the masks for interrupts generated from all transmit statistics counters.

The EMAC_MMC_TXIMSK register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. The EMAC_MMC_TXIMSK register is 32 bits wide.

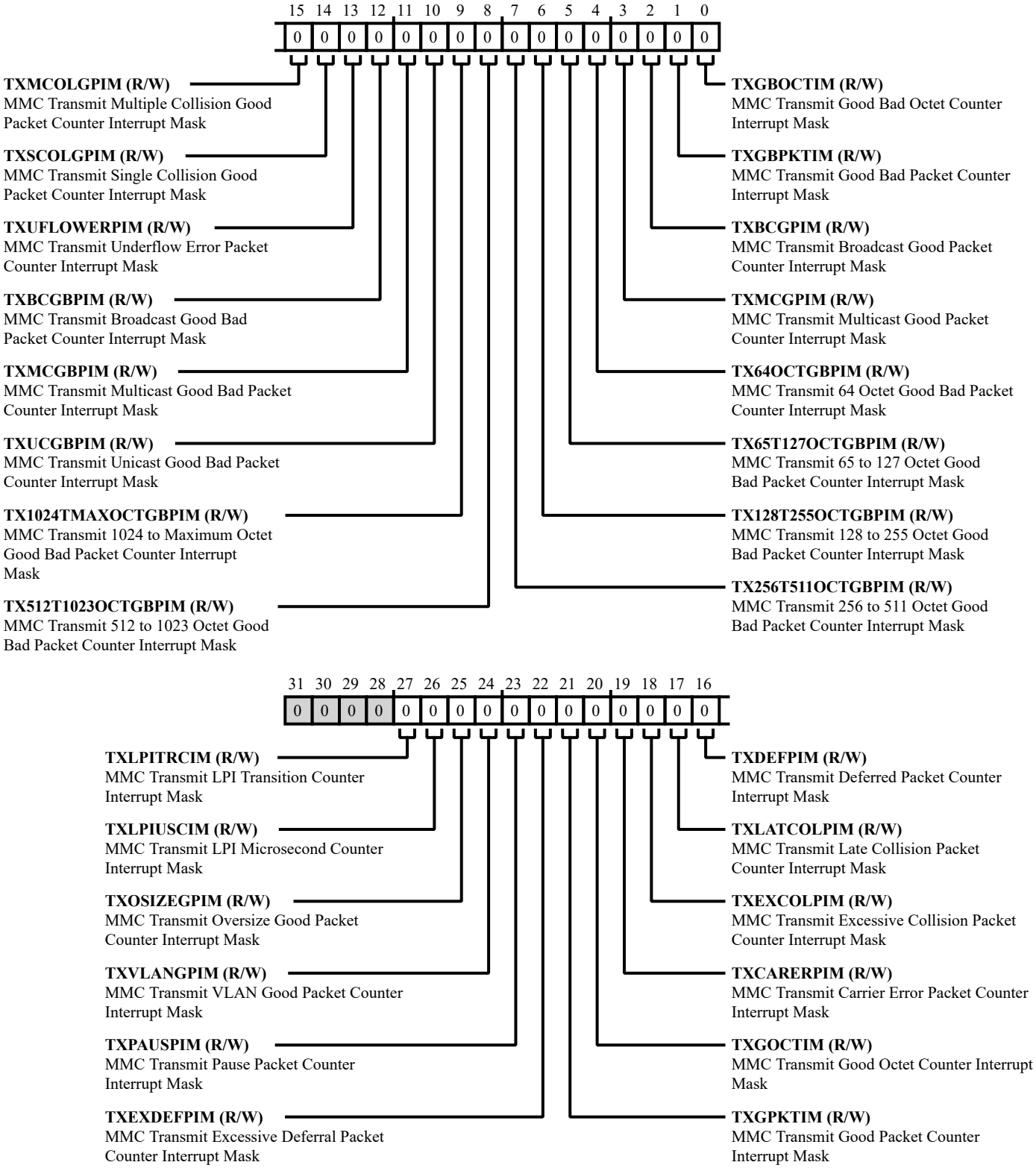


Figure 28-203: EMAC_MMC_TXIMSK Register Diagram

Table 28-258: EMAC_MMC_TXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	TXLPITRCIM	MMC Transmit LPI Transition Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXLPITRCIM` bit masks the interrupt when the EMAC_TX_LPITRAN_CNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit LPI Transition Counter Interrupt Mask is Disabled
		1	MMC Transmit LPI Transition Counter Interrupt Mask is Enabled
26 (R/W)	TXLPIUSCIM	MMC Transmit LPI Microsecond Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXLPIUSCIM` bit masks the interrupt when the EMAC_TX_LPIUSEC_CNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit LPI Microsecond Counter Interrupt Mask is Disabled
		1	MMC Transmit LPI Microsecond Counter Interrupt Mask is Enabled
25 (R/W)	TXOSIZEGPIM	MMC Transmit Oversize Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXOSIZEGPIM` bit masks the interrupt when the EMAC_TX_OSIZE_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Oversize Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Oversize Good Packet Counter Interrupt Mask is Enabled
24 (R/W)	TXVLANGPIM	MMC Transmit VLAN Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXVLANGPIM` bit masks the interrupt when the EMAC_TX_VLAN_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit VLAN Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit VLAN Good Packet Counter Interrupt Mask is Enabled

Table 28-258: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	TXPAUSPIM	MMC Transmit Pause Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXPAUSPIM` bit masks the interrupt when the EMAC_TX_PAUSE_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Pause Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Pause Packet Counter Interrupt Mask is Enabled
22 (R/W)	TXEXDEFPIM	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXEXDEFPIM` bit masks the interrupt when the EMAC_TX_EXDFRL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask is Enabled
21 (R/W)	TXGPKTIM	MMC Transmit Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGPKTIM` bit masks the interrupt when the EMAC_TX_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Packet Counter Interrupt Mask is Enabled
20 (R/W)	TXGOCTIM	MMC Transmit Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGOCTIM` bit masks the interrupt when the EMAC_TX_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Octet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Octet Counter Interrupt Mask is Enabled

Table 28-258: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	TXCARERPIM	MMC Transmit Carrier Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXCARERPIM` bit masks the interrupt when the EMAC_TX_CRERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Carrier Error Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Carrier Error Packet Counter Interrupt Mask is Enabled
18 (R/W)	TXEXCOLPIM	MMC Transmit Excessive Collision Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXEXCOLPIM` bit masks the interrupt when the EMAC_TX_EXCOL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Excessive Collision Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Excessive Collision Packet Counter Interrupt Mask is Enabled
17 (R/W)	TXLATCOLPIM	MMC Transmit Late Collision Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXLATCOLPIM` bit masks the interrupt when the EMAC_TX_LTCOL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Late Collision Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Late Collision Packet Counter Interrupt Mask is Enabled
16 (R/W)	TXDEFPIM	MMC Transmit Deferred Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXDEFPIM` bit masks the interrupt when the EMAC_TX_DFRD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Deferred Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Deferred Packet Counter Interrupt Mask is Enabled

Table 28-258: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	TXMCOLGPIM	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXMCOLGPIM` bit masks the interrupt when the EMAC_TX_MCOL_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask is Enabled
14 (R/W)	TXSCOLGPIM	MMC Transmit Single Collision Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXSCOLGPIM` bit masks the interrupt when the EMAC_TX_SNGLCOL_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Single Collision Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Single Collision Good Packet Counter Interrupt Mask is Enabled
13 (R/W)	TXUFLOWERPIM	MMC Transmit Underflow Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXUFLOWERPIM` bit masks the interrupt when the EMAC_TX_URFL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Underflow Error Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Underflow Error Packet Counter Interrupt Mask is Enabled
12 (R/W)	TXBCGBPIM	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXBCGBPIM` bit masks the interrupt when the EMAC_TX_BCST_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask is Enabled

Table 28-258: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	TXMCGBPIM	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXMCGBPIM` bit masks the interrupt when the EMAC_TX_MCST_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask is Enabled
10 (R/W)	TXUCGBPIM	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXUCGBPIM` bit masks the interrupt when the EMAC_TX_UCST_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask is Enabled
9 (R/W)	TX1024TMAXOCTGB-PIM	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX1024TMAXOCTGBPIM` bit masks the interrupt when the EMAC_TX1024TOMAX_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Enabled
8 (R/W)	TX512T1023OCTGBPIM	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX512T1023OCTGBPIM` bit masks the interrupt when the EMAC_TX512TO1023_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 28-258: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	TX256T511OCTGBPIM	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX256T511OCTGBPIM` bit masks the interrupt when the EMAC_TX256TO511_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Enabled
6 (R/W)	TX128T255OCTGBPIM	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX128T255OCTGBPIM` bit masks the interrupt when the EMAC_TX128TO255_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Enabled
5 (R/W)	TX65T127OCTGBPIM	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX65T127OCTGBPIM` bit masks the interrupt when the EMAC_TX65TO127_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Enabled
4 (R/W)	TX64OCTGBPIM	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX64OCTGBPIM` bit masks the interrupt when the EMAC_TX64_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 28-258: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TXMCGPIM	MMC Transmit Multicast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXMCGPIM` bit masks the interrupt when the EMAC_TX_MCST_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Multicast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Multicast Good Packet Counter Interrupt Mask is Enabled
2 (R/W)	TXBCGPIM	MMC Transmit Broadcast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXBCGPIM` bit masks the interrupt when the EMAC_TX_BCST_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Broadcast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Broadcast Good Packet Counter Interrupt Mask is Enabled
1 (R/W)	TXGBPKTIM	MMC Transmit Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGBPKTIM` bit masks the interrupt when the EMAC_TX_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Bad Packet Counter Interrupt Mask is Enabled
0 (R/W)	TXGBOCTIM	MMC Transmit Good Bad Octet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGBOCTIM` bit masks the interrupt when the EMAC_TX_OCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Bad Octet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Bad Octet Counter Interrupt Mask is Enabled

EST Control Register

The EMAC_EST_CTL register controls the operation of Enhancements to Scheduled Transmission (EST) (IEEE802.1Qbv).

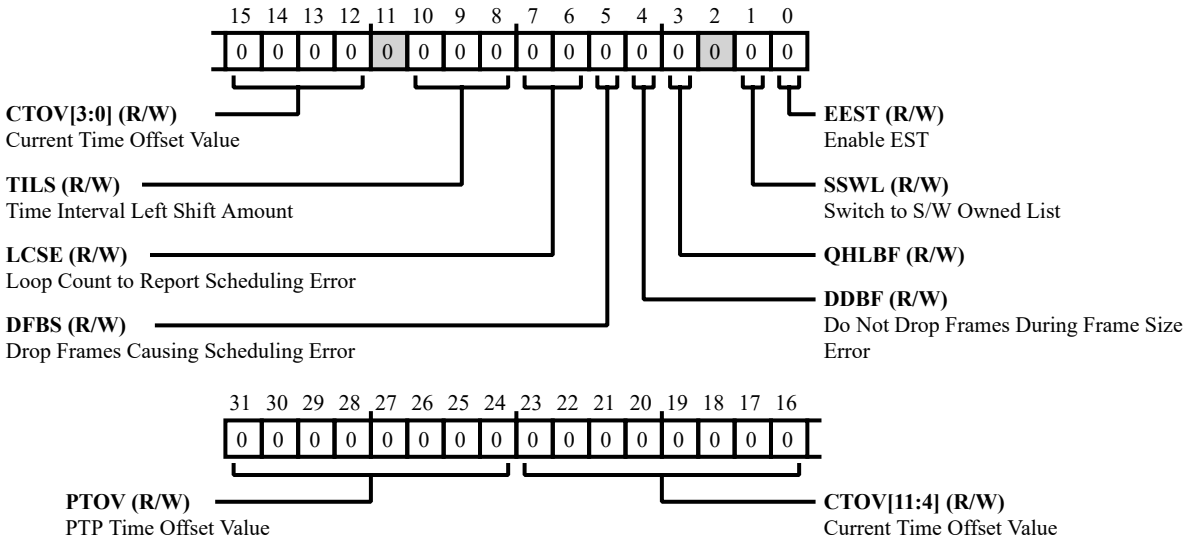


Figure 28-204: EMAC_EST_CTL Register Diagram

Table 28-259: EMAC_EST_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PTOV	PTP Time Offset Value. The EMAC_EST_CTL.PTOV field indicates the value of the PTP clock period multiplied by 6 in nanoseconds. This value is needed to avoid transmission overruns at the beginning of the installation of a new GCL.
23:12 (R/W)	CTOV	Current Time Offset Value. The EMAC_EST_CTL.CTOV field provides a 12 bit time offset value in nano second that is added to the current time to compensate for all the implementation pipeline delays such as the CDC sync delay, buffering delays, data path delays and so on. This offset helps to ensure that the impact of gate controls is visible on the line exactly at the pre-determined schedule (or as close to the schedule as possible).

Table 28-259: EMAC_EST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:8 (R/W)	TILS	Time Interval Left Shift Amount. The EMAC_EST_CTL.TILS field provides the left shift amount for the programmed time interval values used in the Gate Control Lists. - 000: No left shift needed (equal to x1ns) - 001: Left shift TI by 1 bit (equal to x2ns) - 010: Left shift TI by 2 bits (equal to x4ns) - ... - ... - 100: Left shift TI by 7 bits (equal to x128ns) Based on the configuration one or more bits of this field should be treated as reserved/read-only.	
7:6 (R/W)	LCSE	Loop Count to Report Scheduling Error. The EMAC_EST_CTL.LCSE field indicates a programmable number of GCL list iterations before reporting the EMAC_EST_STAT.HLBS bit.	
		0	4 Iterations
		1	8 Iterations
		2	16 Iterations
		3	32 Iterations
5 (R/W)	DFBS	Drop Frames Causing Scheduling Error. When the EMAC_EST_CTL.DFBS bit is set (=1), frames reported to cause HOL blocking due to not getting scheduled (EMAC_EST_STAT.HLBS) after 4,8,16,32 (based on LCSE field of this register) GCL iterations are dropped.	
		0	Do Not Drop Frames Causing Scheduling Error
		1	Drop Frames Causing Scheduling Error
4 (R/W)	DDBF	Do Not Drop Frames During Frame Size Error. When the EMAC_EST_CTL.DDBF bit is set (=1), frames are not dropped during head-of-line blocking due to a frame size error (EMAC_EST_STAT.HLBF).	
		0	Drop Frames During Frame Size Error
		1	Do Not Drop Frames During Frame Size Error
3 (R/W)	QHLBF	. Quick Assertion of HLBF Error When set, Time Window for Head-of-Line blocking due to Frame Size Error is 1 to 2 loop count of GCL list.On reset,Time Window for Head-of-Line blocking due to Frame Size Error is 2 to 3 loop counts of GCL list.	

Table 28-259: EMAC_EST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SSWL	Switch to S/W Owned List. When set (=1), the EMAC_EST_CTL.SSWL bit indicates that the software has programmed that list that it currently owns (EMAC_EST_STAT.SWOL) and the hardware should switch to the new list based on the new BTR. Hardware clears the EMAC_EST_CTL.SSWL bit when the switch to the EMAC_EST_STAT.SWOL bit happens to indicate the completion of the switch or a BTR error (EMAC_EST_STAT.BTRE = 1). When EMAC_EST_STAT.BTRE = 1, the EMAC_EST_CTL.SSWL bit is cleared, but EMAC_EST_STAT.SWOL is not updated as the switch was not successful. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Switch to S/W Owned List is Disabled
		1	Switch to S/W Owned List is Enabled
0 (R/W)	EEST	Enable EST. When set (=1), the EMAC_EST_CTL.EEST bit enables EST. It must be set for the hardware to start processing the gate control lists. When cleared, the gate control list processing is halted and all gates are assumed to be in the Open state. During the toggle from 0 to 1, the gate control list processing starts only after EMAC_EST_CTL.SSWL =1.	
		0	EST is Disabled
		1	EST is Enabled

EST Extended Control Register

The EMAC_EST_EXT_CTL register indicates the number of overhead bytes for EST related scheduling.

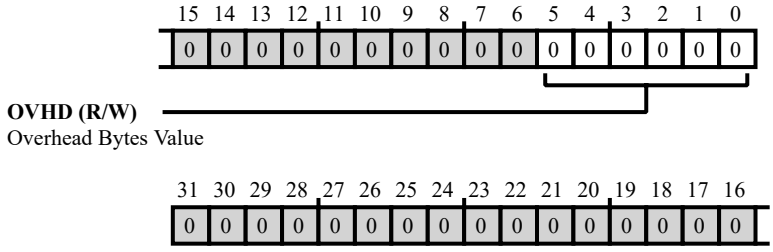


Figure 28-205: EMAC_EST_EXT_CTL Register Diagram

Table 28-260: EMAC_EST_EXT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	OVHD	Overhead Bytes Value. The EMAC_EST_EXT_CTL.OVHD field indicates the fixed overhead for every packet to account for EST Scheduler Delay, IPG or EIPG, and Preamble bytes.

EST Frame Size Capture Register

The EMAC_EST_FRMSZ_CAP register captures the frame size and queue number of the first occurrence of the frame size-related error. Upon clearing, it captures the data of immediate next occurrence of a similar error.

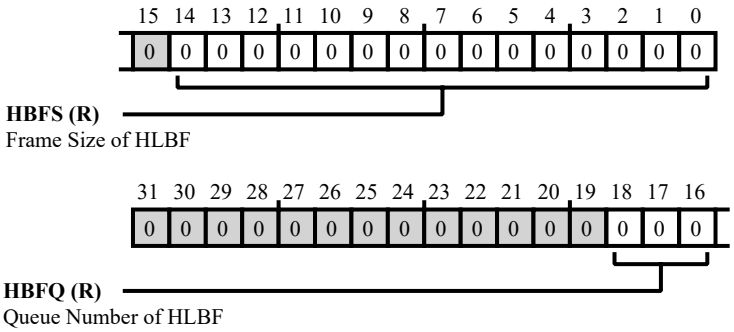


Figure 28-206: EMAC_EST_FRMSZ_CAP Register Diagram

Table 28-261: EMAC_EST_FRMSZ_CAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18:16 (R/NW)	HBFQ	Queue Number of HLBF. The EMAC_EST_FRMSZ_CAP.HBFQ field indicates the binary value of the of the first queue (number) experiencing HLBF error (see HLBF field of status register). Value once written is not altered by any subsequent queue errors of similar nature. Once cleared the queue number of the next occurring HLBF error is captured. The width is based on the number of Tx queues configured; remaining bits are read-only. The EMAC_EST_FRMSZ_CAP.HBFQ field is cleared when the EMAC_EST_FRMSZ_ERR register is all zeros.
14:0 (R/NW)	HBFS	Frame Size of HLBF. The EMAC_EST_FRMSZ_CAP.HBFS field indicates the frame size of the dropped frame related to queue number indicated in EMAC_EST_FRMSZ_CAP.HBFQ field. The contents of the register is considered invalid, if the EMAC_EST_FRMSZ_CAP.HBFS field is zero. The EMAC_EST_FRMSZ_CAP.HBFS field cleared when the EMAC_EST_FRMSZ_ERR register is all zeros.

EST Frame Size Error Register

The EMAC_EST_FRMSZ_ERR register provides the one hot encoded queue numbers that are having the frame size-related error.

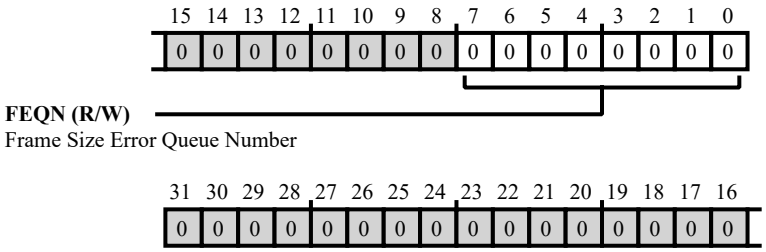


Figure 28-207: EMAC_EST_FRMSZ_ERR Register Diagram

Table 28-262: EMAC_EST_FRMSZ_ERR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FEQN	Frame Size Error Queue Number. The EMAC_EST_FRMSZ_ERR.FEQN field indicates the one hot encoded queue numbers that have experienced the error described in the HLBF field of status register. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.

EST Gate Control List Control Register

The EMAC_EST_GCL_CTL register provides the control information for reading/writing to the gate control lists.

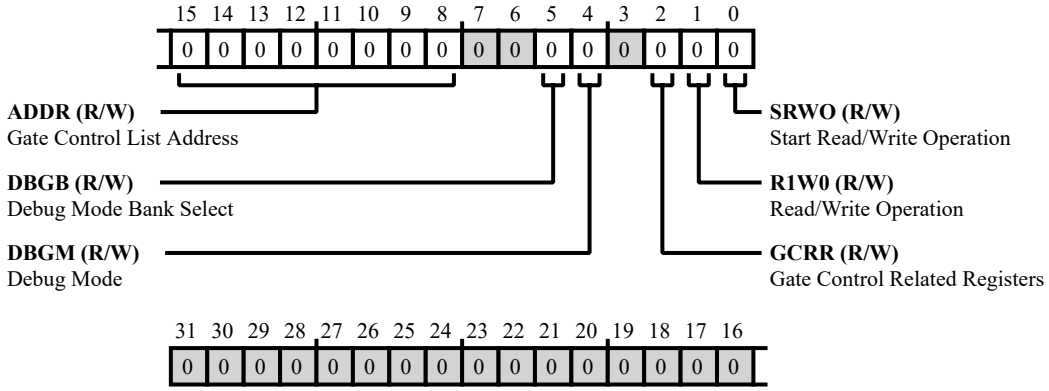


Figure 28-208: EMAC_EST_GCL_CTL Register Diagram

Table 28-263: EMAC_EST_GCL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	ADDR	Gate Control List Address. When set (=1), the EMAC_EST_GCL_CTL.ADDR field indicates the address (row number) of the GCL at which the R/W operation must be performed. By default, the GCL pointed by EMAC_EST_STAT.SWOL is selected for R/W. However, if EMAC_EST_GCL_CTL.DBGM=1, a debug mode access is given to R/W from EMAC_EST_GCL_CTL.DBGB. The width of this field is dependent on DWC_EQOS_EST_DEP; unused bits should be treated as read only. Gate Control List Related Registers Address: (GCRA when GCRR =1). By default, the GCL-related register set pointed by EMAC_EST_STAT.SWOL is selected for R/W. However, if EMAC_EST_GCL_CTL.DBGM=1, debug mode access is given to R/W from EMAC_EST_GCL_CTL.DBGB. Lower 3 bits are only used in this mode, higher order bits are treated as dont care. 000: BTR Low (31:0) 001: BTR High (63:31) 010: CTR Low (31:0) 011: CTR High (39:32) 100: TER (31:0) 101: LLR (8:0) Others: Reserved

Table 28-263: EMAC_EST_GCL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DBGB	Debug Mode Bank Select. When the EMAC_EST_GCL_CTL.DBGB bit is cleared (=0), it indicates R/W in debug mode is directed to bank 0 (GCL0 and corresponding time-related registers). When set (=1), the EMAC_EST_GCL_CTL.DBGB bit indicates R/W in debug mode should be directed to bank 1 (GCL1 and corresponding time-related registers). This value is used when EMAC_EST_GCL_CTL.DBGB = 1 and overrides by value of SWOL (which is normally used).	
		0	R/W in Debug Mode Should Be Directed to Bank 0
		1	R/W in Debug Mode Should Be Directed to Bank 1
4 (R/W)	DBGM	Debug Mode. When set (=1), the EMAC_EST_GCL_CTL.DBGM bit indicates R/W in debug mode where the memory bank (for GCL and time related registers) is explicitly provided by DBGB value. When the EMAC_EST_GCL_CTL.DBGM bit is cleared (=0), the SWOL bit is used to determine which bank to use.	
		0	Debug Mode is Disabled
		1	Debug Mode is Enabled
2 (R/W)	GCRR	Gate Control Related Registers. When set (=1), the bit indicates that the R/W access is for the GCL related registers (BTR, CTR, TER, LLR) whose address is provided by GCRA. When cleared (=0), the bit indicates that the R/W is directed to GCL from the address provided by GCLA.	
		0	Gate Control Related Registers are Disabled
		1	Gate Control Related Registers are Enabled
1 (R/W)	R1W0	Read/Write Operation. The EMAC_EST_GCL_CTL.R1W0 bit indicates a read or write operation.	
		0	Write Operation
		1	Read Operation

Table 28-263: EMAC_EST_GCL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	SRWO	Start Read/Write Operation. When set (=1), the bit indicates the start/progress of a read/write operation. When cleared (=0) by hardware, the bit indicates that the read/write operation has completed or an error has occurred (when bit 20 is set). For read operations, data can be read from EMAC_EST_GCL_DAT register after `EMAC_EST_GCL_CTL.SRWO` = 0. For write operations, the EMAC_EST_GCL_DAT register should be programmed with write data before setting the `EMAC_EST_GCL_CTL.SRWO` bit. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Start Read/Write Op Disabled
		1	Start Read/Write Op Enabled

EST Gate Control List Data Register

The EMAC_EST_GCL_DAT register holds the read data or write data in case of reads and writes respectively.

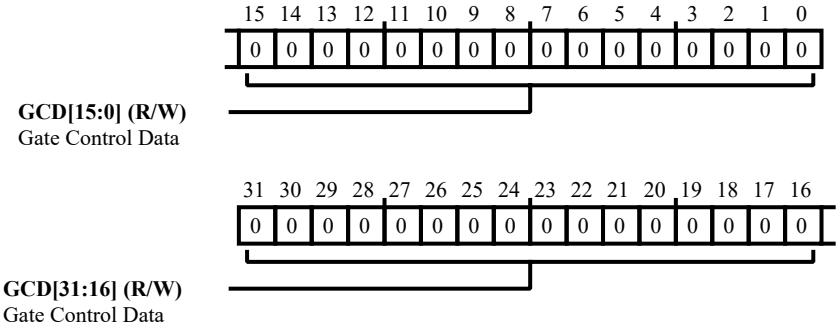


Figure 28-209: EMAC_EST_GCL_DAT Register Diagram

Table 28-264: EMAC_EST_GCL_DAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	GCD	Gate Control Data. The EMAC_EST_GCL_DAT.GCD field contains data corresponding to the address selected in the EMAC_EST_GCL_DAT register. It is used for both read and write operations.

EST Interrupt Enable Register

The EMAC_EST_IEN register implements the interrupt enable bits for the various events that generate an interrupt. Bit positions have a 1-to-1 correlation with the status bit positions in the EMAC_EST_STAT register.

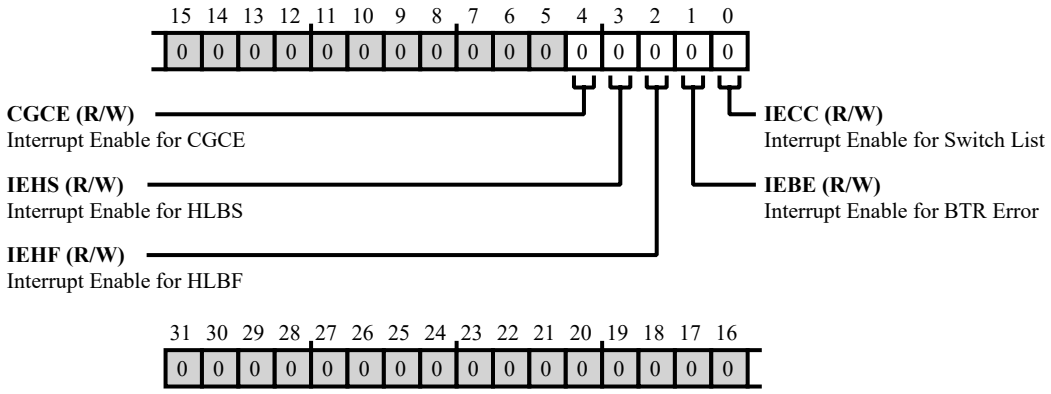


Figure 28-210: EMAC_EST_IEN Register Diagram

Table 28-265: EMAC_EST_IEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CGCE	Interrupt Enable for CGCE. When the EMAC_EST_IEN.CGCE bit is set (=1), an interrupt is generated when the constant gate control error occurs and is indicated in the status. When the EMAC_EST_IEN.CGCE bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for CGCE is Disabled
		1	Interrupt for CGCE is Enabled
3 (R/W)	IEHS	Interrupt Enable for HLBS. When the EMAC_EST_IEN.IEHS bit is set (=1), an interrupt is generated when the head-of-line blocking due to a scheduling issue and is indicated in the status. When the EMAC_EST_IEN.IEHS bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for HLBS is Disabled
		1	Interrupt for HLBS is Enabled
2 (R/W)	IEHF	Interrupt Enable for HLBF. When the EMAC_EST_IEN.IEHF bit is set (=1), an interrupt is generated when the head-of-line blocking due to frame size error occurs and is indicated in the status. When the EMAC_EST_IEN.IEHF bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for HLBF is Disabled
		1	Interrupt for HLBF is Enabled

Table 28-265: EMAC_EST_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	IEBE	Interrupt Enable for BTR Error. When the `EMAC_EST_IEN.IEBE` bit is set (=1), an interrupt is generated when the BTR error occurs and is indicated in the status. When the `EMAC_EST_IEN.IEBE` bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for BTR Error is Disabled
		1	Interrupt for BTR Error is Enabled
0 (R/W)	IECC	Interrupt Enable for Switch List. When the `EMAC_EST_IEN.IECC` bit is set (=1), an interrupt is generated when the configuration change is successful, and the hardware has switched to the new list. When cleared (= 0), no interrupt is generated.	
		0	Interrupt for Switch List is Disabled
		1	Interrupt for Switch List is Enabled

EST Scheduling Error Register

The EMAC_EST_SCH_ERR register provides the one hot encoded queue numbers that are having the scheduling-related error (timeout).

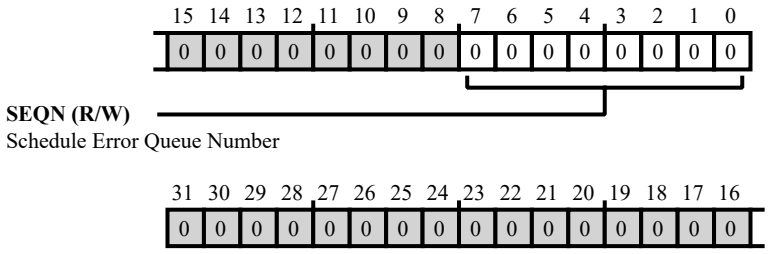


Figure 28-211: EMAC_EST_SCH_ERR Register Diagram

Table 28-266: EMAC_EST_SCH_ERR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	SEQN	Schedule Error Queue Number. The EMAC_EST_SCH_ERR.SEQN field indicates the one hot encoded queue numbers that have experienced error/timeout described in EMAC_EST_STAT.HLBS field. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.

EST Status Register

The EMAC_EST_STAT register provides status information related to Enhancements to Scheduled Transmission (EST) (IEEE802.1Qbv).

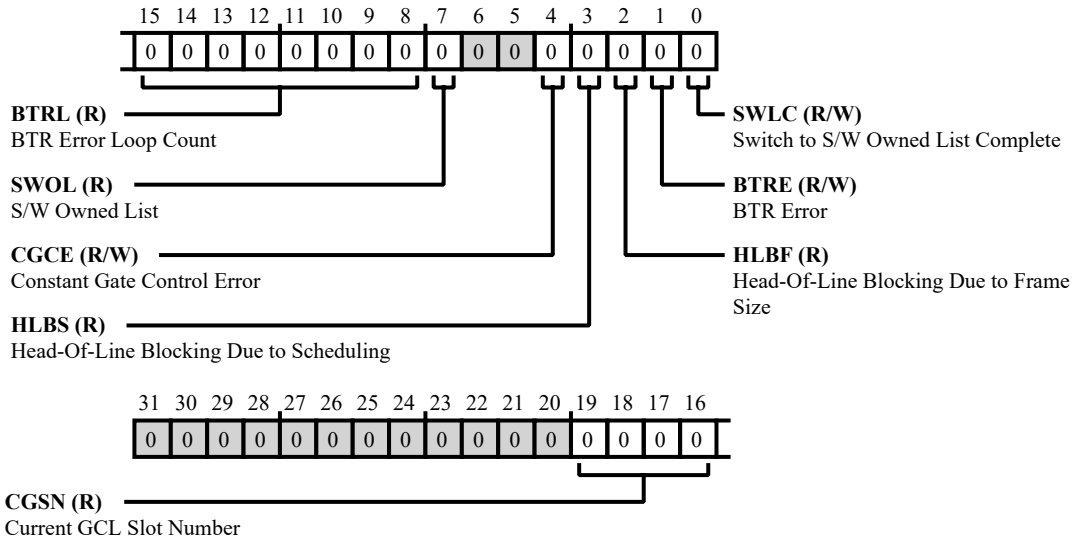


Figure 28-212: EMAC_EST_STAT Register Diagram

Table 28-267: EMAC_EST_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/NW)	CGSN	Current GCL Slot Number. The EMAC_EST_STAT.CGSN field indicates the slot number of the GCL list. Slot number is a modulo 16 count of the GCL list loops executed so far. Even if a new GCL list is installed, the count is incremental.
15:8 (R/NW)	BTRL	BTR Error Loop Count. The EMAC_EST_STAT.BTRL field provides the minimum count (N) for which the equation Current Time =< New BTR + (N * New Cycle Time) becomes true. N = "11111111" indicates the iterations exceeded the value of 128 and the hardware was not able to update New BTR to be equal to or greater than Current Time. Software intervention is needed to update the new BTR. Value is cleared when EMAC_EST_STAT.BTRE = 0.

Table 28-267: EMAC_EST_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SWOL	S/W Owned List. When the `EMAC_EST_STAT.SWOL` bit is cleared (=0), it indicates that the gate control list number "0" is owned by software. When set (=1), the `EMAC_EST_STAT.SWOL` bit indicates that Gate Control list "1" is owned by the software. Any reads/writes by the software (using indirect access through GCL_Control) is directed to the list indicated by this value by default. The inverse of this value is treated as HWOL. R/W operations performed by hardware are directed to the list pointed by HWOL by default.	
		0	Gate Control List Number "0" is Owned by Software
		1	Gate Control List Number "1" is Owned by Software
4 (R/W)	CGCE	Constant Gate Control Error. When set (=1), the `EMAC_EST_STAT.CGCE` bit indicates that an error occurs when the list length (LLR) is 1 and the Cycle Time (CTR) is less than or equal to the programmed Time Interval (TI) value after the optional left shifting. This implies that gates are either always closed or always open based on the Gate Control values; the same effect can be achieved by other simpler (non TSN) programming mechanisms. Since the implementation does not support such a programming an error is reported. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Constant Gate Control Error Not Detected
		1	Constant Gate Control Error Detected
3 (R/NW)	HLBS	Head-Of-Line Blocking Due to Scheduling. When set (=1), the `EMAC_EST_STAT.HLBS` bit indicates when the frame is not able to win arbitration and get scheduled even after four iterations of the GCL. Indicates to software a potential programming error. The one hot encoded values of the queue numbers that are not able to make progress are indicated in the EMAC_EST_SCH_ERR register. The bit is cleared when the EMAC_EST_SCH_ERR register is all zeros.	
		0	Head-Of-Line Blocking Due to Scheduling Not Detected
		1	Head-Of-Line Blocking Due to Scheduling Detected

Table 28-267: EMAC_EST_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	HLBF	Head-Of-Line Blocking Due to Frame Size. When set (=1), the EMAC_EST_STAT.HLBF bit indicates HOL Blocking is noticed on one or more queues as a result of none of the time intervals of gate open in the GCL being greater than or equal to the duration needed for frame size (or frame fragment size when preemption is enabled) transmission. The one hot encoded queue numbers that are experiencing HLBF are indicated in the EMAC_EST_FRMSZ_ERR register. Additionally, the first queue number that experienced HLBF along with the frame size is captured in EMAC_EST_FRMSZ_CAP register. The EMAC_EST_STAT.HLBF bit is cleared when EMAC_EST_FRMSZ_ERR register is all zeros.	
		0	Head-Of-Line Blocking Due to Frame Size Not Detected
		1	Head-Of-Line Blocking Due to Frame Size Detected
1 (R/W)	BTRE	BTR Error. When set (=1), the EMAC_EST_STAT.BTRE bit indicates a programming error in the BTR of SWOL where the programmed value is less than current time. If EMAC_EST_STAT.BTRL = "11111111", EMAC_EST_STAT.SWOL is not updated and software should reprogram the BTR to a value greater than current time and then set the SSWL to reinitiate the switch to EMAC_EST_STAT.SWOL. Else, if the value of EMAC_EST_STAT.BTRL < "11111111", the EMAC_EST_STAT.SWOL bit is updated and this field indicates the number of iterations (of + CycleTime) taken by hardware to update the BTR to a value greater than Current Time. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	BTR Error Not Detected
		1	BTR Error Detected
0 (R/W)	SWLC	Switch to S/W Owned List Complete. When EMAC_EST_STAT.SWLC= 1, the hardware has successfully switched to the SWOL, and the EMAC_EST_STAT.SWOL bit has been updated to that effect. The EMAC_EST_STAT.SWLC bit is cleared when the EMAC_EST_CTL.SSWL bit transitions from 0 to 1, or on a software write. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Switch to S/W Owned List Complete Not Detected
		1	Switch to S/W Owned List Complete Detected

Frame Preemption Advance Register

The EMAC_FPE_ADV register contains the hold and release advance time.

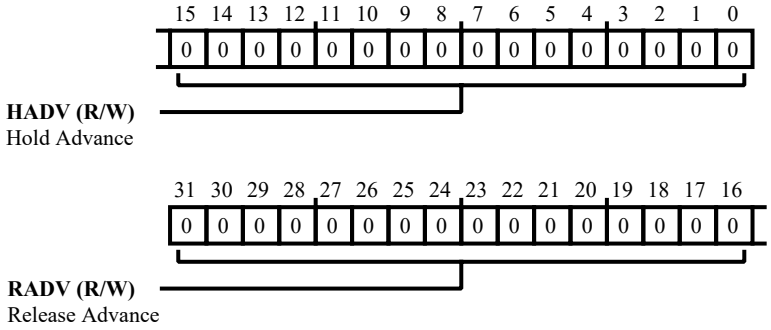


Figure 28-213: EMAC_FPE_ADV Register Diagram

Table 28-268: EMAC_FPE_ADV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RADV	Release Advance. The EMAC_FPE_ADV.RADV field indicates the maximum time in nanoseconds that can elapse between issuing a RELEASE to the MAC and the MAC being ready to resume transmission of preemptable frames, in the absence of there being any express frames available for transmission.
15:0 (R/W)	HADV	Hold Advance. The EMAC_FPE_ADV.HADV field indicates the maximum time in nanoseconds that can elapse between issuing a HOLD to the MAC, and the MAC ceasing to transmit any preemptable frame that is in the process of transmission or any preemptable frames that are queued for transmission.

MTL Frame Preemption Control and Status Register

The EMAC_FPE_CTRLSTS register controls the operation of, and provides status for frame preemption (IEEE802.1Qbu/802.3br).

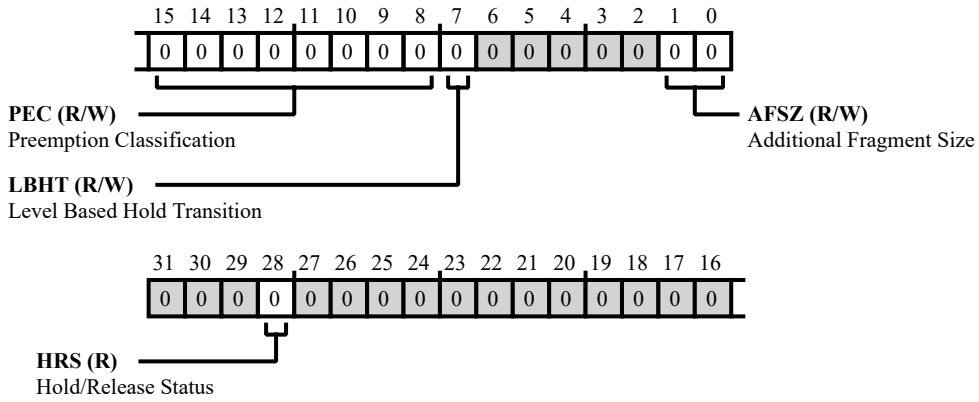


Figure 28-214: EMAC_FPE_CTRLSTS Register Diagram

Table 28-269: EMAC_FPE_CTRLSTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/NW)	HRS	Hold/Release Status. The EMAC_FPE_CTRLSTS.HRS bit indicates whether a set-and-hold MAC operation was last executed and the pMAC is in hold state or a set-and-release MAC operation was last executed and the pMAC is in the release state.	
		0	Release
		1	Hold
15:8 (R/W)	PEC	Preemption Classification. When set (=1), the EMAC_FPE_CTRLSTS.PEC field indicates that the corresponding queue must be classified as preemptable, when '0' queue is classified as express. When both EST (Qbv) and preemption are enabled, queue 0 is always assumed to be preemptable. When EST (Qbv) is enabled, queues categorized as preemptable here are always assumed to be in "Open" state in the Gate Control List.	
7 (R/W)	LBHT	Level Based Hold Transition. WhenEMAC_FPE_CTRLSTS.LBHT =1, the bit enables a hold operation in the second row of the EST GCL when a hold request is set in the second GCL row for the first loop of a new GCL, even though hold request is also set in the first GCL row. This is a deviation from the original behavior, where a transition from a release request to a hold request is required. When cleared (= 0), enables hold operation only on transition from release request to hold request.	

Table 28-269: EMAC_FPE_CTRLSTS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	AFSZ	Additional Fragment Size. The `EMAC_FPE_CTRLSTS.AFSZ` field indicates, in units of 64 bytes, the minimum number of bytes over 64 bytes required in non-final fragments of preempted frames. The minimum non-final fragment size is (AFSZ +1) * 64 bytes.

MTL Interrupt Status Register

The software driver (application) reads the EMAC_MTL_ISTAT register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

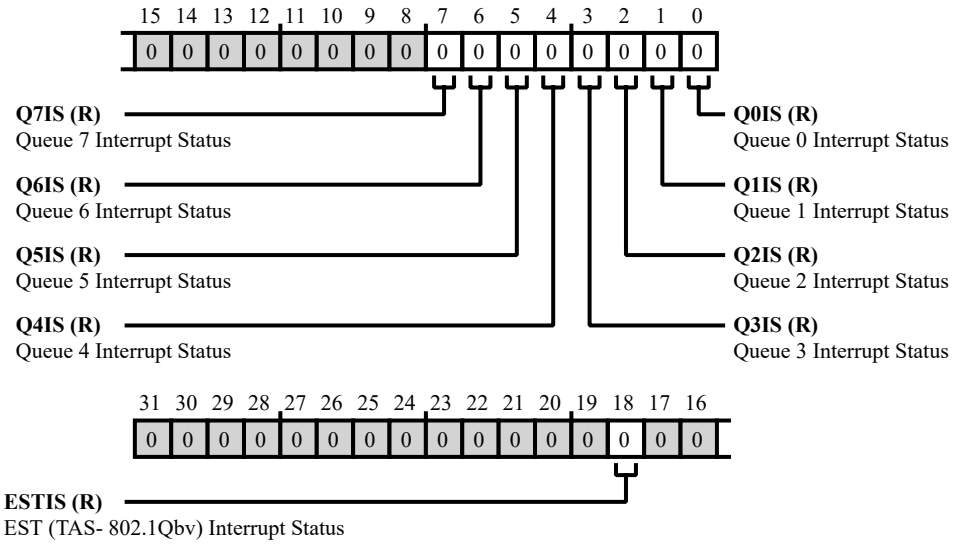


Figure 28-215: EMAC_MTL_ISTAT Register Diagram

Table 28-270: EMAC_MTL_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	ESTIS	EST (TAS- 802.1Qbv) Interrupt Status. The EMAC_MTL_ISTAT.ESTIS bit indicates an interrupt event during the operation of 802.1Qbv. To reset the EMAC_MTL_ISTAT.ESTIS bit, the application must clear the error/event that has caused the Interrupt.	
		0	EST (TAS- 802.1Qbv) Interrupt Status Not Detected
		1	EST (TAS- 802.1Qbv) Interrupt Status Detected
7 (R/NW)	Q7IS	Queue 7 Interrupt Status. The EMAC_MTL_ISTAT.Q7IS bit indicates that there is an interrupt from queue 7. To reset the EMAC_MTL_ISTAT.Q7IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected

Table 28-270: EMAC_MTL_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	Q6IS	Queue 6 Interrupt Status. The EMAC_MTL_ISTAT.Q6IS bit indicates that there is an interrupt from queue 6. To reset the EMAC_MTL_ISTAT.Q6IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
5 (R/NW)	Q5IS	Queue 5 Interrupt Status. The EMAC_MTL_ISTAT.Q5IS bit indicates that there is an interrupt from queue 5. To reset the EMAC_MTL_ISTAT.Q5IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
4 (R/NW)	Q4IS	Queue 4 Interrupt Status. The EMAC_MTL_ISTAT.Q4IS bit indicates that there is an interrupt from queue 4. To reset the EMAC_MTL_ISTAT.Q4IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
3 (R/NW)	Q3IS	Queue 3 Interrupt Status. The EMAC_MTL_ISTAT.Q3IS bit indicates that there is an interrupt from queue 3. To reset the EMAC_MTL_ISTAT.Q3IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
2 (R/NW)	Q2IS	Queue 2 Interrupt Status. The EMAC_MTL_ISTAT.Q2IS bit indicates that there is an interrupt from queue 2. To reset the EMAC_MTL_ISTAT.Q2IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected

Table 28-270: EMAC_MTL_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	Q1IS	Queue 1 Interrupt Status. The `EMAC_MTL_ISTAT.Q1IS` bit indicates that there is an interrupt from queue 1. To reset the `EMAC_MTL_ISTAT.Q1IS` bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
0 (R/NW)	Q0IS	Queue 0 Interrupt Status. The `EMAC_MTL_ISTAT.Q0IS` bit indicates that there is an interrupt from queue 0. To reset the `EMAC_MTL_ISTAT.Q0IS` bit, the application must read the EMAC_Q0_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected

MTL Operation Mode Register

The EMAC_MTL_OPMODE register establishes the transmit and receive operating modes and commands.

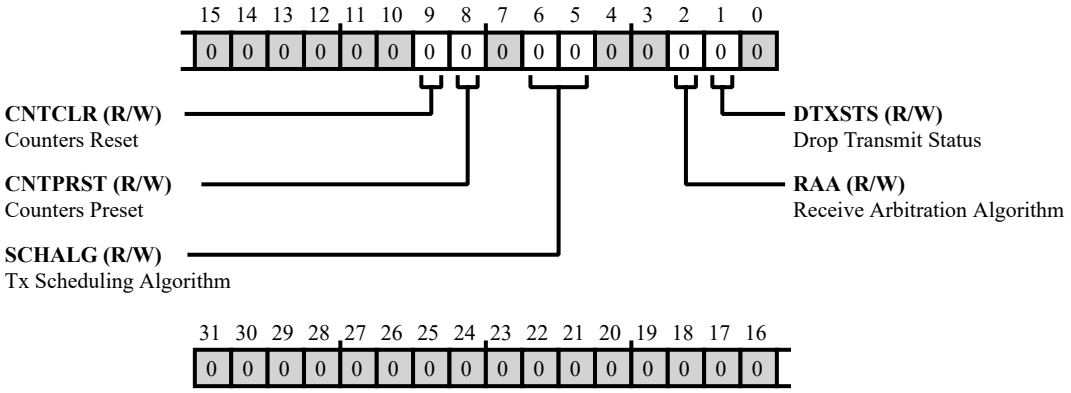


Figure 28-216: EMAC_MTL_OPMODE Register Diagram

Table 28-271: EMAC_MTL_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	CNTCLR	Counters Reset. When EMAC_MTL_OPMODE.CNTCLR bit = 1, all counters are reset. The EMAC_MTL_OPMODE.CNTCLR bit is cleared automatically after 1 clock cycle. If EMAC_MTL_OPMODE.CNTCLR and EMAC_MTL_OPMODE.CNTPRST are both set (=1), the EMAC_MTL_OPMODE.CNTPRST bit has precedence. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Counters are Not Reset
		1	All Counters are Reset
8 (R/W)	CNTPRST	Counters Preset. When set (=1), the EMAC_MTL_OPMODE.CNTPRST bit indicates that the EMAC_TQ[n]_URFL and EMAC_RQ[n]_MSPKTOF_CNT registers are initialized and preset to 12'h7F0. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable

Table 28-271: EMAC_MTL_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	SCHALG	Tx Scheduling Algorithm. The `EMAC_MTL_OPMODE.SCHALG` field indicates the algorithm for Tx scheduling.	
		0	WRR Algorithm
		1	WFQ Algorithm
		2	DWRR Algorithm
		3	Strict Priority Algorithm
2 (R/W)	RAA	Receive Arbitration Algorithm. The `EMAC_MTL_OPMODE.RAA` bit indicates the arbitration algorithm for the Rx side.	
		0	Strict Priority (SP). Queue 0 has the lowest priority and the last queue has the highest priority.
		1	Weighted Strict Priority (WSP)
1 (R/W)	DTXSTS	Drop Transmit Status. When set, the `EMAC_MTL_OPMODE.DTXSTS` bit indicates that the Tx packet status received from the MAC is dropped in the MTL. When cleared (=0), the packet status is forwarded to the application.	
		0	Disable
		1	Enable

Queue 0 Interrupt Control and Status Register

The EMAC_Q0_INT_CTLSTAT register contains the interrupt enable and status bits for the queue 0 interrupts.

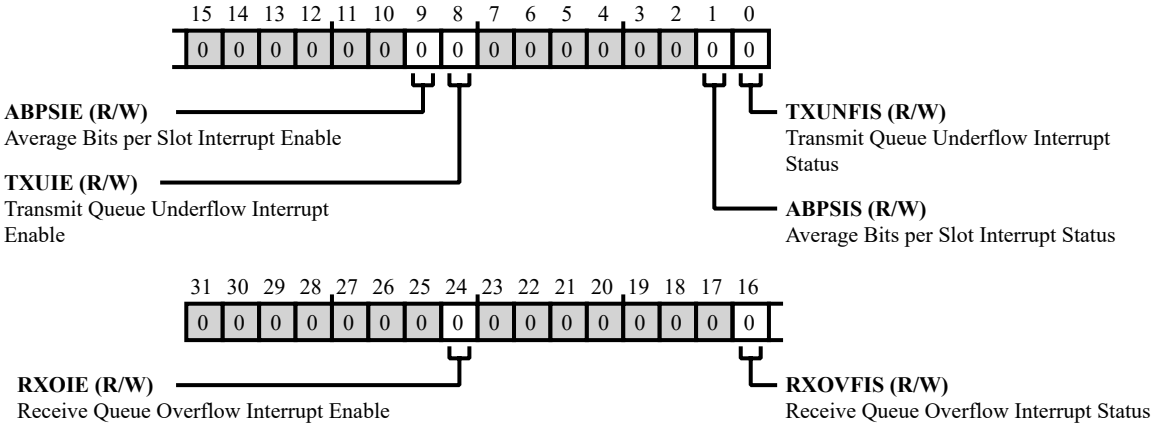


Figure 28-217: EMAC_Q0_INT_CTLSTAT Register Diagram

Table 28-272: EMAC_Q0_INT_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	RXOIE	Receive Queue Overflow Interrupt Enable. When the EMAC_Q0_INT_CTLSTAT.RXOIE bit is set, the Receive Queue Overflow interrupt is enabled. When EMAC_Q0_INT_CTLSTAT.RXOIE = 0, the Receive Queue Overflow interrupt is disabled.	
		0	Receive Queue Overflow Interrupt is Disabled
		1	Receive Queue Overflow Interrupt is Enabled
16 (R/W)	RXOVFIS	Receive Queue Overflow Interrupt Status. The EMAC_Q0_INT_CTLSTAT.RXOVFIS bit indicates that the receive queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. The EMAC_Q0_INT_CTLSTAT.RXOVFIS bit is cleared when the application writes 1 to the EMAC_Q0_INT_CTLSTAT.RXOVFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Queue Overflow Interrupt Status Not Detected
		1	Receive Queue Overflow Interrupt Status Detected

Table 28-272: EMAC_Q0_INT_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ABPSIE	Average Bits per Slot Interrupt Enable. When the EMAC_Q0_INT_CTLSTAT.ABPSIE bit is set, the MAC asserts the EMAC_STAT or EMAC_MAC interrupt when the average bits per slot status is updated. When the EMAC_Q0_INT_CTLSTAT.ABPSIE bit is cleared, the interrupt is not asserted for such an event.	
		0	Average Bits per Slot Interrupt is Disabled
		1	Average Bits per Slot Interrupt is Enabled
8 (R/W)	TXUIE	Transmit Queue Underflow Interrupt Enable. When the EMAC_Q0_INT_CTLSTAT.TXUIE bit is set, the Transmit Queue Underflow interrupt is enabled. When EMAC_Q0_INT_CTLSTAT.TXUIE = 0, the Transmit Queue Underflow interrupt is disabled.	
		0	Transmit Queue Underflow Interrupt Status is Disabled
		1	Transmit Queue Underflow Interrupt Status is Enabled
1 (R/W)	ABPSIS	Average Bits per Slot Interrupt Status. When set, the EMAC_Q0_INT_CTLSTAT.ABPSIS bit indicates that the MAC has updated the ABS value. The EMAC_Q0_INT_CTLSTAT.ABPSIS bit is cleared when the application writes 1 to the EMAC_Q0_INT_CTLSTAT.ABPSIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Average Bits per Slot Interrupt Status Not Detected
		1	Average Bits per Slot Interrupt Status Detected
0 (R/W)	TXUNFIS	Transmit Queue Underflow Interrupt Status. The EMAC_Q0_INT_CTLSTAT.TXUNFIS bit indicates that the transmit queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. The EMAC_Q0_INT_CTLSTAT.TXUNFIS bit is cleared when the application writes 1 to the EMAC_Q0_INT_CTLSTAT.TXUNFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Queue Underflow Interrupt Status Not Detected
		1	Transmit Queue Underflow Interrupt Status Detected

Queue n Interrupt Control and Status Register

The EMAC_Q[n]_INT_CTLSTAT register contains the interrupt enable and status bits for the queue 1 interrupts.

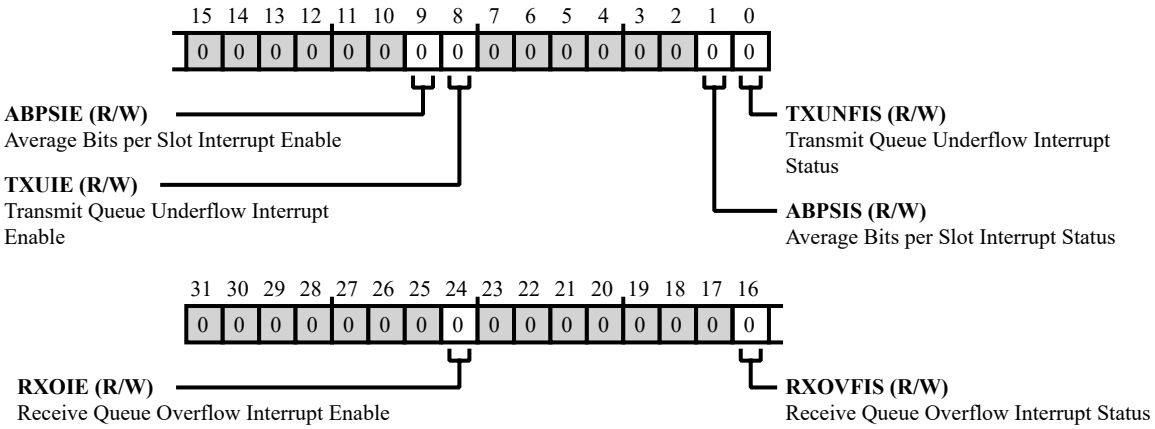


Figure 28-218: EMAC_Q[n]_INT_CTLSTAT Register Diagram

Table 28-273: EMAC_Q[n]_INT_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	RXOIE	Receive Queue Overflow Interrupt Enable. When the EMAC_Q[n]_INT_CTLSTAT.RXOIE bit is set, the receive queue overflow interrupt interrupt is enabled. When EMAC_Q[n]_INT_CTLSTAT.RXOIE = 0, the receive queue overflow interrupt is disabled.	
		0	Disabled
		1	Enabled
16 (R/W)	RXOVFIS	Receive Queue Overflow Interrupt Status. The EMAC_Q[n]_INT_CTLSTAT.RXOVFIS bit indicates that the receive queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. The EMAC_Q[n]_INT_CTLSTAT.RXOVFIS bit is cleared when the application writes 1 to the EMAC_Q[n]_INT_CTLSTAT.RXOVFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Status Not Detected
		1	Status Detected

Table 28-273: EMAC_Q[n]_INT_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ABPSIE	Average Bits per Slot Interrupt Enable. When the EMAC_Q[n]_INT_CTLSTAT.ABPSIE bit is set, the MAC asserts the EMAC_STAT or EMAC_MAC interrupt when the average bits per slot status is updated. When the EMAC_Q[n]_INT_CTLSTAT.ABPSIE bit is cleared, the interrupt is not asserted for such an event.	
		0	Disabled
		1	Enabled
8 (R/W)	TXUIE	Transmit Queue Underflow Interrupt Enable. When the EMAC_Q[n]_INT_CTLSTAT.TXUIE bit is set, the transmit queue underflow interrupt is enabled. When EMAC_Q[n]_INT_CTLSTAT.TXUIE = 0, the transmit queue underflow interrupt is disabled.	
		0	Disabled
		1	Enabled
1 (R/W)	ABPSIS	Average Bits per Slot Interrupt Status. When set, the EMAC_Q[n]_INT_CTLSTAT.ABPSIS bit indicates that the MAC has updated the ABS value. The EMAC_Q[n]_INT_CTLSTAT.ABPSIS bit is cleared when the application writes 1 to the EMAC_Q[n]_INT_CTLSTAT.ABPSIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Status Not Detected
		1	Status Detected
0 (R/W)	TXUNFIS	Transmit Queue Underflow Interrupt Status. The EMAC_Q[n]_INT_CTLSTAT.TXUNFIS bit indicates that the transmit queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. The EMAC_Q[n]_INT_CTLSTAT.TXUNFIS bit is cleared when the application writes 1 to the EMAC_Q[n]_INT_CTLSTAT.TXUNFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Status Not Detected
		1	Status Detected

Receive Queue Control Register

The EMAC_RQ0_CTL register controls the receive arbitration and passing of received packets to the application.

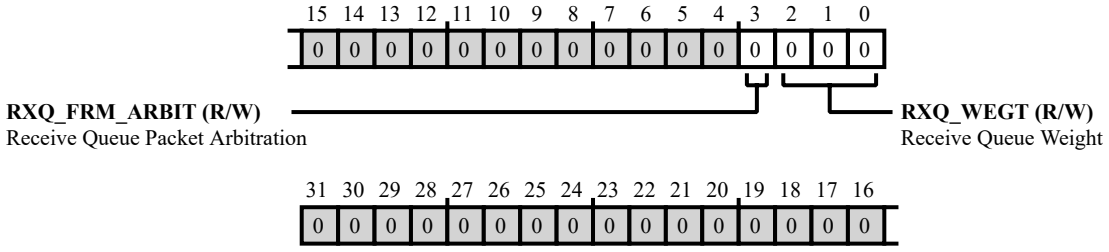


Figure 28-219: EMAC_RQ0_CTL Register Diagram

Table 28-274: EMAC_RQ0_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXQ_FRM_ARBIT	Receive Queue Packet Arbitration. When the EMAC_RQ0_CTL.RXQ_FRM_ARBIT bit is set, the EMAC module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When EMAC_RQ0_CTL.RXQ_FRM_ARBIT = 0, the EMAC module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: - PBL amount of data or - Complete data of a packet The status and the time stamp are not a part of the PBL data. Therefore, the EMAC module drives the complete status (including time stamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in threshold mode).	
		0	Receive Queue Packet Arbitration is Disabled
		1	Receive Queue Packet Arbitration is Enabled

Table 28-274: EMAC_RQ0_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	RXQ_WEGT	Receive Queue Weight. The EMAC_RQ0_CTL.RXQ_WEGT field indicates the weight assigned to the Rx queue 0. The field must be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the EMAC_RQ0_CTL.RXQ_WEGT value to take effect at the start, the EMAC_RQ[n]_CTL registers must be programmed before the EMAC_MTL_OPMODE register.

Receive Queue 0 Debug Register

The EMAC_RQ0_DBG register gives the debug status of various blocks related to the receive queue.

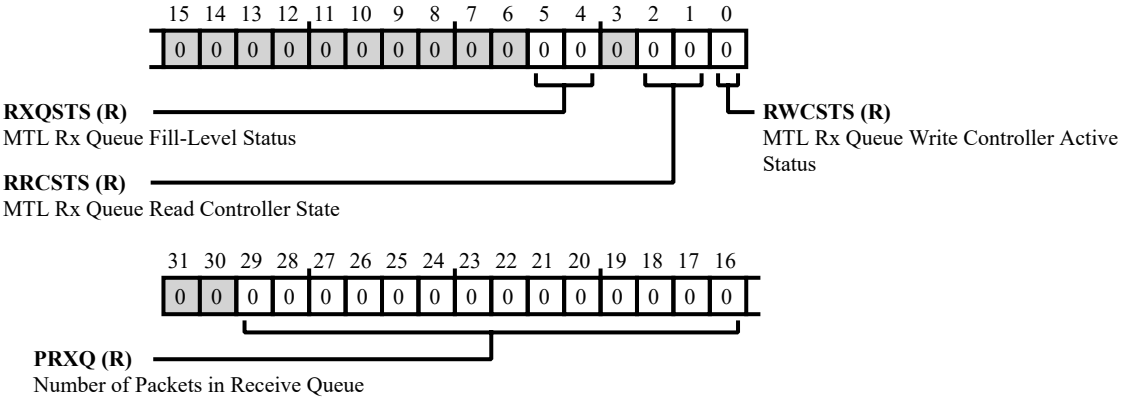


Figure 28-220: EMAC_RQ0_DBG Register Diagram

Table 28-275: EMAC_RQ0_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:16 (R/NW)	PRXQ	Number of Packets in Receive Queue. The EMAC_RQ0_DBG.PRXQ field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.	
5:4 (R/NW)	RXQSTS	MTL Rx Queue Fill-Level Status. The EMAC_RQ0_DBG.RXQSTS field gives the status of the fill-level of the Rx queue:	
		0	Rx Queue Empty
		1	Rx Queue Fill-level Below Flow-control Deactivate Threshold
		2	Rx Queue Fill-level Above Flow-control Activate Threshold
		3	Rx Queue Full
2:1 (R/NW)	RRCSTS	MTL Rx Queue Read Controller State. The EMAC_RQ0_DBG.RRCSTS field gives the state of the Rx queue read controller:	
		0	Idle State
		1	Reading Packet Data
		2	Reading Packet Status (or Timestamp)
		3	Flushing the Packet Data and Status

Table 28-275: EMAC_RQ0_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	RWCSTS	MTL Rx Queue Write Controller Active Status. When high, the `EMAC_RQ0_DBG.RWCSTS` bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.	
		0	MTL Rx Queue Write Controller Active Status Not Detected
		1	MTL Rx Queue Write Controller Active Status Detected

Receive Queue 0 Missed Packet and Overflow Counter Register

The EMAC_RQ0_MSPKTOF_CNT register contains the counter for packets missed because of receive queue packet flush and packets discarded because of receive queue overflow.

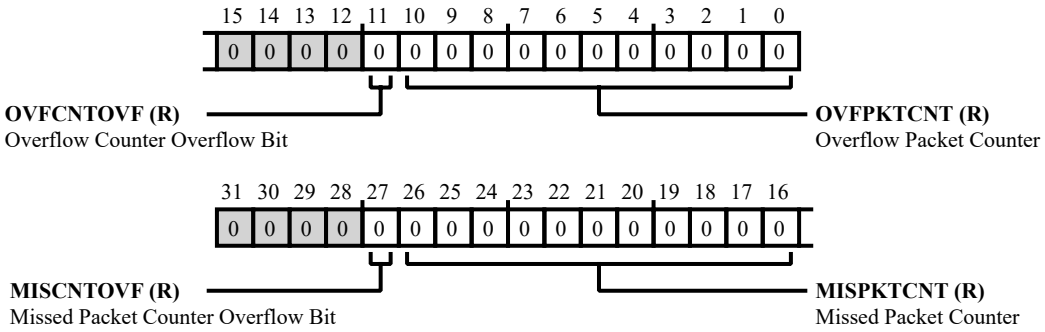


Figure 28-221: EMAC_RQ0_MSPKTOF_CNT Register Diagram

Table 28-276: EMAC_RQ0_MSPKTOF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	MISCNTOVF	Missed Packet Counter Overflow Bit. When set, the EMAC_RQ0_MSPKTOF_CNT.MISCNTOVF bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Missed Packet Counter Overflow Not Detected
		1	Missed Packet Counter Overflow Detected
26:16 (R/NW)	MISPKTCNT	Missed Packet Counter. The EMAC_RQ0_MSPKTOF_CNT.MISPKTCNT field indicates the number of packets missed by the EMAC module because the application asserted EMAC_DMA[n]_RXCTL.RPF for this queue. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
11 (R/NW)	OVFCNTOVF	Overflow Counter Overflow Bit. When set, the EMAC_RQ0_MSPKTOF_CNT.OVFCNTOVF bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Counter Overflow Not Detected
		1	Overflow Counter Overflow Detected

Table 28-276: EMAC_RQ0_MSPKTOF_CNT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	OVFPKTCNT	Overflow Packet Counter. The `EMAC_RQ0_MSPKTOF_CNT.OVFPKTCNT` field indicates the number of packets discarded by the EMAC module because of receive queue overflow. This counter is incremented each time the EMAC module discards an incoming packet because of overflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Receive Queue 0 Operation Mode Register

The EMAC_RQ0_OPMODE register establishes the receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release.

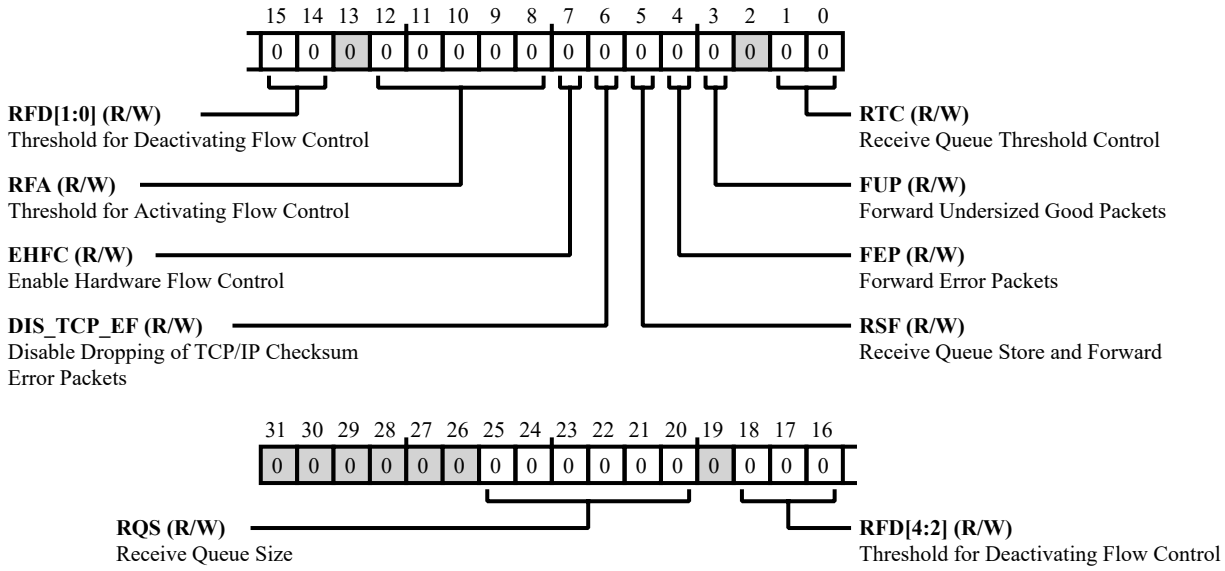


Figure 28-222: EMAC_RQ0_OPMODE Register Diagram

Table 28-277: EMAC_RQ0_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:20 (R/W)	RQS	Receive Queue Size. The EMAC_RQ0_OPMODE.RQS field indicates the size of the allocated receive queues in blocks of 256 bytes. The field is read-write only if the number of Rx queues more than one, the reset value is 0x0 and indicates size of 256 bytes. For example, the value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (RQS+1)*256 bytes. When the number of Rx queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value.

Table 28-277: EMAC_RQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18:14 (R/W)	RFD	Threshold for Deactivating Flow Control. The EMAC_RQ0_OPMODE.RFD field indicates the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation. It is valid in half-duplex and full-duplex mode). 0: Full minus 1 KB, that is, FULL 1 KB 1: Full minus 1.5 KB, that is, FULL 1.5 KB 2: Full minus 2 KB, that is, FULL 2 KB 3: Full minus 2.5 KB, that is, FULL 2.5 KB ... 30: Full minus 16 KB, that is, FULL 16 KB 31: Full minus 16.5 KB, that is, FULL 16.5 KB The deassertion is effective only after flow control is asserted. Note: The value must be programmed in such a way to make sure that the threshold is a positive number. When the EMAC_RQ0_OPMODE.EHFC bit is set (=1), these values are applicable only when the Rx queue size determined by the EMAC_RQ0_OPMODE.RQS field, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the EMAC_RQ0_OPMODE.RQS field.	
12:8 (R/W)	RFA	Threshold for Activating Flow Control. The EMAC_RQ0_OPMODE.RFA field indicates the threshold (fill-level of Rx queue) at which the flow control is activated. It is valid in half-duplex and full-duplex mode). For more information on encoding for this field, see RFD.	
7 (R/W)	EHFC	Enable Hardware Flow Control. When the EMAC_RQ0_OPMODE.EHFC bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When cleared (=0), the flow control operation is disabled.	
		0	Hardware Flow Control is Disabled
		1	Hardware Flow Control is Enabled

Table 28-277: EMAC_RQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets. When the EMAC_RQ0_OPMODE.DIS_TCP_EF bit is set, the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When EMAC_RQ0_OPMODE.DIS_TCP_EF = 0, all error packets are dropped if EMAC_RQ0_OPMODE.FEP = 0.	
		0	Dropping of TCP/IP Checksum Error Packets is Enabled
		1	Dropping of TCP/IP Checksum Error Packets is Disabled
5 (R/W)	RSF	Receive Queue Store and Forward. When the EMAC_RQ0_OPMODE.RSF bit is set, the EMAC module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the EMAC_RQ0_OPMODE.RTC field. When EMAC_RQ0_OPMODE.RSF = 0, the Rx queue operates in the threshold (cut-through) mode, subject to the threshold specified by the EMAC_RQ0_OPMODE.RTC field.	
		0	Receive Queue Store and Forward is Disabled
		1	Receive Queue Store and Forward is Enabled
4 (R/W)	FEP	Forward Error Packets. When EMAC_RQ0_OPMODE.FEP = 0, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in threshold mode), the packet is not dropped. When the EMAC_RQ0_OPMODE.FEP bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the EMAC_RQ0_OPMODE.RSF = 1 and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of the EMAC_RQ0_OPMODE.FEP bit. However, if the EMAC_RQ0_OPMODE.RSF = 0 and the Rx queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA.	
		0	Forward Error Packets is Disabled
		1	Forward Error Packets is Enabled

Table 28-277: EMAC_RQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FUP	Forward Undersized Good Packets. When the EMAC_RQ0_OPMODE.FUP bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When EMAC_RQ0_OPMODE.FUP = 0, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx threshold, for example, RTC = 01.	
		0	Forward Undersized Good Packets is Disabled
		1	Forward Undersized Good Packets is Enabled
1:0 (R/W)	RTC	Receive Queue Threshold Control. The EMAC_RQ0_OPMODE.RTC field indicates the threshold level of the MTL Rx queue (in bytes). The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. The EMAC_RQ0_OPMODE.RTC field is valid only when EMAC_RQ0_OPMODE.RSF = 0. The EMAC_RQ0_OPMODE.RTC field is ignored when EMAC_RQ0_OPMODE.RSF = 1.	
		0	64
		1	32
		2	96
		3	128

Receive Queue n Control Register

The EMAC_RQ[n]_CTL register controls the receive arbitration and passing of received packets to the application.

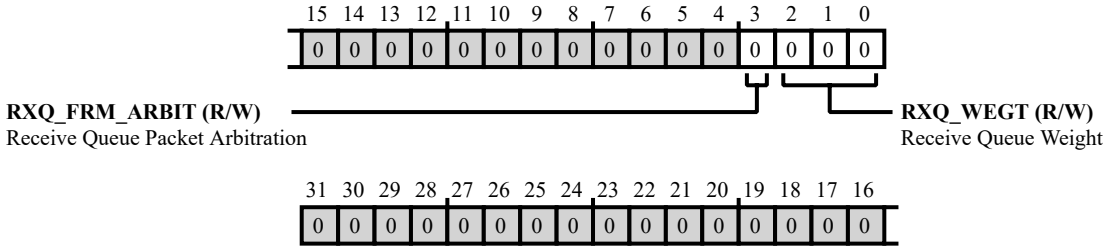


Figure 28-223: EMAC_RQ[n]_CTL Register Diagram

Table 28-278: EMAC_RQ[n]_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXQ_FRM_ARBIT	Receive Queue Packet Arbitration. When the EMAC_RQ[n]_CTL.RXQ_FRM_ARBIT bit is set, the EMAC module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When EMAC_RQ[n]_CTL.RXQ_FRM_ARBIT = 0, the EMAC module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: - PBL amount of data or - Complete data of a packet The status and the time stamp are not a part of the PBL data. Therefore, the EMAC module drives the complete status (including time stamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in threshold mode).	
		0	Disabled
		1	Enabled

Table 28-278: EMAC_RQ[n]_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	RXQ_WEGT	Receive Queue Weight. The `EMAC_RQ[n]_CTL.RXQ_WEGT` field indicates the weight assigned to the Rx queue 0. The `EMAC_RQ[n]_CTL.RXQ_WEGT` field needs to be programmed with one value less than the required weight. For example, a reset value of 0 indicates a weight of 1, a value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the EMAC_RQ[n]_CTL registers must be programmed before the EMAC_MTL_OPMODE register.

Receive Queue n Debug Register

The EMAC_RQ[n]_DBG register gives the debug status of various blocks related to the receive queue.

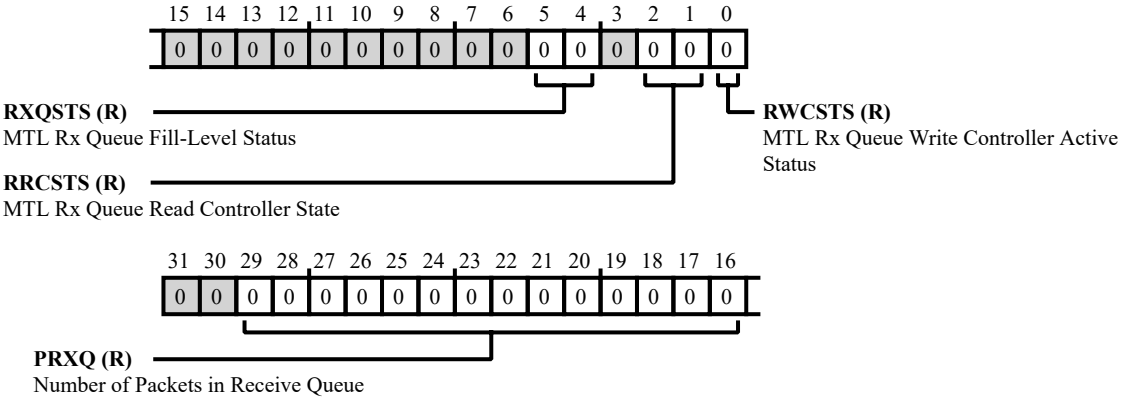


Figure 28-224: EMAC_RQ[n]_DBG Register Diagram

Table 28-279: EMAC_RQ[n]_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:16 (R/NW)	PRXQ	Number of Packets in Receive Queue. The EMAC_RQ[n]_DBG.PRXQ field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256KB/16B = 16K packets, that is, Max_Queue_Size/Min_Packet_Size.	
5:4 (R/NW)	RXQSTS	MTL Rx Queue Fill-Level Status. The EMAC_RQ[n]_DBG.RXQSTS field gives the status of the fill-level of the Rx queue:	
		0	Empty
		1	Below Flow Control Deactivate Threshold
		2	Above Flow Control Activate Threshold
		3	Full
2:1 (R/NW)	RRCSTS	MTL Rx Queue Read Controller State. The EMAC_RQ[n]_DBG.RRCSTS field gives the state of the Rx queue read controller.	
		0	Idle State
		1	Reading Packet Data
		2	Reading Packet Status (or Time Stamp)
		3	Flushing the Packet Data and Status

Table 28-279: EMAC_RQ[n]_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	RWCSTS	MTL Rx Queue Write Controller Active Status. When high, the `EMAC_RQ[n]_DBG.RWCSTS` bit indicates that the MTL Rx queue write controller is active, and it is transferring a received packet to the Rx queue.	
		0	Status Not Detected
		1	Status Detected

Receive Queue n Missed Packet and Overflow Counter Register

The EMAC_RQ[n]_MSPKTOF_CNT register contains the counter for packets missed because of receive queue packet flush and packets discarded because of receive queue overflow.

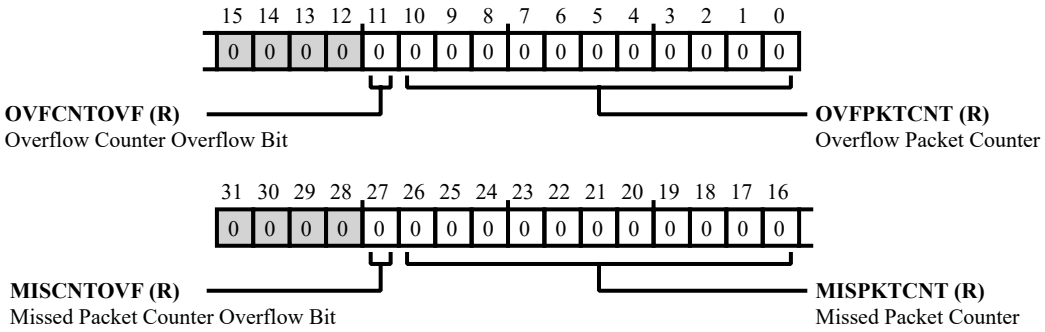


Figure 28-225: EMAC_RQ[n]_MSPKTOF_CNT Register Diagram

Table 28-280: EMAC_RQ[n]_MSPKTOF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	MISCNTOVF	Missed Packet Counter Overflow Bit. When set, the EMAC_RQ[n]_MSPKTOF_CNT.MISCNTOVF bit indicates that the Rx queue missed packet counter crossed the maximum limit. Access restriction applies. Clears (=0) on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected
		1	Overflow Detected
26:16 (R/NW)	MISPKTCNT	Missed Packet Counter. The EMAC_RQ[n]_MSPKTOF_CNT.MISPKTCNT field indicates the number of packets missed by the EMAC module because the application asserted EMAC_DMA[n]_RXCTL.RPF for this queue. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
11 (R/NW)	OVFCNTOVF	Overflow Counter Overflow Bit. When set, the EMAC_RQ[n]_MSPKTOF_CNT.OVFCNTOVF bit indicates that the Rx queue overflow packet counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected
		1	Overflow Detected

Table 28-280: EMAC_RQ[n]_MSPKTOF_CNT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	OVFPKTCNT	Overflow Packet Counter. The `EMAC_RQ[n]_MSPKTOF_CNT.OVFPKTCNT` field indicates the number of packets discarded by the EMAC module because of receive queue overflow. This counter is incremented each time the EMAC module discards an incoming packet because of overflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Receive Queue n Operation Mode Register

The EMAC_RQ[n]_OPMODE register establishes the receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

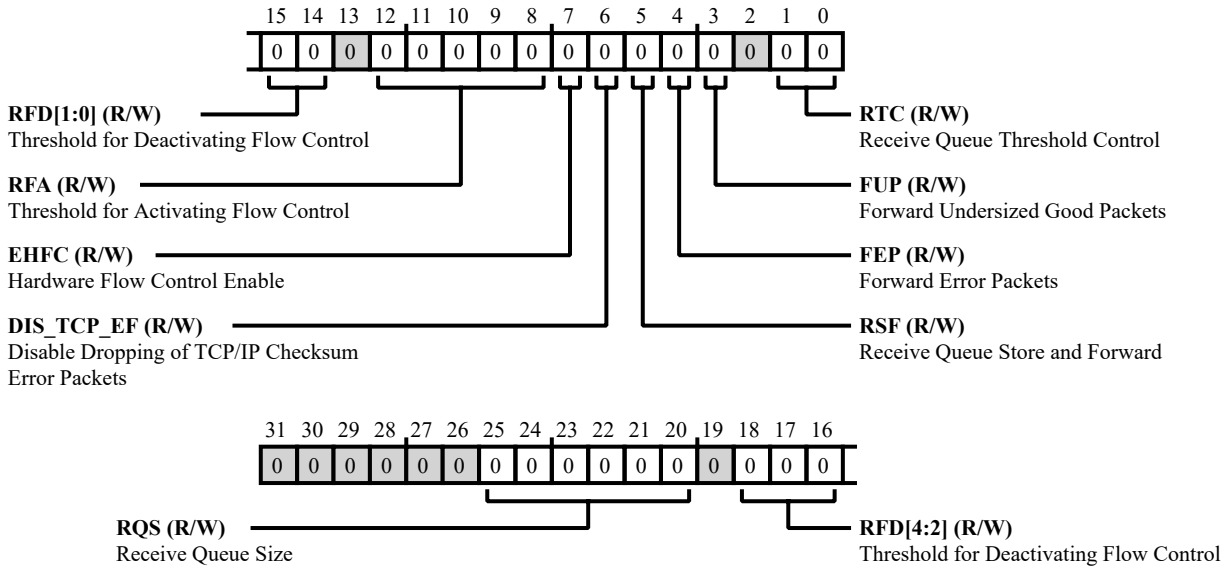


Figure 28-226: EMAC_RQ[n]_OPMODE Register Diagram

Table 28-281: EMAC_RQ[n]_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:20 (R/W)	RQS	Receive Queue Size. The EMAC_RQ[n]_OPMODE.RQS field indicates the size of the allocated receive queues in blocks of 256 bytes. The field is read-write only if the number of Rx queues is more than one; the reset value is 0x0 and indicates size of 256 bytes. For example, a value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. Program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (RQS+1)*256 bytes. When the number of Rx queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value.

Table 28-281: EMAC_RQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18:14 (R/W)	RFD	Threshold for Deactivating Flow Control. The EMAC_RQ[n]_OPMODE.RFD field indicates the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation (in half-duplex and full-duplex modes): - 0: Full minus 1 KB, that is, FULL 1 KB - 1: Full minus 1.5 KB, that is, FULL 1.5 KB - 2: Full minus 2 KB, that is, FULL 2 KB - 3: Full minus 2.5 KB, that is, FULL 2.5 KB - ... - 30: Full minus 16 KB, that is, FULL 16 KB - 31: Full minus 16.5 KB, that is, FULL 16.5 KB The deassertion is effective only after flow control is asserted. Note: The value must be programmed in such a way to make sure that the threshold is a positive number. When EMAC_RQ[n]_OPMODE.EHFC =1, these values are applicable only when the Rx queue size determined by the EMAC_RQ[n]_OPMODE.RQS field, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here, the term FULL and QSIZE refers to the queue size determined by the EMAC_RQ[n]_OPMODE.RQS field.	
12:8 (R/W)	RFA	Threshold for Activating Flow Control. The EMAC_RQ[n]_OPMODE.RFA field indicates the threshold (fill-level of Rx queue) at which the flow control is activated (in half-duplex and full-duplex modes). For more information on encoding for this field, see RFD.	
7 (R/W)	EHFC	Hardware Flow Control Enable. When the EMAC_RQ[n]_OPMODE.EHFC bit is set (=1), the flow control signal operation, based on the fill-level of Rx queue, is enabled. When cleared (=0), the flow control operation is disabled.	
		0	Disabled
		1	Enabled

Table 28-281: EMAC_RQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets. When the EMAC_RQ[n]_OPMODE.DIS_TCP_EF bit is set (=1), the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When EMAC_RQ[n]_OPMODE.DIS_TCP_EF = 0, all error packets are dropped if EMAC_RQ[n]_OPMODE.FEP = 0.	
		0	Enabled
		1	Disabled
5 (R/W)	RSF	Receive Queue Store and Forward. When the EMAC_RQ[n]_OPMODE.RSF bit is set (=1), the EMAC module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the EMAC_RQ[n]_OPMODE.RTC field. When EMAC_RQ[n]_OPMODE.RSF = 0, the Rx queue operates in the threshold (cut-through) mode, subject to the threshold specified by the EMAC_RQ[n]_OPMODE.RTC field.	
		0	Disabled
		1	Enabled
4 (R/W)	FEP	Forward Error Packets. When EMAC_RQ[n]_OPMODE.FEP = 0, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in threshold mode), the packet is not dropped. When the EMAC_RQ[n]_OPMODE.FEP bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of the EMAC_RQ[n]_OPMODE.FEP bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA.	
		0	Disabled
		1	Enabled

Table 28-281: EMAC_RQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FUP	Forward Undersized Good Packets. When the EMAC_RQ[n]_OPMODE.FUP bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When EMAC_RQ[n]_OPMODE.FUP = 0, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx threshold, for example, RTC = 01.	
		0	Disabled
		1	Enabled
1:0 (R/W)	RTC	Receive Queue Threshold Control. The EMAC_RQ[n]_OPMODE.RTC field indicates the threshold level of the MTL Rx queue (in bytes). The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. The EMAC_RQ[n]_OPMODE.RTC field is valid only when the RSF bit is zero. The EMAC_RQ[n]_OPMODE.RTC field is ignored when the RSF bit is set (=1).	
		0	64
		1	32
		2	96
		3	128

Receive Queue to DMA Channel Mapping Register 0

The EMAC_RXQ_DMA_MAP0 register is reserved in EQOS-CORE and EQOS-MTL configurations.

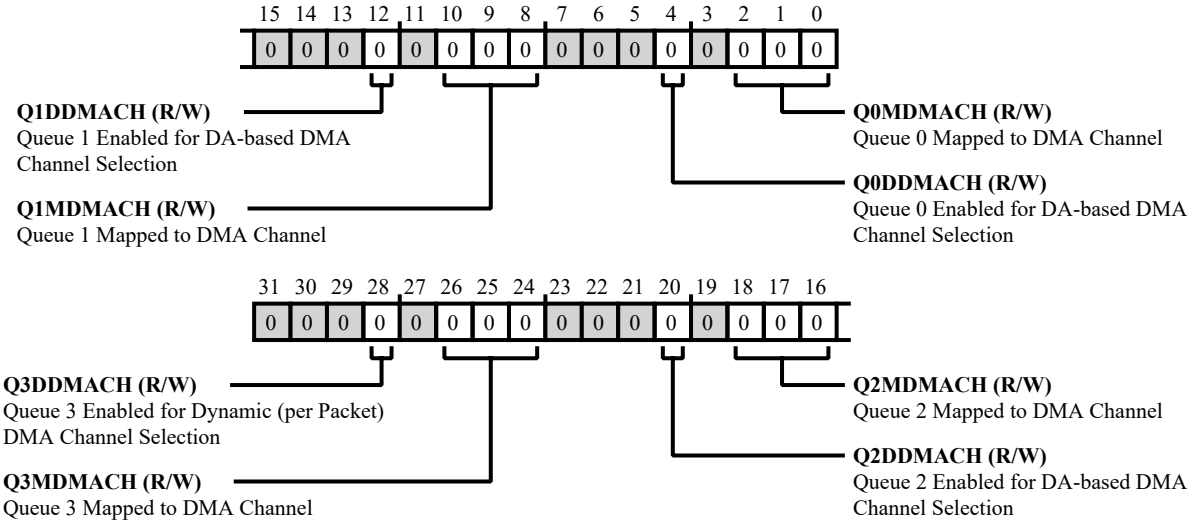


Figure 28-227: EMAC_RXQ_DMA_MAP0 Register Diagram

Table 28-282: EMAC_RXQ_DMA_MAP0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	Q3DDMACH	Queue 3 Enabled for Dynamic (per Packet) DMA Channel Selection. When set, the EMAC_RXQ_DMA_MAP0.Q3DDMACH bit indicates that the packets received in queue 3 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the EMAC_RXQ_DMA_MAP0.Q3DDMACH bit indicates that the packets received in queue 3 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q3MDMACH field.	
		0	Disable
		1	Enable
26:24 (R/W)	Q3MDMACH	Queue 3 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP0.Q3MDMACH field controls the routing of the received packet in queue 3 to the DMA channel. The EMAC_RXQ_DMA_MAP0.Q3MDMACH field is valid when the EMAC_RXQ_DMA_MAP0.Q3DDMACH field is reset. Note: The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the others are reserved	
		0	DMA Channel 0

Table 28-282: EMAC_RXQ_DMA_MAP0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	DMA Channel 0
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
20 (R/W)	Q2DDMACH	Queue 2 Enabled for DA-based DMA Channel Selection. When set, the EMAC_RXQ_DMA_MAP0.Q2DDMACH bit indicates that the packets received in queue 2 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the EMAC_RXQ_DMA_MAP0.Q2DDMACH bit indicates that the packets received in queue 2 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q2MDMACH field.	
		0	Disable
		1	Enable
18:16 (R/W)	Q2MDMACH	Queue 2 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP0.Q2MDMACH field controls the routing of the received packet in queue 2 to the DMA channel. The EMAC_RXQ_DMA_MAP0.Q2MDMACH field is valid when the EMAC_RXQ_DMA_MAP0.Q2DDMACH field is reset.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

Table 28-282: EMAC_RXQ_DMA_MAP0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	Q1DDMACH	Queue 1 Enabled for DA-based DMA Channel Selection. When set (=1), the EMAC_RXQ_DMA_MAP0.Q1DDMACH bit indicates that the packets received in queue 1 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the EMAC_RXQ_DMA_MAP0.Q1DDMACH bit indicates that the packets received in queue 1 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q1MDMACH field.	
		0	Disable
		1	Enable
10:8 (R/W)	Q1MDMACH	Queue 1 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP0.Q1MDMACH field controls the routing of the received packet in queue 1 to the DMA channel. The EMAC_RXQ_DMA_MAP0.Q1MDMACH field is valid when the EMAC_RXQ_DMA_MAP0.Q1DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
4 (R/W)	Q0DDMACH	Queue 0 Enabled for DA-based DMA Channel Selection. When set, the EMAC_RXQ_DMA_MAP0.Q0DDMACH bit indicates that the packets received in queue 0 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the EMAC_RXQ_DMA_MAP0.Q0DDMACH bit indicates that the packets received in queue 0 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q0MDMACH field.	
		0	Disable
		1	Enable

Table 28-282: EMAC_RXQ_DMA_MAP0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	Q0MDMACH	Queue 0 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP0.Q0MDMACH` field controls the routing of the packet received in queue 0 to the DMA channel. The `EMAC_RXQ_DMA_MAP0.Q0MDMACH` field is valid when the EMAC_RXQ_DMA_MAP0.Q0DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

Receive Queue to DMA Channel Mapping Register 1

The EMAC_RXQ_DMA_MAP1 register is reserved in EQOS-CORE and EQOS-MTL configurations.

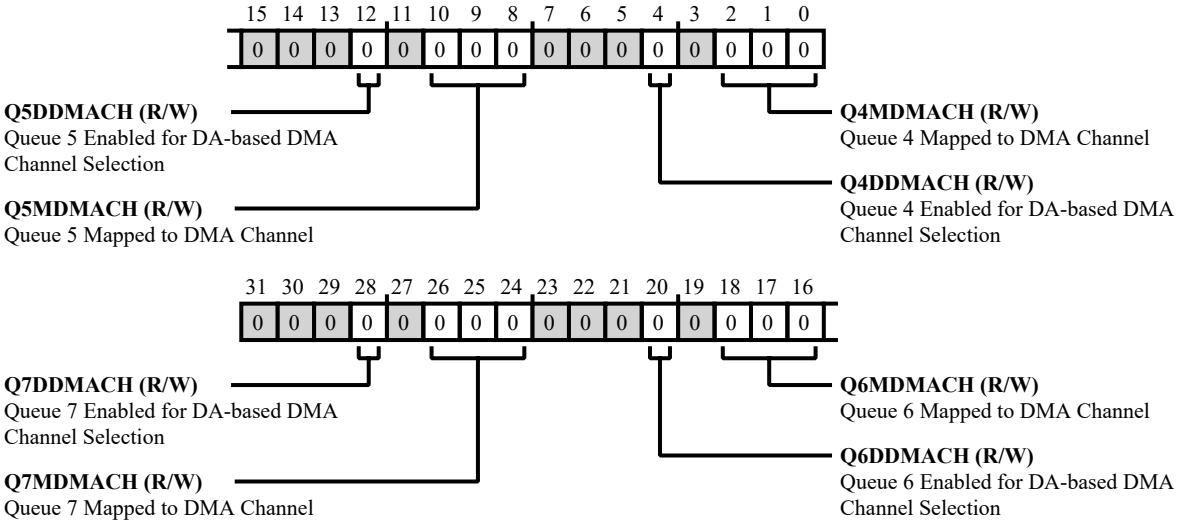


Figure 28-228: EMAC_RXQ_DMA_MAP1 Register Diagram

Table 28-283: EMAC_RXQ_DMA_MAP1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	Q7DDMACH	Queue 7 Enabled for DA-based DMA Channel Selection. When set (=1), the EMAC_RXQ_DMA_MAP1.Q7DDMACH bit indicates that the packets received in queue 7 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (= 0), the EMAC_RXQ_DMA_MAP1.Q7DDMACH bit indicates that the packets received in queue 7 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q7MDMACH field.	
		0	Disable
		1	Enable
26:24 (R/W)	Q7MDMACH	Queue 7 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP1.Q7MDMACH field controls the routing of the packet received in queue 7 to the DMA channel. The EMAC_RXQ_DMA_MAP1.Q7MDMACH field is valid when the EMAC_RXQ_DMA_MAP1.Q7DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0

Table 28-283: EMAC_RXQ_DMA_MAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
20 (R/W)	Q6DDMACH	Queue 6 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP1.Q6DDMACH` bit indicates that the packets received in queue 6 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the `EMAC_RXQ_DMA_MAP1.Q6DDMACH` bit indicates that the packets received in queue 6 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q6MDMACH field.	
		0	Disable
		1	Enable
18:16 (R/W)	Q6MDMACH	Queue 6 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP1.Q6MDMACH` field controls the routing of the packet received in queue 6 to the DMA channel. The `EMAC_RXQ_DMA_MAP1.Q6MDMACH` field is valid when the EMAC_RXQ_DMA_MAP1.Q6DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

Table 28-283: EMAC_RXQ_DMA_MAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	Q5DDMACH	Queue 5 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP1.Q5DDMACH` bit indicates that the packets received in queue 5 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (= 0), the `EMAC_RXQ_DMA_MAP1.Q5DDMACH` bit indicates that the packets received in queue 5 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q5MDMACH field.	
		0	Disable
		1	Enable
10:8 (R/W)	Q5MDMACH	Queue 5 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP1.Q5MDMACH` field controls the routing of the packets received in queue 5 to the DMA channel The `EMAC_RXQ_DMA_MAP1.Q5MDMACH` field is valid when the EMAC_RXQ_DMA_MAP1.Q5DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
4 (R/W)	Q4DDMACH	Queue 4 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP1.Q4DDMACH` bit indicates that the packets received in queue 4 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (= 0), the `EMAC_RXQ_DMA_MAP1.Q4DDMACH` bit indicates that the packets received in queue 4 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q4MDMACH field.	
		0	Disable
		1	Enable

Table 28-283: EMAC_RXQ_DMA_MAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	Q4MDMACH	Queue 4 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP1.Q4MDMACH` field controls the routing of the packet received in queue 4 to the DMA channel. The `EMAC_RXQ_DMA_MAP1.Q4MDMACH` field is valid when the EMAC_RXQ_DMA_MAP1.Q4DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

MTL Time Based Scheduling Control Register

The EMAC_TBS_CTRL register controls the operation of Time-Based Scheduling (TBS).

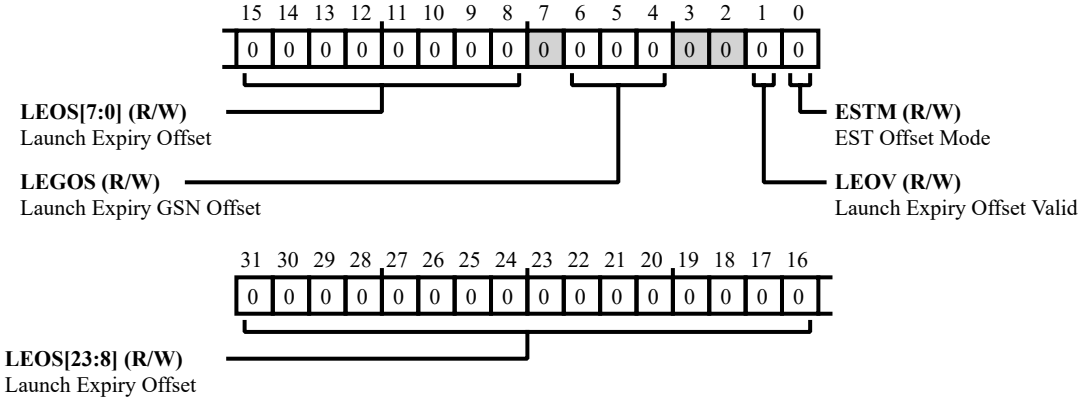


Figure 28-229: EMAC_TBS_CTRL Register Diagram

Table 28-284: EMAC_TBS_CTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	LEOS	Launch Expiry Offset. The field contains the value in units of 256 nanoseconds that must be added to the launch time to compute the launch expiry time. The value is valid only when EMAC_TBS_CTRL.LEOV = 1. The max value, 999,999,999 ns, should be smaller than the CTR-1 value when ESTM mode is enabled, because this value is a modulo CTR value.	
6:4 (R/W)	LEGOS	Launch Expiry GSN Offset. The EMAC_TBS_CTRL.LEGOS field indicates the number GSN slots that must be added to the launch GSN to compute the launch expiry time. The value is valid only when EMAC_TBS_CTRL.LEOV = 1.	
1 (R/W)	LEOV	Launch Expiry Offset Valid. When set (=1), the EMAC_TBS_CTRL.LEOV bit indicates that the EMAC_TBS_CTRL.LEOS field is valid. When not set, it indicates that the launch expiry offset is invalid and the MTL must not check for launch expiry time.	
		0	LEOS Field is Invalid
		1	LEOS Field is Valid

Table 28-284: EMAC_TBS_CTRL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	ESTM	EST Offset Mode. When the `EMAC_TBS_CTRL.ESTM` bit is set, the launch time value used in time based scheduling is interpreted as an EST offset value and is added to the Base Time Register (BTR) of the current list. When cleared (=0), the launch time value is used as an absolute value that should be compared with the system time [39:8].	
		0	Disabled
		1	Enabled

Transmit Queue 0 Debug Register

The EMAC_TQ0_DBG register gives the debug status of various blocks related to the transmit queue.

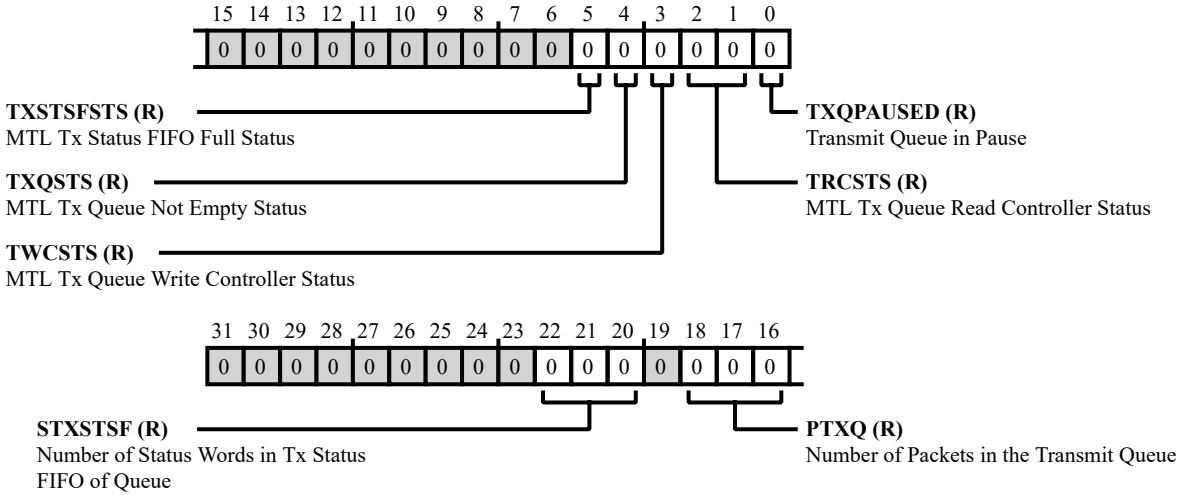


Figure 28-230: EMAC_TQ0_DBG Register Diagram

Table 28-285: EMAC_TQ0_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22:20 (R/NW)	STXSTSF	Number of Status Words in Tx Status FIFO of Queue. The EMAC_TQ0_DBG.STXSTSF field indicates the current number of status in the Tx status FIFO of this queue. When EMAC_MTL_OPMODE.DTXSTS = 1, this field does not reflect the number of status words in Tx status FIFO.	
18:16 (R/NW)	PTXQ	Number of Packets in the Transmit Queue. The EMAC_TQ0_DBG.PTXQ field indicates the current number of packets in the Tx queue. When EMAC_MTL_OPMODE.DTXSTS = 1, this field does not reflect the number of packets in the transmit queue.	
5 (R/NW)	TXSTSFSTS	MTL Tx Status FIFO Full Status. When set (=1), the EMAC_TQ0_DBG.TXSTSFSTS bit indicates that the MTL Tx status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.	
		0	MTL Tx Status FIFO Full Status is Not Detected
		1	MTL Tx Status FIFO Full Status is Detected

Table 28-285: EMAC_TQ0_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	TXQSTS	MTL Tx Queue Not Empty Status. When the EMAC_TQ0_DBG.TXQSTS bit is set (=1), it indicates that the MTL Tx queue is not empty and some data is left for transmission.	
		0	MTL Tx Queue Not Empty Status is Not Detected
		1	MTL Tx Queue Not Empty Status is Detected
3 (R/NW)	TWCSTS	MTL Tx Queue Write Controller Status. When set (=1), the EMAC_TQ0_DBG.TWCSTS bit indicates that the MTL Tx queue write controller is active, and it is transferring the data to the Tx queue.	
		0	MTL Tx Queue Write Controller Status is Not Detected
		1	MTL Tx Queue Write Controller Status is Detected
2:1 (R/NW)	TRCSTS	MTL Tx Queue Read Controller Status. The EMAC_TQ0_DBG.TRCSTS field indicates the state of the Tx queue read controller:	
		0	Idle State
		1	Read State (Transferring Data to the MAC Transmitter)
		2	Waiting for Pending Tx Status from the MAC Transmitter
		3	Flushing the Tx Queue Because of the Packet Abort Request from the MAC
0 (R/NW)	TXQPAUSED	Transmit Queue in Pause. When the EMAC_TQ0_DBG.TXQPAUSED bit is set (=1) and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled. Reception of 802.3x Pause packet when PFC is disabled.	
		0	Transmit Queue in Pause Status is Not Detected
		1	Transmit Queue in Pause Status is Detected

Transmit Queue 0 ETS Status Register

The EMAC_TQ0_ETS_STAT register provides the average traffic transmitted in queue 0.

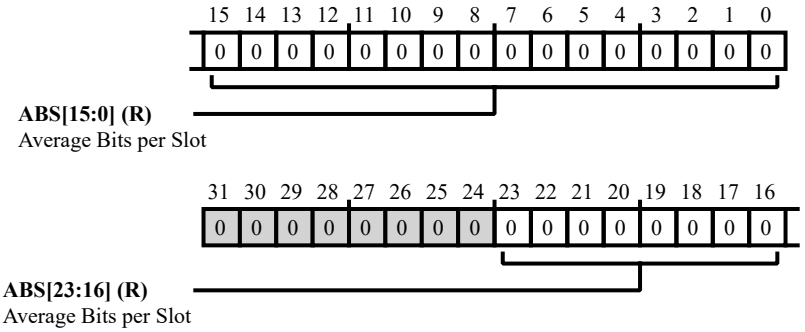


Figure 28-231: EMAC_TQ0_ETS_STAT Register Diagram

Table 28-286: EMAC_TQ0_ETS_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/NW)	ABS	Average Bits per Slot. The EMAC_TQ0_ETS_STAT.ABS field contains the average transmitted bits per slot. When the DCB operation is enabled for queue 0, this field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680.

Transmit Queue 0 Operation Mode Register

The EMAC_TQ0_OPMODE register establishes the transmit queue operating modes and commands.

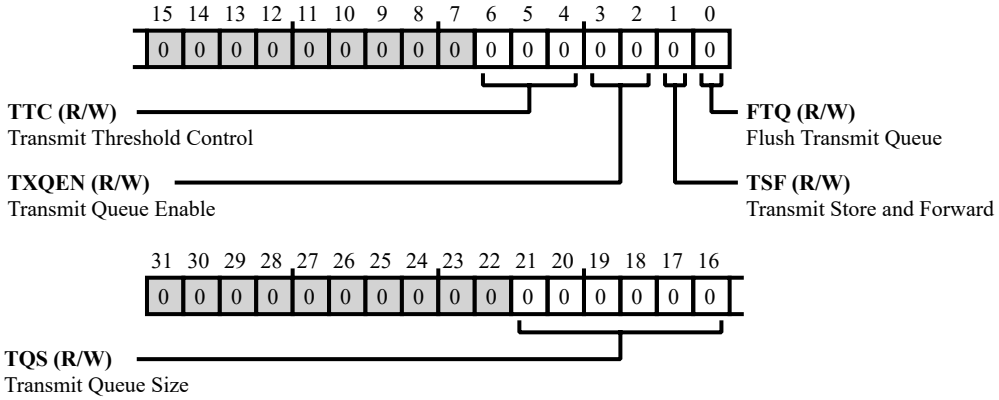


Figure 28-232: EMAC_TQ0_OPMODE Register Diagram

Table 28-287: EMAC_TQ0_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/W)	TQS	Transmit Queue Size. The EMAC_TQ0_OPMODE.TQS field indicates the size of the allocated transmit queues in blocks of 256 bytes. The field is read-write only if the number of Tx queues more than one, the reset value is 0x0 and indicates size of 256 bytes. For example, a value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (TQS+1)*256 bytes. When the number of Tx queues is one, the field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value.	
6:4 (R/W)	TTC	Transmit Threshold Control. These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.	
		0	32
		1	64
		2	96
		3	128
		4	192
		5	256
		6	384

Table 28-287: EMAC_TQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		7	512
3:2 (R/W)	TXQEN	Transmit Queue Enable. The EMAC_TQ0_OPMODE.TXQEN field is used to enable/disable the transmit queue 0. The EMAC_TQ0_OPMODE.TXQEN field is read Only in Single queue configurations and read write in multiple queue configurations. Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.	
		0	Not Enabled
		1	Enable in AV Mode (Reserved in Non-AV)
		2	Enabled
		3	Reserved
1 (R/W)	TSF	Transmit Store and Forward. When the EMAC_TQ0_OPMODE.TSF bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When the EMAC_TQ0_OPMODE.TSF bit is set, the EMAC_TQ0_OPMODE.TTC field is ignored. The EMAC_TQ0_OPMODE.TSF bit should be changed only when the transmission is stopped.	
		0	Transmit Store and Forward is Disabled
		1	Transmit Store and Forward is Enabled
0 (R/W)	FTQ	Flush Transmit Queue. When the EMAC_TQ0_OPMODE.FTQ bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. The EMAC_TQ0_OPMODE.FTQ bit is internally reset when the flushing operation is complete. Until EMAC_TQ0_OPMODE.FTQ = 0, do not write to the EMAC_TQ[n]_OPMODE register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flush Transmit Queue is Disabled
		1	Flush Transmit Queue is Enabled

Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register

The EMAC_TQ0_QTMWGT register contains the quantum value for Deficit Weighted Round Robin (DWRR), weights for the Weighted Round Robin (WRR), and Weighted Fair Queuing (WFQ) for queue 0.

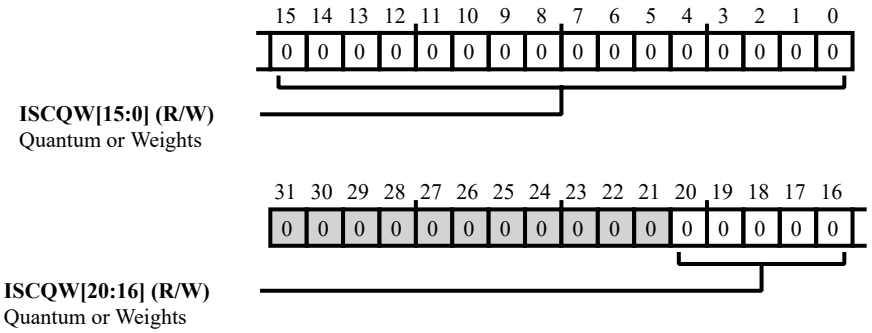


Figure 28-233: EMAC_TQ0_QTMWGT Register Diagram

Table 28-288: EMAC_TQ0_QTMWGT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:0 (R/W)	ISCQW	Quantum or Weights. When the DCB operation is enabled with DWRR algorithm for queue 0 traffic, the EMAC_TQ0_QTMWGT.ISCQW field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. When DCB operation is enabled with WFQ algorithm for queue 0 traffic, the EMAC_TQ0_QTMWGT.ISCQW field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. The higher the programmed weights lesser the bandwidth allocated for the particular transmit queue. This is because the weights are used to compute the packet finish time (weights * packet_size). Lesser the finish time, higher the probability of the packet getting scheduled first and using more bandwidth. When DCB operation or generic queuing operation is enabled with WRR algorithm for queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero.

Transmit Queue 0 Underflow Packet Counter Register

The EMAC_TQ0_URFL register contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush.

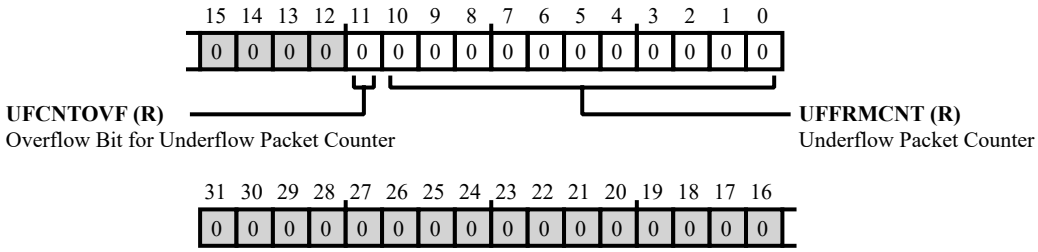


Figure 28-234: EMAC_TQ0_URFL Register Diagram

Table 28-289: EMAC_TQ0_URFL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	UFCNTOVF	Overflow Bit for Underflow Packet Counter. The EMAC_TQ0_URFL.UFCNTOVF bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and the EMAC_TQ0_URFL.UFCNTOVF bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected for Underflow Packet Counter
		1	Overflow Detected for Underflow Packet Counter
10:0 (R/NW)	UFFRMCNT	Underflow Packet Counter. The EMAC_TQ0_URFL.UFFRMCNT field indicates the number of packets aborted by the controller because of Tx queue underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.	

Transmit Queue n Debug Register

The EMAC_TQ[n]_DBG register gives the debug status of various blocks related to the transmit queue.

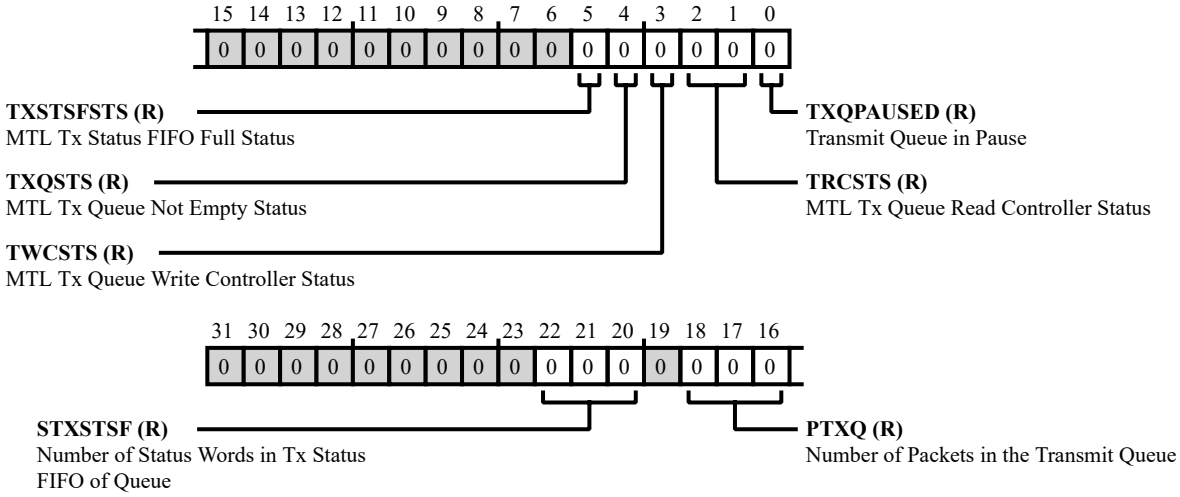


Figure 28-235: EMAC_TQ[n]_DBG Register Diagram

Table 28-290: EMAC_TQ[n]_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22:20 (R/NW)	STXSTSF	Number of Status Words in Tx Status FIFO of Queue. The EMAC_TQ[n]_DBG.STXSTSF field indicates the current number of status in the Tx Status FIFO of this queue. When EMAC_MTL_OPMODE.DTXSTS = 1, this field does not reflect the number of status words in Tx Status FIFO.	
18:16 (R/NW)	PTXQ	Number of Packets in the Transmit Queue. The EMAC_TQ[n]_DBG.PTXQ field indicates the current number of packets in the Tx queue. When EMAC_MTL_OPMODE.DTXSTS = 1, the field does not reflect the number of packets in the transmit queue.	
5 (R/NW)	TXSTSFSTS	MTL Tx Status FIFO Full Status. When set (=1), the EMAC_TQ[n]_DBG.TXSTSFSTS bit indicates that the MTL Tx status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.	
		0	Status Not Detected
		1	Status Detected

Table 28-290: EMAC_TQ[n]_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	TXQSTS	MTL Tx Queue Not Empty Status. When the EMAC_TQ[n]_DBG.TXQSTS bit is set (=1), it indicates that the MTL Tx queue is not empty and some data is left for transmission.	
		0	Status Not Detected
		1	Status Detected
3 (R/NW)	TWCSTS	MTL Tx Queue Write Controller Status. When set (=1), the EMAC_TQ[n]_DBG.TWCSTS bit indicates that the MTL Tx queue write controller is active (status detected), and it is transferring the data to the Tx queue.	
		0	Status Not Detected
		1	Status Detected
2:1 (R/NW)	TRCSTS	MTL Tx Queue Read Controller Status. The EMAC_TQ[n]_DBG.TRCSTS field indicates the state of the Tx queue read controller.	
		0	Idle State
		1	Read State (Transferring data to the MAC transmitter)
		2	Wait for pending Tx status from the MAC transmitter
		3	Flush the Tx queue because of the packet abort request from the MAC
0 (R/NW)	TXQPAUSED	Transmit Queue in Pause. When the EMAC_TQ[n]_DBG.TXQPAUSED bit is set (=1) and the Rx flow control is enabled, it indicates that the Tx queue is in the pause condition (in the full-duplex only mode) because of the following: - Reception of the PFC packet for the priorities assigned to the Tx queue, when PFC is enabled. - Reception of the 802.3x pause packet, when PFC is disabled.	
		0	Transmit Queue in Pause Status is Not Detected
		1	Transmit Queue in Pause Status is Detected

Transmit Queue n ETS Control Register

The EMAC_TQ[n]_ETS_CTL register controls the enhanced transmission selection operation.

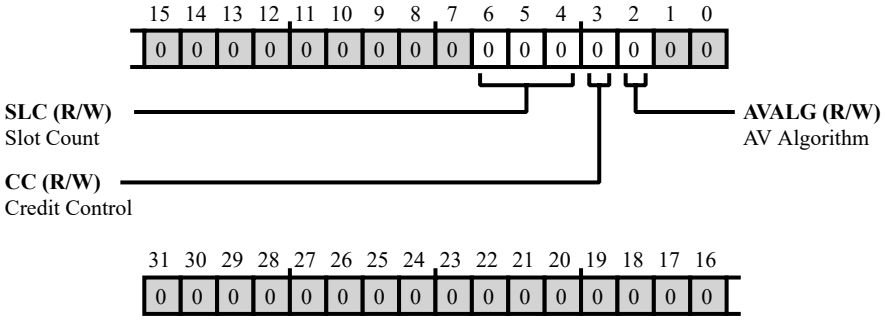


Figure 28-236: EMAC_TQ[n]_ETS_CTL Register Diagram

Table 28-291: EMAC_TQ[n]_ETS_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	SLC	Slot Count. If the credit-based shaper algorithm is enabled, the software can program the EMAC_TQ[n]_ETS_CTL.SLC field (of a duration programmed in EMAC_DMA[n]_SFN_CTLSTAT register). The average transmitted bits per slot must be computed for the queue; it is provided in the EMAC_TQ[n]_ETS_STAT register.	
		0	1 Slot
		1	2 Slots
		2	4 Slots
		3	8 Slots
		4	16 Slots
		5	Reserved
3 (R/W)	CC	Credit Control. When the EMAC_TQ[n]_ETS_CTL.CC bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in channel 1. The credit accumulates even when there is no packet waiting in channel 1 and another channel is transmitting. When EMAC_TQ[n]_ETS_CTL.CC = 0, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in channel 1. When there is no packet waiting in channel 1 and other channel is transmitting, no credit is accumulated.	
		0	Credit Control is Disabled
		1	Credit Control is Enabled

Table 28-291: EMAC_TQ[n]_ETS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	AVALG	AV Algorithm. When queue 1 is programmed for AV, the `EMAC_TQ[n]_ETS_CTL.AVALG` field configures the scheduling algorithm for the queue. The `EMAC_TQ[n]_ETS_CTL.AVALG` bit when set, a credit-based shaper algorithm (CBS) is selected for queue 1 traffic. When cleared (=0), strict priority is selected.	
		0	CBS Algorithm is Disabled
		1	CBS Algorithm is Enabled

Transmit Queue n ETS Status Register

The EMAC_TQ[n]_ETS_STAT register provides the average traffic transmitted in queue 1.

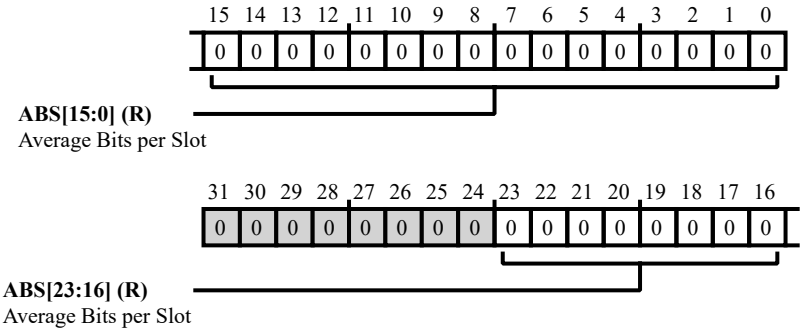


Figure 28-237: EMAC_TQ[n]_ETS_STAT Register Diagram

Table 28-292: EMAC_TQ[n]_ETS_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/NW)	ABS	Average Bits per Slot. The EMAC_TQ[n]_ETS_STAT.ABS field contains the average transmitted bits per slot. If AV operation is enabled, the EMAC_TQ[n]_ETS_STAT.ABS field is computed over the number of slots, programmed in EMAC_TQ[n]_ETS_CTL.SLC. The maximum value of this field is 0x6_4000 in 100 Mbps, 0x3E_8000 in 1000 Mbps and 9C_4000 in 2500 Mbps mode respectively. When the DCB operation is enabled for queue, the EMAC_TQ[n]_ETS_STAT.ABS field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680.

Transmit Queue n CBS HiCredit Register

The EMAC_TQ[n]_HICRDT register contains the HiCredit value required for the credit-based shaper algorithm for the queue.

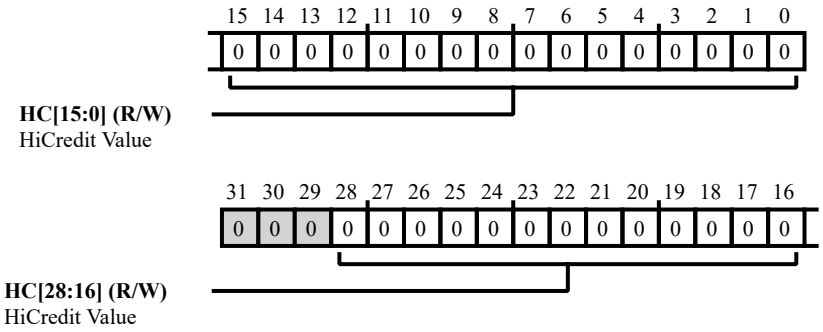


Figure 28-238: EMAC_TQ[n]_HICRDT Register Diagram

Table 28-293: EMAC_TQ[n]_HICRDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	HC	HiCredit Value. When the AV feature is enabled, the EMAC_TQ[n]_HICRDT.HC field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. It is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is 131,072 * 1,024 = 134,217,728 or 0x0800_0000.

Transmit Queue n CBS LoCredit Register

The EMAC_TQ[n]_LOCRDT register contains the loCredit value required for the credit-based shaper algorithm for the queue.

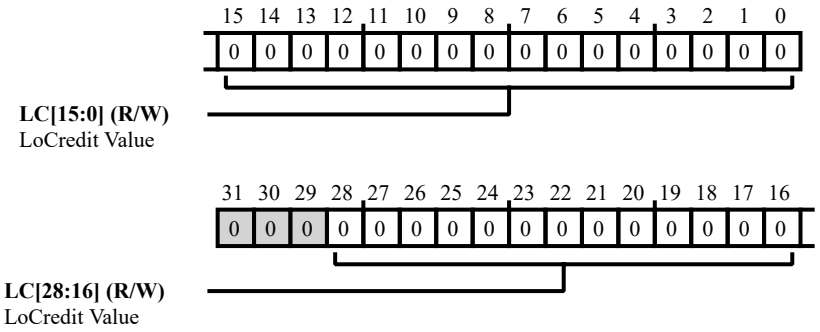


Figure 28-239: EMAC_TQ[n]_LOCRDT Register Diagram

Table 28-294: EMAC_TQ[n]_LOCRDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	LC	LoCredit Value. When the AV operation is enabled, the EMAC_TQ[n]_LOCRDT.LC field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. It is specified in bits scaled by 1,024. The maximum value to be programmed corresponds to twice the maxFrameSize transmitted from this queue. If the maxFrameSize is 8192 bytes, then (8192*2) * 8 * 1024 = 134,217,728 or 0x0800_0000. Because it is a negative value, the programmed value is the 2's complement of the value, that is, 0x1800_0000.

Transmit Queue n Operation Mode Register

The EMAC_TQ[n]_OPMODE register establishes the transmit queue operating modes and commands.

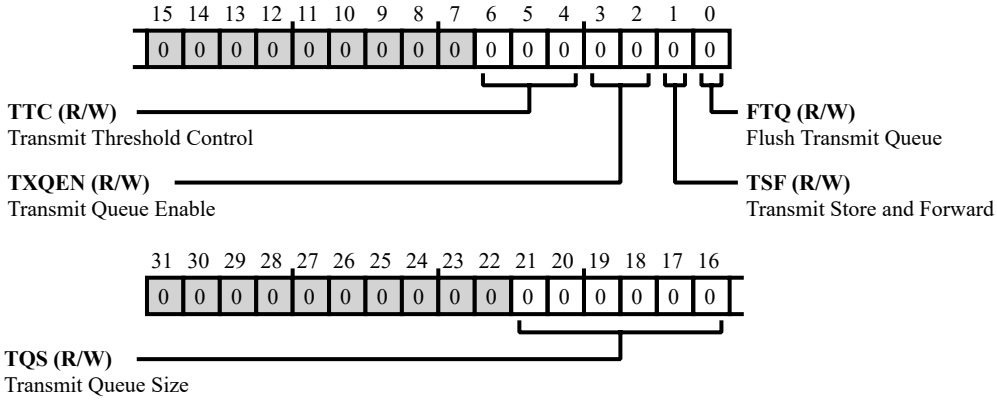


Figure 28-240: EMAC_TQ[n]_OPMODE Register Diagram

Table 28-295: EMAC_TQ[n]_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/W)	TQS	Transmit Queue Size. The EMAC_TQ[n]_OPMODE.TQS field indicates the size of the allocated transmit queues in blocks of 256 bytes. The EMAC_TQ[n]_OPMODE.TQS field is read-write only if the number of Tx queues more than one; the reset value is 0x0 and indicates size of 256 bytes. For example, a value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. Program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (TQS+1)*256 bytes. When the number of Tx queues is one, the EMAC_TQ[n]_OPMODE.TQS field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value.	
6:4 (R/W)	TTC	Transmit Threshold Control. The EMAC_TQ[n]_OPMODE.TTC field controls the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. The EMAC_TQ[n]_OPMODE.TTC field is used only when EMAC_TQ[n]_OPMODE.TSF = 0.	
		0	32
		1	64
		2	96
		3	128
		4	192

Table 28-295: EMAC_TQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	256
		6	384
		7	512
3:2 (R/W)	TXQEN	Transmit Queue Enable. The `EMAC_TQ[n]_OPMODE.TXQEN` field is used to enable/disable the transmit queue 0. Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.	
		0	Not Enabled
		1	Enable in AV Mode (Reserved in Non-AV)
		2	Enabled
		3	Reserved
1 (R/W)	TSF	Transmit Store and Forward. When the `EMAC_TQ[n]_OPMODE.TSF` bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When the `EMAC_TQ[n]_OPMODE.TSF` bit is set, the TTC values specified in bits[6:4] of this register are ignored. The `EMAC_TQ[n]_OPMODE.TSF` bit should be changed only when the transmission is stopped.	
		0	Transmit Store and Forward is Disabled
		1	Transmit Store and Forward is Enabled
0 (R/W)	FTQ	Flush Transmit Queue. When the `EMAC_TQ[n]_OPMODE.FTQ` bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. The `EMAC_TQ[n]_OPMODE.FTQ` bit is internally reset when the flushing operation is complete. Until EMAC_TQ[0]_OPMODE= 0, do not write to the EMAC_TQ[1]_OPMODE register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flush Transmit Queue is Disabled
		1	Flush Transmit Queue is Enabled

Transmit Queue n IdleSlopeCredit, Quantum or Weights Register

The EMAC_TQ[n]_QTMWGT register provides the average traffic transmitted in queue 1.

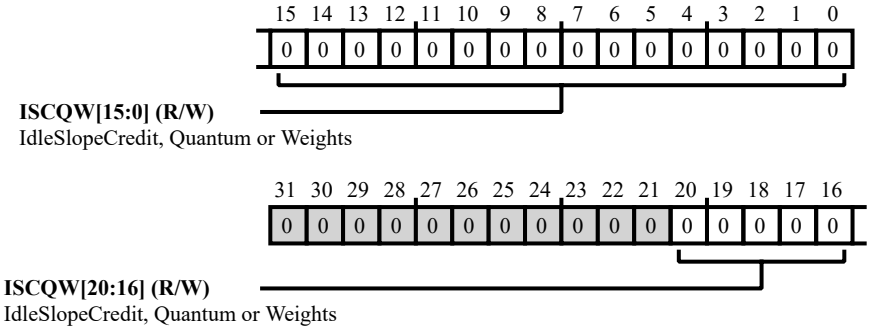


Figure 28-241: EMAC_TQ[n]_QTMWGT Register Diagram

Table 28-296: EMAC_TQ[n]_QTMWGT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:0 (R/W)	ISCQW	IdleSlopeCredit, Quantum or Weights. When AV feature is enabled, the `EMAC_TQ[n]_QTMWGT.ISCQW` field contains the idleSlopeCredit value required for the credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps; 3.2 ns for 2500 Mbps) when the credit is increasing. The software should program the `EMAC_TQ[n]_QTMWGT.ISCQW` field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. Bits[20:14] must be written to zero. Quantum When the DCB operation is enabled with DWRR algorithm for queue 1 traffic, the `EMAC_TQ[n]_QTMWGT.ISCQW` field field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. Weights When DCB operation is enabled with WFQ algorithm for queue 1 traffic, the `EMAC_TQ[n]_QTMWGT.ISCQW` field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. When DCB operation or generic queuing operation is enabled with WRR algorithm for queue 1 traffic, the `EMAC_TQ[n]_QTMWGT.ISCQW` field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero. Note 1: In multiple queue configuration this field in respective per queue register must be programmed to some non-zero value when multiple queues are enabled or single queue other than Q0 is enabled. The `EMAC_TQ[n]_QTMWGT.ISCQW` field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both receive and transmit. In receive, the register is EMAC_MTL_OPMODE register. Note 2: For WFQ algorithm, higher the programmed weights lesser the bandwidth allocated for that transmit queue. The finish time is not a function of particular packet alone but it is as per the formula: (previous_finish_time of particular transmit queue + (weights*packet_size)) Note 3: The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue with respect to total BW or time.

Transmit Queue n SendSlopeCredit Register

The EMAC_TQ[n]_SSCRDT register contains the sendSlope credit value required for the credit-based shaper algorithm for the queue.

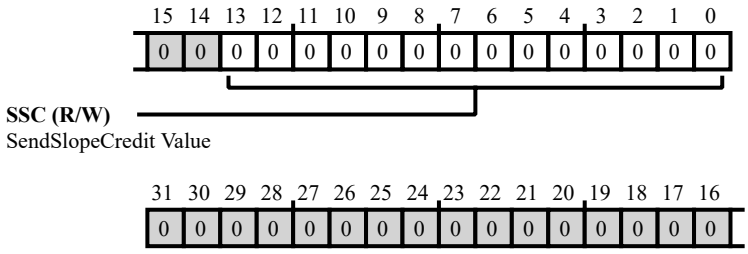


Figure 28-242: EMAC_TQ[n]_SSCRDT Register Diagram

Table 28-297: EMAC_TQ[n]_SSCRDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:0 (R/W)	SSC	SendSlopeCredit Value. When AV operation is enabled, the EMAC_TQ[n]_SSCRDT.SSC field contains the sendSlopeCredit value required for credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns, 8 ns and 3.2 ns for 100 Mbps, 1000 Mbps and 2500 Mbps respectively) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. The EMAC_TQ[n]_SSCRDT.SSC field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when channel 1 is selected for transmission.

Transmit Queue n Underflow Packet Counter Register

The EMAC_TQ[n]_URFL register contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush.

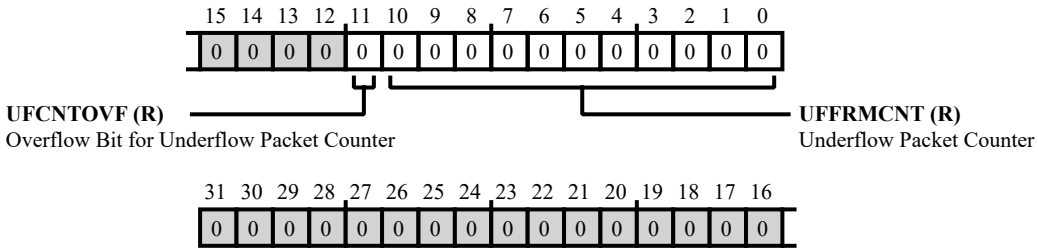


Figure 28-243: EMAC_TQ[n]_URFL Register Diagram

Table 28-298: EMAC_TQ[n]_URFL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	UFCNTOVF	Overflow Bit for Underflow Packet Counter. The EMAC_TQ[n]_URFL.UFCNTOVF bit is set every time the Tx queue underflow packet counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and the EMAC_TQ[n]_URFL.UFCNTOVF bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected for Underflow Packet Counter
		1	Overflow Detected for Underflow Packet Counter
10:0 (R/NW)	UFFRMCNT	Underflow Packet Counter. The EMAC_TQ[n]_URFL.UFFRMCNT field indicates the number of packets aborted by the controller because of Tx queue underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.	

Receive ICMP Error Octets Counter Register

The EMAC_RXICMP_ERR_OCNT register provides the number of bytes received by the EMAC module in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

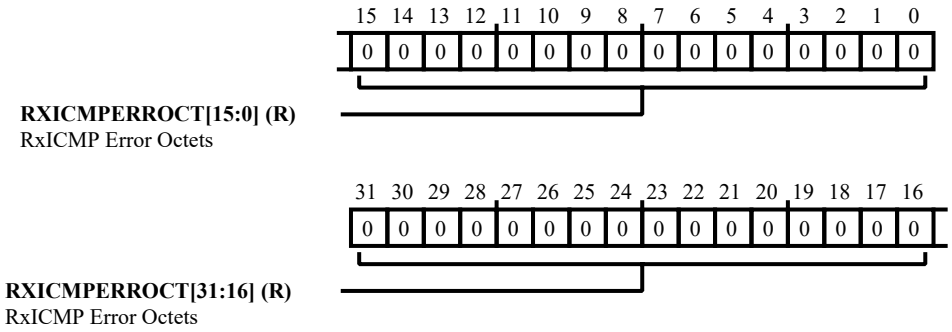


Figure 28-244: EMAC_RXICMP_ERR_OCNT Register Diagram

Table 28-299: EMAC_RXICMP_ERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXICMPERROCT	RxICMP Error Octets. The EMAC_RXICMP_ERR_OCNT.RXICMPERROCT field indicates the number of bytes received in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

Receive ICMP Error Packets Counter Register

The EMAC_RXICMP_ERR_PCNT register provides the number of good IP datagrams received by the EMAC module whose ICMP payload has a checksum error.

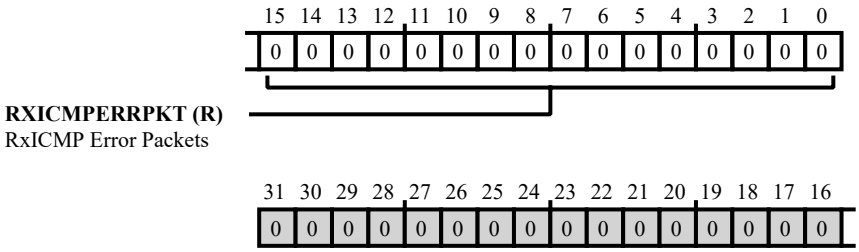


Figure 28-245: EMAC_RXICMP_ERR_PCNT Register Diagram

Table 28-300: EMAC_RXICMP_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXICMPERRPKT	RxICMP Error Packets. The EMAC_RXICMP_ERR_PCNT.RXICMPERRPKT field indicates the number of good IP datagrams received whose ICMP payload has a checksum error.

Receive ICMP Good Octets Counter Register

The EMAC_RXICMP_OCNT_G register provides the number of bytes received by the EMAC module in a good ICMP segment. This counter does not count IP header bytes.

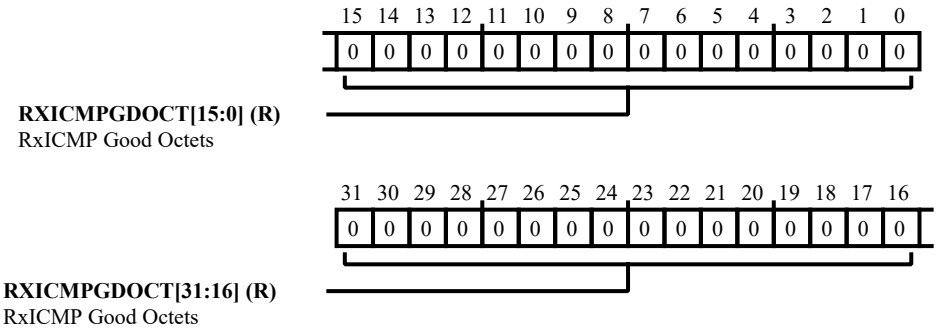


Figure 28-246: EMAC_RXICMP_OCNT_G Register Diagram

Table 28-301: EMAC_RXICMP_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXICMPGDOCT	RxICMP Good Octets. The EMAC_RXICMP_OCNT_G.RXICMPGDOCT field indicates the number of bytes received in a good ICMP segment. This counter does not count IP header bytes.

Receive ICMP Good Packets Counter Register

The EMAC_RXICMP_PCNT_G register provides the number of good IP datagrams received by the EMAC module with a good ICMP payload.

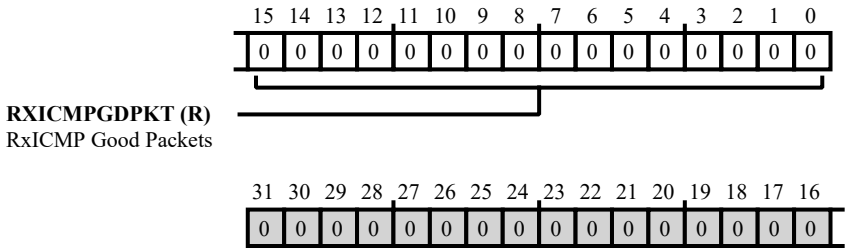


Figure 28-247: EMAC_RXICMP_PCNT_G Register Diagram

Table 28-302: EMAC_RXICMP_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXICMPGDPKT	RxICMP Good Packets. The EMAC_RXICMP_PCNT_G.RXICMPGDPKT field indicates the number of good IP datagrams received with a good ICMP payload.

Receive IPv4 Fragmented Octets Counter Register

The EMAC_RXIPV4_FRGMT_OCNT register provides the number of bytes received by the EMAC module in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

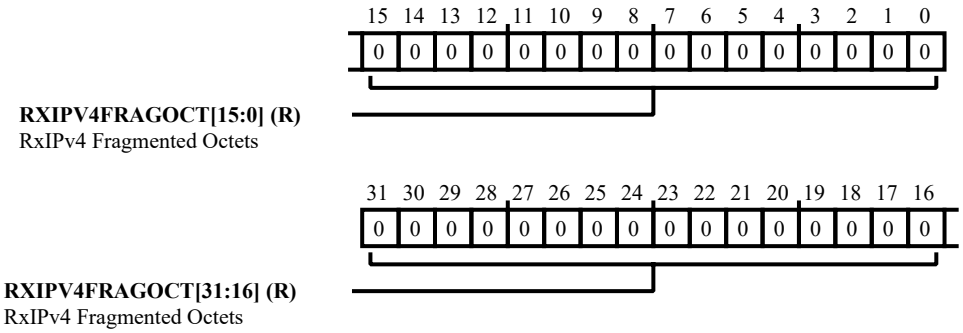


Figure 28-248: EMAC_RXIPV4_FRGMT_OCNT Register Diagram

Table 28-303: EMAC_RXIPV4_FRGMT_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4FRAGOCT	RxIPv4 Fragmented Octets. The EMAC_RXIPV4_FRGMT_OCNT.RXIPV4FRAGOCT field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Fragmented Packets Counter Register

The EMAC_RXIPV4_FRGMT_PCNT register provides the number of good IPv4 datagrams received by the EMAC module with fragmentation.

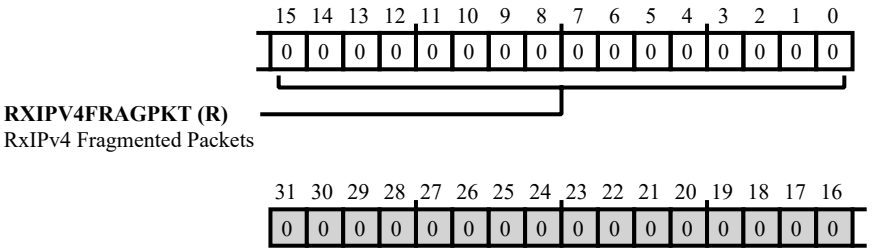


Figure 28-249: EMAC_RXIPV4_FRGMT_PCNT Register Diagram

Table 28-304: EMAC_RXIPV4_FRGMT_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4FRAGPKT	RxIPv4 Fragmented Packets. The EMAC_RXIPV4_FRGMT_PCNT.RXIPV4FRAGPKT field indicates the number of good IPv4 datagrams received with fragmentation.

Receive IPv4 Good Octets Counter Register

The EMAC_RXIPV4_OCNT_G register provides the number of bytes received by the EMAC module in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

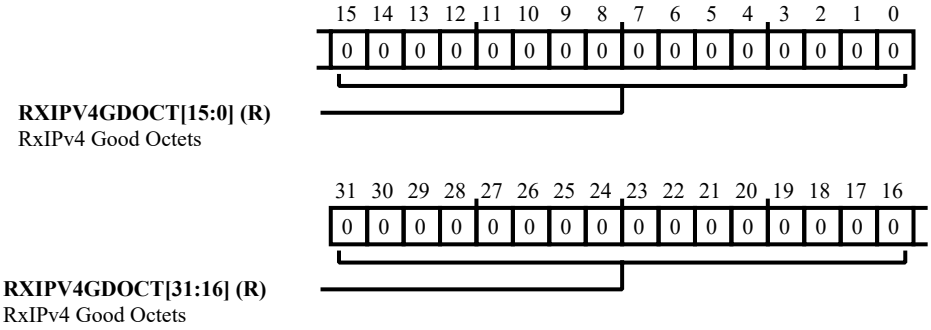


Figure 28-250: EMAC_RXIPV4_OCNT_G Register Diagram

Table 28-305: EMAC_RXIPV4_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4GDOCT	RxIPv4 Good Octets. The EMAC_RXIPV4_OCNT_G.RXIPV4GDOCT field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Good Packets Counter Register

The EMAC_RXIPV4_PCNT_G register provides the number of good IPv4 datagrams received by the EMAC module with the TCP, UDP, or ICMP payload.

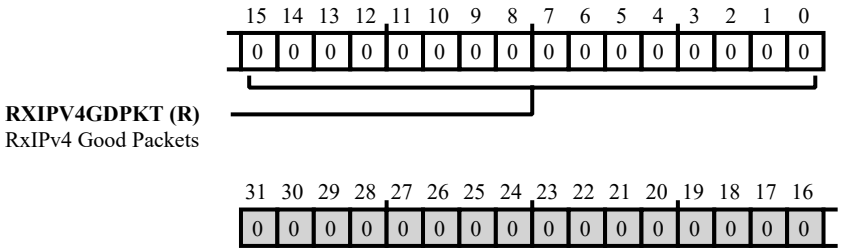


Figure 28-251: EMAC_RXIPV4_PCNT_G Register Diagram

Table 28-306: EMAC_RXIPV4_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4GDPKT	RxIPv4 Good Packets. The EMAC_RXIPV4_PCNT_G.RXIPV4GDPKT field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

Receive IPv4 Header Error Octets Counter Register

The EMAC_RXIPV4_HDRERR_OCNT register provides the number of bytes received by the EMAC module in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

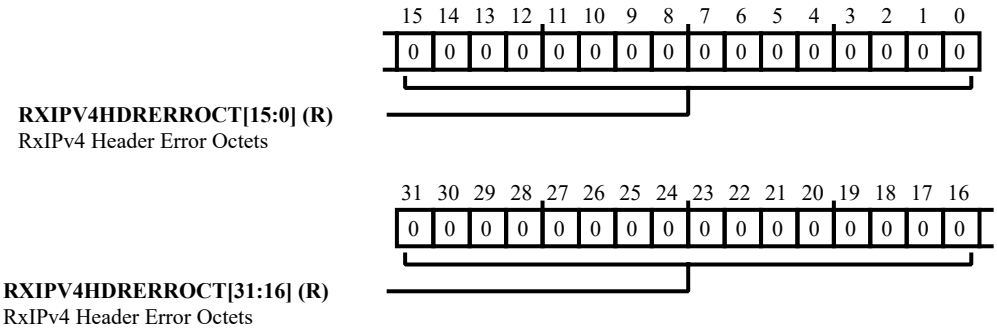


Figure 28-252: EMAC_RXIPV4_HDRERR_OCNT Register Diagram

Table 28-307: EMAC_RXIPV4_HDRERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4HDRERROCT	RxIPv4 Header Error Octets. The EMAC_RXIPV4_HDRERR_OCNT.RXIPV4HDRERROCT field indicates the number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Header Error Packets Counter Register

The EMAC_RXIPV4_HDRERR_PCNT register provides the number of IPv4 datagrams received by the EMAC module with header (checksum, length, or version mismatch) errors.

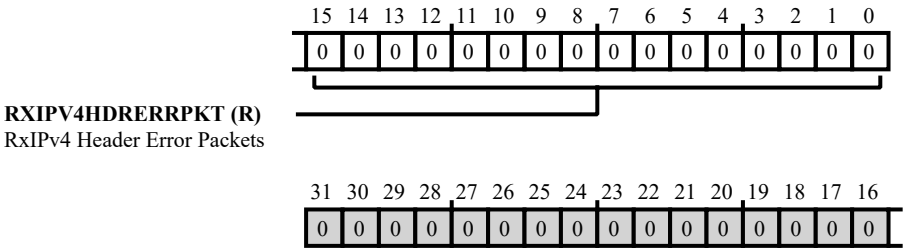


Figure 28-253: EMAC_RXIPV4_HDRERR_PCNT Register Diagram

Table 28-308: EMAC_RXIPV4_HDRERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4HDRERRPKT	RxIPv4 Header Error Packets. The EMAC_RXIPV4_HDRERR_PCNT.RXIPV4HDRERRPKT field indicates the number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors.

Receive IPv4 Payload Octets Counter Register

The EMAC_RXIPV4_PYLD_OCNT register provides the number of bytes received by the EMAC module in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

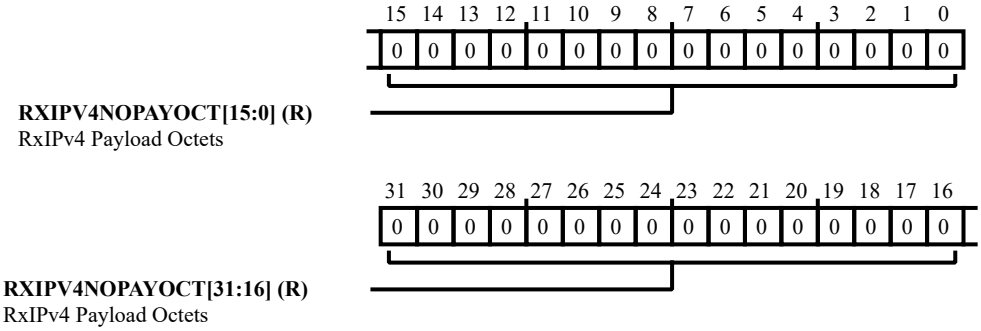


Figure 28-254: EMAC_RXIPV4_PYLD_OCNT Register Diagram

Table 28-309: EMAC_RXIPV4_PYLD_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4NOPAYOCT	RxIPv4 Payload Octets. The EMAC_RXIPV4_PYLD_OCNT.RXIPV4NOPAYOCT field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Payload Packets Counter Register

The EMAC_RXIPV4_PYLD_PCNT register provides the number of IPv4 datagram packets received by EMAC module that did not have a TCP, UDP, or ICMP payload.

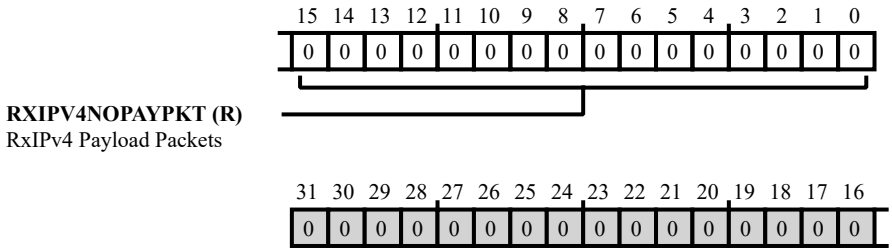


Figure 28-255: EMAC_RXIPV4_PYLD_PCNT Register Diagram

Table 28-310: EMAC_RXIPV4_PYLD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4NOPAYPKT	RxIPv4 Payload Packets. The EMAC_RXIPV4_PYLD_PCNT.RXIPV4NOPAYPKT field indicates the number of IPv4 datagram packets received that did not have a TCP, UDP, or ICMP payload.

Receive IPv4 UDP Checksum Disable Octets Counter Register

The EMAC_RXIPV4_UDPCSD_OCNT register provides the number of bytes received by the EMAC module in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

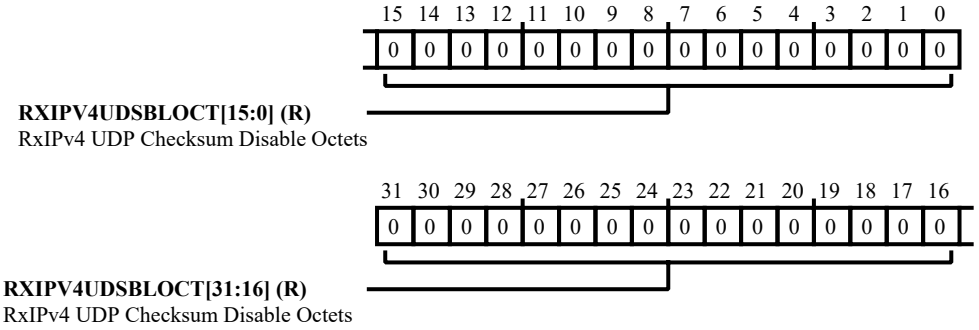


Figure 28-256: EMAC_RXIPV4_UDPCSD_OCNT Register Diagram

Table 28-311: EMAC_RXIPV4_UDPCSD_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4UDSBLOCT	RxIPv4 UDP Checksum Disable Octets. The EMAC_RXIPV4_UDPCSD_OCNT.RXIPV4UDSBLOCT field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 UDP Checksum Disable Packets Counter Register

The EMAC_RXIPV4_UDPCSD_PCNT register provides the number of good IPv4 datagrams received by the EMAC module that had a UDP payload with checksum disabled.

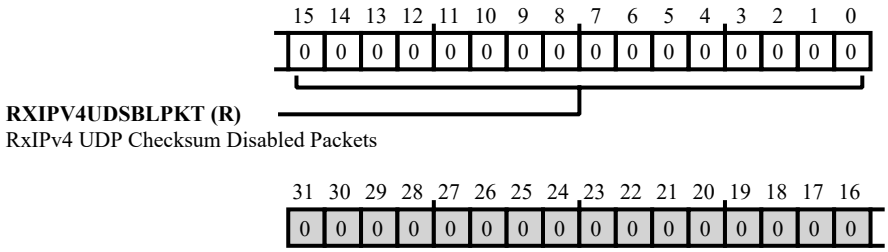


Figure 28-257: EMAC_RXIPV4_UDPCSD_PCNT Register Diagram

Table 28-312: EMAC_RXIPV4_UDPCSD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4UDSBLPKT	RxIPv4 UDP Checksum Disabled Packets. The EMAC_RXIPV4_UDPCSD_PCNT.RXIPV4UDSBLPKT field indicates the number of good IPv4 datagrams received that had a UDP payload with checksum disabled.

Receive IPv6 Good Octets Counter Register

The EMAC_RXIPV6_OCNT_G register provides the number of bytes received by the EMAC module in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

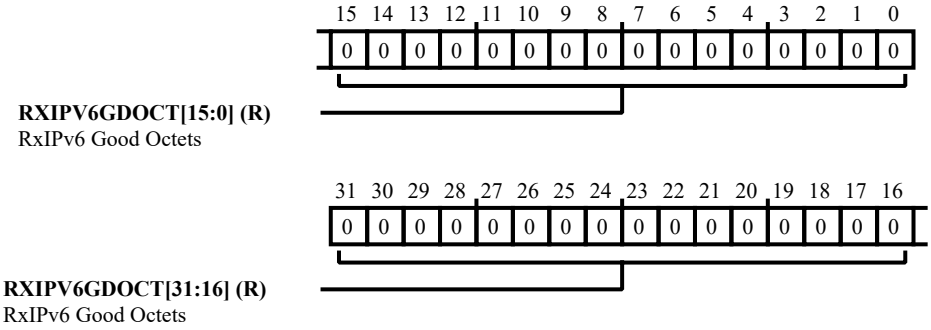


Figure 28-258: EMAC_RXIPV6_OCNT_G Register Diagram

Table 28-313: EMAC_RXIPV6_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV6GDOCT	RxIPv6 Good Octets. The EMAC_RXIPV6_OCNT_G.RXIPV6GDOCT field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv6 Good Packets Counter Register

The EMAC_RXIPV6_PCNT_G register provides the number of good IPv6 datagrams received by the EMAC module with the TCP, UDP, or ICMP payload.

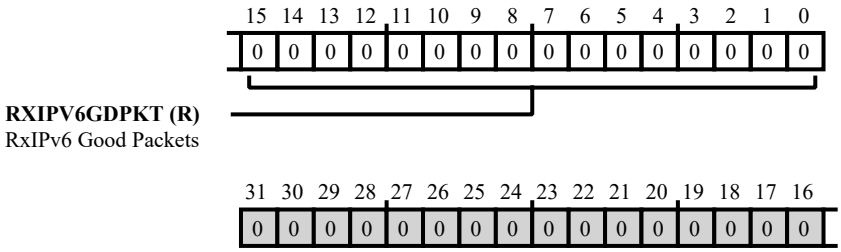


Figure 28-259: EMAC_RXIPV6_PCNT_G Register Diagram

Table 28-314: EMAC_RXIPV6_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV6GDPKT	RxIPv6 Good Packets. The EMAC_RXIPV6_PCNT_G.RXIPV6GDPKT field indicates the number of good IPv6 datagrams received with the TCP, UDP, or ICMP payload.

Receive IPv6 Header Error Octets Counter Register

The EMAC_RXIPV6_HDRERR_OCNT register provides the number of bytes received by the EMAC module in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

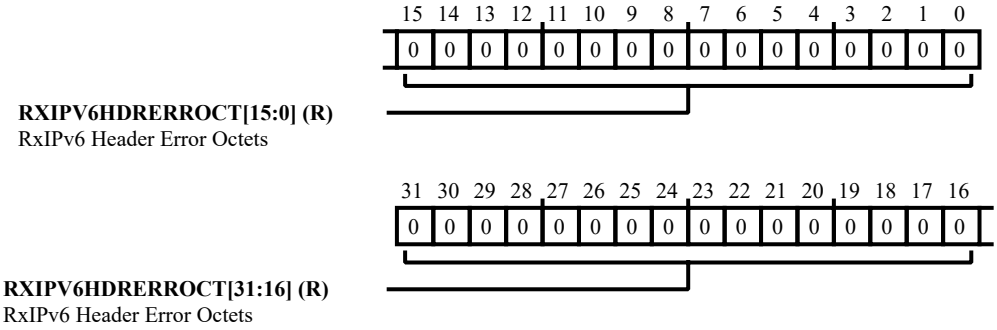


Figure 28-260: EMAC_RXIPV6_HDRERR_OCNT Register Diagram

Table 28-315: EMAC_RXIPV6_HDRERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV6HDRERROCT	RxIPv6 Header Error Octets. The EMAC_RXIPV6_HDRERR_OCNT.RXIPV6HDRERROCT field indicates the number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv6 Header Error Packets Counter Register

The EMAC_RXIPV6_HDRERR_PCNT register provides the number of IPv6 datagrams received by the EMAC module with header (length or version mismatch) errors.

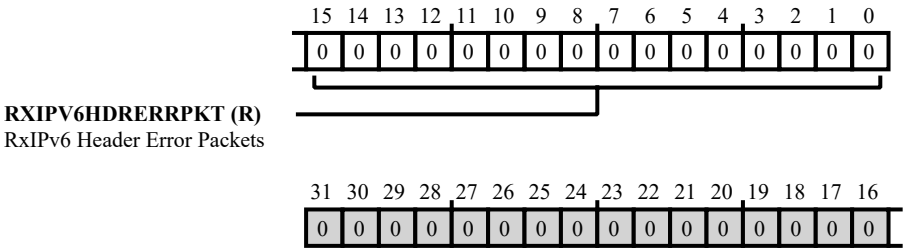


Figure 28-261: EMAC_RXIPV6_HDRERR_PCNT Register Diagram

Table 28-316: EMAC_RXIPV6_HDRERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV6HDRERRPKT	RxIPv6 Header Error Packets. The EMAC_RXIPV6_HDRERR_PCNT.RXIPV6HDRERRPKT field indicates the number of IPv6 datagrams received with header (length or version mismatch) errors.

Receive IPv6 Payload Octets Counter Register

The EMAC_RXIPV6_PYLD_OCNT register provides the number of bytes received by the EMAC module in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

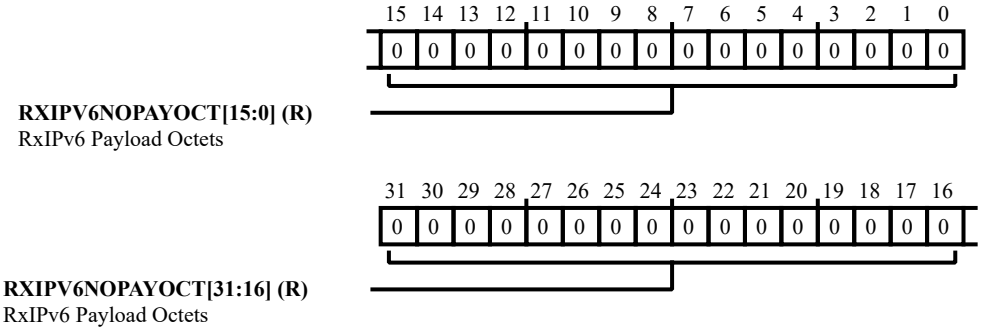


Figure 28-262: EMAC_RXIPV6_PYLD_OCNT Register Diagram

Table 28-317: EMAC_RXIPV6_PYLD_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV6NOPAYOCT	RxIPv6 Payload Octets. The EMAC_RXIPV6_PYLD_OCNT.RXIPV6NOPAYOCT field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv6 Payload Packets Counter Register

The EMAC_RXIPV6_PYLD_PCNT register provides the number of IPv6 datagram packets received by the EMAC module that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

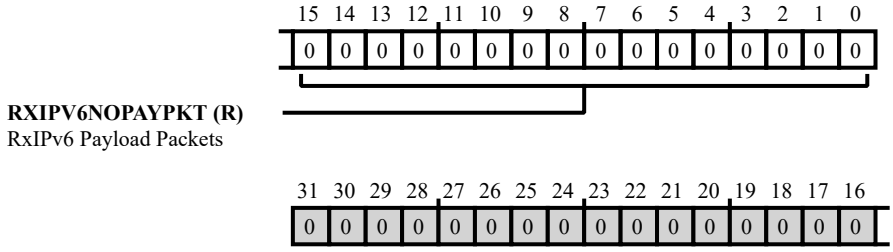


Figure 28-263: EMAC_RXIPV6_PYLD_PCNT Register Diagram

Table 28-318: EMAC_RXIPV6_PYLD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV6NOPAYPKT	RxIPv6 Payload Packets. The EMAC_RXIPV6_PYLD_PCNT.RXIPV6NOPAYPKT field indicates the number of IPv6 datagram packets received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

Receive TCP Error Octets Counter Register

The EMAC_RXTCP_ERR_OCNT register provides the number of bytes received by the EMAC module in a TCP segment that had checksum errors. This counter does not count IP header bytes.

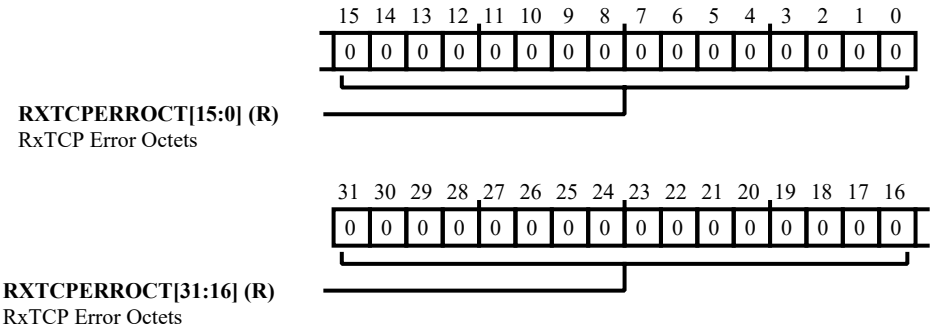


Figure 28-264: EMAC_RXTCP_ERR_OCNT Register Diagram

Table 28-319: EMAC_RXTCP_ERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXTCPERROCT	RxTCP Error Octets. The EMAC_RXTCP_ERR_OCNT.RXTCPERROCT field indicates the number of bytes received in a TCP segment that had checksum errors. This counter does not count IP header bytes.

Receive TCP Error Packets Counter Register

The EMAC_RXTCP_ERR_PCNT register provides the number of good IP datagrams received by the EMAC module whose TCP payload has a checksum error.

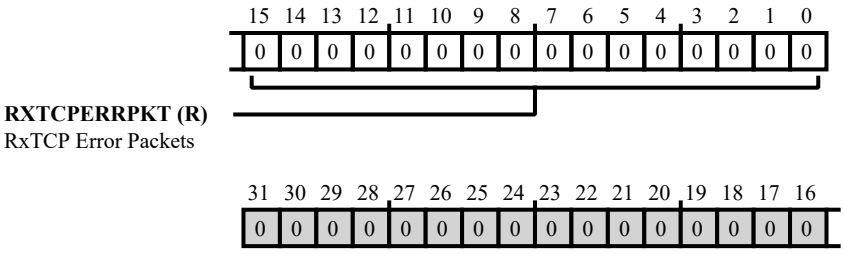


Figure 28-265: EMAC_RXTCP_ERR_PCNT Register Diagram

Table 28-320: EMAC_RXTCP_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXTCPERRPKT	RxTCP Error Packets. The EMAC_RXTCP_ERR_PCNT.RXTCPERRPKT field indicates the number of good IP datagrams received whose TCP payload has a checksum error.

Receive TCP Good Octets Counter Register

The EMAC_RXTCP_OCNT_G register provides the number of bytes received by the EMAC module in a good TCP segment. This counter does not count IP header bytes.

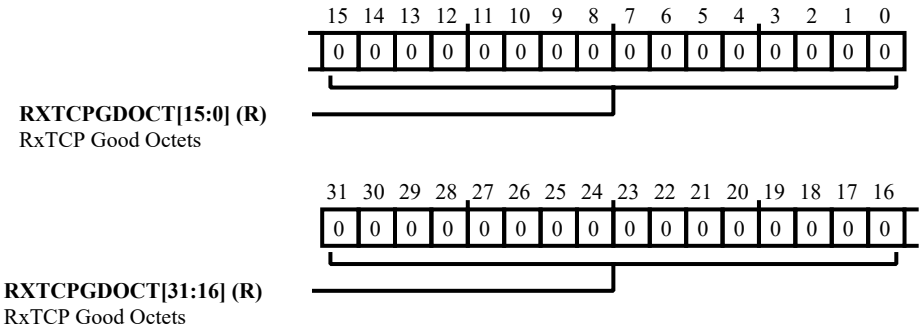


Figure 28-266: EMAC_RXTCP_OCNT_G Register Diagram

Table 28-321: EMAC_RXTCP_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXTCPGDOCT	RxTCP Good Octets. The EMAC_RXTCP_OCNT_G.RXTCPGDOCT field indicates the number of bytes received in a good TCP segment. This counter does not count IP header bytes.

Receive TCP Good Packets Counter Register

The EMAC_RXTCP_PCNT_G register provides the number of good IP datagrams received by the EMAC module with a good TCP payload.

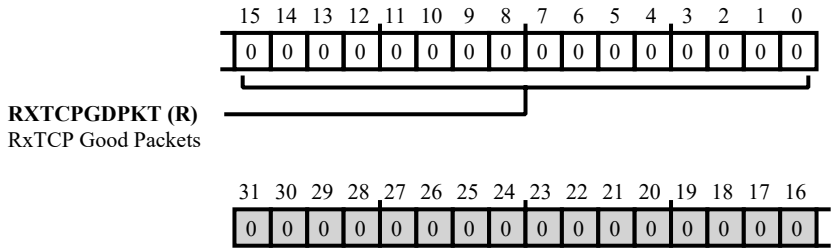


Figure 28-267: EMAC_RXTCP_PCNT_G Register Diagram

Table 28-322: EMAC_RXTCP_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXTCPGDPKT	RxTCP Good Packets. The EMAC_RXTCP_PCNT_G.RXTCPGDPKT field indicates the number of good IP datagrams received with a good TCP payload.

Receive UDP Error Octets Counter Register

The EMAC_RXUDP_ERR_OCNT register provides the number of bytes received by the EMAC module in a UDP segment that had checksum errors. This counter does not count IP header bytes.

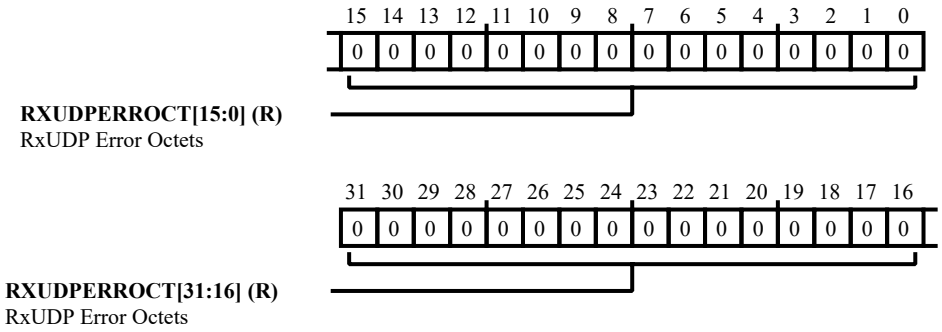


Figure 28-268: EMAC_RXUDP_ERR_OCNT Register Diagram

Table 28-323: EMAC_RXUDP_ERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUDPERROCT	RxUDP Error Octets. The EMAC_RXUDP_ERR_OCNT.RXUDPERROCT field indicates the number of bytes received in a UDP segment that had checksum errors. This counter does not count IP header bytes.

Receive UDP Error Packets Counter Register

The EMAC_RXUDP_ERR_PCNT register provides the number of good IP datagrams received by the EMAC module whose UDP payload has a checksum error.

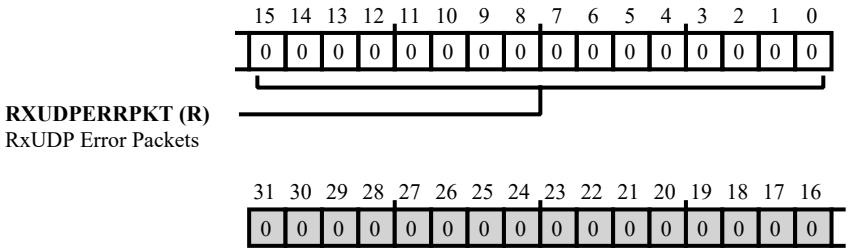


Figure 28-269: EMAC_RXUDP_ERR_PCNT Register Diagram

Table 28-324: EMAC_RXUDP_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXUDPERRPKT	RxUDP Error Packets. The EMAC_RXUDP_ERR_PCNT.RXUDPERRPKT field indicates the number of good IP datagrams received whose UDP payload has a checksum error.

Receive UDP Good Octets Counter Register

The EMAC_RXUDP_OCNT_G register provides the number of bytes received by the EMAC module in a good UDP segment. This counter does not count IP header bytes.

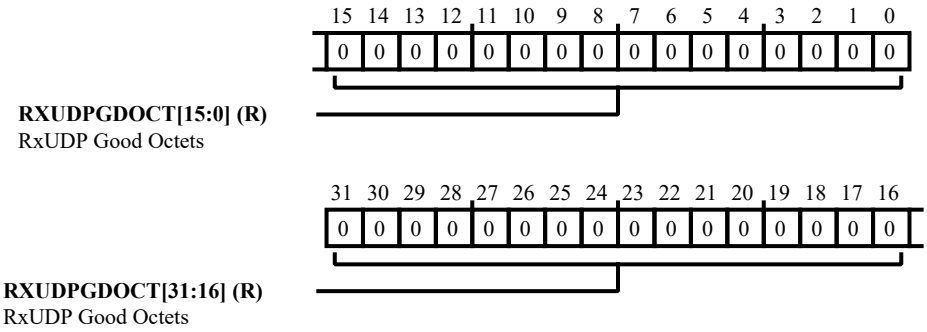


Figure 28-270: EMAC_RXUDP_OCNT_G Register Diagram

Table 28-325: EMAC_RXUDP_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUDPGDOCT	RxUDP Good Octets. The EMAC_RXUDP_OCNT_G.RXUDPGDOCT field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes.

Receive UDP Good Packets Counter Register

The EMAC_RXUDP_PCNT_G register provides the number of good IP datagrams received by the EMAC module with a good UDP payload. This counter is not updated when the EMAC_RXIPV4_UDPCSD_PCNT counter is incremented.

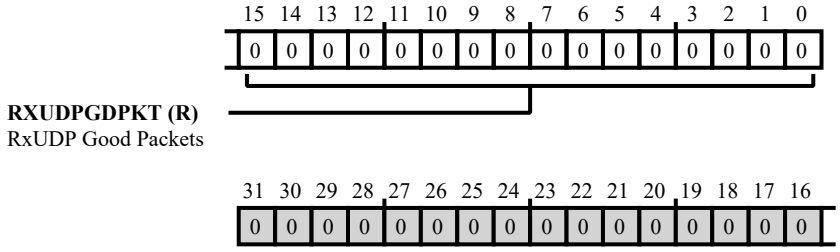


Figure 28-271: EMAC_RXUDP_PCNT_G Register Diagram

Table 28-326: EMAC_RXUDP_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXUDPGDPKT	RxUDP Good Packets. The EMAC_RXUDP_PCNT_G.RXUDPGDPKT field indicates the number of good IP datagrams received with a good UDP payload. This counter is not updated when the EMAC_RXIPV4_UDPCSD_PCNT counter is incremented.

Receive 1024 - Maximum Octets Packets Good Bad Counter Register

The EMAC_RX1024TOMAX_GB register provides the number of good and bad packets received by the EMAC module with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

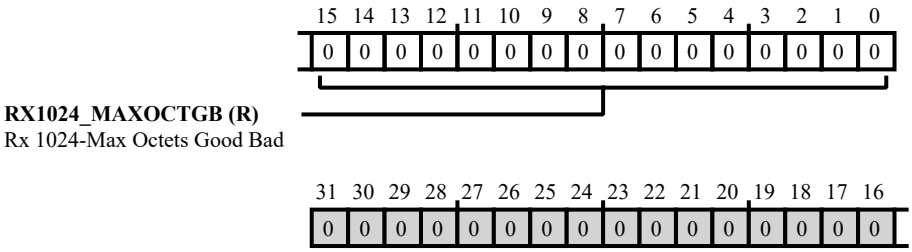


Figure 28-272: EMAC_RX1024TOMAX_GB Register Diagram

Table 28-327: EMAC_RX1024TOMAX_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX1024_MAXOCTGB	Rx 1024-Max Octets Good Bad. The EMAC_RX1024TOMAX_GB.RX1024_MAXOCTGB field indicates the number of good and bad packets received with a length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

Receive 128 - 255 Octets Packets Good Bad Counter Register

The EMAC_RX128TO255_GB register provides the number of good and bad packets received by the EMAC module with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

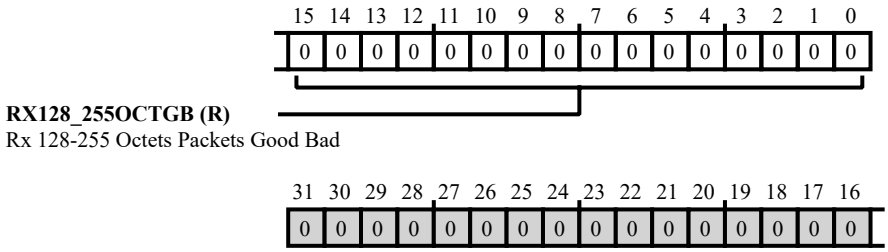


Figure 28-273: EMAC_RX128TO255_GB Register Diagram

Table 28-328: EMAC_RX128TO255_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX128_255OCTGB	Rx 128-255 Octets Packets Good Bad. The EMAC_RX128TO255_GB.RX128_255OCTGB field indicates the number of good and bad packets received with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

Receive 256 - 511 Octets Packets Good Bad Counter Register

The EMAC_RX256TO511_GB register provides the number of good and bad packets received by the EMAC module with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

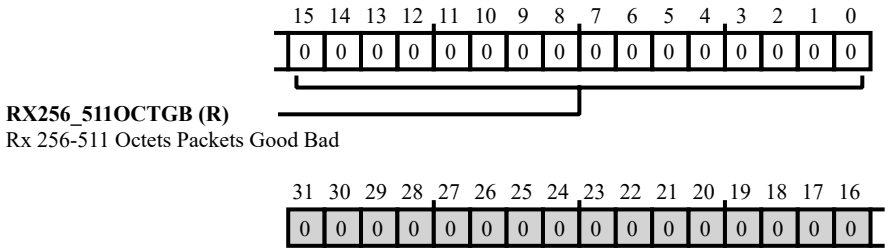


Figure 28-274: EMAC_RX256TO511_GB Register Diagram

Table 28-329: EMAC_RX256TO511_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX256_511OCTGB	Rx 256-511 Octets Packets Good Bad. The EMAC_RX256TO511_GB.RX256_511OCTGB field indicates the number of good and bad packets received with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

Receive 512 - 1023 Octets Packets Good Bad Counter Register

The EMAC_RX512TO1023_GB register provides the number of good and bad packets received by the EMAC module with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

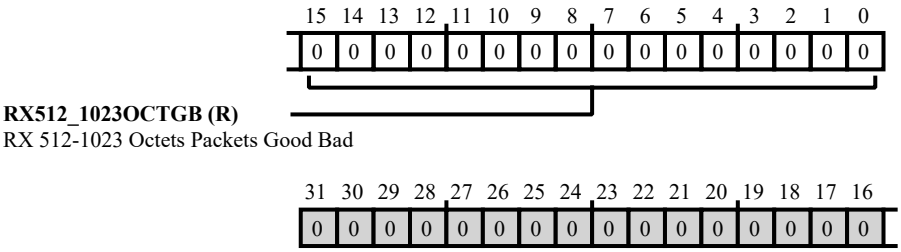


Figure 28-275: EMAC_RX512TO1023_GB Register Diagram

Table 28-330: EMAC_RX512TO1023_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX512_1023OCTGB	RX 512-1023 Octets Packets Good Bad. The EMAC_RX512TO1023_GB.RX512_1023OCTGB field indicates the number of good and bad packets received with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

Receive 64 Octets Packets Good Bad Counter Register

The EMAC_RX64_GB register provides the number of good and bad packets received by the EMAC module with length 64 bytes, exclusive of the preamble.

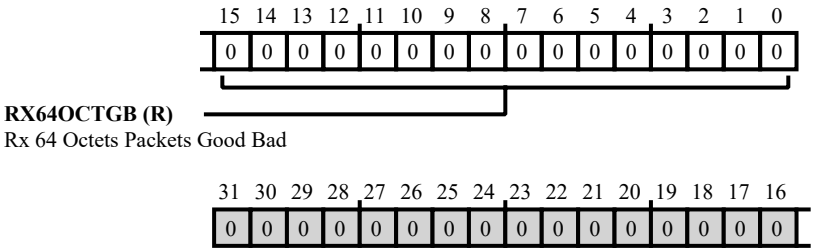


Figure 28-276: EMAC_RX64_GB Register Diagram

Table 28-331: EMAC_RX64_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX64OCTGB	Rx 64 Octets Packets Good Bad. The EMAC_RX64_GB.RX64OCTGB field indicates the number of good and bad packets received with length 64 bytes, exclusive of the preamble.

Receive 65 - 127 Octets Packets Good Bad Counter Register

The EMAC_RX65TO127_GB register provides the number of good and bad packets received by the EMAC module with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

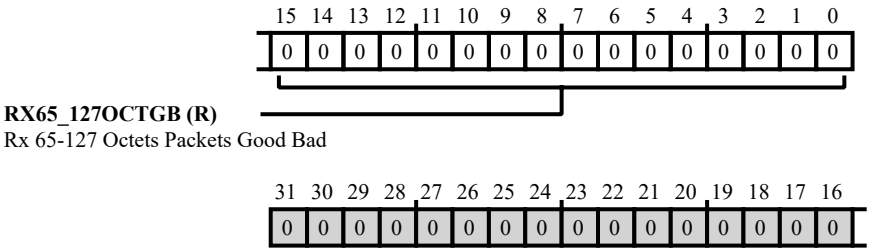


Figure 28-277: EMAC_RX65TO127_GB Register Diagram

Table 28-332: EMAC_RX65TO127_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX65_127OCTGB	Rx 65-127 Octets Packets Good Bad. The EMAC_RX65TO127_GB.RX65_127OCTGB field indicates the number of good and bad packets received with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

Receive Alignment Error Packets Counter Register

The EMAC_RX_ALGNERR_PCNT register provides the number of packets received by the EMAC module with alignment (dribble) error. It is valid only in 10/100 mode.

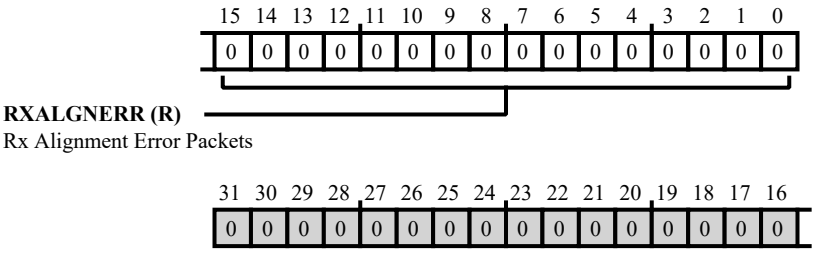


Figure 28-278: EMAC_RX_ALGNERR_PCNT Register Diagram

Table 28-333: EMAC_RX_ALGNERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXALGNERR	Rx Alignment Error Packets. The EMAC_RX_ALGNERR_PCNT.RXALGNERR field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Receive Broadcast Packets Good Counter Register

The EMAC_RX_BCERR_PCNT_G register provides the number of good broadcast packets received by the EMAC module.

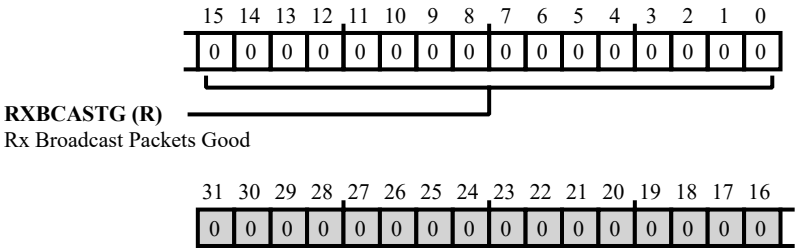


Figure 28-279: EMAC_RX_BCERR_PCNT_G Register Diagram

Table 28-334: EMAC_RX_BCERR_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXBCASTG	Rx Broadcast Packets Good. The EMAC_RX_BCERR_PCNT_G.RXBCASTG field indicates the number of good broadcast packets received.

Receive CRC Error Packets Counter Register

The EMAC_RX_CRCERR_PCNT register provides the number of packets received by the EMAC module with CRC error.

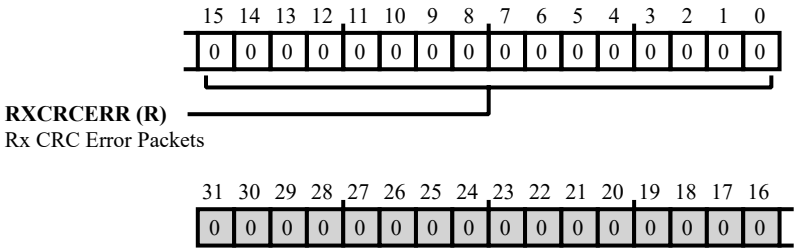


Figure 28-280: EMAC_RX_CRCERR_PCNT Register Diagram

Table 28-335: EMAC_RX_CRCERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXCRCERR	Rx CRC Error Packets. The EMAC_RX_CRCERR_PCNT.RXCRCERR field indicates the number of packets received with CRC error.

Receive Control Packets Good Counter Register

The EMAC_RX_CTL_PCNT_G register provides the number of good control packets received by the EMAC module.

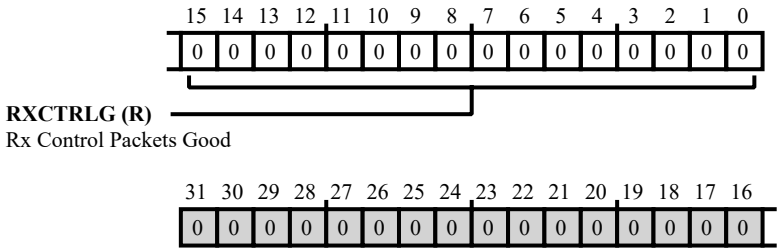


Figure 28-281: EMAC_RX_CTL_PCNT_G Register Diagram

Table 28-336: EMAC_RX_CTL_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXCTRLG	Rx Control Packets Good. The EMAC_RX_CTL_PCNT_G.RXCTRLG field indicates the number of good control packets received.

Receive FIFO Overflow Packets Counter Register

The EMAC_RX_FIFOOVFL_PCNT register provides the number of missed received packets because of FIFO overflow in the EMAC module.

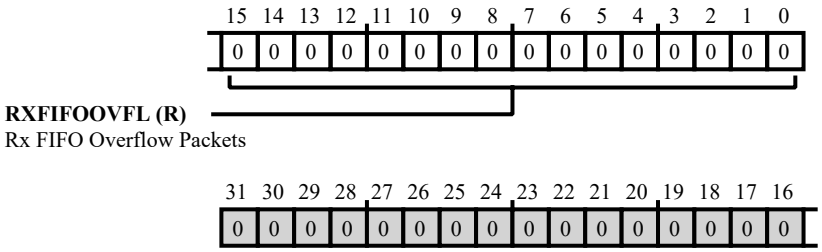


Figure 28-282: EMAC_RX_FIFOOVFL_PCNT Register Diagram

Table 28-337: EMAC_RX_FIFOOVFL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXFIFOOVFL	Rx FIFO Overflow Packets. The EMAC_RX_FIFOOVFL_PCNT.RXFIFOOVFL field indicates the number of missed received packets because of FIFO overflow.

Receive Jabber Error Packets Counter Register

The EMAC_RX_JBERR_PCNT register provides the number of giant packets received by the EMAC module with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. When jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1), packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

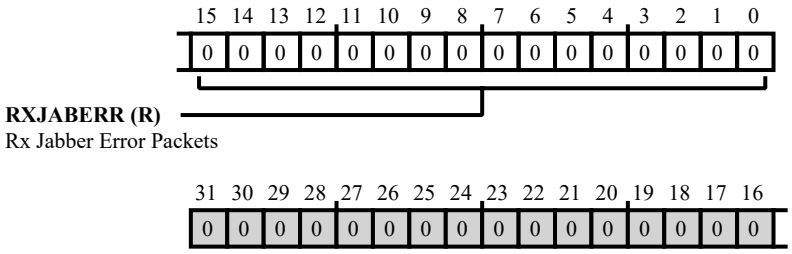


Figure 28-283: EMAC_RX_JBERR_PCNT Register Diagram

Table 28-338: EMAC_RX_JBERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXJABERR	Rx Jabber Error Packets. The EMAC_RX_JBERR_PCNT.RXJABERR field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1), packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

Receive LPI Transition Counter Register

The EMAC_RX_LPITRAN_CNT register provides the number of times the EMAC module has entered Rx LPI.

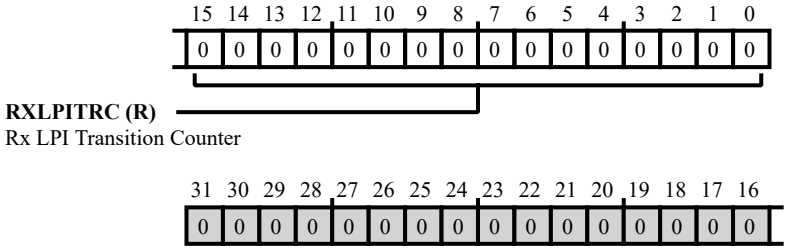


Figure 28-284: EMAC_RX_LPITRAN_CNT Register Diagram

Table 28-339: EMAC_RX_LPITRAN_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXLPITRC	Rx LPI Transition Counter. The EMAC_RX_LPITRAN_CNT.RXLPITRC field indicates the number of times Rx LPI entry has occurred.

Receive LPI Microseconds Counter Register

The EMAC_RX_LPIUSEC_CNT register provides the number of microseconds Rx LPI is sampled by the EMAC module.

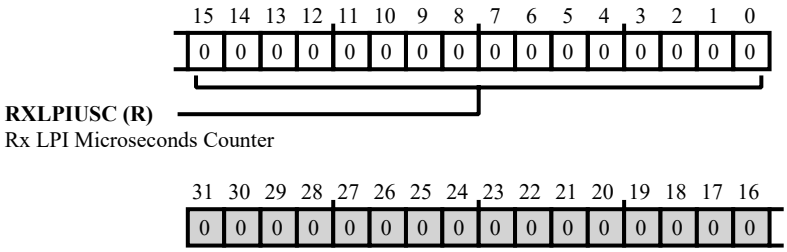


Figure 28-285: EMAC_RX_LPIUSEC_CNT Register Diagram

Table 28-340: EMAC_RX_LPIUSEC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXLPIUSC	Rx LPI Microseconds Counter. The EMAC_RX_LPIUSEC_CNT.RXLPIUSC field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI entry and exit, the timer value can have an error of +/- 1 microsecond.

Receive Length Error Packets Counter Register

The EMAC_RX_LENERR_PCNT register provides the number of packets received by the EMAC module with length error (Length Type field not equal to packet size), for all packets with valid length field.

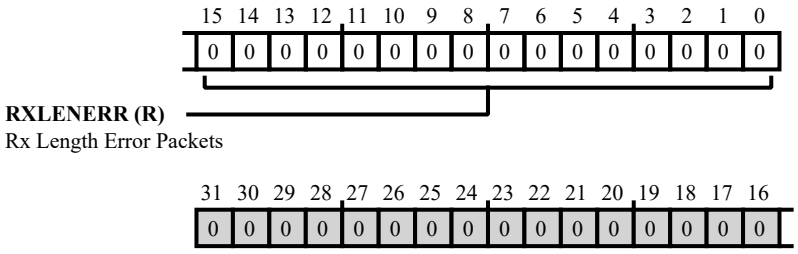


Figure 28-286: EMAC_RX_LENERR_PCNT Register Diagram

Table 28-341: EMAC_RX_LENERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXLENERR	Rx Length Error Packets. The EMAC_RX_LENERR_PCNT.RXLENERR field indicates the number of packets received with length error (length type field not equal to packet size), for all packets with valid length field.

Receive Multicast Packets Good Counter Register

The EMAC_RX_MCST_PCNT_G register provides the number of good multicast packets received by the EMAC module.

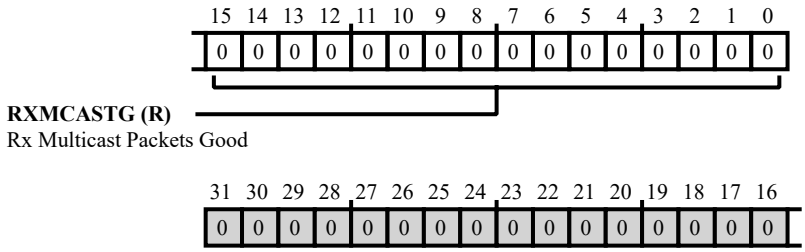


Figure 28-287: EMAC_RX_MCST_PCNT_G Register Diagram

Table 28-342: EMAC_RX_MCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXMCASTG	Rx Multicast Packets Good. The EMAC_RX_MCST_PCNT_G.RXMCASTG field indicates the number of good multicast packets received.

Receive Octet Good Counter Register

The EMAC_RX_OCNT_G register provides the number of bytes received by the EMAC module, exclusive of preamble, only in good packets.

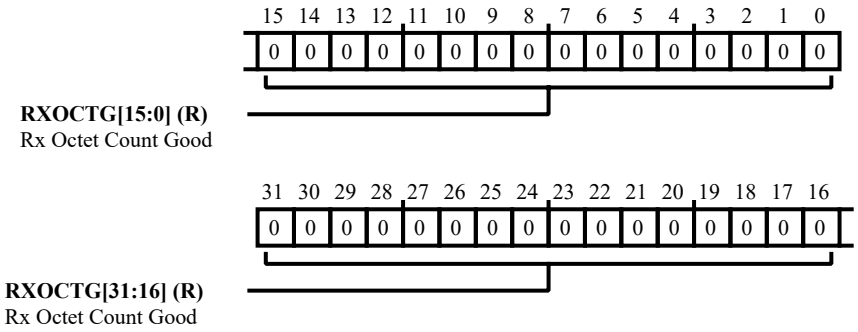


Figure 28-288: EMAC_RX_OCNT_G Register Diagram

Table 28-343: EMAC_RX_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXOCTG	Rx Octet Count Good. The EMAC_RX_OCNT_G.RXOCTG field indicates the number of bytes received, exclusive of preamble, only in good packets.

Receive Octet Good Bad Counter Register

The EMAC_RX_OCNT_GB register provides the number of bytes received by the EMAC module, exclusive of preamble, in good and bad packets.

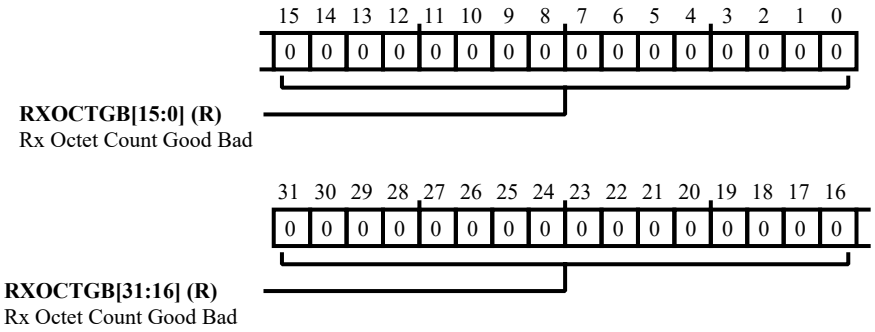


Figure 28-289: EMAC_RX_OCNT_GB Register Diagram

Table 28-344: EMAC_RX_OCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXOCTGB	Rx Octet Count Good Bad. The EMAC_RX_OCNT_GB.RXOCTGB field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

Receive Out of Range Type Packets Counter Register

The EMAC_RX_ORTYPE_PCNT register provides the number of packets received by the EMAC module with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

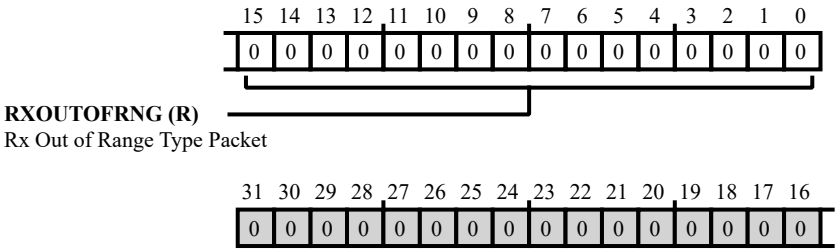


Figure 28-290: EMAC_RX_ORTYPE_PCNT Register Diagram

Table 28-345: EMAC_RX_ORTYPE_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXOUTOFRNG	Rx Out of Range Type Packet. The EMAC_RX_ORTYPE_PCNT.RXOUTOFRNG field indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

Receive Oversize Packets Good Counter Register

The EMAC_RX_OVRSZ_PCNT register provides the number of packets received by the EMAC module without errors, with length greater than the max size (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes when EMAC_MAC_CFG MAC.S2KP = 1).

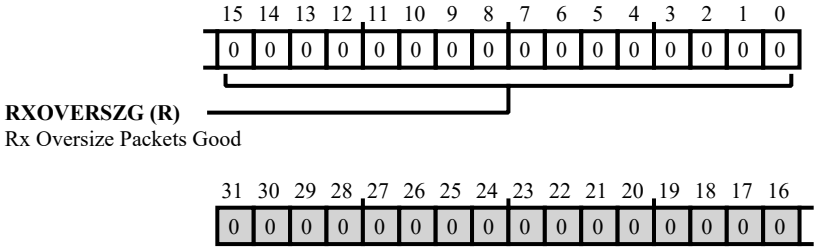


Figure 28-291: EMAC_RX_OVRSZ_PCNT Register Diagram

Table 28-346: EMAC_RX_OVRSZ_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXOVERSZG	Rx Oversize Packets Good. The EMAC_RX_OVRSZ_PCNT.RXOVERSZG field indicates the number of packets received without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes when EMAC_MAC_CFG MAC.S2KP = 1).

Receive Packets Good Bad Counter Register

The EMAC_RX_PCNT_GB register provides the number of good and bad packets received by the EMAC module.

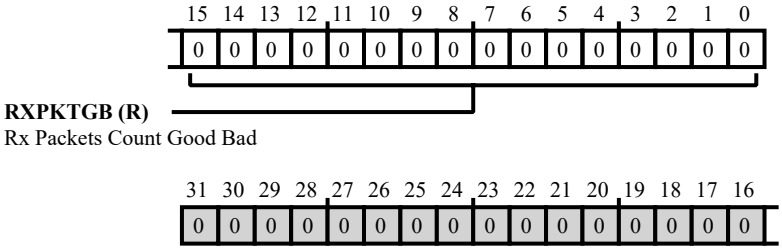


Figure 28-292: EMAC_RX_PCNT_GB Register Diagram

Table 28-347: EMAC_RX_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXPKTGB	Rx Packets Count Good Bad. The EMAC_RX_PCNT_GB.RXPKTGB field indicates the number of good and bad packets received.

Receive Pause Packets Counter Register

The EMAC_RX_PAUSE_PCNT register provides the number of good and valid Pause packets received by the EMAC module.

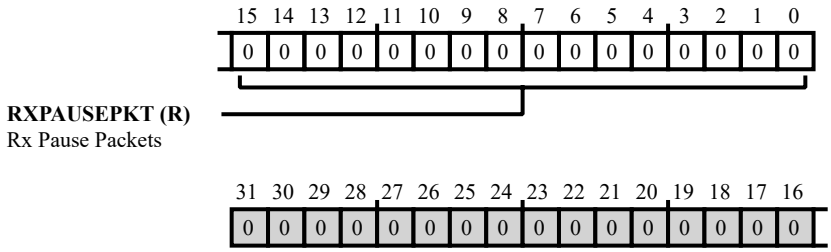


Figure 28-293: EMAC_RX_PAUSE_PCNT Register Diagram

Table 28-348: EMAC_RX_PAUSE_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXPAUSEPKT	Rx Pause Packets. The EMAC_RX_PAUSE_PCNT.RXPAUSEPKT field indicates the number of good and valid Pause packets received.

Receive Error Packets Counter Register

The EMAC_RX_ERR_PCNT register provides the number of packets received by the EMAC module with receive error or packet extension error on the GMII or MII interface.

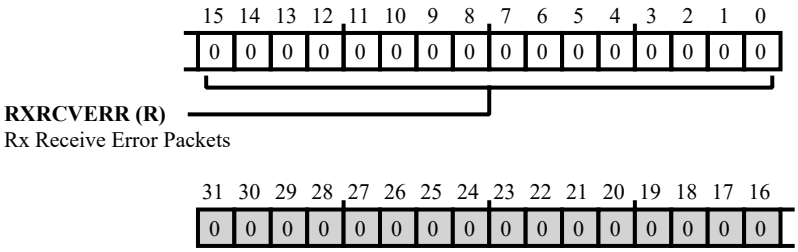


Figure 28-294: EMAC_RX_ERR_PCNT Register Diagram

Table 28-349: EMAC_RX_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXRCVERR	Rx Receive Error Packets. The EMAC_RX_ERR_PCNT.RXRCVERR field indicates the number of packets received with Receive error or Packet Extension error on the GMII or MII interface.

Receive Runt Error Packets Counter Register

The EMAC_RX_RNTERR_PCNT register provides the number of packets received by the EMAC module with runt (length less than 64 bytes and CRC error) error.

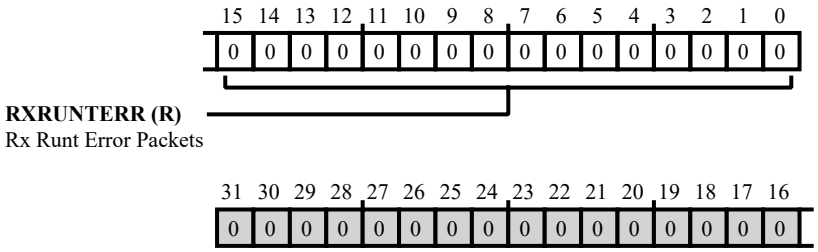


Figure 28-295: EMAC_RX_RNTERR_PCNT Register Diagram

Table 28-350: EMAC_RX_RNTERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXRUNTERR	Rx Runt Error Packets. The EMAC_RX_RNTERR_PCNT.RXRUNTERR field indicates the number of packets received with runt (length less than 64 bytes and CRC error) error.

Receive Undersize Packets Good Counter Register

The EMAC_RX_UNDRSZ_PCNT_G register provides the number of packets received by the EMAC module with length less than 64 bytes, without any errors.

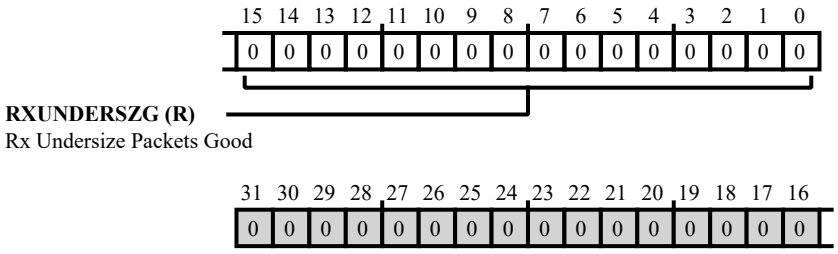


Figure 28-296: EMAC_RX_UNDRSZ_PCNT_G Register Diagram

Table 28-351: EMAC_RX_UNDRSZ_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXUNDERSZG	Rx Undersize Packets Good. The EMAC_RX_UNDRSZ_PCNT_G.RXUNDERSZG field indicates the number of packets received with length less than 64 bytes, without any errors.

Receive Unicast Packets Good Counter Register

The EMAC_RX_UCST_PCNT_G register provides the number of good unicast packets received by the EMAC module.

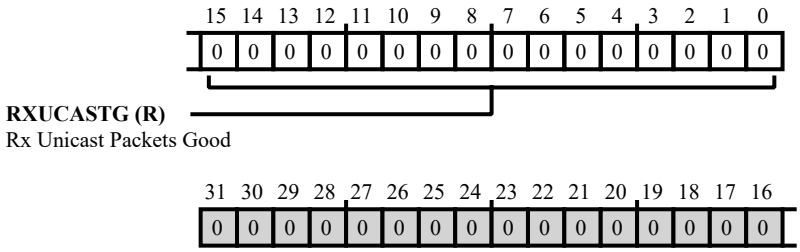


Figure 28-297: EMAC_RX_UCST_PCNT_G Register Diagram

Table 28-352: EMAC_RX_UCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXUCASTG	Rx Unicast Packets Good. The EMAC_RX_UCST_PCNT_G.RXUCASTG field indicates the number of good unicast packets received.

Receive VLAN Packets Good Bad Counter Register

The EMAC_RX_VLAN_PCNT_GB register provides the number of good and bad VLAN packets received by the EMAC module.

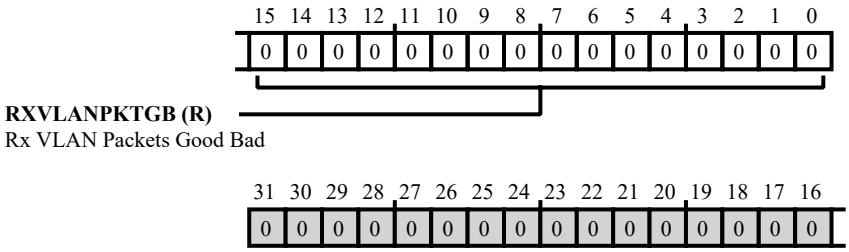


Figure 28-298: EMAC_RX_VLAN_PCNT_GB Register Diagram

Table 28-353: EMAC_RX_VLAN_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXVLANPKTGB	Rx VLAN Packets Good Bad. The EMAC_RX_VLAN_PCNT_GB.RXVLANPKTGB field indicates the number of good and bad VLAN packets received.

Receive Watchdog Error Packets Counter Register

The EMAC_RX_WDTERR_PCNT register provides the number of packets received by the EMAC module with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when EMAC_MAC_CFG.JE and EMAC_MAC_CFG.WD = 0), 10,240 bytes (EMAC_MAC_CFG.JE = 1 and EMAC_MAC_CFG.WD = 0), 16,384 bytes (when EMAC_MAC_CFG.WD = 1) or the value programmed in the EMAC_WDT_TMOUT register).

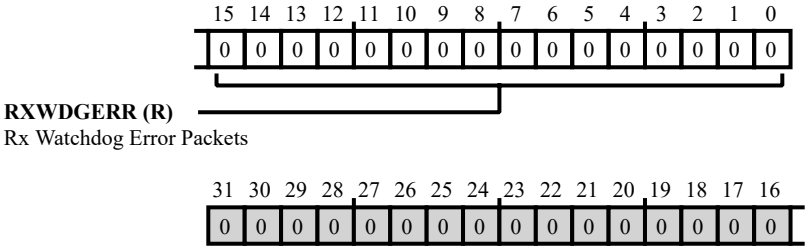


Figure 28-299: EMAC_RX_WDTERR_PCNT Register Diagram

Table 28-354: EMAC_RX_WDTERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXWDGERR	Rx Watchdog Error Packets. The EMAC_RX_WDTERR_PCNT.RXWDGERR field indicates the number of packets received with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when EMAC_MAC_CFG MAC.JE = 0 and EMAC_MAC_CFG MAC.WD = 0), 10,240 bytes (when EMAC_MAC_CFG MAC.JE = 1 and EMAC_MAC_CFG MAC.WD = 0), 16,384 bytes (when EMAC_MAC_CFG MAC.WD = 1) or the value programmed in the EMAC_WDT_TMOUT register).

Transmit 1024-Max Octets Packets Good Bad Counter Register

The EMAC_TX1024TOMAX_GB register provides the number of good and bad packets transmitted by the EMAC module with length 1024 to maxsize (inclusive) bytes, exclusive of preamble and retried packets.

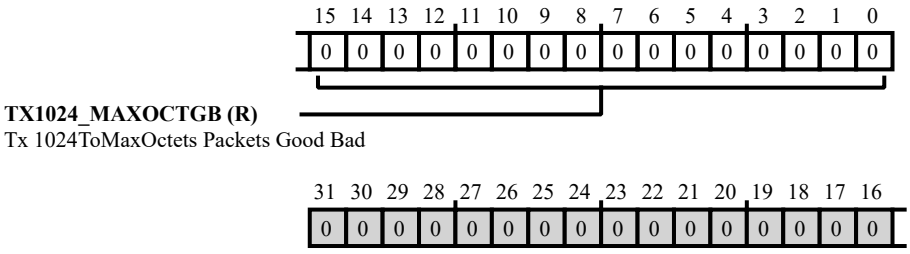


Figure 28-300: EMAC_TX1024TOMAX_GB Register Diagram

Table 28-355: EMAC_TX1024TOMAX_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TX1024_MAXOCTGB	Tx 1024ToMaxOctets Packets Good Bad. The EMAC_TX1024TOMAX_GB.TX1024_MAXOCTGB field indicates the number of good and bad packets transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 128-255 Octets Packets Good Bad Counter Register

The EMAC_TX128TO255_GB register provides the number of good and bad packets transmitted by the EMAC module with length between 128 to 255 (inclusive) bytes, exclusive of preamble and retried packets.

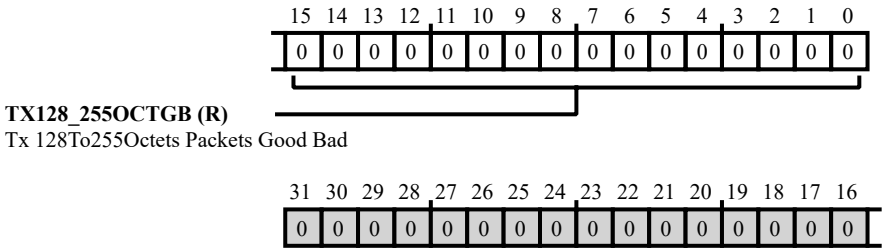


Figure 28-301: EMAC_TX128TO255_GB Register Diagram

Table 28-356: EMAC_TX128TO255_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TX128_255OCTGB	Tx 128To255Octets Packets Good Bad. The EMAC_TX128TO255_GB.TX128_255OCTGB field indicates the number of good and bad packets transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 256-511 Octets Packets Good Bad Counter Register

The EMAC_TX256TO511_GB register provides the number of good and bad packets transmitted by the EMAC module with length between 256 to 511 (inclusive) bytes, exclusive of preamble and retried packets.

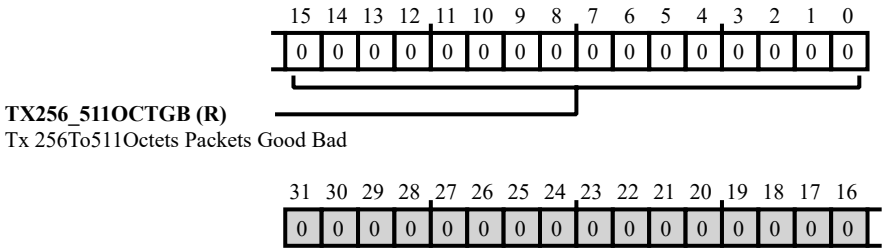


Figure 28-302: EMAC_TX256TO511_GB Register Diagram

Table 28-357: EMAC_TX256TO511_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TX256_511OCTGB	Tx 256To511Octets Packets Good Bad. The EMAC_TX256TO511_GB.TX256_511OCTGB field indicates the number of good and bad packets transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 512-1023 Octets Packets Good Bad Counter Register

The EMAC_TX512TO1023_GB register provides the number of good and bad packets transmitted by the EMAC module with length 512 to 1023 (inclusive) bytes, exclusive of preamble and retried packets.

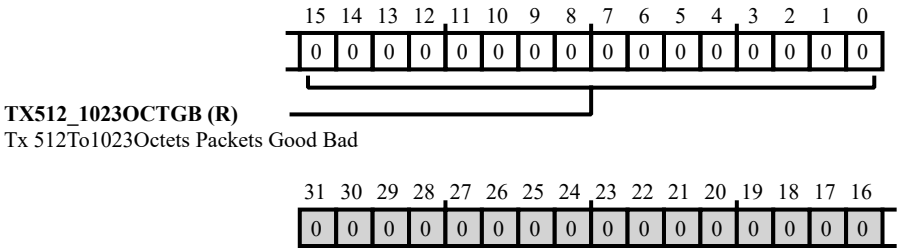


Figure 28-303: EMAC_TX512TO1023_GB Register Diagram

Table 28-358: EMAC_TX512TO1023_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TX512_1023OCTGB	Tx 512To1023Octets Packets Good Bad. The EMAC_TX512TO1023_GB.TX512_1023OCTGB field indicates the number of good and bad packets transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 64 Octets Packets Good Bad Counter Register

The EMAC_TX64_GB register provides the number of good and bad packets transmitted by the EMAC module with length 64 bytes, exclusive of preamble and retried packets.

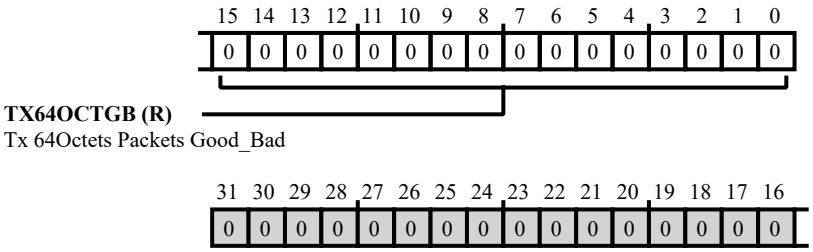


Figure 28-304: EMAC_TX64_GB Register Diagram

Table 28-359: EMAC_TX64_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TX64OCTGB	Tx 64Octets Packets Good_Bad. The EMAC_TX64_GB.TX64OCTGB field indicates the number of good and bad packets transmitted with length 64 bytes, exclusive of preamble and retried packets.

Transmit 65-127 Octets Packets Good Bad Counter Register

The EMAC_TX65TO127_GB register provides the number of good and bad packets transmitted by the EMAC module with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

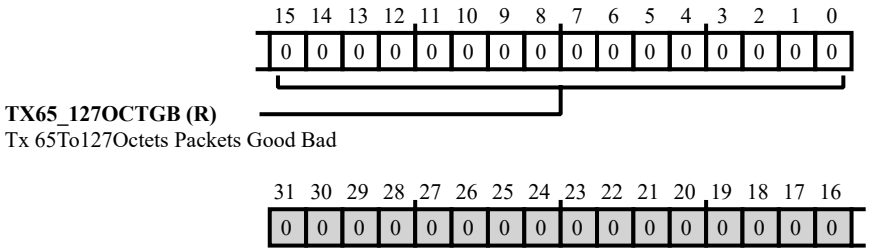


Figure 28-305: EMAC_TX65TO127_GB Register Diagram

Table 28-360: EMAC_TX65TO127_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TX65_127OCTGB	Tx 65To127Octets Packets Good Bad. The EMAC_TX65TO127_GB.TX65_127OCTGB field indicates the number of good and bad packets transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit Broadcast Packets Good Counter Register

The EMAC_TX_BCST_PCNT_G register provides the number of good broadcast packets transmitted by the EMAC module.

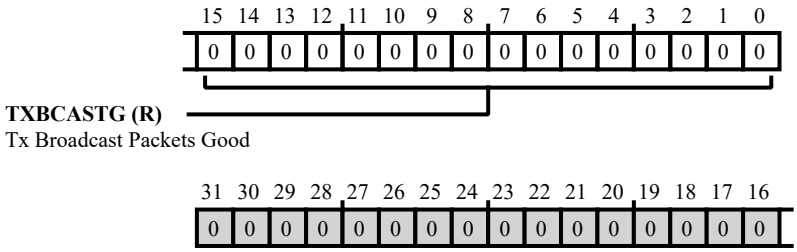


Figure 28-306: EMAC_TX_BCST_PCNT_G Register Diagram

Table 28-361: EMAC_TX_BCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXBCASTG	Tx Broadcast Packets Good. The EMAC_TX_BCST_PCNT_G.TXBCASTG field indicates the number of good broadcast packets transmitted.

Transmit Broadcast Packets Good Bad Counter Register

The EMAC_TX_BCST_PCNT_GB register provides the number of good and bad broadcast packets transmitted by the EMAC module.

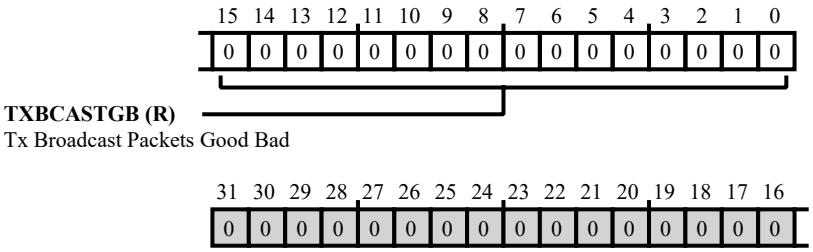


Figure 28-307: EMAC_TX_BCST_PCNT_GB Register Diagram

Table 28-362: EMAC_TX_BCST_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXBCASTGB	Tx Broadcast Packets Good Bad. The EMAC_TX_BCST_PCNT_GB.TXBCASTGB field indicates the number of good and bad broadcast packets transmitted.

Transmit Carrier Error Packets Counter Register

The EMAC_TX_CRERR_PCNT register provides the number of packets aborted by the EMAC module because of carrier sense error (no carrier or loss of carrier).

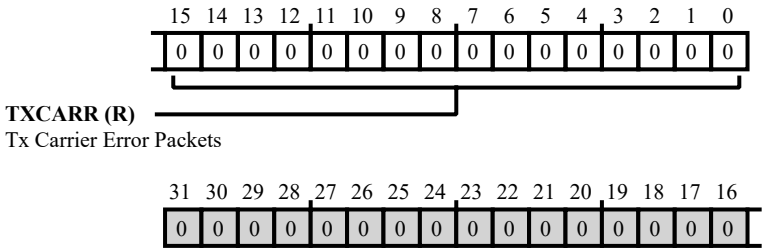


Figure 28-308: EMAC_TX_CRERR_PCNT Register Diagram

Table 28-363: EMAC_TX_CRERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXCARR	Tx Carrier Error Packets. The EMAC_TX_CRERR_PCNT.TXCARR field indicates the number of packets aborted because of carrier sense error (no carrier or loss of carrier).

Transmit Deferred Packets Counter Register

The EMAC_TX_DFRD_PCNT register provides the number of successfully transmitted by the EMAC module after a deferral in the half-duplex mode.

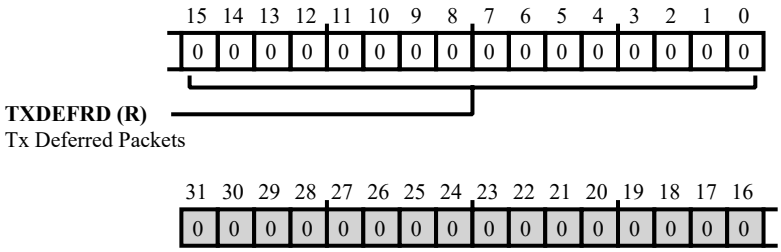


Figure 28-309: EMAC_TX_DFRD_PCNT Register Diagram

Table 28-364: EMAC_TX_DFRD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXDEFRD	Tx Deferred Packets. The EMAC_TX_DFRD_PCNT.TXDEFRD field indicates the number of successfully transmitted after a deferral in the half-duplex mode.

Transmit Excessive Collision Packets Counter Register

The EMAC_TX_EXCOL_PCNT register provides the number of packets aborted by the EMAC module because of excessive (16) collision errors.

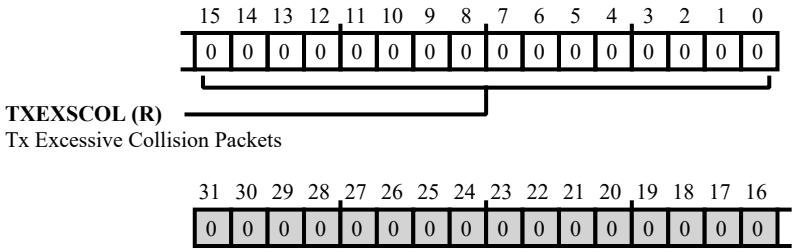


Figure 28-310: EMAC_TX_EXCOL_PCNT Register Diagram

Table 28-365: EMAC_TX_EXCOL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXEXSCOL	Tx Excessive Collision Packets. The EMAC_TX_EXCOL_PCNT.TXEXSCOL field indicates the number of packets aborted because of excessive (16) collision errors.

Transmit Excessive Deferral Error Counter Register

The EMAC_TX_EXDFRL_PCNT register provides the number of packets aborted by the EMAC module because of excessive deferral error (deferred for more than two max-sized packet times).

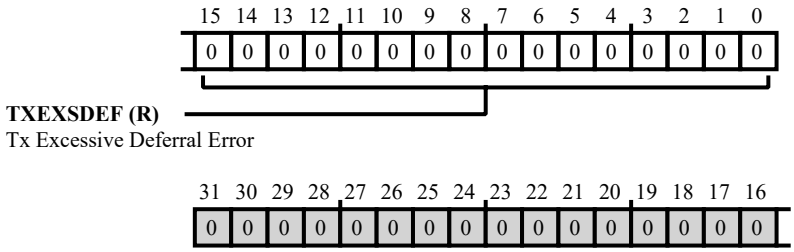


Figure 28-311: EMAC_TX_EXDFRL_PCNT Register Diagram

Table 28-366: EMAC_TX_EXDFRL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXEXSDEF	Tx Excessive Deferral Error. The EMAC_TX_EXDFRL_PCNT.TXEXSDEF field indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times).

Transmit LPI Transition Counter Register

The EMAC_TX_LPITRAN_CNT register provides the number of times the EMAC module has entered Tx LPI.

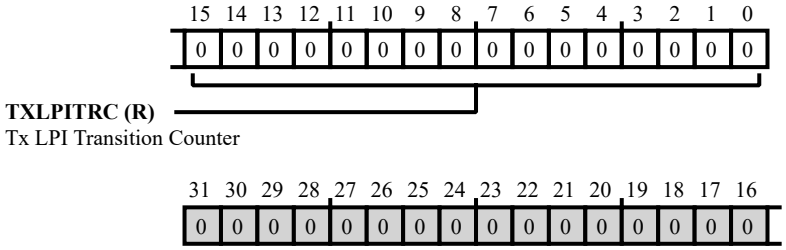


Figure 28-312: EMAC_TX_LPITRAN_CNT Register Diagram

Table 28-367: EMAC_TX_LPITRAN_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXLPITRC	Tx LPI Transition Counter. The EMAC_TX_LPITRAN_CNT.TXLPITRC field indicates the number of times Tx LPI entry has occurred. Even if Tx LPI entry occurs in automate mode (EMAC_LPI_CTLSTAT.LPITXA = 1), the counter increments.

Transmit LPI Microseconds Counter Register

The EMAC_TX_LPIUSEC_CNT register provides the number of microseconds Tx LPI is asserted by the EMAC module.

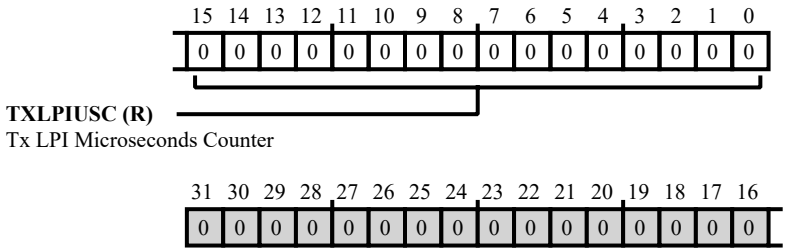


Figure 28-313: EMAC_TX_LPIUSEC_CNT Register Diagram

Table 28-368: EMAC_TX_LPIUSEC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXLPIUSC	Tx LPI Microseconds Counter. The EMAC_TX_LPIUSEC_CNT.TXLPIUSC field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI entry and exit, the timer value can have an error of +/- 1 microsecond.

Transmit Late Collision Packets Counter Register

The EMAC_TX_LTCOL_PCNT register provides the number of packets aborted by the EMAC module because of late collision error.

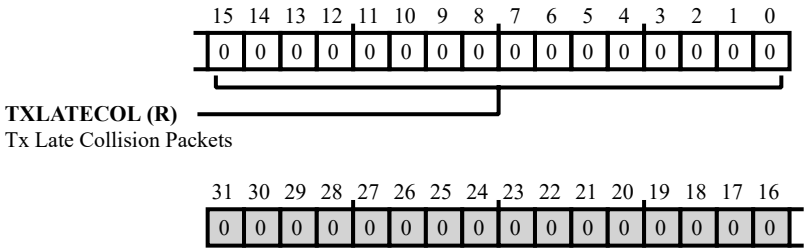


Figure 28-314: EMAC_TX_LTCOL_PCNT Register Diagram

Table 28-369: EMAC_TX_LTCOL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXLATECOL	Tx Late Collision Packets. The EMAC_TX_LTCOL_PCNT.TXLATECOL field indicates the number of packets aborted because of late collision error.

Transmit Multicast Packets Good Counter Register

The EMAC_TX_MCST_PCNT_G register provides the number of good multicast packets transmitted by the EMAC module.

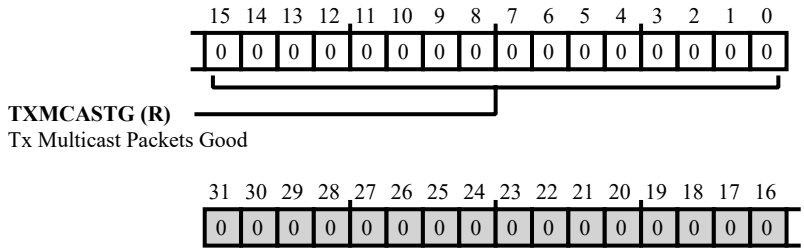


Figure 28-315: EMAC_TX_MCST_PCNT_G Register Diagram

Table 28-370: EMAC_TX_MCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXMCASTG	Tx Multicast Packets Good. The EMAC_TX_MCST_PCNT_G.TXMCASTG field indicates the number of good multicast packets transmitted.

Transmit Multicast Packets Good Bad Counter Register

The EMAC_TX_MCST_PCNT_GB register provides the number of good and bad multicast packets transmitted by the EMAC module.

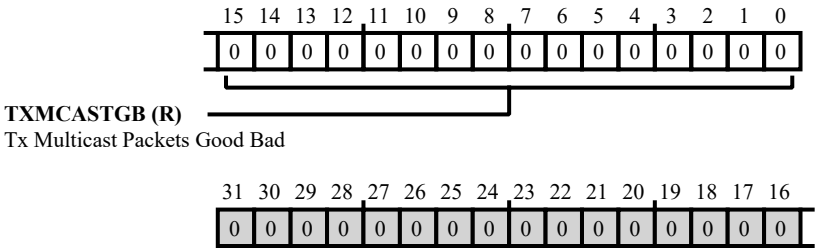


Figure 28-316: EMAC_TX_MCST_PCNT_GB Register Diagram

Table 28-371: EMAC_TX_MCST_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXMCASTGB	Tx Multicast Packets Good Bad. The EMAC_TX_MCST_PCNT_GB.TXMCASTGB field indicates the number of good and bad multicast packets transmitted.

Transmit Multiple Collision Packets Good Counter Register

The EMAC_TX_MCOL_PCNT_G register provides the number of successfully transmitted packets by the EMAC module after multiple collisions in the half-duplex mode.

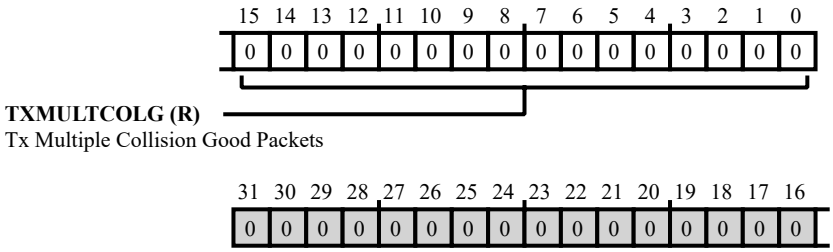


Figure 28-317: EMAC_TX_MCOL_PCNT_G Register Diagram

Table 28-372: EMAC_TX_MCOL_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXMULTCOLG	Tx Multiple Collision Good Packets. The EMAC_TX_MCOL_PCNT_G.TXMULTCOLG field indicates the number of successfully transmitted packets after multiple collisions in the half-duplex mode.

Transmit Osize Packets Good Counter Register

The EMAC_TX_OSIZE_PCNT_G register provides the number of packets transmitted by the EMAC module without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in EMAC_MAC_CFG.S2KP).

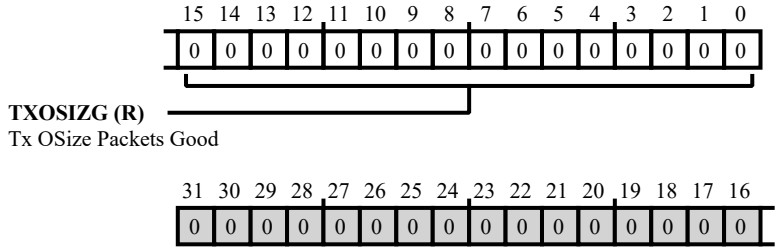


Figure 28-318: EMAC_TX_OSIZE_PCNT_G Register Diagram

Table 28-373: EMAC_TX_OSIZE_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXOSIZG	Tx OSize Packets Good. The EMAC_TX_OSIZE_PCNT_G.TXOSIZG field indicates the number of packets transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes when EMAC_MAC_CFG MAC.S2KP = 1).

Transmit Octet Good Counter Register

The EMAC_TX_OCNT_G register provides the number of bytes transmitted by the EMAC module, exclusive of preamble, only in good packets.

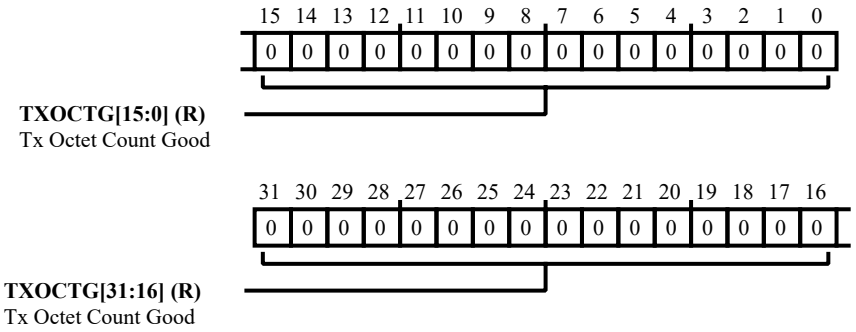


Figure 28-319: EMAC_TX_OCNT_G Register Diagram

Table 28-374: EMAC_TX_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXOCTG	Tx Octet Count Good. The EMAC_TX_OCNT_G.TXOCTG field indicates the number of bytes transmitted, exclusive of preamble, only in good packets.

Transmit Octet Good Bad Counter Register

The EMAC_TX_OCNT_GB register provides the number of bytes transmitted by the EMAC module, exclusive of preamble and retried bytes, in good and bad packets.

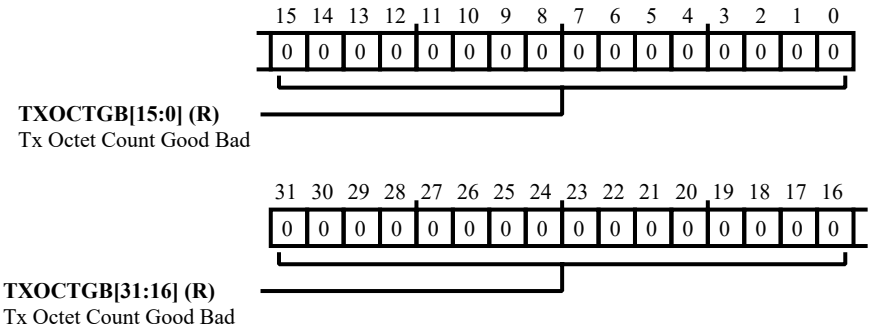


Figure 28-320: EMAC_TX_OCNT_GB Register Diagram

Table 28-375: EMAC_TX_OCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXOCTGB	Tx Octet Count Good Bad. The EMAC_TX_OCNT_GB.TXOCTGB field indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets.

Transmit Packet Good Counter Register

The EMAC_TX_PCNT_G register provides the number of good packets transmitted by the EMAC module.

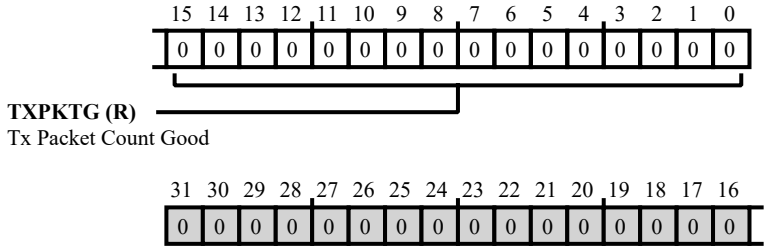


Figure 28-321: EMAC_TX_PCNT_G Register Diagram

Table 28-376: EMAC_TX_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXPKTG	Tx Packet Count Good. The EMAC_TX_PCNT_G.TXPKTG field indicates the number of good packets transmitted.

Transmit Packet Good Bad Counter Register

The EMAC_TX_PCNT_GB register provides the number of good and bad packets transmitted by the EMAC module, exclusive of retried packets.

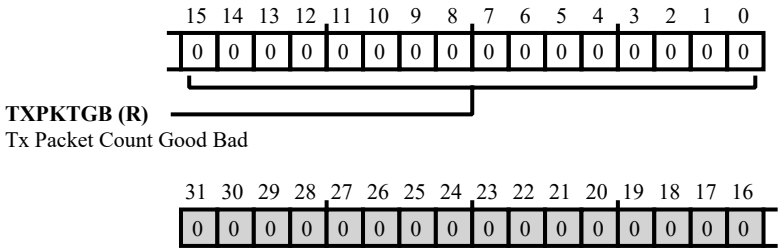


Figure 28-322: EMAC_TX_PCNT_GB Register Diagram

Table 28-377: EMAC_TX_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXPKTGB	Tx Packet Count Good Bad. The EMAC_TX_PCNT_GB.TXPKTGB field indicates the number of good and bad packets transmitted, exclusive of retried packets.

Transmit Pause Packets Counter Register

The EMAC_TX_PAUSE_PCNT register provides the number of good pause packets transmitted by the EMAC module.

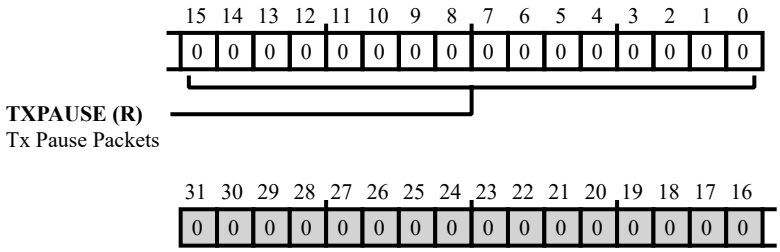


Figure 28-323: EMAC_TX_PAUSE_PCNT Register Diagram

Table 28-378: EMAC_TX_PAUSE_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXPAUSE	Tx Pause Packets. The EMAC_TX_PAUSE_PCNT.TXPAUSE field indicates the number of good Pause packets transmitted.

Transmit Single Collision Packets Good Counter Register

The EMAC_TX_SNGLCOL_PCNT_G register provides the number of successfully transmitted packets by the EMAC module after a single collision in the half-duplex mode.

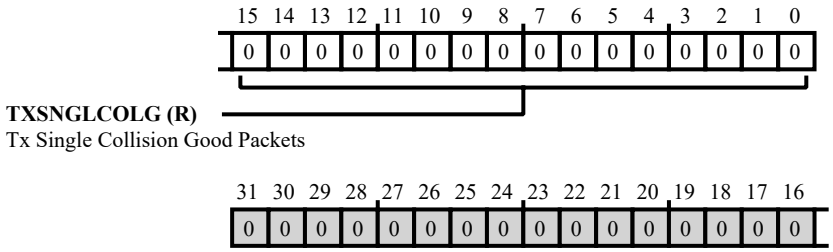


Figure 28-324: EMAC_TX_SNGLCOL_PCNT_G Register Diagram

Table 28-379: EMAC_TX_SNGLCOL_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXSNGLCOLG	Tx Single Collision Good Packets. The EMAC_TX_SNGLCOL_PCNT_G.TXSNGLCOLG field indicates the number of successfully transmitted packets after a single collision in the half-duplex mode.

Transmit Underflow Error Packets Counter Register

The EMAC_TX_URFL_PCNT register provides the number of packets aborted by the EMAC module because of packets underflow error.

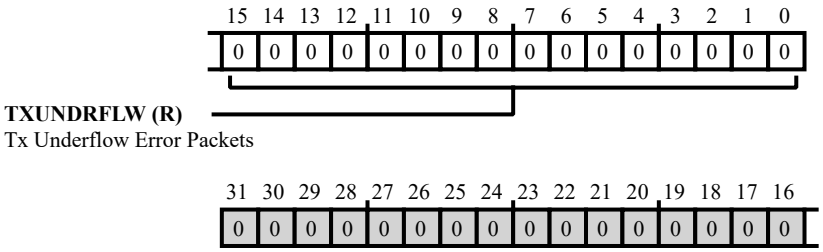


Figure 28-325: EMAC_TX_URFL_PCNT Register Diagram

Table 28-380: EMAC_TX_URFL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXUNDRFLW	Tx Underflow Error Packets. The EMAC_TX_URFL_PCNT.TXUNDRFLW field indicates the number of packets aborted because of packets underflow error.

Transmit Unicast Packets Good Bad Counter Register

The EMAC_TX_UCST_PCNT_GB register provides the number of good and bad unicast packets transmitted by the EMAC module.

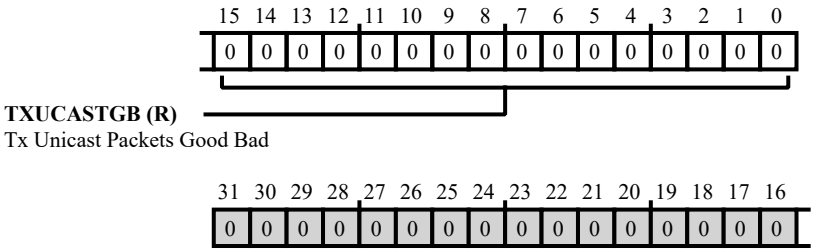


Figure 28-326: EMAC_TX_UCST_PCNT_GB Register Diagram

Table 28-381: EMAC_TX_UCST_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXUCASTGB	Tx Unicast Packets Good Bad. The EMAC_TX_UCST_PCNT_GB.TXUCASTGB field indicates the number of good and bad unicast packets transmitted.

Transmit VLAN Packets Good Counter Register

The EMAC_TX_VLAN_PCNT_G register provides the number of good VLAN packets transmitted by the EMAC module.

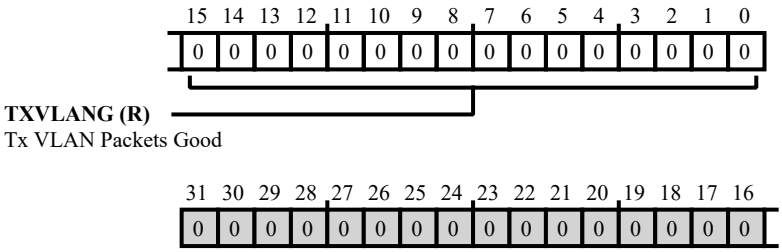


Figure 28-327: EMAC_TX_VLAN_PCNT_G Register Diagram

Table 28-382: EMAC_TX_VLAN_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXVLANG	Tx VLAN Packets Good. The EMAC_TX_VLAN_PCNT_G.TXVLANG field provides the number of good VLAN packets transmitted.

29 Digital Audio Interface (DAI)

The Digital Audio Interfaces (DAIn) are comprised of groups of identical peripherals and their respective Signal Routing Units (SRUn). The SRU connects inputs and outputs of the DAI peripherals with each other and to the external pins. This configuration allows peripherals to be interconnected to accommodate a wide variety of systems without making external pin connections.

In a typical processor, static (multiplexed) pins are assigned to specific peripherals. When certain peripherals are not required for an application, these pins are unnecessary and expensive. The pins may need to be defined as high or low to prevent any illegal conditions. The signal routing units on the SHARC processors addresses this situation by controlling a number of general-purpose pins which can be assigned flexibly (a virtual connectivity between peripherals) depending on system requirements. This virtual connectivity includes pin buffers and routing logic (multiplexer). It also allows the SHARC processors to include an arbitrary number and variety of peripherals while retaining high levels of compatibility without increasing pin count.

SRU Features

The SRU has the following features and capabilities.

- Flexible connections can be made through software and during run time; no hard wiring is required.
- At reset, a default routing scheme is already programmed.
- Connectivity can be made internally between peripherals, externally between pin buffers, or a mix of both.
- Status of the pin buffers can be programmed for conditional execution or interrupts.
- Some pin buffers allow control of signal polarity changes.
- No fan-out limitation, a peripheral or pin buffer output that can be routed to multiple peripheral or pin buffer inputs.
- Two independent routing systems are available the DAI0 and the DAI1. Signals cannot be interconnected between both routing units with the exception of the following modules under some specific scenarios.
    - The precision clock generator (PCG)
    - A pair of DAI pin buffers from each DAI unit
    - Asynchronous Sample Rate Converter (ASRC) data for daisy chaining across the DAI units

NOTE: Each of the DAIs has 20 pin buffers in the HPC package and 14 pin buffers in LPC package.

Functional Description

The fundamental timing clock of the DAI modules is SCLK.

The DAI Block Diagram shows how the DAI pin buffers are connected through the SRUn. This configuration allows for flexible signal routing.

The DAI units are comprised of four primary blocks:

- Peripherals (A/B/C) associated with DAIn
- Signal Routing Units (SRUn)
- DAIn I/O pin buffers
- Miscellaneous buffers

The peripherals shown in DAI Block Diagram can have up to three connections (if requester or completer capable); one acts as a signal input, one as a signal output and the third as an output enable. The SRUs are based on a group of multiplexers which are controlled by registers to establish the desired interconnections. The DAI pin buffers have three signals which are used for input and output to or from off-chip and the third for output enable.

The miscellaneous buffers have an input and an output and are used for group interconnection.

The figures are a simplified representation of a DAI system. In a real representation, the SRU and DAI would show several types of data being routed from several sources including the following:

- Serial ports (SPORT)
- Precision clock generators (PCG)
- Asynchronous sample rate converters (SRC)
- S/PDIF transmitter
- S/PDIF receiver
- DAI interrupts (miscellaneous)
- PDM interface

ADSP-2184x DAI Register List

The Digital Audio Interfaces (DAIn) contain groups of identical peripherals which can be connected internally between peripherals, externally between pin buffers, or a mix of both. This module contains the following registers.

Table 29-1: ADSP-2184x DAI Register List

Name	Description
DAI_CLK0	Clock Routing Control Register 0
DAI_CLK1	Clock Routing Control Register 1
DAI_CLK2	Clock Routing Control Register 2
DAI_CLK3	Clock Routing Control Register 3
DAI_CLK4	Clock Routing Control Register 4
DAI_CLK5	Clock Routing Control Register 5
DAI_CLK6	Clock Routing Control Register 6
DAI_CLK7	Clock Routing Control Register 7
DAI_DAI_MISC_CTL	DAI MISC Control Register
DAI_DAT0	Serial Data Routing Control Register 0
DAI_DAT1	Serial Data Routing Control Register 1
DAI_DAT2	Serial Data Routing Control Register 2
DAI_DAT3	Serial Data Routing Control Register 3
DAI_DAT4	Serial Data Routing Control Register 4
DAI_DAT5	Serial Data Routing Control Register 5
DAI_DAT6	Serial Data Routing Control Register 6
DAI_DAT7	Serial Data Routing Control Register 7
DAI_DAT8	Serial Data Routing Control Register 8
DAI_EXTD_CLK0	Extended Clock Routing Control Register 0
DAI_EXTD_CLK1	Extended Clock Routing Control Register 1
DAI_EXTD_CLK2	Extended Clock Routing Control Register 2
DAI_EXTD_CLK3	Extended Clock Routing Control Register 3
DAI_EXTD_CLK4	Extended Clock Routing Control Register 4
DAI_EXTD_CLK5	Extended Clock Routing Control Register 5
DAI_EXTD_CLK6	Extended Clock Routing Control Register 6
DAI_EXTD_CLK7	Extended Clock Routing Control Register 7
DAI_EXTD_DAT0	Extended Serial Data Routing Control Register 0
DAI_EXTD_DAT1	Extended Serial Data Routing Control Register 1
DAI_EXTD_DAT2	Extended Serial Data Routing Control Register 2
DAI_EXTD_DAT3	Extended Serial Data Routing Control Register 3
DAI_EXTD_DAT4	Extended Serial Data Routing Control Register 4

Table 29-1: ADSP-2184x DAI Register List (Continued)

Name	Description
DAI_EXTD_DAT5	Extended Serial Data Routing Control Register 5
DAI_EXTD_DAT6	Extended Serial Data Routing Control Register 6
DAI_EXTD_DAT7	Extended Serial Data Routing Control Register 7
DAI_EXTD_DAT8	Extended Serial Data Routing Control Register 8
DAI_EXTD_FS0	Extended Frame Sync Routing Control Register 0
DAI_EXTD_FS1	Extended Frame Sync Routing Control Register 1
DAI_EXTD_FS2	Extended Frame Sync Routing Control Register 2
DAI_EXTD_FS4	Extended Frame Sync Routing Control Register 4
DAI_EXTD_FS5	Extended Frame Sync Routing Control Register 5
DAI_EXTD_FS6	Extended Frame Sync Routing Control Register 6
DAI_EXTD_MISC0	Extended Miscellaneous Control Register 0
DAI_EXTD_MISC1	Extended Miscellaneous Control Register 1
DAI_EXTD_MISC2	Extended Miscellaneous Control Register 2
DAI_EXTD_PBEN0	Extended Pin Buffer Enable Register 0
DAI_EXTD_PBEN1	Extended Pin Buffer Enable Register 1
DAI_EXTD_PBEN2	Extended Pin Buffer Enable Register 2
DAI_EXTD_PBEN3	Extended Pin Buffer Enable Register 3
DAI_EXTD_PIN0	Extended Pin Buffer Assignment Register 0
DAI_EXTD_PIN1	Extended Pin Buffer Assignment Register 1
DAI_EXTD_PIN2	Extended Pin Buffer Assignment Register 2
DAI_EXTD_PIN3	Extended Pin Buffer Assignment Register 3
DAI_EXTD_PIN4	Extended Pin Buffer Assignment Register 4
DAI_FS0	Frame Sync Routing Control Register 0
DAI_FS1	Frame Sync Routing Control Register 1
DAI_FS2	Frame Sync Routing Control Register 2
DAI_FS4	Frame Sync Routing Control Register 4
DAI_FS5	Frame Sync Routing Control Register 5
DAI_FS6	Frame Sync Routing Control Register 6
DAI_GBL_INT_EN	Global SPORT Interrupt Grouping Register
DAI_GBL_PCG_EN	Global PCG Enable Control Register
DAI_GBL_SP_EN	Global SPORT Enable Register

Table 29-1: ADSP-2184x DAI Register List (Continued)

Name	Description
DAI_IMSK_FE	Falling-Edge Interrupt Mask Register
DAI_IMSK_PRI	Core Interrupt Priority Assignment Register
DAI_IMSK_RE	Rising-Edge Interrupt Mask Register
DAI_IRPTL_H	High Priority Interrupt Latch Register
DAI_IRPTL_HS	Shadow High Priority Interrupt Latch Register
DAI_IRPTL_L	Low Priority Interrupt Latch Register
DAI_IRPTL_LS	Shadow Low Priority Interrupt Latch Register
DAI_MISC0	Miscellaneous Control Register 0
DAI_MISC1	Miscellaneous Control Register 1
DAI_MISC2	Miscellaneous Control Register 1
DAI_PBEN0	Pin Buffer Enable Register 0
DAI_PBEN1	Pin Buffer Enable Register 1
DAI_PBEN2	Pin Buffer Enable Register 2
DAI_PBEN3	Pin Buffer Enable Register 3
DAI_PIN0	Pin Buffer Assignment Register 0
DAI_PIN1	Pin Buffer Assignment Register 1
DAI_PIN2	Pin Buffer Assignment Register 2
DAI_PIN3	Pin Buffer Assignment Register 3
DAI_PIN4	Pin Buffer Assignment Register 4
DAI_PIN5	Dummy Pin Buffer Assignment Register
DAI_PIN_STAT	Pin Status Register

ADSP-2184x DAI Interrupt List

Table 29-2: ADSP-2184x DAI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
64	DAI0_IRQH	DAI0 High Priority Interrupt	None	
65	DAI1_IRQH	DAI1 High Priority Interrupt	None	
66	DAI0_IRQL	DAI0 Low Priority Interrupt	None	
67	DAI1_IRQL	DAI1 Low Priority Interrupt	None	
286	DAI0_GBL_SPORT_INT0	DAI0 Global SPORT Interrupt 0	None	

Table 29-2: ADSP-2184x DAI Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
287	DAI0_GBL_SPORT_INT1	DAI0 Global SPORT Interrupt 1	None	
288	DAI1_GBL_SPORT_INT0	DAI1 Global SPORT Interrupt 0	None	
289	DAI1_GBL_SPORT_INT1	DAI1 Global SPORT Interrupt 1	None	

ADSP-2184x DAI Trigger List

Table 29-3: ADSP-2184x DAI Trigger List Generators

Trigger ID	Name	Description	Sensitivity
141	DAI0_GBL_SPORT_TRG_O0	DAI0 SPORT GROUP0 Trigger Output	None
142	DAI0_GBL_SPORT_TRG_O1	DAI0 SPORT GROUP1 Trigger Output	None
143	DAI1_GBL_SPORT_TRG_O0	DAI1 SPORT GROUP2 Trigger Output	None
144	DAI1_GBL_SPORT_TRG_O1	DAI1 SPORT Group3 Trigger Output	None

Table 29-4: ADSP-2184x DAI Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
173	DAI0_GBL_SPORT_TRG_I0	DAI0 SPORT GROUP0 Trigger Input	Pulse
174	DAI0_GBL_SPORT_TRG_I1	DAI0 SPORT GROUP1 Trigger Input	Pulse
175	DAI1_GBL_SPORT_TRG_I0	DAI1 SPORT GROUP2 Trigger Input	Pulse
176	DAI1_GBL_SPORT_TRG_I1	DAI1 SPORT GROUP3 Trigger Input	Pulse

DAI Block Diagram

The ***DAI Functional Block Diagram*** and the ***Digital Audio Interconnect Unit*** figures show the functional blocks within the DAI and the unit connections to the peripherals.

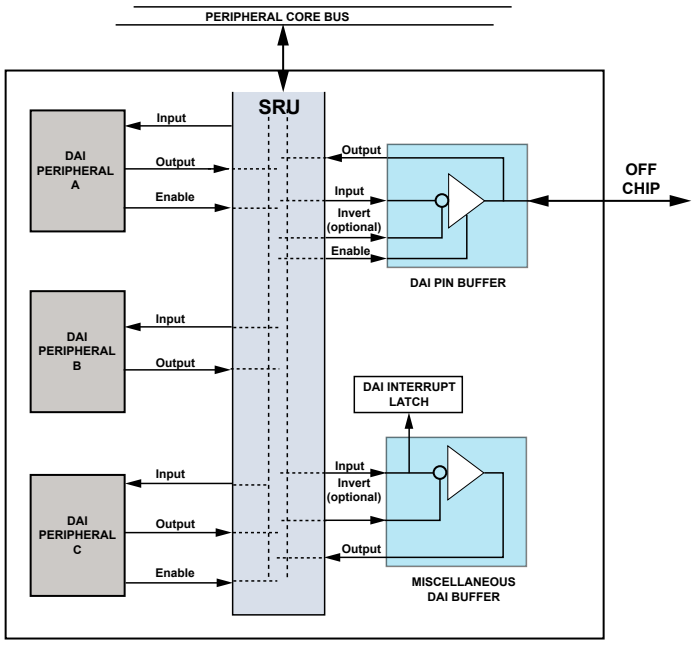


Figure 29-1: DAI Functional Block Diagram

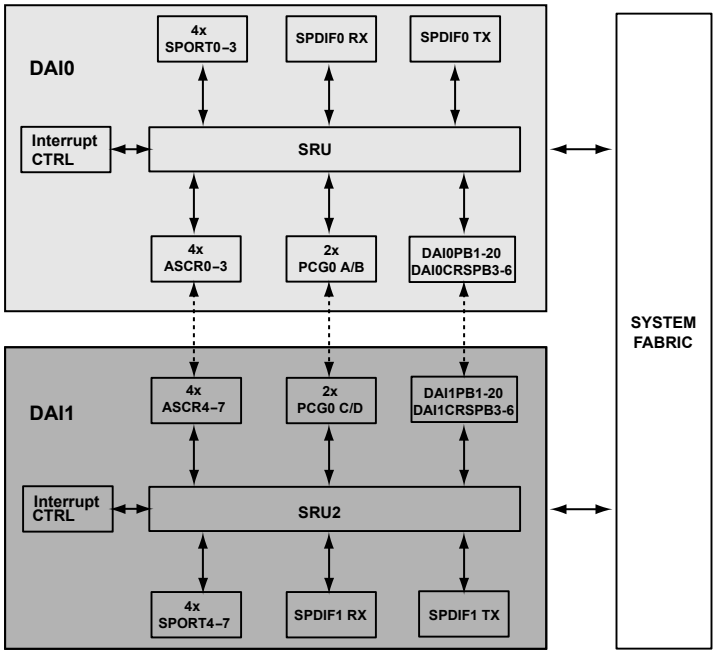


Figure 29-2: Digital Audio Interconnect Unit

DAI Signal Naming Conventions

The peripherals associated with the DAI do not have any dedicated I/O pins for off-chip communication. Instead, the I/O pin is only accessible in the chip internally and is known as an *internal node*. Every internal node of a DAI peripheral (input or output) is given a unique mnemonic. The convention is to begin the name with an identifier for the peripheral that the signal is coming to or from, followed by the function of the signal.

A number is included if the DAI contains more than one peripheral type (for example, serial ports), or if the peripheral has more than one signal that performs this function (for example, serial ports). The mnemonic always ends with *_I* if the signal is an input, or with *_O* if the signal is an output. An example is shown in the ***Example DAI Mnemonics*** figure.

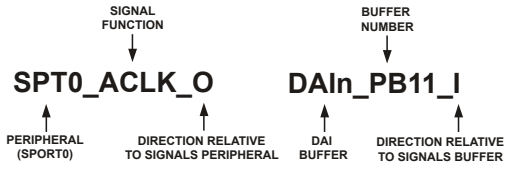


Figure 29-3: Example DAI Mnemonics

I/O Pin Buffers

Within the context of the SRU, physical connections to the DAI pins are replaced by a logical interface known as a *pin buffer*. This three terminal active device is capable of sourcing or sinking output current when its driver is enabled, and passing external input signals when disabled. Each pin has an input, an output, and an enable as shown in the ***Pin Buffer Example*** figure. The inputs and the outputs are defined with respect to the pin, like a peripheral device. This naming convention is consistent with the SRU naming convention.

Pin Buffer Signals

The pin buffer is based on three signals shown in the ***Pin Buffer Example*** figure and described in the following sections.

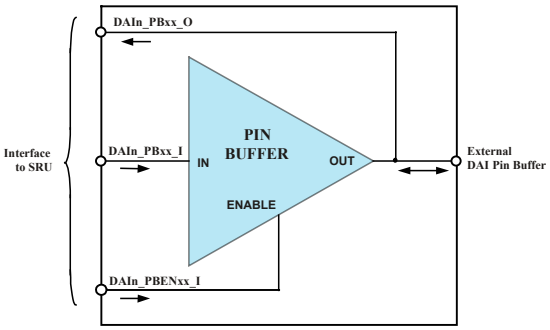


Figure 29-4: Pin Buffer Example

Pin Buffer Input Signal

A pin buffer input (`DAIn_PBxx_I`) is driven as an output from the processor when the pin buffer enable is set (=1). Each physical pin (connected to a bonded pad) can be connected through the SRU to any of the outputs of the DAI peripherals, based on the bit field values. The SRU can also be used to route signals that control the pins in other ways. Many signals can be configured for use as control signals.

Pin Buffer Enable Signal

When a pin buffer enable (`DAIn_PBENxx_I`) is set (=1), the signal present at the corresponding pin buffer input (`DAIn_PBxx_I`) is driven off-chip as an output. When a pin buffer enable is cleared (=0), the signal present at the corresponding pin buffer input is ignored. The pin enable control registers activate the drive buffer for each of the

DAI pins. When the pins are not enabled (driven), they can be used as inputs. There are two options to control the pin buffer enable signal; setting the level high for a static solution or connecting the dedicated peripheral's pin buffer output enable signal to its pin buffer, which automatically enables the pin buffer.

Pin Buffer Functions

Pin buffers can be configured as inputs or outputs as described in the following sections.

Pin Buffers as Signal Input

When the DAI pin is used only as an input, connect the corresponding pin buffer enable to logic low as shown in the ***Pin Buffer as Input*** figure. This configuration disables the buffer amplifier and allows an off-chip source to drive the value present on the DAI pin and at the pin buffer output. When the pin buffer enable (for example, `DAI_PBEN0.PB01`) is cleared (= 0), the pin buffer output (`DAIn_PBxx_O`) is the signal driven onto the DAI pin by an external source, and the pin buffer input signal (`DAIn_PBxx_I`) is not used.

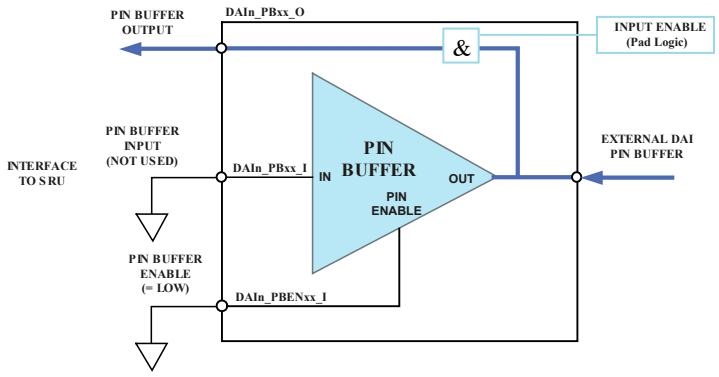


Figure 29-5: Pin Buffer as Input

NOTE: Whether programmed as input or output, a DAI buffer input always routes the same signal to an output internally. DAI pins have programmable internal pull-up resistors. For more information, see the General-Purpose Ports (PORT) chapter for details.

Pin Buffers As Signal Output

In a typical embedded system, most pins are designated as either inputs or outputs when the circuit is designed, even though they can be used in either direction. Each of the DAI pins can be used as either an output or an input. Although the direction of a DAI pin is set simply by writing to a memory-mapped register, most often the direction of the pin is dictated by the designated use of that pin.

When the DAI pin is used only as an output, connect the corresponding pin buffer enable to logic high as shown in the ***Pin Buffer as Output*** figure. This configuration enables the buffer amplifier to operate as a current source and to drive the value present at the pin buffer input onto the DAI pin and off-chip. When the pin buffer enable bits are set (in the `DAIn_PBxx_I` registers) (=1), the pin buffer output (`DAIn_PBxx_O`) is the same signal as the pin buffer input (`DAIn_PBxx_I`), and this signal is driven as an output.

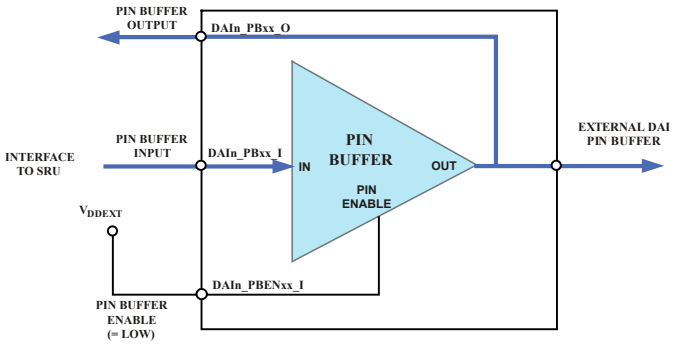


Figure 29-6: Pin Buffer as Output

DAI Pin Buffer Status

The signal levels on the DAI pins can be read with the `DAI_PIN_STAT` registers.

DAIn Peripherals

There are two categories of peripherals associated with the DAI units. These are described in the following sections.

Output Signals With Pin Buffer Enable Control

Many peripherals within the DAIn that have bidirectional pins generate a corresponding pin enable signal. Typically, the settings within the control registers of a peripheral determine if a bidirectional pin is an input or an output. The pin is then driven accordingly.

Though most peripherals can operate bidirectionally, it is not required that all of the `_I` and `_O` signals of a peripheral be connected to the pin buffer. When the system design only uses a signal in one direction, it is simpler to connect the pin buffer accordingly.

NOTE: All available pin buffer output enables must be routed to their pin buffer input enable signals in cases where data streaming connections are used. This arrangement guarantees timing requirements.

NOTE: In some cases, it is necessary to use a peripheral's dedicated pin buffer enable signal instead of static levels. For example, SPORT TDM mode requires the SPORT's dedicated data pin buffer enable signal to be used for the SPORT's data pin to three-state the data pins on inactive channels.

Output Signals Without Pin Buffer Enable Control

Some peripherals have signal outputs without an automated pin buffer control enable signal. The operation of these peripherals is simplified. The routing to a DAIn pin buffer enable input requires a static high from the SRUn. To disable the pin buffer output, software must clear the pin buffer enable input.

Signal Routing Units (SRUs)

The following sections provide details specific to the SRUs.

Signal Routing Matrix by Groups

The SRU is similar to a set of patch bays, which contain a bank of inputs and a bank of outputs. For each input (destination), there is a set of permissible output (source) options. Outputs can feed to any number of inputs in parallel, but every input must be patched to exactly one valid output source. Together, the set of inputs and outputs are called a group. The signal's inputs and outputs that comprise each group all serve similar purposes. They are compatible such that almost any output-to-input patch makes functional sense. With the grouping, the multiplexing scheme becomes highly efficient since it does not make sense (for example) to route a frame sync signal to a data signal.

The SRU for the DAI contains six groups named A through F. Each group routes a unique set of signals with a specific purpose:

- Group A routes clock signals
- Group B routes serial data signals
- Group C routes frame sync signals
- Group D routes pin signals
- Group E routes miscellaneous signals
- Group F routes pin output enable signals

Together, the six groups of the SRU include all of the inputs and outputs of the DAI peripherals, a number of additional signals from the core, and all of the connections to the DAI pins.

NOTE: It is not possible to connect a signal in one group directly to a signal in a different group (analogous to wiring from one patch bay to another). However, group D (DAI) is largely devoted to routing in this vein.

DAI Group Routing

Each group has a unique encoding for its associated output signals and a set of configuration registers. For example, DAI group A is used to route clock signals. The memory-mapped group A registers, `DAI_CLK0` through `DAI_CLK5`, contain bit fields corresponding to the clock inputs of various peripherals. The values written to these bit fields specify a signal source that is an output from another peripheral. All of the possible encodings represent sources that are clock signals (or at least could be clock signals in some systems). The ***Example DAI Group A Multiplexing (DAI_CLKx)*** diagrams the input signals that are controlled by the group A registers. All bit fields in the SRU configuration registers correspond to inputs. The value written to the bit field specifies the signal source. This value is also an output from some other component within the SRU.

The SRU is similar to a set of patch bays. Each bay routes a distinct set of outputs to compatible inputs. These connections are implemented as a set of memory-mapped registers with a bit field for each input. The outputs are implemented as a set of bit encodings. Conceptually, a patch cord is used to connect an output to an input. In the SRU, a bit pattern that is associated with a signal output (shown in the ***Example DAI Group A Multiplexing (DAI_CLKx)*** figure) is written to a bit field corresponding to a signal input.

The same encoding can be written to any number of bit fields in the same group. It is not possible to run out of patch points for an output signal.

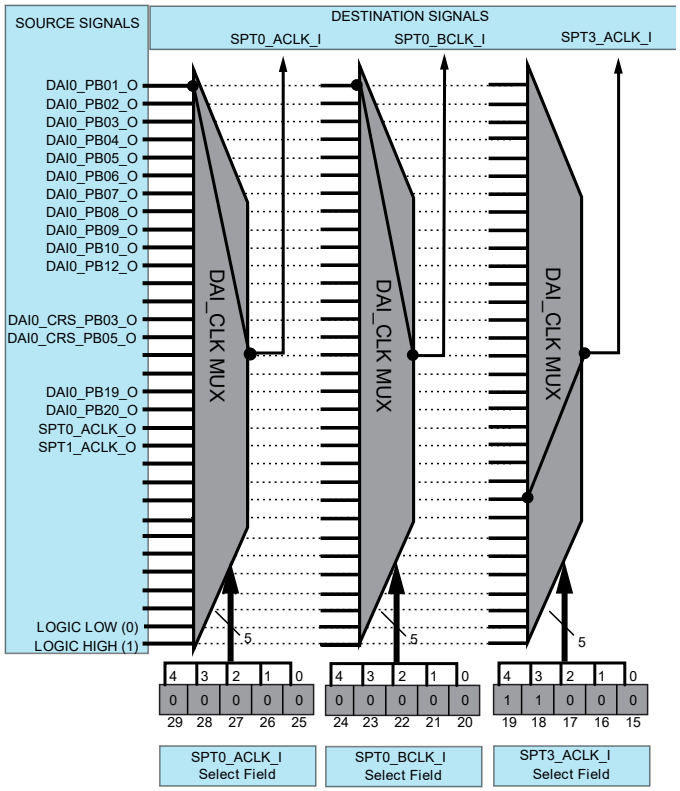


Figure 29-7: Example DAI Group A Multiplexing (DAIn_CLKx)

Just as group A routes clock signals, each of the other groups route a collection of compatible signals. Group B routes serial data streams while group C routes frame sync signals. All of the groups have an encoding that allow a signal to flow from a pin output to the input being specified by the bit field.

Group D routes signals to pins so that they may be driven off-chip (required to route a signal to the pin input). Group F routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input. The input of one pin can be patched to the output of another pin, allowing board-level routing under software control.

Rules for SRU Connections

There are two rules which apply to all routing:

1. One source (output node) can drive different destinations (input nodes).
2. One destination (input node) can only be assigned to one source (output node).

As an example from the ***Example DAI Group A Multiplexing (DAI_CLKx)*** figure:

- `DAI0_PB02_O` is routed to `SPT0_ACLK_I`
- `DAI0_PIN02_O` is routed to `SPT0_BCLK_I`

- SPT2_ACLK_O is routed to SPT3_ACLK_I

NOTE: Inputs may only be connected to outputs.

Miscellaneous Buffers and Functions

The SRU group E provides miscellaneous buffers used for group interconnect.

DAI group E connections are slightly different from the others in that the inputs and outputs being routed vary considerably in function. This group routes control signals and provides a means of connecting signals between groups.

In the ***DAI MISCAx SRU Signal Connections*** table, the DAIn_MISCAx_I signals appear as inputs in group E (also connected to the DAI interrupt logic), but do not directly feed any peripheral. Rather, the MISCAx_O signals reappear as outputs in group F.

Table 29-5: DAI MISCAx SRU Signal Connections

MISCA Source	DAI Connection	MISCA Destination
	Group E	DAIn_MISCA5-0_I
DAIn_MISCA5-0_O	Group F	

Additional connections among groups provide a great amount of utility. Since the output groups F (DAI) dictate pin direction, these few signal paths enable a number of possible uses and connections for the DAI pins. Other examples include:

- A pin input can be patched to another pin's enable, allowing an off-chip signal to gate an output from the processor.
- Any of the DAI pins can be used as interrupt sources or general-purpose I/O (GPIO) signals.

In summary, the SRU enables many possible functional changes, both within the processor as well as externally. Used creatively, it allows system designers to radically change functionality at run time, and to potentially reuse circuit boards across many products.

DAI Routing Unit (DRU)

The DAI routing unit provides all possible connections across DAIs by interconnecting two DAI units through a multiplexer.

- For every input signal of an SRU, there is a control bit that selects either the original signal from the native DAI or its corresponding signal from the other DAI. For example, either SP0_ACLK_O from DAI0 or SP0_ACLK_O from DAI1 is selected and routed to the SP0_ACLK_O input of the SRU of DAI0 based on the programmed control bit.
- By default, all native signals are connected to their respective SRU (for example, SP0_ACLK_O from DAI0 is connected to SP0_ACLK_O input of the SRU of DAI0.

As a part of fully cross-connected DAI Routing unit, following source signals are added to the existing SRU's source group.

- All the source signals of Group A/B/C/D/E/F from the other DAI are added as source signals to the native group A/B/C/D/E/F's respectively.

The *DAI Routing Unit System* figure depicts the block level view of the DAI Routing Unit connecting DAI0 and DAI1 through a multiplexer.

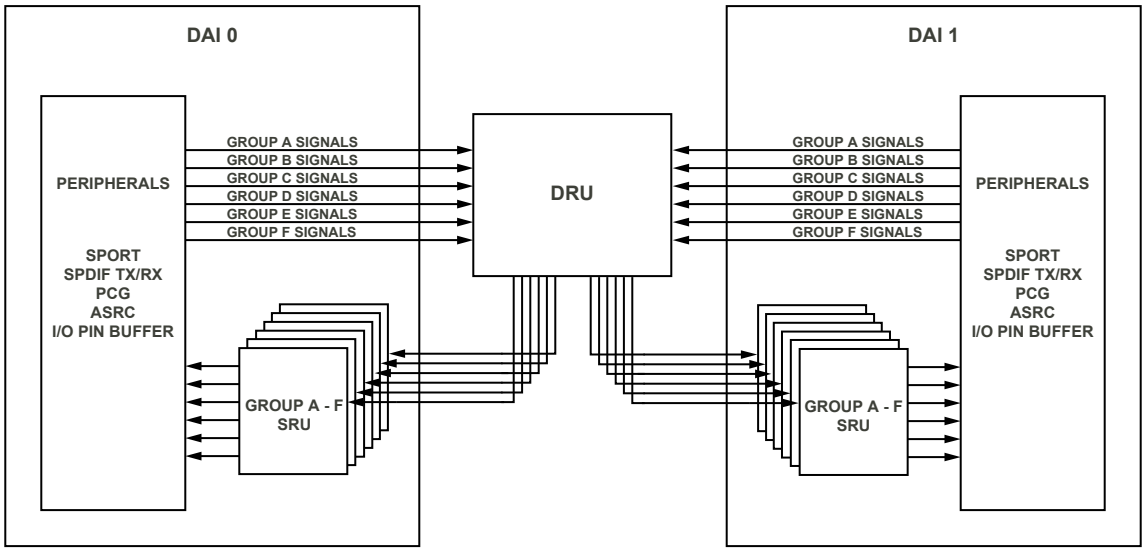


Figure 29-8: DAI Routing Unit System

DAI Routing Capabilities

This section describes the routing options to aid in designing a system using the DAI units. The DAI Default Routing section provides diagrams that show how that DAIs connect at default. The DAI Group tables provide the source signals and selections codes. Finally, the xxx provides information about configuring destinations.

The ***DAI Routing Capabilities*** tables provide an overview of the different routing capabilities for the DAI unit. For information on an individual peripherals routing, see the "SRU Programming" section of the specific peripheral chapter.

Table 29-6: DAI0 Routing Capabilities

Source Signals - Output (xxxx_O)		DAI0 Group	Destination Signals - Input (xxxx_I)
SPORT2B–0A (clocks) PCG A, B S/PDIF0 Rx (clock, TDM clock) SPORT3B-3A, PCG E, F PCG A, B, E, F (Inverted) PDM0 All Group A source signals from other DAI	DAI Pin Buffer20–1 Logic level high Logic level low	A – clocks	SPORT3B–0A (clocks) SRC3–0 (clocks) PCG A–B ext. clock, ext. sync S/PDIF0 Tx (clock, HF clock) PCG E, F PDM0 (clk, bclk)
SPORT3B–0A (data) SRC3–0 (data, TDM data) SRC8-11 (data, TDM data) S/PDIF0 Tx/Rx (data) PDM0 All Group B source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	B – data	SPORT3B–0A (data) SRC3–0 (data, TDM data) SRC8-11 (data, TDM data) S/PDIF0 Tx/Rx (data) PDM0 (data0/1)
SPORT2B–0A (FS) PCG A, B (FS) S/PDIF0 Rx (FS) SPORT3A-3B, PCG E, F PCG A, B, E, F (Inverted) All Group C source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	C – frame sync	SPORT3B–0A (FS) SRC3–0 (FS) PDM0 (lrclk) TMR_ACI14
SPORT3B–0A (clock, FS, TDV, data) PCG A,B (clock, FS) S/PDIF0 Rx (clock, TDM clock, FS, data) S/PDIF0 Tx (data, block start) SRC0-3 (data) SRC4-7 (data) PCG E, F (clock, FS) PCG A, B, E, F (inverted-Clock, FS) PDM0 (clock, FS) All Group D source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	D – pin buffer inputs	DAI0 pin buffer 20–1 Options: DAI0 pin buffer 20–19 Polarity change

Table 29-6: DAI0 Routing Capabilities (Continued)

Source Signals - Output (xxxx_O)		DAI0 Group	Destination Signals - Input (xxxx_I)
SPORT2B–0A (FS) PCG A (clock) PCG B (clock, FS) S/PDIF0 Tx (block start) SPORT3B-3A (FS) PCG E, F (FS) PCG A, B, E, F (trigger receiver) All Group E source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	E – miscellaneous signals	***DAI Interrupt 31-22*** MISCA5–0 Options: MISCA5–4 Polarity change PCG A, B, E, F (trigger generator)
SPORT3B–0A (clock OE, FS OE, data OE, TDV) MISCA5–0 PDM0 (clock OE, FS OE) All Group F source signals from other DAI	Logic level high Logic level low	F – pin buffer enable	DAI Pin Buffer Enable 20-1

Table 29-7: DAI1 Routing Capabilities

Source Signals - Output (xxxx_O)		DAI1 Group	Destination Signals - Input (xxxx_I)
SPORT6B–4A (clocks) PCG C, D (clocks) S/PDIF1 Rx (clock, TDM clock) SPORT7B-7A, PCG G, H PCG C, D, G, H (Inverted) PDM1 All Group A source signals from other DA	DAI1 Pin Buffer 20-1 Logic level high Logic level low	A – clocks	SPORT7B–4A (clocks) SRC4–7 (clocks) PCG C–D Ext. clock, Ext. sync S/PDIF1 Tx (clock, HF clock) PCG G, H PDM1 (clk, bclk)
SPORT7B–4A (data A, B) SRC4–7 (data, TDM data) SRC12-15 (data, TDM da- ta) S/PDIF1 Tx/Rx (data) PDM1 All Group B source signals from other DA	DAI1 Pin Buffer 20–1 Logic level high Logic level low	B – data	SPORT7B–4A (data) SRC4–7 (data, TDM data) SRC12-15 (data, TDM da- ta) S/PDIF1 Tx/Rx (data) PDM1 (data0/1)

Table 29-7: DAI1 Routing Capabilities (Continued)

Source Signals - Output (xxxx_O)		DAI1 Group	Destination Signals - Input (xxxx_I)
SPORT6B–4A (FS) PCG C, D (FS) S/PDIF1 Rx (FS) SPORT7B-7A, PCG G, H PCG C, D, G, H (Inverted) All Group C source signals from other DA	DAI1 Pin Buffer 20–1 Logic level high Logic level low	C – frame sync	SPORT7B–4A (FS) SRC4–7 (FS) PDM1 (lrclk) TMR_ACI15
SPORT7B–4A (clock, FS, TDV, data) PCG C, D (clock, FS) S/PDIF1 Rx (clock, TDM clock, FS, data) S/PDIF1 Tx (data, block start) SRC11-15 (data) PCG G, H (clock, FS) PCG C, D, G, H (inverted-Clock, FS) PDM1 (clock, FS) All Group D source signals from other DAI	DAI1 Pin Buffer 20-1 Logic level high Logic level low	D – pin buffer inputs	DAI1 Pin Buffer 20–1 Options: DAI1 Pin Buffer 20–19 Polarity Change
SPORT6B–4A (FS) PCG C (clock) PCG D (clock, FS) S/PDIF1 Tx (block start) SPORT7B-7A (FS) PCG C, D (FS) PCG C, D, G, H (trigger receiver) All Group E source signals from other DAI	DAI1 Pin Buffer 20–1 Logic level high Logic level low	E – miscellaneous signals	***DAI Interrupt 31-22*** MISCA5–0 Options: MISCA5–4 Polarity Change PCG C, D, G, H (trigger generator)
SPORT7B–4A (clock, FS, data, TDV) PDM1 (clock OE, FS OE) All Group F source signals from other DAI.	Logic level high Logic level low	F – pin buffer enable	DAI1 Pin Buffer Enable 20–1

DAI Default Routing

When the processor comes out of reset, the SPORT junctions are bidirectional to the DAI pin buffers. This configuration allows systems to use the SPORTs as either requester or completer (without changing the routing scheme). Therefore, programs only need to use the SPORT control register settings to configure requester or completer operations. Note that all DAI inputs which are not routed by default are tied to signal low.

NOTE: All DAI input buffers which are not routed by default are driven low and all DAI pin buffer enable signals are driven low.

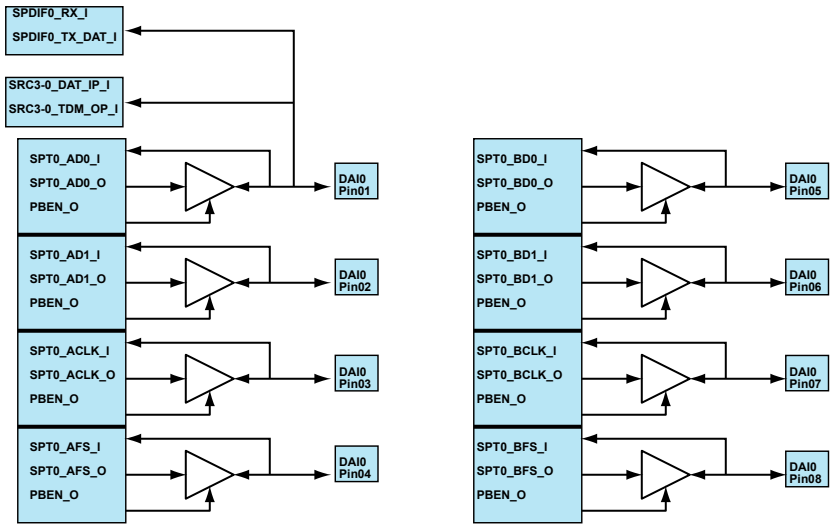


Figure 29-9: DAI0 Default Routing Pins 01-08

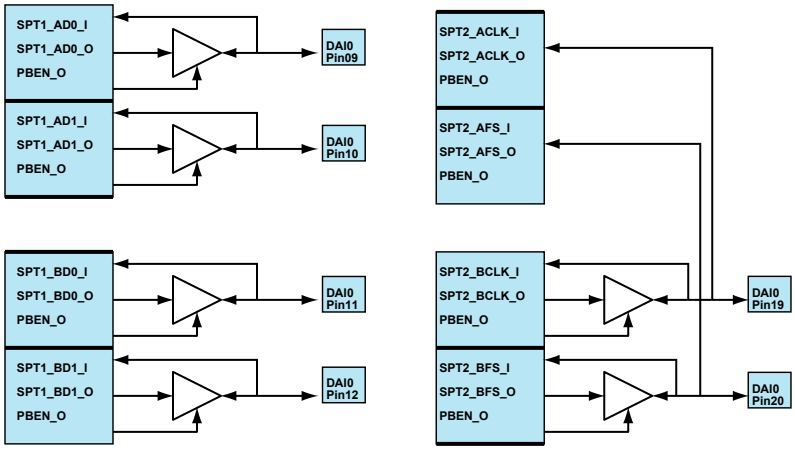


Figure 29-10: DAI0 Default Routing Pins 09-20

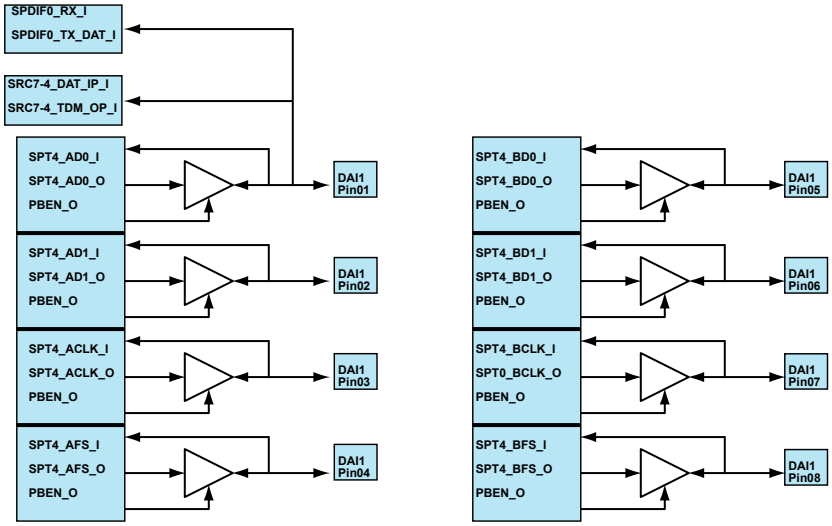


Figure 29-11: DAI1 Default Routing Pins 01-08

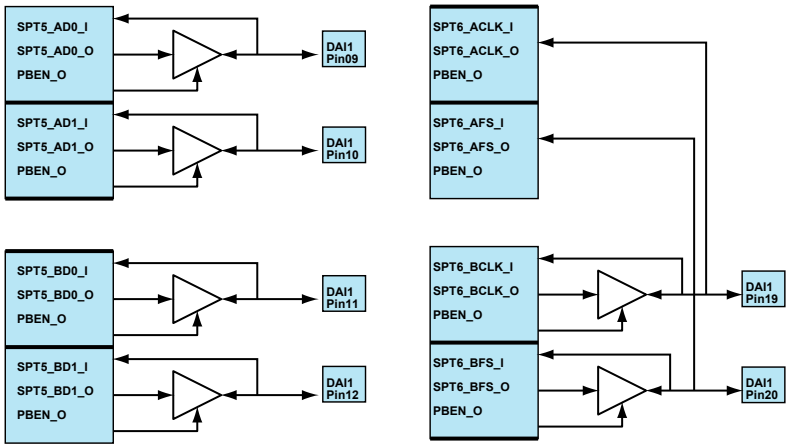


Figure 29-12: DAI1 Default Routing Pins 09-20

Unused DAI Connections

The SRUs have a default general-purpose routing scheme which can be modified to accommodate any number of different system designs. Regardless of the system design, it is good practice to tie all unused inputs to a high or low level to reduce dynamic power consumption.

DAI Operating Modes

Some buffers allow polarity changes, described as follows.

DAI Pin Buffer Polarity

As shown in the ***Pin Buffer Polarity*** figure, the DAI pin buffer 20-19 can change the polarity of the input signal if the corresponding control bits (`DAI_PIN4.INV20`, `DAI_PIN4.INV19`) are set. These bits can be set during run time and the buffer should not loopback to itself.

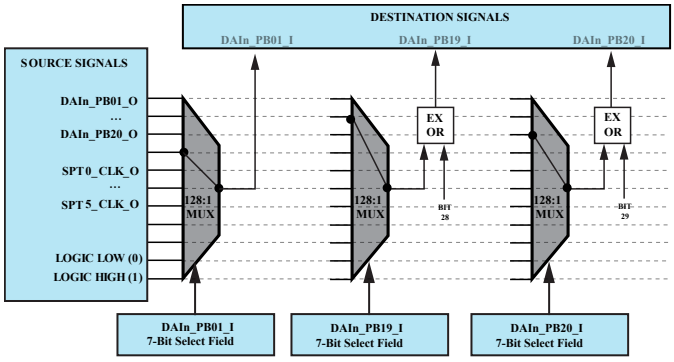


Figure 29-13: Pin Buffer Polarity

DAI Miscellaneous Buffer Polarity

As shown in the ***Pin Buffer Polarity*** figure, the A5-4 miscellaneous buffers can change the polarity of the input signal if the corresponding control bits (`DAI_MISC1.IN5`, `DAI_MISC1.IN4`) are set. Both buffers are not connected to the DAI interrupt latch register. These bits can be set during run time.

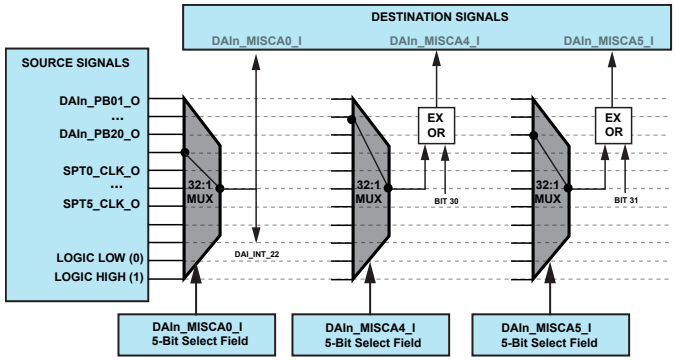


Figure 29-14: Miscellaneous Buffer Polarity

DAI System Interrupt Controller (SIC)

The DAI module incorporates a system interrupt controller (SIC) which is connected to the SEC as seen in the ***DAI System Interrupt Controller*** figure.

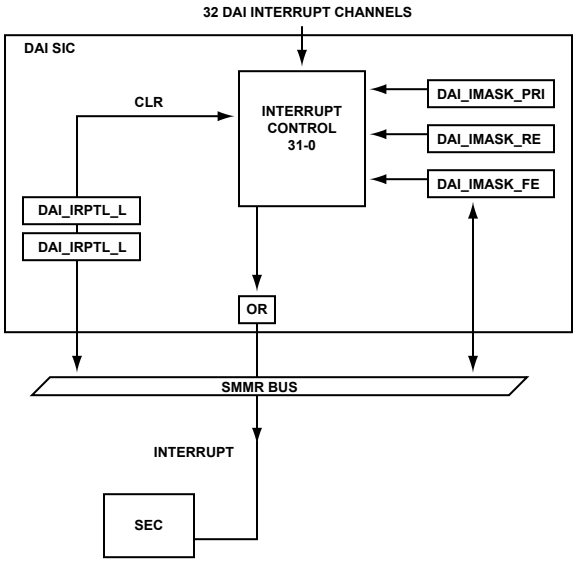


Figure 29-15: DAI System Interrupt Controller

The DAI has its own system interrupt controllers that indicate to the core when DAI audio peripheral-related events have occurred. Since audio events occur infrequently relative to the SHARC core, the DAI interrupt controller reduces all of its interrupts onto two interrupt signals within the core's primary interrupt systems. One interrupt is mapped with DAI low priority. The second interrupt is mapped with DAI high priority. This configuration allows programs to broadly indicate priority. In this way, the DAI SIC provides 32 independently configurable sources or channels. The output bus interrupt signals are logically OR'ed into one interrupt line and fed to the interrupt controller logic of the core.

Three registers are used to configure the DAI interrupt controller. Each of the 32 interrupt sources can be independently configured to trigger on a rising edge, falling edge, both edges, or neither edge of an incoming signal. All DAI interrupt control registers are memory-mapped registers and are accessed through the peripheral bus.

Interrupt Sources

The DAI's five peripheral sources are multiplexed into 32 interrupt sources and are labeled DAI_INT31-0 (***DAI Interrupt Sources*** table).

NOTE: There are two naming conventions. The DAI interrupt controller register bits are labeled DAI_31-0_INT. Their corresponding SRU routing signals are labeled DAI_INT_31-0_I.

Table 29-8: DAI Interrupt Sources

Interrupt Source	Description	Signal Response
DAI_INT2–0, DAI_INT4	S/PDIF RX, 4 channels	Event
DAI_INT2–0, DAI_INT4	S/PDIF RX, 4 channels	Waveform
DAI_INT21–18	ASRC, 4 channels	
DAI_INT31–22	Miscellaneous, S/PDIF TX, 9 channels	

Interrupt Latch Priority Option

The DAI_IMSK_PRI register specifies the priority for the DAI interrupt channels. DAI system interrupt controller has a pair of interrupt latch registers, DAI_IRPTL_H and DAI_IRPTL_L. The configuration of the DAI_IMSK_PRI register also determines the interrupt latch mapping for a particular DAI interrupt. When a DAI interrupt is configured as low priority (DAI_IMSK_PRI bit cleared, default setting), interrupts are mapped to the DAI_INTR_IRQL signal and when an interrupt occurs, the corresponding bit of the DAI_IRPTL_L register is set. When a DAI interrupt is configured as high priority (DAI_IMSK_PRI bit set), interrupts are mapped to the DAI_INTR_IRQH signal and the interrupt is latched to the DAI_IRPTL_H register. The low priority DAI interrupt (INTR_DAI_IRQL) and high priority DAI interrupt (INTR_DAI_IRQH) are connected to the SEC .

Interrupt Mask for Waveforms

The DAI_IMSK_RE and DAI_IMSK_FE registers allow programs to mask or unmask interrupts for specific edges of a signal mapped to the channel. It can be configured for rising edges, falling edges, both rising and falling edges, or neither rising nor falling edges by masking them separately. Signals from the SRU can be used to generate interrupts. For example, when the DAI_IMSK_FE.MISCINT9 bit is set to one, any falling edge signals from the external channel generate an interrupt and the interrupt latch is set.

Interrupt Mask for Events

The system interrupt controller needs information about a peripheral's interrupt sources that correspond to event signals (see the ***DAI Interrupt Sources*** table). As a result, the rising edge is used as an interrupt source only. For DAI peripherals marked as events, programs may unmask an interrupt source on the rising edge only.

Shadow Interrupt Register

The DAI interrupt controller has shadow registers to simplify debug activities since these registers do not manipulate status control. Any read of the shadow registers (DAI_IRPTL_HS, DAI_IRPTL_LS), provides the same data as a read of the DAI_IRPTL_H and DAI_IRPTL_L registers. However, reads of the DAI shadow registers do not change the interrupt acknowledge status to the core interrupt controller.

Interrupt Service

The interrupt acknowledge operates differently when multiple channels are multiplexed into one interrupt output signal. When an interrupt from the DAI must be serviced, any of the two interrupt service routines (INTR_DAI_IRQL, INTR_DAI_IRQH) must query the SIC to determine the source(s). Sources can be any one or more of the DAI channels (DAI_INT31-0).

- When the DAI_IRPTL_H register is read, the high priority latched interrupts are cleared.
- When the DAI_IRPTL_L register is read, the low priority latched interrupts are cleared.

If an interrupt occurs in the same cycle as a latch register is cleared, the clear mechanism has lower priority and the new interrupt is registered.

Signal Routing Unit Effect Latency

After the DAIy registers are configured the effect latency is 2 *SCLK0* cycles minimum and 3 *SCLK0* cycles maximum.

DAI Programming Model

As discussed in the previous sections, the signal routing unit is controlled by writing values that provide a plug-in tool in CCES so that configuring the SRU is done graphically. Analog Devices offers macros that are included with the CrossCore or VisualDSP++ tools, greatly easing code development in the SRU.

There is a macro that has been created to connect peripherals used in a DAI configuration. This code can be used in both assembly and C code. See the INCLUDE file SRU.H. In practice the macro is provided and forms the style `SRU(source_O, destination_I)` for DAI0. Example appears as: `SRU(DAI0_PB12_O, SPT0_ACLK_I);`

Debug Features

The following section describes the feature that can be used to help in debugging the DAI.

Loopback Routing

The SPORT serial peripheral supports an internal loopback mode. If the loopback bit for each peripheral is enabled, it connects the transmitter with the receiver block internally (does not signal off-chip). The SRU can be used for this purpose. The ***Loopback Routing*** table describes the different possible routings based on the peripheral.

NOTE: The peripheral's loopback mode for debug is independent from both signal routing units.

Table 29-9: Loopback Routing

Peripheral	Loopback Mode	SRU1–0 Internal Routing for Loopback	SRU1–0 External Routing for Loopback
SPORT	Yes	SPTx_xx_O → SPTx_xx_I	SPTx_xx_O → DAIn_PBxx_I DAIn_PBxx_O → SPTx_xx_I
S/PDIF Tx/Rx	No	SPDIFn_TX_O → SPDIFn_RX_I	SPDIFn_TX_O → DAI_PBxx_I DAI_PBxx_O → SPDIFn_RX_I
SRC	No	SRCx_DAT_OP_O → SRCx_DAT_IP_I	SRCx_DAT_OP_O → DAI_PBxx_I DAI_PBxx_O → SRCx_DAT_IP_I

DAI Sources Overview

The following tables provide information of the various DAI sources sorted by group.

Group A – Clock Routing Signals

The group A clock routing sources are listed in the following table.

Table 29-10: Group A – Clock Routing Sources

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	00000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	0	00001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	0	00010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	0	00011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	0	00100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	0	00101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	0	00110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	0	00111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	0	01000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	0	01001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	0	01010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	0	01011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	0	01100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	0	01101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	0	01110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	0	01111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	0	10000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	0	10001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	0	10010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	0	10011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	0	10100 (0x14)	SPT0_ACLK_O	SPORT 0 Clock A
0	0	10101 (0x15)	SPT0_BCLK_O	SPORT 0 Clock B
0	0	10110 (0x16)	SPT1_ACLK_O	SPORT 1 Clock A
0	0	10111 (0x17)	SPT1_BCLK_O	SPORT 1 Clock B
0	0	11000 (0x18)	SPT2_ACLK_O	SPORT 2 Clock A
0	0	11001 (0x19)	SPT2_BCLK_O	SPORT 2 Clock B

Table 29-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	11010 (0x1A)	SPDIF0_RX_CLK_O	SPDIF 0 Receive Clock Output
0	0	11011 (0x1B)	SPDIF0_RX_TDMCLK_O	SPDIF 0 Receive TDM Clock Output
0	0	11100 (0x1C)	PCG_CLKA_O	Precision Clock A Output
0	0	11101 (0x1D)	PCG_CLKB_O	Precision Clock B Output
0	0	11110 (0x1E)	LOW	Logic Level Low (0)
0	0	11111 (0x1F)	HIGH	Logic Level High (1)
0	1	00000 (0x0)	SPT3_ACLK_O	SPORT 3 clock A
0	1	00001 (0x1)	SPT3_BCLK_O	SPORT 3 clock B
0	1	00010 (0x2)	PCG_CLKE_O	Precision Clock E output
0	1	00011 (0x3)	PCG_CLKF_O	Precision Clock F output
0	1	00100 (0x4)	PCG_CLKA_INV_O	Inverted Precision Clock A output
0	1	00101 (0x5)	PCG_CLKB_INV_O	Inverted Precision Clock B output
0	1	00110 (0x6)	PCG_CLKE_INV_O	Inverted Precision Clock E output
0	1	00111 (0x7)	PCG_CLKF_INV_O	Inverted Precision Clock F output
0	1	01000 (0x8)	PDM0_CLK0_O	PDM0 Clock0 output
0	1	01001 (0x9)	DAI0_DUMMY_PB01_O	Dummy Pin Buffer1 output
0	1	01010 (0xA)	DAI0_DUMMY_PB02_O	Dummy Pin Buffer2 output
0	1	01011 (0xB)	DAI0_DUMMY_PB03_O	Dummy Pin Buffer3 output
0	1	01100 (0xC)	DAI0_DUMMY_PB04_O	Dummy Pin Buffer4 output
0	1	01101 (0xD) -11111 (0x1F)	Reserved	
1	0	00000 (0x0)	DAI1_PB01_O	Pin Buffer 1 (from other DAI)
1	0	00001(0x1)	DAI1_PB02_O	Pin Buffer 2 (from other DAI)
1	0	00010 (0x2)	DAI1_PB03_O	Pin Buffer 3 (from other DAI). This pin buffer is used for sharing clocks between DAI modules. DAI0_PIN03 and DAI1_PIN03 can be used to share clock signal across DAI. When DAI0_PIN03 is configured as input clock pin then this clock signal is available in Group A of both DAIs. This source signal is DAI0_CRS_PB03_O in group A of DAI

Table 29-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
				0 and DAI1_CRS_PB03_O in group A of DAI1.
1	0	00011 (0x3)	DAI1_PB04_O	Pin Buffer 4 (from other DAI)
1	0	00100 (0x4)	DAI1_PB05_O	Pin Buffer 5 (from other DAI). This pin buffer is used for sharing clocks between DAI modules. DAI0_PIN05 and DAI1_PIN05 can be used to share clock signal across DAI. When DAI0_PIN05 is configured as input clock pin then this clock signal is available in Group A of both DAIs. This source signal is DAI0_CRS_PB05_O in group A of DAI 0 and DAI1_CRS_PB05_O in group A of DAI1.
1	0	00101 (0x5)	DAI1_PB06_O	Pin Buffer 6 (from other DAI)
1	0	00110 (0x6)	DAI1_PB07_O	Pin Buffer 7 (from other DAI)
1	0	00111 (0x7)	DAI1_PB08_O	Pin Buffer 8 (from other DAI)
1	0	01000 (0x8)	DAI1_PB09_O	Pin Buffer 9 (from other DAI)
1	0	01001 (0x9)	DAI1_PB10_O	Pin Buffer 10 (from other DAI)
1	0	01010 (0xA)	DAI1_PB11_O	Pin Buffer 11 (from other DAI)
1	0	01011 (0xB)	DAI1_PB12_O	Pin Buffer 12 (from other DAI)
1	0	01100 (0xC)	DAI1_PB13_O	Pin Buffer 13 (from other DAI)
1	0	01101 (0xD)	DAI1_PB14_O	Pin Buffer 14 (from other DAI)
1	0	01110 (0xE)	DAI1_PB15_O	Pin Buffer 15 (from other DAI)
1	0	01111 (0xF)	DAI1_PB16_O	Pin Buffer 16 (from other DAI)
1	0	10000 (0x10)	DAI1_PB17_O	Pin Buffer 17 (from other DAI)
1	0	10001 (0x11)	DAI1_PB18_O	Pin Buffer 18 (from other DAI)
1	0	10010 (0x12)	DAI1_PB19_O	Pin Buffer 19 (from other DAI)
1	0	10011 (0x13)	DAI1_PB20_O	Pin Buffer 20 (from other DAI)
1	0	10100 (0x14)	SPT4_ACLK_O	SPORT 4 Clock A (from other DAI)
1	0	10101 (0x15)	SPT4_BCLK_O	SPORT 4 Clock B (from other DAI)
1	0	10110 (0x16)	SPT5_ACLK_O	SPORT 5 Clock A (from other DAI)
1	0	10111 (0x17)	SPT5_BCLK_O	SPORT 5 Clock B (from other DAI)

Table 29-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	0	11000 (0x18)	SPT6_ACLK_O	SPORT 6 Clock A (from other DAI)
1	0	11001 (0x19)	SPT6_BCLK_O	SPORT 6 Clock B (from other DAI)
1	0	11010 (0x1A)	SPDIF1_RX_CLK_O	SPDIF 0 Receive Clock Output (from other DAI)
1	0	11011 (0x1B)	SPDIF1_RX_TDMCLK_O	SPDIF 0 Receive TDM Clock Output (from other DAI)
1	0	11100 (0x1C)	PCG_CLKC_O	Precision Clock C Output (from other DAI)
1	0	11101 (0x1D)	PCG_CLKD_O	Precision Clock D Output (from other DAI)
1	0	11110 (0x1E)	Reserved	
1	0	11111 (0x1F)	Reserved	
1	1	00000 (0x0)	SPT7_ACLK_O	SPORT 3 clock A (from other DAI)
1	1	00001 (0x1)	SPT7_BCLK_O	SPORT 3 clock B (from other DAI)
1	1	00010 (0x2)	PCG_CLKG_O	Precision Clock G output (from other DAI)
1	1	00011 (0x3)	PCG_CLKH_O	Precision Clock H output (from other DAI)
1	1	00100 (0x4)	PCG_CLKC_INV_0	Inverted Precision Clock C output (from other DAI)
1	1	00101 (0x5)	PCG_CLKD_INV_0	Inverted Precision Clock D output (from other DAI)
1	1	00110 (0x6)	PCG_CLKG_INV_0	Inverted Precision Clock G output (from other DAI)
1	1	00111 (0x7)	PCG_CLKH_INV_0	Inverted Precision Clock H output (from other DAI)
1	1	01000 (0x8)	PDM1_CLK0_O	PDM1 Clock0 Output (from other DAI)
1	1	01001 (0x9)	DAI1_DUMMY_PB01_O	Dummy Pin Buffer1 output (from other DAI)
1	1	01010 (0xA)	DAI1_DUMMY_PB02_O	Dummy Pin Buffer2 output (from other DAI)
1	1	01011 (0xB)	DAI1_DUMMY_PB03_O	Dummy Pin Buffer3 output (from other DAI)
1	1	01100 (0xC)	DAI1_DUMMY_PB04_O	Dummy Pin Buffer4 output (from other DAI)

Table 29-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	1	01101 (0xD) -11111 (0x1F)	Reserved	

Group B – Serial Data Source Signals

The group B data sources are listed in the following table. The group B data destinations are configured using DAI_PBEN0 through DAI_PBEN3.

Table 29-11: Group B – Serial Data Signals

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	Reserved	000000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	Reserved	000001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	Reserved	000010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	Reserved	000011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	Reserved	000100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	Reserved	000101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	Reserved	000110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	Reserved	000111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	Reserved	001000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	Reserved	001001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	Reserved	001010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	Reserved	001011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	Reserved	001100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	Reserved	001101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	Reserved	001110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	Reserved	001111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	Reserved	010000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	Reserved	010001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	Reserved	010010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	Reserved	010011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	Reserved	010100 (0x14)	SPT0_AD0_O	SPORT 0 Data AD0
0	Reserved	010101 (0x15)	SPT0_AD1_O	SPORT 0 Data AD1
0	Reserved	010110 (0x16)	SPT0_BD0_O	SPORT 0 Data BD0
0	Reserved	010111 (0x17)	SPT0_BD1_O	SPORT 0 Data BD1
0	Reserved	011000 (0x18)	SPT1_AD0_O	SPORT 1 Data AD0
0	Reserved	011001 (0x19)	SPT1_AD1_O	SPORT 1 Data AD1

Table 29-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	011010 (0x1A)	SPT1_BD0_O	SPORT 1 Data BD0
0	Reserved	011011 (0x1B)	SPT1_BD1_O	SPORT 1 Data BD1
0	Reserved	011100 (0x1C)	SPT2_AD0_O	SPORT 2 Data AD0
0	Reserved	011101 (0x1D)	SPT2_AD1_O	SPORT 2 Data AD1
0	Reserved	011110 (0x1E)	SPT2_BD0_O	SPORT 2 Data BD0
0	Reserved	011111 (0x1F)	SPT2_BD1_O	SPORT 2 Data BD1
0	Reserved	100000 (0x20)	SRC0_DAT_OP_O	SRC0 Data Out
0	Reserved	100001 (0x21)	SRC1_DAT_OP_O	SRC1 Data Out
0	Reserved	100010 (0x22)	SRC2_DAT_OP_O	SRC2 Data Out
0	Reserved	100011 (0x23)	SRC3_DAT_OP_O	SRC3 Data Out
0	Reserved	100100 (0x24)	SRC0_TDM_IP_O	SRC0 Data Out
0	Reserved	100101 (0x25)	SRC1_TDM_IP_O	SRC1 Data Out
0	Reserved	100110 (0x26)	SRC2_TDM_IP_O	SRC2 Data Out
0	Reserved	100111 (0x27)	SRC3_TDM_IP_O	SRC3 Data Out
0	Reserved	101000 (0x28)	SPDIF0_RX_DAT_O	SPDIF 0 RX Serial Data Out
0	Reserved	101001 (0x29)–101011 (0x2B)	Reserved	
0	Reserved	101100 (0x2C)	SPT3_AD0_O	SPORT 3 Data AD0
0	Reserved	101101 (0x2D)	SPT3_AD1_O	SPORT 3 Data AD1
0	Reserved	101110 (0x2E)	SPT3_BD0_O	SPORT 3 Data BD0
0	Reserved	101111 (0x2F)	SPT3_BD1_O	SPORT 3 Data BD1
0	Reserved	110000 (0x30)	SPDIF0_TX_O	SPDIF 0 TX Biphase Stream
0	Reserved	110001 (0x31)	Reserved	
0	Reserved	110010 (0x32)	Reserved	
0	Reserved	110011 (0x33)	PDM0_SDATA_O	PDM0 Serial Data Output
0	Reserved	110100 (0x34)	SRC8_DAT_OP_O	SRC8 Data Out
0	Reserved	110101 (0x35)	SRC9_DAT_OP_O	SRC9 Data Out
0	Reserved	110110 (0x36)	SRC10_DAT_OP_O	SRC10 Data Out
0	Reserved	110111 (0x37)	SRC11_DAT_OP_O	SRC11 Data Out
0	Reserved	111000 (0x38)	SRC8_TDM_IP_O	SRC8 TDM Data Out

Table 29-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	111001 (0x39)	SRC9_TDM_IP_O	SRC9 TDM Data Out
0	Reserved	111010 (0x3A)	SRC10_TDM_IP_O	SRC10 TDM Data Out
0	Reserved	111011 (0x3B)	SRC11_TDM_IP_O	SRC11 TDM Data Out
0	Reserved	111100 (0x3C) - 111101 (0x3D)	Reserved	
0	Reserved	111110 (0x3E)	LOW	Logic Level Low (0)
0	Reserved	111111 (0x3F)	HIGH	Logic Level High (1)
1	Reserved	000000 (0x0)	DAI1_PB01_O	Pin Buffer 1 (From other DAI)
1	Reserved	000001 (0x1)	DAI1_PB02_O	Pin Buffer 2 (From other DAI)
1	Reserved	000010 (0x2)	DAI1_PB03_O	Pin Buffer 3 (From other DAI)
1	Reserved	000011 (0x3)	DAI1_PB04_O	Pin Buffer 4 (From other DAI)
1	Reserved	000100 (0x4)	DAI1_PB05_O	Pin Buffer 5 (From other DAI)
1	Reserved	000101 (0x5)	DAI1_PB06_O	Pin Buffer 6 (From other DAI)
1	Reserved	000110 (0x6)	DAI1_PB07_O	Pin Buffer 7 (From other DAI)
1	Reserved	000111 (0x7)	DAI1_PB08_O	Pin Buffer 8 (From other DAI)
1	Reserved	001000 (0x8)	DAI1_PB09_O	Pin Buffer 9 (From other DAI)
1	Reserved	001001 (0x9)	DAI1_PB10_O	Pin Buffer 10 (From other DAI)
1	Reserved	001010 (0xA)	DAI1_PB11_O	Pin Buffer 11 (From other DAI)
1	Reserved	001011 (0xB)	DAI1_PB12_O	Pin Buffer 12 (From other DAI)
1	Reserved	001100 (0xC)	DAI1_PB13_O	Pin Buffer 13 (From other DAI)
1	Reserved	001101 (0xD)	DAI1_PB14_O	Pin Buffer 14 (From other DAI)
1	Reserved	001110 (0xE)	DAI1_PB15_O	Pin Buffer 15 (From other DAI)
1	Reserved	001111 (0xF)	DAI1_PB16_O	Pin Buffer 16 (From other DAI)
1	Reserved	010000 (0x10)	DAI1_PB17_O	Pin Buffer 17 (From other DAI)
1	Reserved	010001 (0x11)	DAI1_PB18_O	Pin Buffer 18 (From other DAI)
1	Reserved	010010 (0x12)	DAI1_PB19_O	Pin Buffer 19 (From other DAI)
1	Reserved	010011 (0x13)	DAI1_PB20_O	Pin Buffer 20 (From other DAI)
1	Reserved	010100 (0x14)	SPT4_AD0_O	SPORT 4 Data AD0 (From other DAI)
1	Reserved	010101 (0x15)	SPT4_AD1_O	SPORT 4 Data AD1 (From other DAI)
1	Reserved	010110 (0x16)	SPT4_BD0_O	SPORT 4 Data BD0 (From other DAI)

Table 29-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	010111 (0x17)	SPT4_BD1_O	SPORT 4 Data BD1 (From other DAI)
1	Reserved	011000 (0x18)	SPT5_AD0_O	SPORT 5 Data AD0 (From other DAI)
1	Reserved	011001 (0x19)	SPT5_AD1_O	SPORT 5 Data AD1 (From other DAI)
1	Reserved	011010 (0x1A)	SPT5_BD0_O	SPORT 5 Data BD0 (From other DAI)
1	Reserved	011011 (0x1B)	SPT5_BD1_O	SPORT 5 Data BD1 (From other DAI)
1	Reserved	011100 (0x1C)	SPT6_AD0_O	SPORT 6 Data AD0 (From other DAI)
1	Reserved	011101 (0x1D)	SPT6_AD1_O	SPORT 6 Data AD1 (From other DAI)
1	Reserved	011110 (0x1E)	SPT6_BD0_O	SPORT 6 Data BD0 (From other DAI)
1	Reserved	011111 (0x1F)	SPT6_BD1_O	SPORT 6 Data BD1 (From other DAI)
1	Reserved	100000 (0x20)	SRC4_DAT_OP_O	SRC4 Data Out (From other DAI)
1	Reserved	100001 (0x21)	SRC5_DAT_OP_O	SRC5 Data Out (From other DAI)
1	Reserved	100010 (0x22)	SRC6_DAT_OP_O	SRC6 Data Out (From other DAI)
1	Reserved	100011 (0x23)	SRC7_DAT_OP_O	SRC7 Data Out (From other DAI)
1	Reserved	100100 (0x24)	SRC4_TDM_IP_O	SRC4 Data Out (From other DAI)
1	Reserved	100101 (0x25)	SRC5_TDM_IP_O	SRC5 Data Out (From other DAI)
1	Reserved	100110 (0x26)	SRC6_TDM_IP_O	SRC6 Data Out (From other DAI)
1	Reserved	100111 (0x27)	SRC7_TDM_IP_O	SRC7 Data Out (From other DAI)
1	Reserved	101000 (0x28)	SPDIF1_RX_DAT_O	SPDIF 1 RX Serial Data Out (From other DAI)
1	Reserved	101001 (0x29)–101011 (0x2B)	Reserved	
1	Reserved	101100 (0x2C)	SPT7_AD0_O	SPORT 7 Data AD0 (From other DAI)
1	Reserved	101101 (0x2D)	SPT7_AD1_O	SPORT 7 Data AD1 (From other DAI)
1	Reserved	101110 (0x2E)	SPT7_BD0_O	SPORT 7 Data BD0 (From other DAI)
1	Reserved	101111 (0x2F)	SPT7_BD1_O	SPORT 7 Data BD1 (From other DAI)
1	Reserved	110000 (0x30)	SPDIF1_TX_O	SPDIF 1 TX Biphase Stream (From other DAI)
1	Reserved	110001 (0x31)	Reserved	
1	Reserved	110010 (0x32)	Reserved	
1	Reserved	110011 (0x33)	PDM1_SDATA_O	PDM1 Serial Data Output
1	Reserved	110100 (0x34)	SRC12_DAT_OP_O	SRC12 Data Out

Table 29-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	Reserved	110101 (0x35)	SRC13_DAT_OP_O	SRC13 Data Out
1	Reserved	110110 (0x36)	SRC14_DAT_OP_O	SRC14 Data Out
1	Reserved	110111 (0x37)	SRC15_DAT_OP_O	SRC15 Data Out
1	Reserved	111000 (0x38)	SRC12_TDM_IP_O	SRC12 TDM Data Out
1	Reserved	111001 (0x39)	SRC13_TDM_IP_O	SRC13 TDM Data Out
1	Reserved	111010 (0x3A)	SRC14_TDM_IP_O	SRC14 TDM Data Out
1	Reserved	111011 (0x3B)	SRC15_TDM_IP_O	SRC15 TDM Data Out
1	Reserved	110001 (0x3C) - 111101 (0x3D)	Reserved	
1	Reserved	111110 (0x3E)	Reserved	
1	Reserved	111111 (0x3F)	Reserved	

Group C – Frame Sync Source Signals

The group C frame sync signal sources are listed in the following table.

Table 29-12: Group C – Frame Sync Signals

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	00000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	0	00001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	0	00010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	0	00011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	0	00100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	0	00101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	0	00110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	0	00111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	0	01000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	0	01001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	0	01010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	0	01011 (0xB)	DAI0_PB12_O	Pin Buffer 12

Table 29-12: Group C – Frame Sync Signals (Continued)

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	0	01100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	0	01101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	0	01110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	0	01111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	0	10000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	0	10001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	0	10010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	0	10011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	0	10100 (0x14)	SPT0_AFS_O	SPORT 0 Frame Sync A
0	0	10101 (0x15)	SPT0_BFS_O	SPORT 0 Frame Sync B
0	0	10110 (0x16)	SPT1_AFS_O	SPORT 1 Frame Sync A
0	0	10111 (0x17)	SPT1_BFS_O	SPORT 1 Frame Sync B
0	0	11000 (0x18)	SPT2_AFS_O	SPORT 2 Frame Sync A
0	0	11001 (0x19)	SPT2_BFS_O	SPORT 2 Frame Sync B
0	0	11010 (0x1A)	SPDIF0_FS_O	SPDIF 0 RX Frame Sync Output
0	0	11011 (0x1B)	Reserved	
0	0	11100 (0x1C)	PCG0_FSA_O	Precision Frame Sync A Output
0	0	11101 (0x1D)	PCG0_FSB_O	Precision Frame Sync B Output
0	0	11110 (0x1E)	LOW	Logic Level Low (0)
0	0	11111 (0x1F)	HIGH	Logic Level High (1)
0	1	00000 (0x0)	SPT3_AFS_O	SPORT 3 Frame Sync A
0	1	00001 (0x1)	SPT3_BFS_O	SPORT 3 Frame Sync B
0	1	00010 (0x2)	PCG_FSE_O	Precision Frame Sync E Output
0	1	00011 (0x3)	PCG_FSF_O	Precision Frame Sync F Output
0	1	00100 (0x4)	PCG_FSA_INV_O	Precision Inverted Frame Sync A Output
0	1	00101 (0x5)	PCG_FSB_INV_O	Precision Inverted Frame Sync B Output
0	1	00110 (0x6)	PCG_FSE_INV_O	Precision Inverted Frame Sync E Output
0	1	00111 (0x7)	PCG_FSF_INV_O	Precision Inverted Frame Sync F Output
0	1	01000 (0x8)	DAI0_DUMMY_PB01_O	Dummy Pin Buffer1 output
0	1	01001 (0x9)	DAI0_DUMMY_PB02_O	Dummy Pin Buffer2 output

Table 29-12: Group C – Frame Sync Signals (Continued)

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	1	01010 (0xA)	DAI0_DUMMY_PB03_O	Dummy Pin Buffer3 output
0	1	01011 (0xB)	DAI0_DUMMY_PB04_O	Dummy Pin Buffer4 output
0	1	01100 (0xC) - 11111 (0x1F)	Reserved	
1	0	00000 (0x0)	DAI1_PB01_O	Pin Buffer 1
1	0	00001 (0x1)	DAI1_PB02_O	Pin Buffer 2
1	0	00010 (0x2)	DAI1_PB03_O	Pin Buffer 3
1	0	00011 (0x3)	DAI1_PB04_O	Pin Buffer 4
1	0	00100 (0x4)	DAI1_PB05_O	Pin Buffer 5
1	0	00101 (0x5)	DAI1_PB06_O	Pin Buffer 6
1	0	00110 (0x6)	DAI1_PB07_O	Pin Buffer 7
1	0	00111 (0x7)	DAI1_PB08_O	Pin Buffer 8
1	0	01000 (0x8)	DAI1_PB09_O	Pin Buffer 9
1	0	01001 (0x9)	DAI1_PB10_O	Pin Buffer 10
1	0	01010 (0xA)	DAI1_PB11_O	Pin Buffer 11
1	0	01011 (0xB)	DAI1_PB12_O	Pin Buffer 12
1	0	01100 (0xC)	DAI1_PB13_O	Pin Buffer 13
1	0	01101 (0xD)	DAI1_PB14_O	Pin Buffer 14
1	0	01110 (0xE)	DAI1_PB15_O	Pin Buffer 15
1	0	01111 (0xF)	DAI1_PB16_O	Pin Buffer 16
1	0	10000 (0x10)	DAI1_PB17_O	Pin Buffer 17
1	0	10001 (0x11)	DAI1_PB18_O	Pin Buffer 18
1	0	10010 (0x12)	DAI1_PB19_O	Pin Buffer 19
1	0	10011 (0x13)	DAI1_PB20_O	Pin Buffer 20
1	0	10100 (0x14)	SPT4_AFS_O	SPORT 4 Frame Sync A
1	0	10101 (0x15)	SPT4_BFS_O	SPORT 4 Frame Sync B
1	0	10110 (0x16)	SPT5_AFS_O	SPORT 5 Frame Sync A
1	0	10111 (0x17)	SPT5_BFS_O	SPORT 5 Frame Sync B
1	0	11000 (0x18)	SPT6_AFS_O	SPORT 6 Frame Sync A
1	0	11001 (0x19)	SPT6_BFS_O	SPORT 6 Frame Sync B

Table 29-12: Group C – Frame Sync Signals (Continued)

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	0	11010 (0x1A)	SPDIF1_FS_O	SPDIF 1 RX Frame Sync Output
1	0	11011 (0x1B)	Reserved	
1	0	11100 (0x1C)	PCG_FSC_O	Precision Frame Sync C Output
1	0	11101 (0x1D)	PCG_FSD_O	Precision Frame Sync D Output
1	0	11110 (0x1E)	Reserved	
1	0	11111 (0x1F)	Reserved	
1	1	00000 (0x0)	SPT7_AFS_O	SPORT 7 Frame Sync A (From other DAI)
1	1	00001 (0x1)	SPT7_BFS_O	SPORT 7 Frame Sync B (From other DAI)
1	1	00010 (0x2)	PCG_FSG_O	Precision Frame Sync G Output (From other DAI)
1	1	00011 (0x3)	PCG_FSH_O	Precision Frame Sync H Output (From other DAI)
1	1	00100 (0x4)	PCG_FSC_INV_O	Precision Inverted Frame Sync C Output (From other DAI)
1	1	00101 (0x5)	PCG_FSD_INV_O	Precision Inverted Frame Sync D Output (From other DAI)
1	1	00110 (0x6)	PCG_FSG_INV_O	Precision Inverted Frame Sync G Output (From other DAI)
1	1	00111 (0x7)	PCG_FSH_INV_O	Precision Inverted Frame Sync H Output (From other DAI)
1	1	01000 (0x8)	DAI0_DUMMY_PB01_O	Dummy Pin Buffer1 output
1	1	01001 (0x9)	DAI0_DUMMY_PB02_O	Dummy Pin Buffer2 output
1	1	01010 (0xA)	DAI0_DUMMY_PB03_O	Dummy Pin Buffer3 output
1	1	01011 (0xB)	DAI0_DUMMY_PB04_O	Dummy Pin Buffer4 output
1	1	01100 (0xC) - 11111 (0x1F)	Reserved	

Group D – Pin Signal Assignment Source Signals

The group D pin signal assignment sources are listed in the following table.

Table 29-13: Group D – Pin Signal Assignments

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	0000000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	Reserved	0000001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	Reserved	0000010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	Reserved	0000011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	Reserved	0000100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	Reserved	0000101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	Reserved	0000110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	Reserved	0000111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	Reserved	0001000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	Reserved	0001001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	Reserved	0001010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	Reserved	0001011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	Reserved	0001100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	Reserved	0001101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	Reserved	0001110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	Reserved	0001111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	Reserved	0010000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	Reserved	0010001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	Reserved	0010010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	Reserved	0010011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	Reserved	0010100 (0x14)	SPT0_AD0_O	SPORT 0 Data AD0
0	Reserved	0010101 (0x15)	SPT0_AD1_O	SPORT 0 Data AD1
0	Reserved	0010110 (0x16)	SPT0_BD0_O	SPORT 0 Data BD0
0	Reserved	0010111 (0x17)	SPT0_BD1_O	SPORT 0 Data BD1
0	Reserved	0011000 (0x18)	SPT1_AD0_O	SPORT 1 Data AD0
0	Reserved	0011001 (0x19)	SPT1_AD1_O	SPORT 1 Data AD1
0	Reserved	0011010 (0x1A)	SPT1_BD0_O	SPORT 1 Data BD0
0	Reserved	0011011 (0x1B)	SPT1_BD1_O	SPORT 1 Data BD1
0	Reserved	0011100 (0x1C)	SPT2_AD0_O	SPORT 2 Data AD0
0	Reserved	0011101 (0x1D)	SPT2_AD1_O	SPORT 2 Data AD1

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	0011110 (0x1E)	SPT2_BD0_O	SPORT 2 Data BD0
0	Reserved	0011111 (0x1F)	SPT2_BD1_O	SPORT 2 Data BD1
0	Reserved	0100000 (0x20)	SPT0_ACLK_O	SPORT 0 Clock A
0	Reserved	0100001 (0x21)	SPT0_BCLK_O	SPORT 0 Clock B
0	Reserved	0100010 (0x22)	SPT1_ACLK_O	SPORT 1 Clock A
0	Reserved	0100011 (0x23)	SPT1_BCLK_O	SPORT 1 Clock B
0	Reserved	0100100 (0x24)	SPT2_ACLK_O	SPORT 2 Clock A
0	Reserved	0100101 (0x25)	SPT2_BCLK_O	SPORT 2 Clock B
0	Reserved	0100110 (0x26)	SPT0_AFS_O	SPORT 0 Frame Sync A
0	Reserved	0100111 (0x27)	SPT0_BFS_O	SPORT 0 Frame Sync B
0	Reserved	0101000 (0x28)	SPT1_AFS_O	SPORT 1 Frame Sync A
0	Reserved	0101001 (0x29)	SPT1_BFS_O	SPORT 1 Frame Sync B
0	Reserved	0101010 (0x2A)	SPT2_AFS_O	SPORT 2 Frame Sync A
0	Reserved	0101011 (0x2B)	SPT2_BFS_O	SPORT 2 Frame Sync B
0	Reserved	0101100 (0x2C)	SPT3_AD0_O	SPORT 3 Data AD0
0	Reserved	0101101 (0x2D)	SPT3_AD1_O	SPORT 3 Data AD1
0	Reserved	0101110 (0x2E)	SPT3_BD0_O	SPORT 3 Data BD0
0	Reserved	0101111 (0x2F)	SPT3_BD1_O	SPORT 3 Data BD1
0	Reserved	0110000 (0x30)	MLB0_CLKOUT	MLB PLL clock output
0	Reserved	0110001 (0x31)	SPDIF0_TX_BLKSTART_O	SPDIF 0 TX Block Start Output
0	Reserved	0110010 (0x32)	Reserved	
0	Reserved	0110011 (0x33)	Reserved	
0	Reserved	0110100 (0x34)	SPT3_ACLK_O	SPORT 3 Clock A
0	Reserved	0110101 (0x35)	SPT3_BCLK_O	SPORT 3 Clock B
0	Reserved	0110110 (0x36)	SPT3_AFS_O	SPORT 3 Frame Sync A
0	Reserved	0110111 (0x37)	SPT3_BFS_O	SPORT 3 Frame Sync B
0	Reserved	0111000 (0x38)	PCG0_CLKA_O	Precision Clock A
0	Reserved	0111001 (0x39)	PCG0_CLKB_O	Precision Clock B
0	Reserved	0111010 (0x3A)	PCG0_FSA_O	Precision Frame Sync A
0	Reserved	0111011 (0x3B)	PCG0_FSB_O	Precision Frame Sync B

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	0111100 (0x3C)	Reserved	
0	Reserved	0111101 (0x3D)	SRC0_DAT_OP_O	SRC0 Data Output
0	Reserved	0111110 (0x3E)	SRC1_DAT_OP_O	SRC1 Data Output
0	Reserved	0111111 (0x3F)	SRC2_DAT_OP_O	SRC2 Data Output
0	Reserved	1000000 (0x40)	SRC3_DAT_OP_O	SRC3 Data Output
0	Reserved	1000001 (0x41)	SPDIF0_RX_DAT_O	SPDIF 0 RX Data Output
0	Reserved	1000010 (0x42)	SPDIF0_FS_O	SPDIF 0 RX Frame Sync Output
0	Reserved	1000011 (0x43)	SPDIF0_RXCLK_O	SPDIF 0 RX Clock Output
0	Reserved	1000100 (0x44)	SPDIF0_RX_TDMCLK_O	SPDIF 0 RX TDM Clock Output
0	Reserved	1000101 (0x45)	SPDIF0_TX_O	SPDIF 0 TX Biphase Encoded Data Output
0	Reserved	1000110 (0x46)	SPT0_ATDV_O	SPORT0 Transmit A Data Valid Output
0	Reserved	1000111 (0x47)	SPT0_BTDV_O	SPORT0 Transmit B Data Valid Output
0	Reserved	1001000 (0x48)	SPT1_ATDV_O	SPORT1 Transmit A Data Valid Output
0	Reserved	1001001 (0x49)	SPT1_BTDV_O	SPORT1 Transmit B Data Valid Output
0	Reserved	1001010 (0x4A)	SPT2_ATDV_O	SPORT2 Transmit A Data Valid Output
0	Reserved	1001011 (0x4B)	SPT2_BTDV_O	SPORT2 Transmit B Data Valid Output
0	Reserved	1001100 (0x4C)	SPT3_ATDV_O	SPORT3 Transmit A Data Valid Output
0	Reserved	1001101 (0x4D)	SPT3_BTDV_O	SPORT3 Transmit B Data Valid Output
0	Reserved	1001110 (0x4E) – 1010101 (0x55)	Reserved	
0	Reserved	1010110 (0x56)	PCG_CLKE_O	Precision Clock E
0	Reserved	1010111 (0x57)	PCG_CLKF_O	Precision Clock F
0	Reserved	1011000 (0x58)	PCG_FSE_O	Precision Frame Sync E
0	Reserved	1011001 (0x59)	PCG_FSF_O	Precision Frame Sync F
0	Reserved	1011010 (0x5A)	PCG_CLKA_INV_O	Inverted Precision Clock A Output
0	Reserved	1011011 (0x5B)	PCG_CLKB_INV_O	Inverted Precision Clock B Output
0	Reserved	1011100 (0x5C)	PCG_CLKE_INV_O	Inverted Precision Clock E Output
0	Reserved	1011101 (0x5D)	PCG_CLKF_INV_O	Inverted Precision Clock F Output
0	Reserved	1011110 (0x5E)	PCG_FSA_INV_O	Inverted Precision Frame Sync A Output

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	1011111 (0x5F)	PCG_FSB_INV_O	Inverted Precision Frame Sync B Output
0	Reserved	1100000 (0x60)	PCG_FSE_INV_O	Inverted Precision Frame Sync E Output
0	Reserved	1100001 (0x61)	PCG_FSF_INV_O	Inverted Precision Frame Sync F Output
0	Reserved	1100010 (0x62)	PDM0_CLK0_O	PDM0 Clock0 Output
0	Reserved	1100011 (0x63)	PDM0_SDATA_O	PDM0 Serial Data Output
0	Reserved	1100100 (0x64)	SRC8_DAT_OP_O	SRC8 Data Output
0	Reserved	1100101 (0x65)	SRC9_DAT_OP_O	SRC9 Data Output
0	Reserved	1100110 (0x66)	SRC10_DAT_OP_O	SRC10 Data Output
0	Reserved	1100111 (0x67)	SRC11_DAT_OP_O	SRC11 Data Output
0	Reserved	1101000 (0x68)	SRU_FRACPLL0_CLK_O	Fractional-N PLL 0 Clock Output
0	Reserved	1101001 (0x69)	SRU_FRACPLL1_CLK_O	Fractional-N PLL 1 Clock Output
0	Reserved	1101010 (0x6A) – 1111101 (0x7D)	Reserved	
0	Reserved	1111110 (0x7E)	LOW	Logic Level Low (0)
0	Reserved	1111111 (0x7F)	HIGH	Logic Level High (1)
1	Reserved	0000000 (0x0)	DAI1_PB01_O	Pin Buffer 1 (from other DAI)
1	Reserved	0000001 (0x1)	DAI1_PB02_O	Pin Buffer 2 (from other DAI)
1	Reserved	0000010 (0x2)	DAI1_PB03_O	Pin Buffer 3 (from other DAI)
1	Reserved	0000011 (0x3)	DAI1_PB04_O	Pin Buffer 4 (from other DAI)
1	Reserved	0000100 (0x4)	DAI1_PB05_O	Pin Buffer 5 (from other DAI)
1	Reserved	0000101 (0x5)	DAI1_PB06_O	Pin Buffer 6 (from other DAI)
1	Reserved	0000110 (0x6)	DAI1_PB07_O	Pin Buffer 7 (from other DAI)
1	Reserved	0000111 (0x7)	DAI1_PB08_O	Pin Buffer 8 (from other DAI)
1	Reserved	0001000 (0x8)	DAI1_PB09_O	Pin Buffer 9 (from other DAI)
1	Reserved	0001001 (0x9)	DAI1_PB10_O	Pin Buffer 10 (from other DAI)
1	Reserved	0001010 (0xA)	DAI1_PB11_O	Pin Buffer 11 (from other DAI)
1	Reserved	0001011 (0xB)	DAI1_PB12_O	Pin Buffer 12 (from other DAI)
1	Reserved	0001100 (0xC)	DAI1_PB13_O	Pin Buffer 13 (from other DAI)
1	Reserved	0001101 (0xD)	DAI1_PB14_O	Pin Buffer 14 (from other DAI)
1	Reserved	0001110 (0xE)	DAI1_PB15_O	Pin Buffer 15 (from other DAI)

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	0001111 (0xF)	DAI1_PB16_O	Pin Buffer 16 (from other DAI)
1	Reserved	0010000 (0x10)	DAI1_PB17_O	Pin Buffer 17 (from other DAI)
1	Reserved	0010001 (0x11)	DAI1_PB18_O	Pin Buffer 18 (from other DAI)
1	Reserved	0010010 (0x12)	DAI1_PB19_O	Pin Buffer 19 (from other DAI)
1	Reserved	0010011 (0x13)	DAI1_PB20_O	Pin Buffer 20 (from other DAI)
1	Reserved	0010100 (0x14)	SPT4_AD0_O	SPORT 4 Data AD0 (from other DAI)
1	Reserved	0010101 (0x15)	SPT4_AD1_O	SPORT 4 Data AD1 (from other DAI)
1	Reserved	0010110 (0x16)	SPT4_BD0_O	SPORT 4 Data BD0 (from other DAI)
1	Reserved	0010111 (0x17)	SPT4_BD1_O	SPORT 4 Data BD1 (from other DAI)
1	Reserved	0011000 (0x18)	SPT5_AD0_O	SPORT 5 Data AD0 (from other DAI)
1	Reserved	0011001 (0x19)	SPT5_AD1_O	SPORT 5 Data AD1 (from other DAI)
1	Reserved	0011010 (0x1A)	SPT5_BD0_O	SPORT 5 Data BD0 (from other DAI)
1	Reserved	0011011 (0x1B)	SPT5_BD1_O	SPORT 5 Data BD1 (from other DAI)
1	Reserved	0011100 (0x1C)	SPT6_AD0_O	SPORT 6 Data AD0 (from other DAI)
1	Reserved	0011101 (0x1D)	SPT6_AD1_O	SPORT 6 Data AD1 (from other DAI)
1	Reserved	0011110 (0x1E)	SPT6_BD0_O	SPORT 6 Data BD0 (from other DAI)
1	Reserved	0011111 (0x1F)	SPT6_BD1_O	SPORT 6 Data BD1 (from other DAI)
1	Reserved	0100000 (0x20)	SPT4_ACLK_O	SPORT 4 Clock A (from other DAI)
1	Reserved	0100001 (0x21)	SPT4_BCLK_O	SPORT 4 Clock B (from other DAI)
1	Reserved	0100010 (0x22)	SPT5_ACLK_O	SPORT 5 Clock A (from other DAI)
1	Reserved	0100011 (0x23)	SPT5_BCLK_O	SPORT 5 Clock B (from other DAI)
1	Reserved	0100100 (0x24)	SPT6_ACLK_O	SPORT 6 Clock A (from other DAI)
1	Reserved	0100101 (0x25)	SPT6_BCLK_O	SPORT 6 Clock B (from other DAI)
1	Reserved	0100110 (0x26)	SPT4_AFS_O	SPORT 4 Frame Sync A (from other DAI)
1	Reserved	0100111 (0x27)	SPT4_BFS_O	SPORT 4 Frame Sync B (from other DAI)
1	Reserved	0101000 (0x28)	SPT5_AFS_O	SPORT 5 Frame Sync A (from other DAI)
1	Reserved	0101001 (0x29)	SPT5_BFS_O	SPORT 5 Frame Sync B (from other DAI)
1	Reserved	0101010 (0x2A)	SPT6_AFS_O	SPORT 6 Frame Sync A (from other DAI)
1	Reserved	0101011 (0x2B)	SPT6_BFS_O	SPORT 6 Frame Sync B (from other DAI)
1	Reserved	0101100 (0x2C)	SPT7_AD0_O	SPORT 7 Data AD0 (from other DAI)

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	0101101 (0x2D)	SPT7_AD1_O	SPORT 7 Data AD1 (from other DAI)
1	Reserved	0101110 (0x2E)	SPT7_BD0_O	SPORT 7 Data BD0 (from other DAI)
1	Reserved	0101111 (0x2F)	SPT7_BD1_O	SPORT 7 Data BD1 (from other DAI)
1	Reserved	0110000 (0x30)	Reserved	
1	Reserved	0110001 (0x31)	SPDIF1_TX_BLKSTART_O	SPDIF 1 TX Block Start Output
1	Reserved	0110010 (0x32)	Reserved	
1	Reserved	0110011 (0x33)	Reserved	
1	Reserved	0110100 (0x34)	SPT7_ACLK_O	SPORT 7 Clock A (from other DAI)
1	Reserved	0110101 (0x35)	SPT7_BCLK_O	SPORT 7 Clock B (from other DAI)
1	Reserved	0110110 (0x36)	SPT7_AFS_O	SPORT 7 Frame Sync A (from other DAI)
1	Reserved	0110111 (0x37)	SPT7_BFS_O	SPORT 7 Frame Sync B (from other DAI)
1	Reserved	0111000 (0x38)	PCG0_CLKC_O	Precision Clock C (from other DAI)
1	Reserved	0111001 (0x39)	PCG0_CLKD_O	Precision Clock D (from other DAI)
1	Reserved	0111010 (0x3A)	PCG0_FSC_O	Precision Frame Sync C (from other DAI)
1	Reserved	0111011 (0x3B)	PCG0_FSD_O	Precision Frame Sync D (from other DAI)
1	Reserved	0111100 (0x3C)	Reserved	
1	Reserved	0111101 (0x3D)	SRC4_DAT_OP_O	SRC0 Data Output (from other DAI)
1	Reserved	0111110 (0x3E)	SRC5_DAT_OP_O	SRC1 Data Output (from other DAI)
1	Reserved	0111111 (0x3F)	SRC6_DAT_OP_O	SRC2 Data Output (from other DAI)
1	Reserved	1000000 (0x40)	SRC7_DAT_OP_O	SRC3 Data Output (from other DAI)
1	Reserved	1000001 (0x41)	SPDIF1_RX_DAT_O	SPDIF 1 RX Data Output (from other DAI)
1	Reserved	1000010 (0x42)	SPDIF1_FS_O	SPDIF 1 RX Frame Sync Output (from other DAI)
1	Reserved	1000011 (0x43)	SPDIF1_RXCLK_O	SPDIF 1 RX Clock Output (from other DAI)
1	Reserved	1000100 (0x44)	SPDIF1_RX_TDMCLK_O	SPDIF 1 RX TDM Clock Output (from other DAI)
1	Reserved	1000101 (0x45)	SPDIF1_TX_O	SPDIF 1 TX Biphase Encoded Data Output (from other DAI)
1	Reserved	1000110 (0x46)	SPT4_ATDV_O	SPORT4 Transmit A Data Valid Output (from other DAI) (from other DAI)

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	1000111 (0x47)	SPT4_BTDV_O	SPORT4 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001000 (0x48)	SPT5_ATDV_O	SPORT5 Transmit A Data Valid Output (from other DAI)
1	Reserved	1001001 (0x49)	SPT5_BTDV_O	SPORT5 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001010 (0x4A)	SPT6_ATDV_O	SPORT6 Transmit A Data Valid Output (from other DAI)
1	Reserved	1001011 (0x4B)	SPT6_BTDV_O	SPORT6 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001100 (0x4C)	SPT7_ATDV_O	SPORT7 Transmit A Data Valid Output (from other DAI)
1	Reserved	1001101 (0x4D)	SPT7_BTDV_O	SPORT7 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001110 (0x4E) – 1010101 (0x55)	Reserved	
1	Reserved	1010110 (0x56)	PCG_CLKG_O	Precision Clock G (from other DAI)
1	Reserved	1010111 (0x57)	PCG_CLKH_O	Precision Clock H (from other DAI)
1	Reserved	1011000 (0x58)	PCG_FSG_O	Precision Frame Sync G (from other DAI)
1	Reserved	1011001 (0x59)	PCG_FSH_O	Precision Frame Sync H (from other DAI)
1	Reserved	1011010 (0x5A)	PCG_CLKC_INV_O	Inverted Precision Clock C Output (from other DAI)
1	Reserved	1011011 (0x5B)	PCG_CLKD_INV_O	Inverted Precision Clock D Output (from other DAI)
1	Reserved	1011100 (0x5C)	PCG_CLKG_INV_O	Inverted Precision Clock G Output (from other DAI)
1	Reserved	1011101 (0x5D)	PCG_CLKH_INV_O	Inverted Precision Clock H Output (from other DAI)
1	Reserved	1011110 (0x5E)	PCG_FSC_INV_O	Inverted Precision Frame Sync C Output (from other DAI)
1	Reserved	1011111 (0x5F)	PCG_FSD_INV_O	Inverted Precision Frame Sync D Output (from other DAI)
1	Reserved	1100000 (0x60)	PCG_FSG_INV_O	Inverted Precision Frame Sync G Output (from other DAI)

Table 29-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	1100001 (0x61)	PCG_FSH_INV_O	Inverted Precision Frame Sync H Output (from other DAI)
1	Reserved	1100010 (0x62)	PDM1_CLK0_O	PDM1 Clock0 Output (from other DAI)
1	Reserved	1100011 (0x63)	PDM1_SDATA_O	PDM1 Serial Data Output (from other DAI)
1	Reserved	1100100 (0x64)	SRC12_DAT_OP_O	SRC12 Data Output
1	Reserved	1100101 (0x65)	SRC13_DAT_OP_O	SRC13 Data Output
1	Reserved	1100110 (0x66)	SRC14_DAT_OP_O	SRC14 Data Output
1	Reserved	1100111 (0x67)	SRC15_DAT_OP_O	SRC15 Data Output
1	Reserved	1100100 (0x68) – 1111111 (0x7F)	Reserved	

Group E – Miscellaneous Source Signals

The following table lists the group E miscellaneous signal sources.

Table 29-14: Group E – Miscellaneous Signals

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	0	00000 (0x0)	DAI0_PB01_O	Pin Buffer 1 Output
0	0	00001 (0x1)	DAI0_PB02_O	Pin Buffer 2 Output
0	0	00010 (0x2)	DAI0_PB03_O	Pin Buffer 3 Output
0	0	00011 (0x3)	DAI0_PB04_O	Pin Buffer 4 Output
0	0	00100 (0x4)	DAI0_PB05_O	Pin Buffer 5 Output
0	0	00101 (0x5)	DAI0_PB06_O	Pin Buffer 6 Output
0	0	00110 (0x6)	DAI0_PB07_O	Pin Buffer 7 Output
0	0	00111 (0x7)	DAI0_PB08_O	Pin Buffer 8 Output
0	0	01000 (0x8)	DAI0_PB09_O	Pin Buffer 9 Output
0	0	01001 (0x9)	DAI0_PB10_O	Pin Buffer 10 Output
0	0	01010 (0xA)	DAI0_PB11_O	Pin Buffer 11 Output
0	0	01011 (0xB)	DAI0_PB12_O	Pin Buffer 12 Output

Table 29-14: Group E – Miscellaneous Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	01100 (0xC)	DAI0_PB13_O	Pin Buffer 13 Output
0	0	01101 (0xD)	DAI0_PB14_O	Pin Buffer 14 Output
0	0	01110 (0xE)	DAI0_PB15_O	Pin Buffer 15 Output
0	0	01111 (0xF)	DAI0_PB16_O	Pin Buffer 16 Output
0	0	10000 (0x10)	DAI0_PB17_O	Pin Buffer 17 Output
0	0	10001 (0x11)	DAI0_PB18_O	Pin Buffer 18 Output
0	0	10010 (0x12)	DAI0_PB19_O	Pin Buffer 19 Output
0	0	10011 (0x13)	DAI0_PB20_O	Pin Buffer 20 Output
0	0	10100 (0x14)	SPT0_AFS_O	SPORT 0 Frame Sync A
0	0	10101 (0x15)	SPT0_BFS_O	SPORT 0 Frame Sync B
0	0	10110 (0x16)	SPT1_AFS_O	SPORT 1 Frame Sync A
0	0	10111 (0x17)	SPT1_BFS_O	SPORT 1 Frame Sync B
0	0	11000 (0x18)	SPT2_AFS_O	SPORT 2 Frame Sync A
0	0	11001 (0x19)	SPT2_BFS_O	SPORT 2 Frame Sync B
0	0	11010 (0x1A)	SPDIF0_TX_BLKSTART_O	SPDIF 0 TX Block Start Output
0	0	11011 (0x1B)	PCG0_FSA_O	Precision Frame Sync A
0	0	11100 (0x1C)	PCG0_CLKB_O	Precision Clock B
0	0	11101 (0x1D)	PCG0_FSB_O	Precision Frame Sync B
0	0	11110 (0x1E)	LOW	Logic Level Low (0) as a Source
0	0	11111 (0x1F)	HIGH	Logic Level High (1) as a Source
0	1	00000 (0x0)	TRGS_PCGA_O	Trigger receiver mapped to PCGA
0	1	00001 (0x1)	TRGS_PCGB_O	Trigger receiver mapped to PCGB
0	1	00010 (0x2)	TRGS_PCGE_O	Trigger receiver mapped to PCGE
0	1	00011 (0x3)	TRGS_PCGF_O	Trigger receiver mapped to PCGF
0	1	00100 (0x4)	SPT3_AFS_O	SPORT 3 Frame Sync A
0	1	00101 (0x5)	SPT3_BFS_O	SPORT 3 Frame Sync B
0	1	00110 (0x6)	PCG_FSE_O	Precision Frame Sync E
0	1	00111 (0x7)	PCG_FSF_O	Precision Frame Sync F
0	1	01000 (0x8)-11111 (0x1F)	Reserved	
1	0	00000 (0x0)	DAI1_PB01_O	Pin Buffer 1 Output (from other DAI)

Table 29-14: Group E – Miscellaneous Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	0	00001 (0x1)	DAI1_PB02_O	Pin Buffer 2 Output (from other DAI)
1	0	00010 (0x2)	DAI1_PB03_O	Pin Buffer 3 Output (from other DAI)
1	0	00011 (0x3)	DAI1_PB04_O	Pin Buffer 4 Output (from other DAI)
1	0	00100 (0x4)	DAI1_PB05_O	Pin Buffer 5 Output (from other DAI)
1	0	00101 (0x5)	DAI1_PB06_O	Pin Buffer 6 Output (from other DAI)
1	0	00110 (0x6)	DAI1_PB07_O	Pin Buffer 7 Output (from other DAI)
1	0	00111 (0x7)	DAI1_PB08_O	Pin Buffer 8 Output (from other DAI)
1	0	01000 (0x8)	DAI1_PB09_O	Pin Buffer 9 Output (from other DAI)
1	0	01001 (0x9)	DAI1_PB10_O	Pin Buffer 10 Output (from other DAI)
1	0	01010 (0xA)	DAI1_PB11_O	Pin Buffer 11 Output (from other DAI)
1	0	01011 (0xB)	DAI1_PB12_O	Pin Buffer 12 Output (from other DAI)
1	0	01100 (0xC)	DAI1_PB13_O	Pin Buffer 13 Output (from other DAI)
1	0	01101 (0xD)	DAI1_PB14_O	Pin Buffer 14 Output (from other DAI)
1	0	01110 (0xE)	DAI1_PB15_O	Pin Buffer 15 Output (from other DAI)
1	0	01111 (0xF)	DAI1_PB16_O	Pin Buffer 16 Output (from other DAI)
1	0	10000 (0x10)	DAI1_PB17_O	Pin Buffer 17 Output (from other DAI)
1	0	10001 (0x11)	DAI1_PB18_O	Pin Buffer 18 Output (from other DAI)
1	0	10010 (0x12)	DAI1_PB19_O	Pin Buffer 19 Output (from other DAI)
1	0	10011 (0x13)	DAI1_PB20_O	Pin Buffer 20 Output (from other DAI)
1	0	10100 (0x14)	SPT4_AFS_O	SPORT 4 Frame Sync A (from other DAI)
1	0	10101 (0x15)	SPT4_BFS_O	SPORT 4 Frame Sync B (from other DAI)
1	0	10110 (0x16)	SPT5_AFS_O	SPORT 5 Frame Sync A (from other DAI)
1	0	10111 (0x17)	SPT5_BFS_O	SPORT 5 Frame Sync B (from other DAI)
1	0	11000 (0x18)	SPT6_AFS_O	SPORT 6 Frame Sync A (from other DAI)
1	0	11001 (0x19)	SPT6_BFS_O	SPORT 6 Frame Sync B (from other DAI)
1	0	11010 (0x1A)	SPDIF1_TX_BLKSTART_O	SPDIF 1 TX Block Start Output (from other DAI)
1	0	11011 (0x1B)	PCG_FSC_O	Precision Frame Sync C (from other DAI)
1	0	11100 (0x1C)	PCG_CLKD_O	Precision Clock D (from other DAI)
1	0	11101 (0x1D)	PCG_FSD_O	Precision Frame Sync D (from other DAI)

Table 29-14: Group E – Miscellaneous Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	0	11110 (0x1E)	Reserved	
1	0	11111 (0x1F)	Reserved	
1	1	00000 (0x0)	TRGS_PCGC_O	Trigger receiver mapped to PCGC
1	1	00001 (0x1)	TRGS_PCGD_O	Trigger receiver mapped to PCGD
1	1	00010 (0x2)	TRGS_PCGG_O	Trigger receiver mapped to PCGG
1	1	00011 (0x3)	TRGS_PCGH_O	Trigger receiver mapped to PCGH
1	1	00100 (0x4)	SPT7_AFS_O	SPORT 7 Frame Sync A
1	1	00101 (0x5)	SPT7_BFS_O	SPORT 7 Frame Sync B
1	1	00110 (0x6)	PCG_FSG_O	Precision Frame Sync G
1	1	00111 (0x7)	PCG_FSH_O	Precision Frame Sync H
1	1	01000 (0x8)-11111 (0x1F)	Reserved	

Group F – Pin Output Enable Source Signals

The group F pin output enable signals are listed in the following table.

Table 29-15: Group F – Pin Output Enable

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	000000 (0x0)	LOW	Logic Level Low (0)
0	Reserved	000001 (0x1)	HIGH	Logic Level High (1)
0	Reserved	000010 (0x2)	DAI0_MISCA0_O	DAI0 Miscellaneous Control A0 Output
0	Reserved	000011 (0x3)	DAI0_MISCA1_O	DAI0 Miscellaneous Control A1 Output
0	Reserved	000100 (0x4)	DAI0_MISCA2_O	DAI0 Miscellaneous Control A2 Output
0	Reserved	000101 (0x5)	DAI0_MISCA3_O	DAI0 Miscellaneous Control A3 Output
0	Reserved	000110 (0x6)	DAI0_MISCA4_O	DAI0 Miscellaneous Control A4 Output
0	Reserved	000111 (0x7)	DAI0_MISCA5_O	DAI0 Miscellaneous Control A5 Output
0	Reserved	001000 (0x8)	SPT0_ACLK_PBEN_O	SPORT 0 Clock A Output Enable
0	Reserved	001001 (0x9)	SPT0_AFS_PBEN_O	SPORT 0 Frame Sync A Output Enable
0	Reserved	001010 (0xA)	SPT0_AD0_PBEN_O	SPORT 0 Data AD0 Output Enable

Table 29-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	001011 (0xB)	SPT0_AD1_PBEN_O	SPORT 0 Data AD1 Output Enable
0	Reserved	001100 (0xC)	SPT0_BCLK_PBEN_O	SPORT 0 Clock B Output Enable
0	Reserved	001101 (0xD)	SPT0_BFS_PBEN_O	SPORT 0 Frame Sync B Output Enable
0	Reserved	001110 (0xE)	SPT0_BD0_PBEN_O	SPORT 0 Data BD0 Output Enable
0	Reserved	001111 (0xF)	SPT0_BD1_PBEN_O	SPORT 0 Data BD1 Output Enable
0	Reserved	010000 (0x10)	SPT1_ACLK_PBEN_O	SPORT 1 Clock A Output Enable
0	Reserved	010001 (0x11)	SPT1_AFS_PBEN_O	SPORT 1 Frame Sync A Output Enable
0	Reserved	010010 (0x12)	SPT1_AD0_PBEN_O	SPORT 1 Data AD0 Output Enable
0	Reserved	010011 (0x13)	SPT1_AD1_PBEN_O	SPORT 1 Data AD1 Output Enable
0	Reserved	010100 (0x14)	SPT1_BCLK_PBEN_O	SPORT 1 Clock B Output Enable
0	Reserved	010101 (0x15)	SPT1_BFS_PBEN_O	SPORT 1 Frame Sync B Output Enable
0	Reserved	010110 (0x16)	SPT1_BD0_PBEN_O	SPORT 1 Data BD0 Output Enable
0	Reserved	010111 (0x17)	SPT1_BD1_PBEN_O	SPORT 1 Data BD1 Output Enable
0	Reserved	011000 (0x18)	SPT2_ACLK_PBEN_O	SPORT 2 Clock A Output Enable
0	Reserved	011001 (0x19)	SPT2_AFS_PBEN_O	SPORT 2 Frame Sync A Output Enable
0	Reserved	011010 (0x1A)	SPT2_AD0_PBEN_O	SPORT 2 Data AD0 Output Enable
0	Reserved	011011 (0x1B)	SPT2_AD1_PBEN_O	SPORT 2 Data AD1 Output Enable
0	Reserved	011100 (0x1C)	SPT2_BCLK_PBEN_O	SPORT 2 Clock B Output Enable
0	Reserved	011101 (0x1D)	SPT2_BFS_PBEN_O	SPORT 2 Frame Sync B Output Enable
0	Reserved	011110 (0x1E)	SPT2_BD0_PBEN_O	SPORT 2 Data BD0 Output Enable
0	Reserved	011111 (0x1F)	SPT2_BD1_PBEN_O	SPORT 2 Data BD1 Output Enable
0	Reserved	100000 (0x20)	SPT3_ACLK_PBEN_O	SPORT 3 Clock A Output Enable
0	Reserved	100001 (0x21)	SPT3_AFS_PBEN_O	SPORT 3 Frame Sync A Output Enable
0	Reserved	100010 (0x22)	SPT3_AD0_PBEN_O	SPORT 3 Data AD0 Output Enable
0	Reserved	100011 (0x23)	SPT3_AD1_PBEN_O	SPORT 3 Data AD1 Output Enable
0	Reserved	100100 (0x24)	SPT3_BCLK_PBEN_O	SPORT 2 Clock B Output Enable
0	Reserved	100101 (0x25)	SPT3_BFS_PBEN_O	SPORT 3 Frame Sync B Output Enable
0	Reserved	100110 (0x26)	SPT3_BD0_PBEN_O	SPORT 3 Data BD0 Output Enable
0	Reserved	100111 (0x27)	SPT3_BD1_PBEN_O	SPORT 3 Data BD1 Output Enable
0	Reserved	101000 (0x28)	SPT0_ATDV_PBEN_O	SPORT 0 A Transmit Data Valid Output

Table 29-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	101001 (0x29)	SPT0_BTDV_PBEN_O	SPORT 0 B Transmit Data Valid Output
0	Reserved	101010 (0x2A)	SPT1_ATDV_PBEN_O	SPORT 1 A Transmit Data Valid Output
0	Reserved	101011 (0x2B)	SPT1_BTDV_PBEN_O	SPORT 1 B Transmit Data Valid Output
0	Reserved	101100 (0x2C)	SPT2_ATDV_PBEN_O	SPORT 2 A Transmit Data Valid Output
0	Reserved	101101 (0x2D)	SPT2_BTDV_PBEN_O	SPORT 2 B Transmit Data Valid Output
0	Reserved	100111 (0x2E)	SPT3_ATDV_PBEN_O	SPORT 3 A Transmit Data Valid Output
0	Reserved	101110 (0x2F)	SPT3_BTDV_PBEN_O	SPORT 3 B Transmit Data Valid Output
0	Reserved	110000 (0x30)	PDM0_CLK0_OE_O	PDM0 Clock0 Output Enable
0	Reserved	110001 (0x31)	PDM0_SDATA_OE_O	PDM0 Serial Data Output Enable
0	Reserved	110010 (0x32)–1111111 (0x3F)	Reserved	
1	Reserved	000000 (0x0)–000001 (0x1)	Reserved	
1	Reserved	000010 (0x2)	DAI1_MISCA0_O	DAI 1 Miscellaneous Control A0 Output
1	Reserved	000011 (0x3)	DAI1_MISCA1_O	DAI 1 Miscellaneous Control A1 Output
1	Reserved	000100 (0x4)	DAI1_MISCA2_O	DAI 1 Miscellaneous Control A2 Output
1	Reserved	000101 (0x5)	DAI1_MISCA3_O	DAI 1 Miscellaneous Control A3 Output
1	Reserved	000110 (0x6)	DAI1_MISCA4_O	DAI 1 Miscellaneous Control A4 Output
1	Reserved	000111 (0x7)	DAI1_MISCA5_O	DAI 1 Miscellaneous Control A5 Output
1	Reserved	001000 (0x8)	SPT4_ACLK_PBEN_O	SPORT 4 Clock A Output Enable (from other DAI)
1	Reserved	001001 (0x9)	SPT4_AFS_PBEN_O	SPORT 4 Frame Sync A Output Enable (from other DAI)
1	Reserved	001010 (0xA)	SPT4_AD0_PBEN_O	SPORT 4 Data AD0 Output Enable (from other DAI)
1	Reserved	001011 (0xB)	SPT4_AD1_PBEN_O	SPORT 4 Data AD1 Output Enable (from other DAI)
1	Reserved	001100 (0xC)	SPT4_BCLK_PBEN_O	SPORT 4 Clock B Output Enable (from other DAI)
1	Reserved	001101 (0xD)	SPT4_BFS_PBEN_O	SPORT 4 Frame Sync B Output Enable (from other DAI) (from other DAI)
1	Reserved	001110 (0xE)	SPT4_BD0_PBEN_O	SPORT 4 Data BD0 Output Enable (from other DAI)

Table 29-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	001111 (0xF)	SPT4_BD1_PBEN_O	SPORT 4 Data BD1 Output Enable (from other DAI)
1	Reserved	010000 (0x10)	SPT5_ACLK_PBEN_O	SPORT 5 Clock A Output Enable (from other DAI)
1	Reserved	010001 (0x11)	SPT5_AFS_PBEN_O	SPORT 5 Frame Sync A Output Enable (from other DAI)
1	Reserved	010010 (0x12)	SPT5_AD0_PBEN_O	SPORT 5 Data AD0 Output Enable (from other DAI)
1	Reserved	010011 (0x13)	SPT5_AD1_PBEN_O	SPORT 5 Data AD1 Output Enable (from other DAI)
1	Reserved	010100 (0x14)	SPT5_BCLK_PBEN_O	SPORT 5 Clock B Output Enable (from other DAI)
1	Reserved	010101 (0x15)	SPT5_BFS_PBEN_O	SPORT 5 Frame Sync B Output Enable (from other DAI)
1	Reserved	010110 (0x16)	SPT5_BD0_PBEN_O	SPORT 5 Data BD0 Output Enable (from other DAI)
1	Reserved	010111 (0x17)	SPT5_BD1_PBEN_O	SPORT 5 Data BD1 Output Enable (from other DAI)
1	Reserved	011000 (0x18)	SPT6_ACLK_PBEN_O	SPORT 6 Clock A Output Enable (from other DAI)
1	Reserved	011001 (0x19)	SPT6_AFS_PBEN_O	SPORT 6 Frame Sync A Output Enable (from other DAI)
1	Reserved	011010 (0x1A)	SPT6_AD0_PBEN_O	SPORT 6 Data AD0 Output Enable (from other DAI)
1	Reserved	011011 (0x1B)	SPT6_AD1_PBEN_O	SPORT 6 Data AD1 Output Enable (from other DAI)
1	Reserved	011100 (0x1C)	SPT6_BCLK_PBEN_O	SPORT 6 Clock B Output Enable (from other DAI)
1	Reserved	011101 (0x1D)	SPT6_BFS_PBEN_O	SPORT 6 Frame Sync B Output Enable (from other DAI)
1	Reserved	011110 (0x1E)	SPT6_BD0_PBEN_O	SPORT 6 Data BD0 Output Enable (from other DAI)
1	Reserved	011111 (0x1F)	SPT6_BD1_PBEN_O	SPORT 6 Data BD1 Output Enable (from other DAI)

Table 29-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	100000 (0x20)	SPT7_ACLK_PBEN_O	SPORT 7 Clock A Output Enable (from other DAI)
1	Reserved	100001 (0x21)	SPT7_AFS_PBEN_O	SPORT 7 Frame Sync A Output Enable (from other DAI)
1	Reserved	100010 (0x22)	SPT7_AD0_PBEN_O	SPORT 7 Data AD0 Output Enable (from other DAI)
1	Reserved	100011 (0x23)	SPT7_AD1_PBEN_O	SPORT 7 Data AD1 Output Enable (from other DAI)
1	Reserved	100100 (0x24)	SPT7_BCLK_PBEN_O	SPORT 6 Clock B Output Enable (from other DAI)
1	Reserved	100101 (0x25)	SPT7_BFS_PBEN_O	SPORT 7 Frame Sync B Output Enable (from other DAI)
1	Reserved	100110 (0x26)	SPT7_BD0_PBEN_O	SPORT 7 Data BD0 Output Enable (from other DAI)
1	Reserved	100111 (0x27)	SPT7_BD1_PBEN_O	SPORT 7 Data BD1 Output Enable (from other DAI)
1	Reserved	101000 (0x28)	SPT4_ATDV_PBEN_O	SPORT 4 A Transmit Data Valid Output (from other DAI)
1	Reserved	101001 (0x29)	SPT4_BTDV_PBEN_O	SPORT 4 B Transmit Data Valid Output (from other DAI)
1	Reserved	101010 (0x2A)	SPT5_ATDV_PBEN_O	SPORT 5 A Transmit Data Valid Output (from other DAI)
1	Reserved	101011 (0x2B)	SPT5_BTDV_PBEN_O	SPORT 5 B Transmit Data Valid Output (from other DAI)
1	Reserved	101100 (0x2C)	SPT6_ATDV_PBEN_O	SPORT 6 A Transmit Data Valid Output (from other DAI)
1	Reserved	101101 (0x2D)	SPT6_BTDV_PBEN_O	SPORT 6 B Transmit Data Valid Output (from other DAI)
1	Reserved	100111 (0x2E)	SPT7_ATDV_PBEN_O	SPORT 7 A Transmit Data Valid Output (from other DAI)
1	Reserved	101110 (0x2F)	SPT7_BTDV_PBEN_O	SPORT 7 B Transmit Data Valid Output (from other DAI)
1	Reserved	110000 (0x30)	PDM1_CLK0_OE_O	PDM1 Clock0 Output Enable (from other DAI)
1	Reserved	110001 (0x31)	PDM1_SDATA_OE_O	PDM1 Serial Data Output Enable

Table 29-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	Reserved	110010 (0x32)–1111111 (0x3F)	Reserved	

DAI Destination Registers Overview

The tables in the DAI Routing Capabilities section provide high level descriptions that illustrate source (output) connections to destinations (inputs) depending on the routing groups A through F. This sections lists the various input fields (INx), which are described in detail in the register descriptions section.

Table 29-16: Clock Destination Registers (Group A)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_CLK0	IN0	SPT0_ACLK_I	SPT4_ACLK_I	SPORT0A or SPORT4A Clock
	IN1	SPT0_BCLK_I	SPT4_BCLK_I	SPORT0B or SPORT4B Clock
	IN2	SPT1_ACLK_I	SPT5_ACLK_I	SPORT1A or SPORT5A Clock
	IN3	SPT1_BCLK_I	SPT5_BCLK_I	SPORT1B or SPORT5B Clock
	IN4	SPT2_ACLK_I	SPT6_ACLK_I	SPORT2A or SPORT6A Clock
	IN5	SPT2_BCLK_I	SPT6_BCLK_I	SPORT2B or SPORT6B Clock
DAI_CLK1	IN0	SRC0_CLK_IP_I	SRC4_CLK_IP_I	SRC0 or SRC4 Clock Input
	IN1	SRC0_CLK_OP_I	SRC4_CLK_OP_I	SRC0 or SRC4 Clock Output
	IN2	SRC1_CLK_IP_I	SRC5_CLK_IP_I	SRC1 or SRC5 Clock Input
	IN3	SRC1_CLK_OP_I	SRC5_CLK_OP_I	SRC1 or SRC5 Clock Output
	IN4	SRC2_CLK_IP_I	SRC6_CLK_IP_I	SRC2 or SRC6 Clock Input
	IN5	SRC2_CLK_OP_I	SRC6_CLK_OP_I	SRC2 or SRC6 Clock Output
DAI_CLK2	IN0	SRC3_CLK_IP_I	SRC7_CLK_IP_I	SRC3 or SRC7 Clock Input
	IN1	SRC3_CLK_OP_I	SRC7_CLK_OP_I	SRC3 or SRC7 Clock Output
	IN2	SPDIF0_TX_CLK_I	SPDIF1_TX_CLK_I	SPDIF0 or SPDIF1 TX Clock
	IN3	PDM0_CLK0_I	PDM1_CLK0_I	PDM0 or PDM1 Clock0 Input
	IN4	PDM0_BCLK_I	PDM1_BCLK_I	PDM0 or PDM1 Bit Clock Input
DAI_CLK3	IN5	SPDIF0_TX_HFCLK_I	SPDIF1_TX_HFCLK_I	SPDIF0 or SPDIF1 TX HF Clock

Table 29-16: Clock Destination Registers (Group A) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_CLK4	IN0	PCG0_EXTCLKA_I	PCG0_EXTCLKC_I	PCG0 External Clock A or C
	IN1	PCG0_EXTCLKB_I	PCG0_EXTCLKD_I	PCG0 External Clock B or D
	IN3	SPDIF0_TX_EXT_SYNC_I	SPDIF1_TX_EXT_SYNC_I	SPDIF0 or SPDIF1 TX External Sync
	IN4	PCG0_SYNC_CLKA_I	PCG0_SYNC_CLKC_I	PCG0 Sync Clock A or C
	IN5	PCG0_SYNC_CLKB_I	PCG0_SYNC_CLKD_I	PCG0 Sync Clock B or D
DAI_CLK5	IN0	SPT3_ACLK_I	SPT7_ACLK_I	SPORT3A or SPORT 7A Clock
	IN1	SPT3_BCLK_I	SPT7_BCLK_I	SPORT3B or SPORT 7B Clock
	IN2	PCG0_SYNC_CLKE_I	PCG0_SYNC_CLKG_I	PCG0 Sync Clock E or G
	IN3	PCG0_SYNC_CLKF_I	PCG0_SYNC_CLKH_I	PCG0 Sync Clock F or H
	IN4	PCG0_EXTCLKE_I	PCG0_EXTCLKG_I	PCG0 External Clock E or G
	IN5	PCG0_EXTCLKF_I	PCG0_EXTCLKH_I	PCG0 External Clock F or H
DAI_CLK6	IN0	SRC8_CLK_IP_I	SRC12_CLK_IP_I	SRC8 or SRC12 Clock Input
	IN1	SRC8_CLK_OP_I	SRC12_CLK_OP_I	SRC8 or SRC12 Clock Output
	IN2	SRC9_CLK_IP_I	SRC13_CLK_IP_I	SRC9 or SRC13 Clock Input
	IN3	SRC9_CLK_OP_I	SRC13_CLK_OP_I	SRC9 or SRC13 Clock Output
	IN4	SRC10_CLK_IP_I	SRC14_CLK_IP_I	SRC10 or SRC14 Clock Input
	IN5	SRC10_CLK_OP_I	SRC14_CLK_OP_I	SRC10 or SRC14 Clock Output
DAI_CLK7	IN0	SRC11_CLK_IP_I	SRC15_CLK_IP_I	SRC11 or SRC15 Clock Input
	IN1	SRC11_CLK_OP_I	SRC15_CLK_OP_I	SRC11 or SRC15 Clock Output

Table 29-17: Data Destination Registers (Group B)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_DAT0	IN0	SPT0_AD0_I	SPT4_AD0_I	SPORT0A or SPORT4A Primary Data
	IN1	SPT0_AD1_I	SPT4_AD1_I	SPORT0A or SPORT4A Secondary Data
	IN2	SPT0_BD0_I	SPT4_BD0_I	SPORT0B or SPORT4B Primary Data
	IN3	SPT0_BD1_I	SPT4_BD1_I	SPORT0B or SPORT4B Secondary Data
	IN4	SPT1_AD0_I	SPT5_AD0_I	SPORT1A or SPORT5A Primary Data

Table 29-17: Data Destination Registers (Group B) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_DAT1	IN0	SPT1_AD1_I	SPT5_AD1_I	SPORT1A or SPORT5A Secondary Data
	IN1	SPT1_BD0_I	SPT5_BD0_I	SPORT1B or SPORT5B Primary Data
	IN2	SPT1_BD1_I	SPT5_BD1_I	SPORT1B or SPORT5B Secondary Data
	IN3	SPT2_AD0_I	SPT6_AD0_I	SPORT2A or SPORT6A Primary Data
	IN4	SPT2_AD1_I	SPT6_AD1_I	SPORT2A or SPORT6A Secondary Data
DAI_DAT2	IN0	SPT2_BD0_I	SPT6_BD0_I	SPORT2B or SPORT6B Primary Data
	IN1	SPT2_BD1_I	SPT6_BD1_I	SPORT2B or SPORT6B Secondary Data
	IN2	SRC0_DAT_IP_I	SRC4_DAT_IP_I	SRC0 or SRC4 Data
	IN3	SRC1_DAT_IP_I	SRC5_DAT_IP_I	SRC1 or SRC5 Data
	IN4	SRC2_DAT_IP_I	SRC6_DAT_IP_I	SRC2 or SRC6 Data
DAI_DAT3	IN0	SRC3_DAT_IP_I	SRC7_DAT_IP_I	SRC3 or SRC7 Data
	IN1	SRC0_DAT_TDM_OP_I	SRC4_DAT_TDM_OP_I	SRC0 or SRC4 TDM output port
	IN2	SRC1_DAT_TDM_OP_I	SRC5_DAT_TDM_OP_I	SRC1 or SRC5 TDM Output port Data
	IN3	SRC2_DAT_TDM_OP_I	SRC6_DAT_TDM_OP_I	SRC2 or SRC6 TDM Output port Data
	IN4	SRC3_DAT_TDM_OP_I	SRC7_DAT_TDM_OP_I	SRC3 or SRC7 TDM Output port Data
DAI_DAT4	IN0	SPDIF0_TX_DAT_I	SPDIF1_TX_DAT_I	SPDIF0 or SPDIF1 serial transmitter Data
	IN1	PDM0_DAT0_I	PDM1_DAT0_I	PDM0 or PDM1 Data0 Input
	IN2	PDM0_DAT1_I	PDM1_DAT1_I	PDM0 or PDM1 Data1 Input
DAI_DAT5	IN4	SPDIF0_RX_I	SPDIF1_RX_I	SPDIF0 or SPDIF1 receiver bi-phase Data

Table 29-17: Data Destination Registers (Group B) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_DAT6	IN0	SPT3_AD0_I	SPT7_AD0_I	SPORT3A or SPORT7A Primary Data
	IN1	SPT3_AD1_I	SPT7_AD1_I	SPORT3A or SPORT7A Secondary Data
	IN2	SPT3_BD0_I	SPT7_BD0_I	SPORT3B or SPORT7B Primary Data
	IN3	SPT3_BD1_I	SPT7_BD1_I	SPORT3B or SPORT7B Secondary Data
	IN4	Reserved	Reserved	
DAI_DAT7	IN0	SRC8_DAT_IP_I	SRC12_DAT_IP_I	SRC8 or SRC12 Data
	IN1	SRC9_DAT_IP_I	SRC13_DAT_IP_I	SRC9 or SRC13 Data
	IN2	SRC10_DAT_IP_I	SRC14_DAT_IP_I	SRC10 or SRC14 Data
	IN3	SRC11_DAT_IP_I	SRC15_DAT_IP_I	SRC11 or SRC15 Data
	IN4	SRC8_DAT_TDM_OP_I	SRC12_DAT_TDM_OP_I	SRC8 or SRC12 TDM Output port
DAI_DAT8	IN0	SRC9_DAT_TDM_OP_I	SRC13_DAT_TDM_OP_I	SRC9 or SRC13 TDM Output port Data
	IN1	SRC10_DAT_TDM_OP_I	SRC14_DAT_TDM_OP_I	SRC10 or SRC14 TDM Output port Data
	IN2	SRC11_DAT_TDM_OP_I	SRC15_DAT_TDM_OP_I	SRC11 or SRC15 TDM Output port Data

Table 29-18: Frame Sync Destination Registers (Group C)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_FS0	IN0	SPT0_AFS_I	SPT4_AFS_I	SPORT0A or SPORT4A Frame Sync
	IN1	SPT0_BFS_I	SPT4_BFS_I	SPORT0B or SPORT4B Frame Sync
	IN2	SPT1_AFS_I	SPT5_AFS_I	SPORT1A or SPORT5A Frame Sync
	IN3	SPT1_BFS_I	SPT5_BFS_I	SPORT1B or SPORT5B Frame Sync
	IN4	SPT2_AFS_I	SPT6_AFS_I	SPORT2A or SPORT6A Frame Sync
	IN5	SPT2_BFS_I	SPT6_BFS_I	SPORT2B or SPORT6B Frame Sync
DAI_FS1	IN0	SRC0_FS_IP_I	SRC4_FS_IP_I	SRC0 or SRC4 Frame Sync Input
	IN1	SRC0_FS_OP_I	SRC4_FS_OP_I	SRC0 or SRC4 Frame Sync Output
	IN2	SRC1_FS_IP_I	SRC5_FS_IP_I	SRC1 or SRC5 Frame Sync Input
	IN3	SRC1_FS_OP_I	SRC5_FS_OP_I	SRC1 or SRC5 Frame Sync Output
	IN4	SRC2_FS_IP_I	SRC6_FS_IP_I	SRC2 or SRC6 Frame Sync Input

Table 29-18: Frame Sync Destination Registers (Group C) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
	IN5	SRC2_FS_OP_I	SRC6_FS_OP_I	SRC2 or SRC6 Frame Sync Output
DAI_FS2	IN0	SRC3_FS_IP_I	SRC7_FS_IP_I	SRC3 or SRC7 Frame Sync Input
	IN1	SRC3_FS_OP_I	SRC7_FS_OP_I	SRC3 or SRC7 Frame Sync Output
	IN2	SPDIF0_TX_FS_I	SPDIF1_TX_FS_I	SPDIF0 or SPDIF1 Transmit Frame Sync
DAI_FS4	IN0	SPT3_AFS_I	SPT7_AFS_I	SPORT3A or SPORT7A Frame Sync
	IN1	SPT3_BFS_I	SPT7_BFS_I	SPORT3B or SPORT7B Frame Sync
	IN2	TM0_ACI14_I	TM0_ACI15_I	ACI 14 or ACI 15 Timer Input
	IN3	PDM0_LRCLK_I	PDM1_LRCLK_I	PDM0 or PDM1 Left/Right Clock or Frame Sync Input
DAI_FS5	IN0	SRC8_FS_IP_I	SRC12_FS_IP_I	SRC8 or SRC12 Frame Sync Input
	IN1	SRC8_FS_OP_I	SRC12_FS_OP_I	SRC8 or SRC12 Frame Sync Output
	IN2	SRC9_FS_IP_I	SRC13_FS_IP_I	SRC9 or SRC13 Frame Sync Input
	IN3	SRC9_FS_OP_I	SRC13_FS_OP_I	SRC9 or SRC13 Frame Sync Output
	IN4	SRC10_FS_IP_I	SRC14_FS_IP_I	SRC10 or SRC14 Frame Sync Input
	IN5	SRC10_FS_OP_I	SRC14_FS_OP_I	SRC10 or SRC14 Frame Sync Output
DAI_FS6	IN0	SRC11_FS_IP_I	SRC15_FS_IP_I	SRC11 or SRC15 Frame Sync Input
	IN1	SRC11_FS_OP_I	SRC15_FS_OP_I	SRC11 or SRC15 Frame Sync Output

Table 29-19: Pin Buffer Assignment Destination Registers (Group D)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_PIN0	IN0	DAI0_PB01_I	DAI1_PB01_I	Pin Buffer 1
	IN1	DAI0_PB02_I	DAI1_PB02_I	Pin Buffer 2
	IN2	DAI0_PB03_I	DAI1_PB03_I	Pin Buffer 3
	IN3	DAI0_PB04_I	DAI1_PB04_I	Pin Buffer 4
DAI_PIN1	IN0	DAI0_PB05_I	DAI1_PB05_I	Pin Buffer 5
	IN1	DAI0_PB06_I	DAI1_PB06_I	Pin Buffer 6
	IN2	DAI0_PB07_I	DAI1_PB07_I	Pin Buffer 7
	IN3	DAI0_PB08_I	DAI1_PB08_I	Pin Buffer 8

Table 29-19: Pin Buffer Assignment Destination Registers (Group D) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_PIN2	IN0	DAI0_PB09_I	DAI1_PB09_I	Pin Buffer 9
	IN1	DAI0_PB10_I	DAI1_PB10_I	Pin Buffer 10
	IN2	DAI0_PB11_I	DAI1_PB11_I	Pin Buffer 11
	IN3	DAI0_PB12_I	DAI1_PB12_I	Pin Buffer 12
DAI_PIN3	IN0	DAI0_PB13_I	DAI1_PB13_I	Pin Buffer 13
	IN1	DAI0_PB14_I	DAI1_PB14_I	Pin Buffer 14
	IN2	DAI0_PB15_I	DAI1_PB15_I	Pin Buffer 15
	IN3	DAI0_PB16_I	DAI1_PB16_I	Pin Buffer 16
DAI_PIN4	IN0	DAI0_PB17_I	DAI1_PB17_I	Pin Buffer 17
	IN1	DAI0_PB18_I	DAI1_PB18_I	Pin Buffer 18
	IN2	DAI0_PB19_I	DAI1_PB19_I	Pin Buffer 19
	IN3	DAI0_PB20_I	DAI1_PB20_I	Pin Buffer 20
	IN4	INV_DAI0_PB19_I	INV_DAI1_PB19_I	Inverted Pin Buffer 19
	IN5	INV_DAI0_PB20_I	INV_DAI1_PB20_I	Inverted Pin Buffer 20
DAI_PIN5	IN0	DAI0_DUMMY_PB01_1	DAI1_DUMMY_PB01_1	Dummy Pin Buffer 1
	IN1	DAI0_DUMMY_PB02_1	DAI1_DUMMY_PB02_1	Dummy Pin Buffer 1
	IN2	DAI0_DUMMY_PB03_1	DAI1_DUMMY_PB03_1	Dummy Pin Buffer 1
	IN3	DAI0_DUMMY_PB04_1	DAI1_DUMMY_PB04_1	Dummy Pin Buffer 1

Miscellaneous Control Destination Registers (Group E)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_MISC0	IN6	DAI0_MISCA0_I/ DAI0_INT_6_I	DAI1_MISCA0_I/ DAI1_INT_6_I	DAIx miscellaneous A0 input, or DAIx interrupt 28
	IN7	DAI0_MISCA1_I/ DAI0_INT_7_I	DAI1_MISCA1_I/ DAI1_INT_7_I	DAIx miscellaneous A1 input, or DAIx interrupt 29
	IN8	DAI0_MISCA2_I/ DAI0_INT_8_I	DAI1_MISCA2_I/ DAI1_INT_8_I	DAIx miscellaneous A2 input, or DAIx interrupt 30
	IN9	DAI0_MISCA3_I/ DAI0_INT_9_I	DAI1_MISCA3_I/ DAI1_INT_9_I	DAIx miscellaneous A3 input, or DAIx interrupt 31
DAI_MISC1	IN0	DAI0_INT_0	DAI1_INT_0	DAIx interrupt 22

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
	IN1	DAI0_INT_1	DAI1_INT_1	DAIx interrupt 23
	IN2	DAI0_INT_2	DAI1_INT_2	DAIx interrupt 24
	IN3	DAI0_INT_3	DAI1_INT_3	DAIx interrupt 25
	IN4	DAI0_INT_4	DAI1_INT_4	DAIx interrupt 26
	IN5	DAI0_INT_5	DAI1_INT_5	DAIx interrupt 27
DAI_MISC2	IN0	PCG_HWA_TRIG_I	PCG_HWC_TRIG_I	PCG hardware trigger input A or C
	IN1	PCG_HWB_TRIG_I	PCG_HWD_TRIG_I	PCG hardware trigger input B or D
	IN2	PCG_HWE_TRIG_I	PCG_HWG_TRIG_I	PCG hardware trigger input E or G
	IN3	PCG_HWF_TRIG_I	PCG_HWH_TRIG_I	PCG hardware trigger input A or C

Table 29-20: Pin Buffer Enable Destination Registers (Group F)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_PBEN0	IN0	DAI0_PBEN01_I	DAI1_PBEN01_I	Pin Buffer 1 Enable
	IN1	DAI0_PBEN02_I	DAI1_PBEN02_I	Pin Buffer 2 Enable
	IN2	DAI0_PBEN03_I	DAI1_PBEN03_I	Pin Buffer 3 Enable
	IN3	DAI0_PBEN04_I	DAI1_PBEN04_I	Pin Buffer 4 Enable
	IN4	DAI0_PBEN05_I	DAI1_PBEN05_I	Pin Buffer 5 Enable
DAI_PBEN1	IN0	DAI0_PBEN06_I	DAI1_PBEN06_I	Pin Buffer 6 Enable
	IN1	DAI0_PBEN07_I	DAI1_PBEN07_I	Pin Buffer 7 Enable
	IN2	DAI0_PBEN08_I	DAI1_PBEN08_I	Pin Buffer 8 Enable
	IN3	DAI0_PBEN09_I	DAI1_PBEN09_I	Pin Buffer 9 Enable
	IN4	DAI0_PBEN10_I	DAI1_PBEN10_I	Pin Buffer 10 Enable
DAI_PBEN2	IN0	DAI0_PBEN11_I	DAI1_PBEN11_I	Pin Buffer 11 Enable
	IN1	DAI0_PBEN12_I	DAI1_PBEN12_I	Pin Buffer 12 Enable
	IN2	DAI0_PBEN13_I	DAI1_PBEN13_I	Pin Buffer 13 Enable
	IN3	DAI0_PBEN14_I	DAI1_PBEN14_I	Pin Buffer 14 Enable
	IN4	DAI0_PBEN15_I	DAI1_PBEN15_I	Pin Buffer 15 Enable

Table 29-20: Pin Buffer Enable Destination Registers (Group F) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_PBEN3	IN0	DAI0_PBEN16_I	DAI1_PBEN16_I	Pin Buffer 16 Enable
	IN1	DAI0_PBEN17_I	DAI1_PBEN17_I	Pin Buffer 17 Enable
	IN2	DAI0_PBEN18_I	DAI1_PBEN18_I	Pin Buffer 18 Enable
	IN3	DAI0_PBEN19_I	DAI1_PBEN19_I	Pin Buffer 19 Enable
	IN4	DAI0_PBEN20_I	DAI1_PBEN20_I	Pin Buffer 20 Enable

The ***Interrupt Events*** table presents the information for the following registers.

- DAI_IMSK_PRI (Core Interrupt Priority Assignment) register
- DAI_IMSK_FE (Falling-Edge Interrupt Mask) and DAI_IMSK_RE (Rising-Edge Interrupt Mask) registers
- DAI_IRPTL_H (High-Priority Interrupt Latch) and DAI_IRPTL_L (Low-Priority Interrupt Latch) registers
- DAI_IRPTL_HS (Shadow High Interrupt Latch) and DAI_IRPTL_LS (Shadow Low Interrupt Latch) registers

Table 29-21: Interrupt Events

Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
RXVALID	SPDIF0_RXVALID_INT	SPDIF1_RXVALID_INT	SPDIF0 or SPDIF1 RX Valid interrupt
RXLOCK	SPDIF0_RXLOCK_INT	SPDIF1_RXLOCK_INT	SPDIF0 or SPDIF1 RX Lock interrupt
RXLOSSOFLOCK	SPDIF0_RXLOSSOFLOCK_INT	SPDIF1_RXLOSSOFLOCK_INT	SPDIF0 or SPDIF1 RX Loss of Lock interrupt
RXNONAUDIO	SPDIF0_RXNONAUDIO_INT	SPDIF1_RXNONAUDIO_INT	SPDIF0 or SPDIF1 RX Non-audio interrupt
SRC0MUTE	SRC0_MUTE_INT	SRC4_MUTE_INT	DAI0 or DAI1 ASRC0 Mute interrupt
SRC1MUTE	SRC1_MUTE_INT	SRC5_MUTE_INT	DAI0 or DAI1 ASRC1 Mute interrupt
SRC2MUTE	SRC2_MUTE_INT	SRC6_MUTE_INT	DAI0 or DAI1 ASRC2 Mute interrupt
SRC3MUTE	SRC3_MUTE_INT	SRC7_MUTE_INT	DAI0 or DAI1 ASRC3 Mute interrupt
MISCINT0	DAI0_INT_00	DAI1_INT_00	DAI0 or DAI1 Miscellaneous Interrupt 0

Table 29-21: Interrupt Events (Continued)

Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
MISCINT1	DAI0_INT_01	DAI1_INT_01	DAI0 or DAI1 Miscellaneous Interrupt 1
MISCINT2	DAI0_INT_02	DAI1_INT_02	DAI0 or DAI1 Miscellaneous Interrupt 2
MISCINT3	DAI0_INT_03	DAI1_INT_03	DAI0 or DAI1 Miscellaneous Interrupt 3
MISCINT4	DAI0_INT_04	DAI1_INT_04	DAI0 or DAI1 Miscellaneous Interrupt 4
MISCINT5	DAI0_INT_05	DAI1_INT_05	DAI0 or DAI1 Miscellaneous Interrupt 5
MISCINT6	DAI0_EXTMISCA0_INT/ DAI0_INT_06	DAI1_EXTMISCA0_INT/ DAI1_INT_06	DAI0 or DAI1 Miscellaneous Interrupt 6 / External Miscellaneous A0 Interrupt
MISCINT7	DAI0_EXTMISCA1_INT/ DAI0_INT_07	DAI1_EXTMISCA1_INT/ DAI1_INT_07	DAI0 or DAI1 Miscellaneous Interrupt 7 / External Miscellaneous A1 Interrupt
MISCINT8	DAI0_EXTMISCA2_INT/ DAI0_INT_08	DAI1_EXTMISCA2_INT/ DAI1_INT_08	DAI0 or DAI1 Miscellaneous Interrupt 8 / External Miscellaneous A2 Interrupt
MISCINT9	DAI0_EXTMISCA3_INT/ DAI0_INT_09	DAI1_EXTMISCA3_INT/ DAI1_INT_09	DAI0 or DAI1 Miscellaneous Interrupt 9 / External Miscellaneous A3 Interrupt

SPORT Grouping with DAI

DAI supports two registers `DAI_GBL_SP_EN` and `DAI_GBL_INT_EN` to group enabling/disabling and interrupt/trigger generation/reception for multiple SPORTs.

For more information, refer to the Grouping of SPORTs section in the Serial Port (SPORT) chapter.

Global PCG Enable

The DAI module includes the `DAI_GBL_PCG_EN` register that enables and disables multiple PCGs globally at the same time. For more information, see Global PCG Enable section in the PCG chapter.

ADSP-2184x DAI Register Descriptions

The Digital Audio Interface (DAIn) registers are used to configure DAI destinations for the DAI sources shown in the Group A - F Routing tables. (DAI) contains the following registers.

Table 29-22: ADSP-2184x DAI Register List

Name	Description
DAI_CLK0	Clock Routing Control Register 0
DAI_CLK1	Clock Routing Control Register 1
DAI_CLK2	Clock Routing Control Register 2
DAI_CLK3	Clock Routing Control Register 3
DAI_CLK4	Clock Routing Control Register 4
DAI_CLK5	Clock Routing Control Register 5
DAI_CLK6	Clock Routing Control Register 6
DAI_CLK7	Clock Routing Control Register 7
DAI_DAI_MISC_CTL	DAI MISC Control Register
DAI_DAT0	Serial Data Routing Control Register 0
DAI_DAT1	Serial Data Routing Control Register 1
DAI_DAT2	Serial Data Routing Control Register 2
DAI_DAT3	Serial Data Routing Control Register 3
DAI_DAT4	Serial Data Routing Control Register 4
DAI_DAT5	Serial Data Routing Control Register 5
DAI_DAT6	Serial Data Routing Control Register 6
DAI_DAT7	Serial Data Routing Control Register 7
DAI_DAT8	Serial Data Routing Control Register 8
DAI_EXTD_CLK0	Extended Clock Routing Control Register 0
DAI_EXTD_CLK1	Extended Clock Routing Control Register 1
DAI_EXTD_CLK2	Extended Clock Routing Control Register 2
DAI_EXTD_CLK3	Extended Clock Routing Control Register 3
DAI_EXTD_CLK4	Extended Clock Routing Control Register 4
DAI_EXTD_CLK5	Extended Clock Routing Control Register 5
DAI_EXTD_CLK6	Extended Clock Routing Control Register 6
DAI_EXTD_CLK7	Extended Clock Routing Control Register 7
DAI_EXTD_DAT0	Extended Serial Data Routing Control Register 0
DAI_EXTD_DAT1	Extended Serial Data Routing Control Register 1
DAI_EXTD_DAT2	Extended Serial Data Routing Control Register 2
DAI_EXTD_DAT3	Extended Serial Data Routing Control Register 3
DAI_EXTD_DAT4	Extended Serial Data Routing Control Register 4

Table 29-22: ADSP-2184x DAI Register List (Continued)

Name	Description
DAI_EXTD_DAT5	Extended Serial Data Routing Control Register 5
DAI_EXTD_DAT6	Extended Serial Data Routing Control Register 6
DAI_EXTD_DAT7	Extended Serial Data Routing Control Register 7
DAI_EXTD_DAT8	Extended Serial Data Routing Control Register 8
DAI_EXTD_FS0	Extended Frame Sync Routing Control Register 0
DAI_EXTD_FS1	Extended Frame Sync Routing Control Register 1
DAI_EXTD_FS2	Extended Frame Sync Routing Control Register 2
DAI_EXTD_FS4	Extended Frame Sync Routing Control Register 4
DAI_EXTD_FS5	Extended Frame Sync Routing Control Register 5
DAI_EXTD_FS6	Extended Frame Sync Routing Control Register 6
DAI_EXTD_MISC0	Extended Miscellaneous Control Register 0
DAI_EXTD_MISC1	Extended Miscellaneous Control Register 1
DAI_EXTD_MISC2	Extended Miscellaneous Control Register 2
DAI_EXTD_PBEN0	Extended Pin Buffer Enable Register 0
DAI_EXTD_PBEN1	Extended Pin Buffer Enable Register 1
DAI_EXTD_PBEN2	Extended Pin Buffer Enable Register 2
DAI_EXTD_PBEN3	Extended Pin Buffer Enable Register 3
DAI_EXTD_PIN0	Extended Pin Buffer Assignment Register 0
DAI_EXTD_PIN1	Extended Pin Buffer Assignment Register 1
DAI_EXTD_PIN2	Extended Pin Buffer Assignment Register 2
DAI_EXTD_PIN3	Extended Pin Buffer Assignment Register 3
DAI_EXTD_PIN4	Extended Pin Buffer Assignment Register 4
DAI_FS0	Frame Sync Routing Control Register 0
DAI_FS1	Frame Sync Routing Control Register 1
DAI_FS2	Frame Sync Routing Control Register 2
DAI_FS4	Frame Sync Routing Control Register 4
DAI_FS5	Frame Sync Routing Control Register 5
DAI_FS6	Frame Sync Routing Control Register 6
DAI_GBL_INT_EN	Global SPORT Interrupt Grouping Register
DAI_GBL_PCG_EN	Global PCG Enable Control Register
DAI_GBL_SP_EN	Global SPORT Enable Register

Table 29-22: ADSP-2184x DAI Register List (Continued)

Name	Description
DAI_IMSK_FE	Falling-Edge Interrupt Mask Register
DAI_IMSK_PRI	Core Interrupt Priority Assignment Register
DAI_IMSK_RE	Rising-Edge Interrupt Mask Register
DAI_IRPTL_H	High Priority Interrupt Latch Register
DAI_IRPTL_HS	Shadow High Priority Interrupt Latch Register
DAI_IRPTL_L	Low Priority Interrupt Latch Register
DAI_IRPTL_LS	Shadow Low Priority Interrupt Latch Register
DAI_MISC0	Miscellaneous Control Register 0
DAI_MISC1	Miscellaneous Control Register 1
DAI_MISC2	Miscellaneous Control Register 1
DAI_PBEN0	Pin Buffer Enable Register 0
DAI_PBEN1	Pin Buffer Enable Register 1
DAI_PBEN2	Pin Buffer Enable Register 2
DAI_PBEN3	Pin Buffer Enable Register 3
DAI_PIN0	Pin Buffer Assignment Register 0
DAI_PIN1	Pin Buffer Assignment Register 1
DAI_PIN2	Pin Buffer Assignment Register 2
DAI_PIN3	Pin Buffer Assignment Register 3
DAI_PIN4	Pin Buffer Assignment Register 4
DAI_PIN5	Dummy Pin Buffer Assignment Register
DAI_PIN_STAT	Pin Status Register

Clock Routing Control Register 0

The DAI_CLK0 register provides clock routing connections for the serial ports (SPORTs).

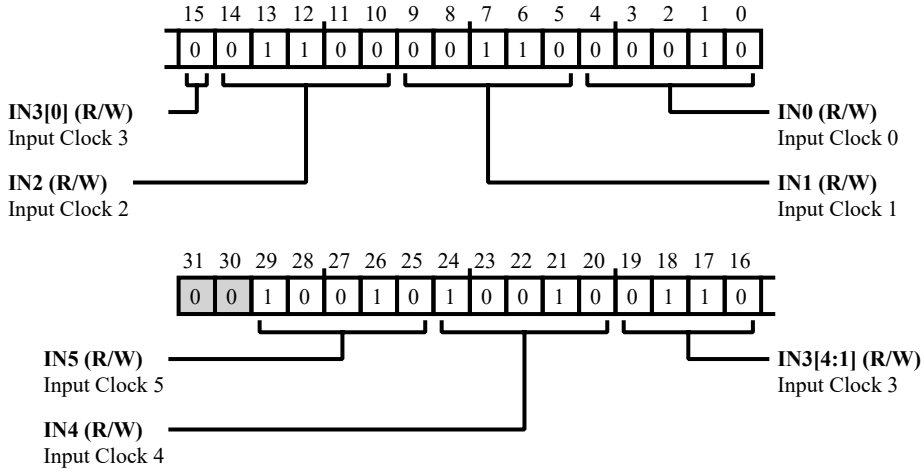


Figure 29-16: DAI_CLK0 Register Diagram

Table 29-23: DAI_CLK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK0.IN5 holds the source signal assignment that will be routed to the DAI_CLK0.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK0.IN4 holds the source signal assignment that will be routed to the DAI_CLK0.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK0.IN3 holds the source signal assignment that will be routed to the DAI_CLK0.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK0.IN2 holds the source signal assignment that will be routed to the DAI_CLK0.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK0.IN1 holds the source signal assignment that will be routed to the DAI_CLK0.IN1 destination. Refer to the Group A Signals table for source and destination mappings.

Table 29-23: DAI_CLK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK0.IN0` holds the source signal assignment that will be routed to the `DAI_CLK0.IN0` destination. Refer to the Group A Signals table for source and destination mappings.

Clock Routing Control Register 1

The DAI_CLK1 register provides clock routing connections for the asynchronous sample rate converters (ASRC).

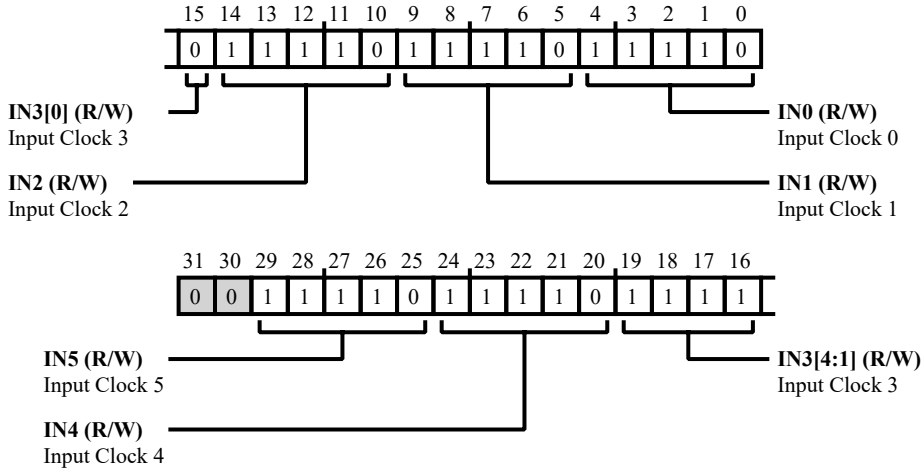


Figure 29-17: DAI_CLK1 Register Diagram

Table 29-24: DAI_CLK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK1.IN5 holds the source signal assignment that will be routed to the DAI_CLK1.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK1.IN4 holds the source signal assignment that will be routed to the DAI_CLK1.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK1.IN3 holds the source signal assignment that will be routed to the DAI_CLK1.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK1.IN2 holds the source signal assignment that will be routed to the DAI_CLK1.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK1.IN1 holds the source signal assignment that will be routed to the DAI_CLK1.IN1 destination. Refer to the Group A Signals table for source and destination mappings.

Table 29-24: DAI_CLK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK1.IN0` holds the source signal assignment that will be routed to the `DAI_CLK1.IN0` destination. Refer to the Group A Signals table for source and destination mappings.

Clock Routing Control Register 2

The DAI_CLK2 register provides clock routing connections for the S/PDIF and ASRC.

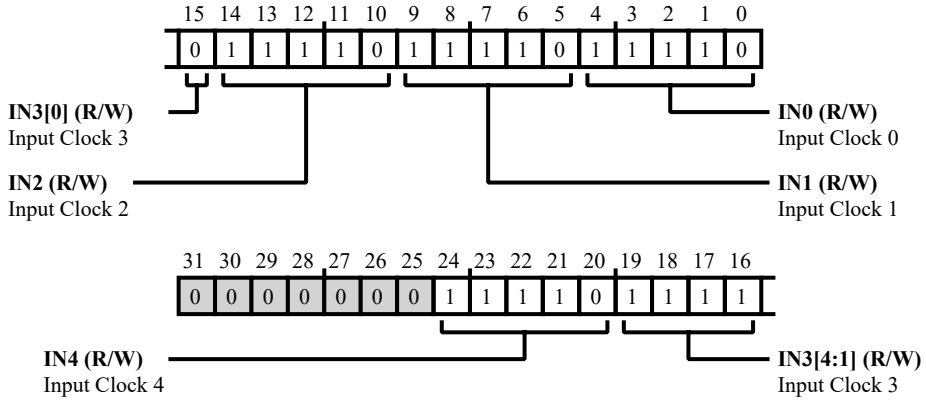


Figure 29-18: DAI_CLK2 Register Diagram

Table 29-25: DAI_CLK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK2.IN4 holds the source signal assignment that will be routed to the DAI_CLK2.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK2.IN3 holds the source signal assignment that will be routed to the DAI_CLK2.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK2.IN2 holds the source signal assignment that will be routed to the DAI_CLK2.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK2.IN1 holds the source signal assignment that will be routed to the DAI_CLK2.IN1 destination. Refer to the Group A Signals table for source and destination mappings.
4:0 (R/W)	IN0	Input Clock 0. DAI_CLK2.IN0 holds the source signal assignment that will be routed to the DAI_CLK2.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Clock Routing Control Register 3

The DAI_CLK3 register provides clock routing connections for the S/PDIF.

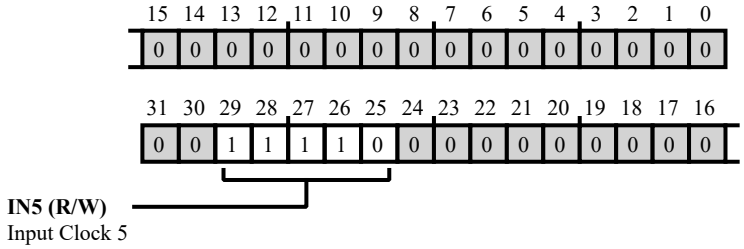


Figure 29-19: DAI_CLK3 Register Diagram

Table 29-26: DAI_CLK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. The DAI_CLK3.IN5 bit field provides the S/PDIF oversampling clock.

Clock Routing Control Register 4

The DAI_CLK4 register provides clock routing connections for the precision clock generators (PCGs).

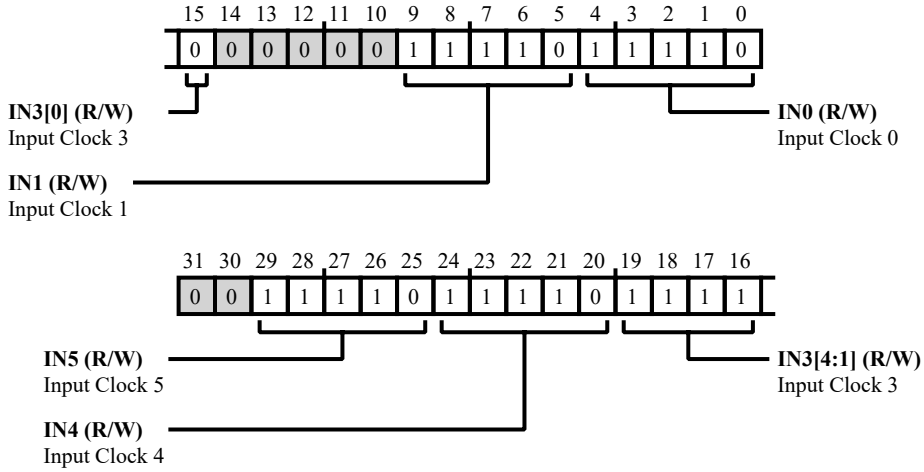


Figure 29-20: DAI_CLK4 Register Diagram

Table 29-27: DAI_CLK4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK4.IN5 holds the source signal assignment that will be routed to the DAI_CLK4.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK4.IN4 holds the source signal assignment that will be routed to the DAI_CLK4.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK4.IN3 holds the source signal assignment that will be routed to the DAI_CLK4.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK4.IN1 holds the source signal assignment that will be routed to the DAI_CLK4.IN1 destination. Refer to the Group A Signals table for source and destination mappings.
4:0 (R/W)	IN0	Input Clock 0. DAI_CLK4.IN0 holds the source signal assignment that will be routed to the DAI_CLK4.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Clock Routing Control Register 5

The DAI_CLK5 register provides clock routing connections for the serial ports.

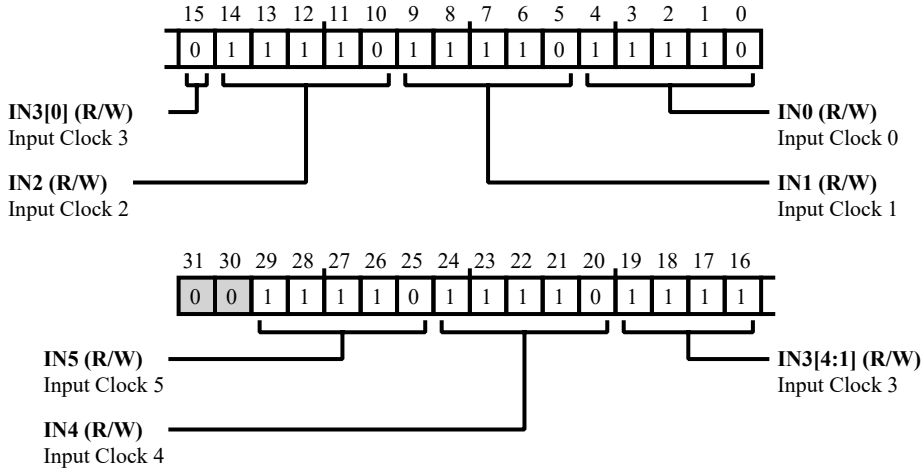


Figure 29-21: DAI_CLK5 Register Diagram

Table 29-28: DAI_CLK5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK5.IN5 holds the source signal assignment that will be routed to the DAI_CLK5.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK5.IN4 holds the source signal assignment that will be routed to the DAI_CLK5.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK5.IN3 holds the source signal assignment that will be routed to the DAI_CLK5.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK5.IN2 holds the source signal assignment that will be routed to the DAI_CLK5.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK5.IN1 holds the source signal assignment that will be routed to the DAI_CLK5.IN1 destination. Refer to the Group A Signals table for source and destination mappings.

Table 29-28: DAI_CLK5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK5.IN0` holds the source signal assignment that will be routed to the `DAI_CLK5.IN0` destination. Refer to the Group A Signals table for source and destination mappings.

Clock Routing Control Register 6

The DAI_CLK6 register provides clock routing connections for the serial ports.

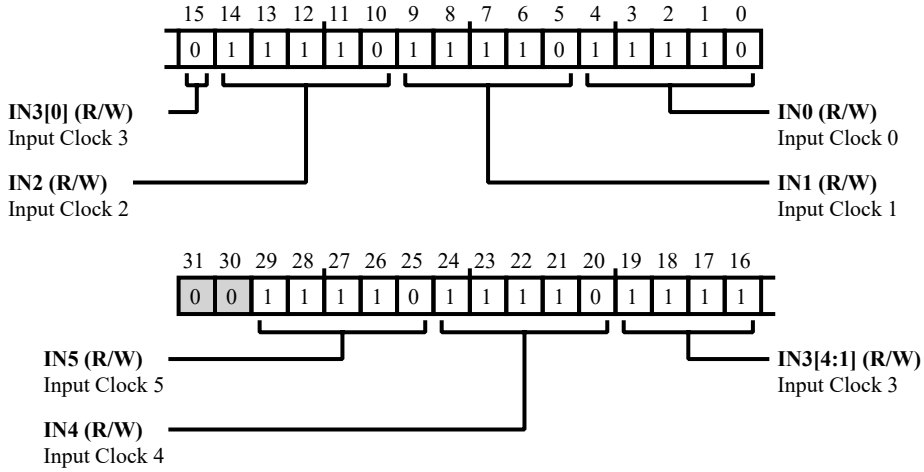


Figure 29-22: DAI_CLK6 Register Diagram

Table 29-29: DAI_CLK6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK6.IN5 holds the source signal assignment that will be routed to the DAI_CLK6.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK6.IN4 holds the source signal assignment that will be routed to the DAI_CLK6.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK6.IN3 holds the source signal assignment that will be routed to the DAI_CLK6.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK6.IN2 holds the source signal assignment that will be routed to the DAI_CLK6.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK6.IN1 holds the source signal assignment that will be routed to the DAI_CLK6.IN1 destination. Refer to the Group A Signals table for source and destination mappings.

Table 29-29: DAI_CLK6 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK6.IN0` holds the source signal assignment that will be routed to the `DAI_CLK6.IN0` destination. Refer to the Group A Signals table for source and destination mappings.

Clock Routing Control Register 7

The DAI_CLK7 register provides clock routing connections for the serial ports.

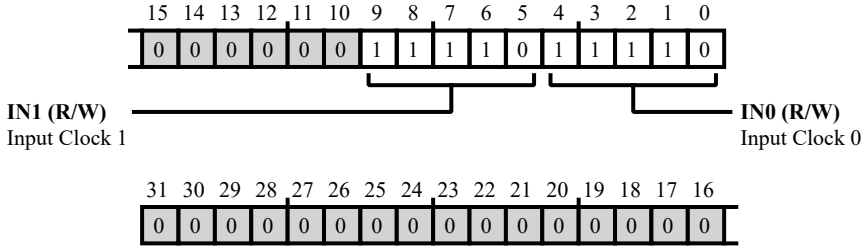


Figure 29-23: DAI_CLK7 Register Diagram

Table 29-30: DAI_CLK7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK7.IN1 holds the source signal assignment that will be routed to the DAI_CLK7.IN1 destination. Refer to the Group A Signals table for source and destination mappings.
4:0 (R/W)	IN0	Input Clock 0. DAI_CLK7.IN0 holds the source signal assignment that will be routed to the DAI_CLK7.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

DAI MISC Control Register

The DAI_DAI_MISC_CTL register provides the misc control signals related to DAI block.

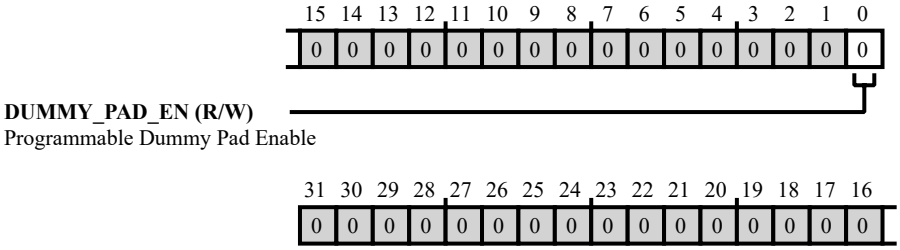


Figure 29-24: DAI_DAI_MISC_CTL Register Diagram

Table 29-31: DAI_DAI_MISC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	DUMMY_PAD_EN	Programmable Dummy Pad Enable. When set (=1), the DAI_DAI_MISC_CTL.DUMMY_PAD_EN bit, enables programmable dummy pads feature in DAI block.

Serial Data Routing Control Register 0

The DAI_DAT0 register routes serial data to the serial ports.

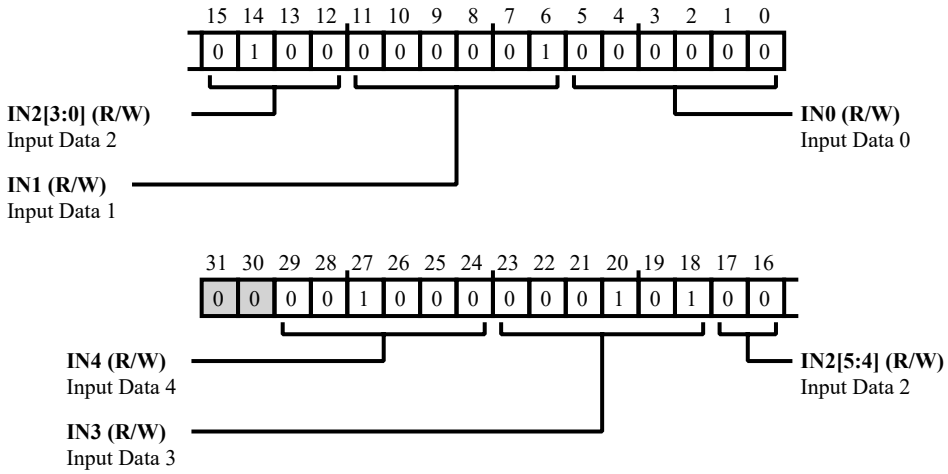


Figure 29-25: DAI_DAT0 Register Diagram

Table 29-32: DAI_DAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT0.IN4 holds the source signal assignment that will be routed to the DAI_DAT0.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT0.IN3 holds the source signal assignment that will be routed to the DAI_DAT0.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT0.IN2 holds the source signal assignment that will be routed to the DAI_DAT0.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT0.IN1 holds the source signal assignment that will be routed to the DAI_DAT0.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT0.IN0 holds the source signal assignment that will be routed to the DAI_DAT0.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 1

The DAI_DAT1 register routes serial data to the serial ports.

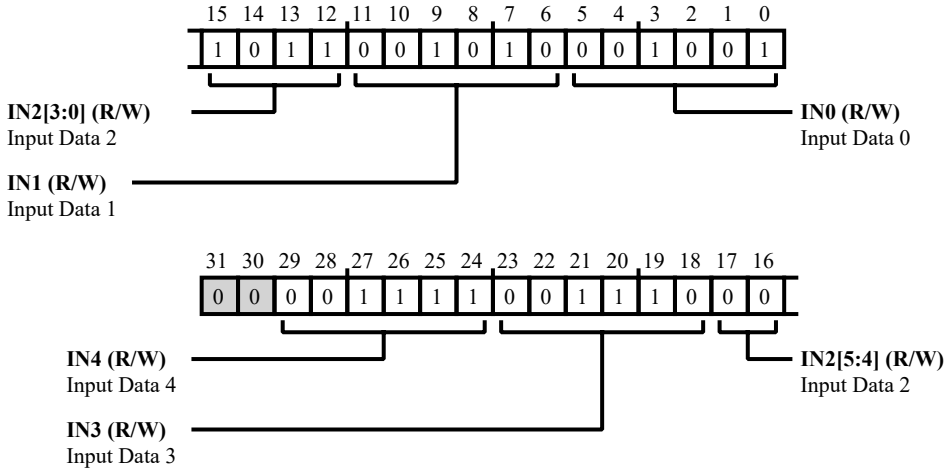


Figure 29-26: DAI_DAT1 Register Diagram

Table 29-33: DAI_DAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT1.IN4 holds the source signal assignment that will be routed to the DAI_DAT1.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT1.IN3 holds the source signal assignment that will be routed to the DAI_DAT1.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT1.IN2 holds the source signal assignment that will be routed to the DAI_DAT1.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT1.IN1 holds the source signal assignment that will be routed to the DAI_DAT1.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT1.IN0 holds the source signal assignment that will be routed to the DAI_DAT1.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 2

The DAI_DAT2 register routes serial data to the serial ports and the asynchronous sample rate converter (ASRC).

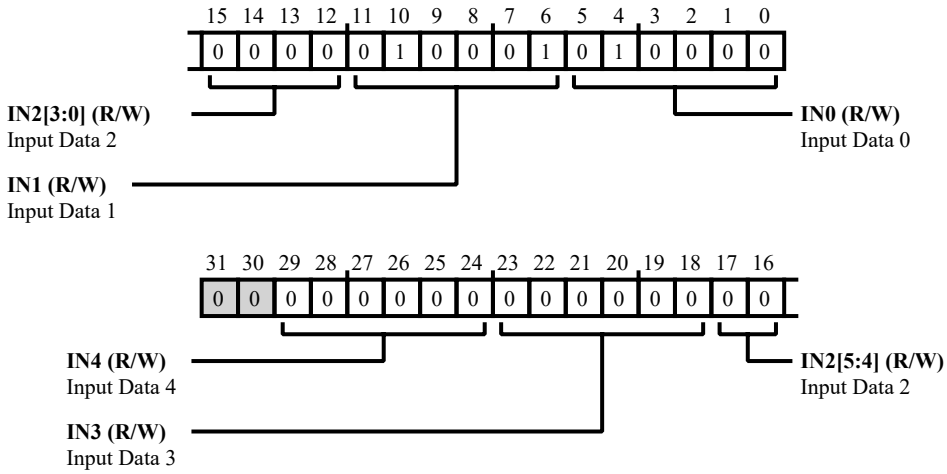


Figure 29-27: DAI_DAT2 Register Diagram

Table 29-34: DAI_DAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT2.IN4 holds the source signal assignment that will be routed to the DAI_DAT2.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT2.IN3 holds the source signal assignment that will be routed to the DAI_DAT2.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT2.IN2 holds the source signal assignment that will be routed to the DAI_DAT2.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT2.IN1 holds the source signal assignment that will be routed to the DAI_DAT2.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT2.IN0 holds the source signal assignment that will be routed to the DAI_DAT2.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 3

The DAI_DAT3 register routes serial data to the asynchronous sample rate converter (ASRC).

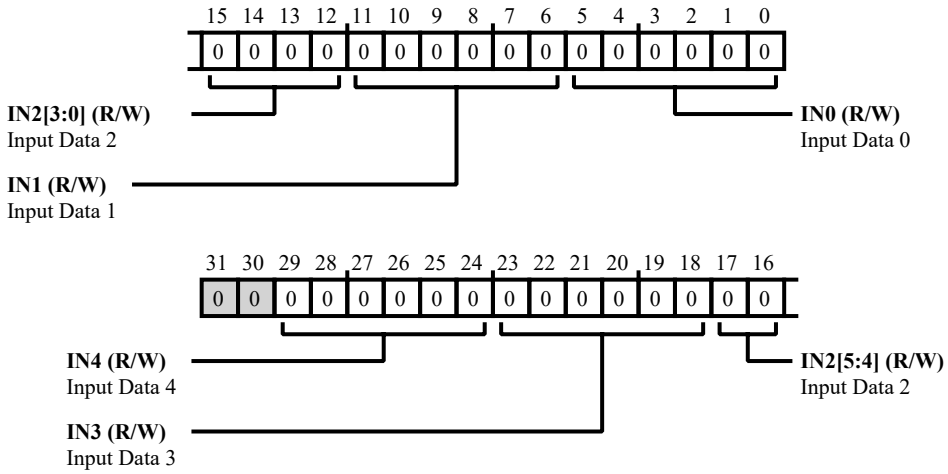


Figure 29-28: DAI_DAT3 Register Diagram

Table 29-35: DAI_DAT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT3.IN4 holds the source signal assignment that will be routed to the DAI_DAT3.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT3.IN3 holds the source signal assignment that will be routed to the DAI_DAT3.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT3.IN2 holds the source signal assignment that will be routed to the DAI_DAT3.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT3.IN1 holds the source signal assignment that will be routed to the DAI_DAT3.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT3.IN0 holds the source signal assignment that will be routed to the DAI_DAT3.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 4

The DAI_DAT4 register routes serial data to the S/PDIF.

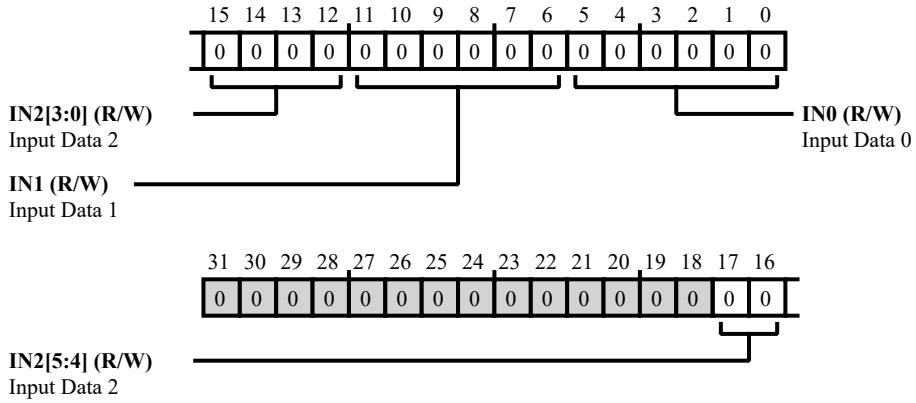


Figure 29-29: DAI_DAT4 Register Diagram

Table 29-36: DAI_DAT4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:12 (R/W)	IN2	Input Data 2. DAI_DAT4.IN2 holds the source signal assignment that will be routed to the DAI_DAT4.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT4.IN1 holds the source signal assignment that will be routed to the DAI_DAT4.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT4.IN0 holds the source signal assignment that will be routed to the DAI_DAT4.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 5

The DAI_DAT5 register routes serial data to the S/PDIF.

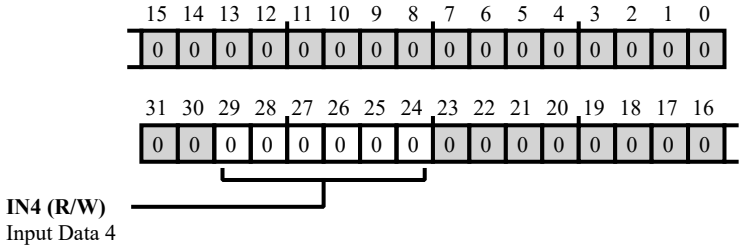


Figure 29-30: DAI_DAT5 Register Diagram

Table 29-37: DAI_DAT5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. The DAI_DAT5.IN4 bit field routes input data to the S/PDIF Biphase receiver stream.

Serial Data Routing Control Register 6

The DAI_DAT6 register routes serial data to the serial ports.

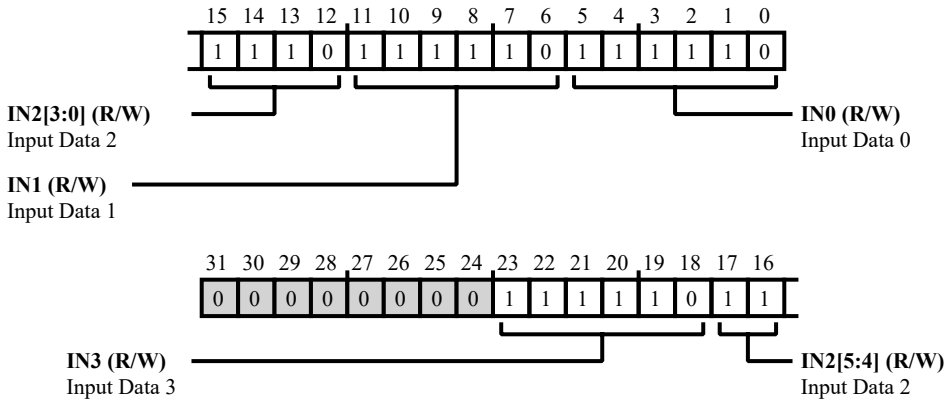


Figure 29-31: DAI_DAT6 Register Diagram

Table 29-38: DAI_DAT6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:18 (R/W)	IN3	Input Data 3. DAI_DAT6.IN3 holds the source signal assignment that will be routed to the DAI_DAT6.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT6.IN2 holds the source signal assignment that will be routed to the DAI_DAT6.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT6.IN1 holds the source signal assignment that will be routed to the DAI_DAT6.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT6.IN0 holds the source signal assignment that will be routed to the DAI_DAT6.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 7

The DAI_DAT7 register routes serial data to the serial ports.

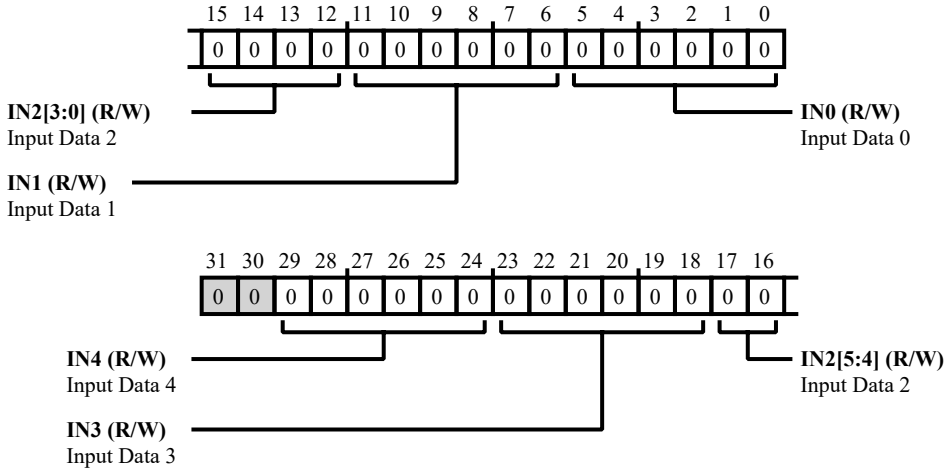


Figure 29-32: DAI_DAT7 Register Diagram

Table 29-39: DAI_DAT7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT7.IN4 holds the source signal assignment that will be routed to the DAI_DAT7.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT7.IN3 holds the source signal assignment that will be routed to the DAI_DAT7.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT7.IN2 holds the source signal assignment that will be routed to the DAI_DAT7.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT7.IN1 holds the source signal assignment that will be routed to the DAI_DAT7.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT7.IN0 holds the source signal assignment that will be routed to the DAI_DAT7.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Serial Data Routing Control Register 8

The DAI_DAT8 register routes serial data to the serial ports.

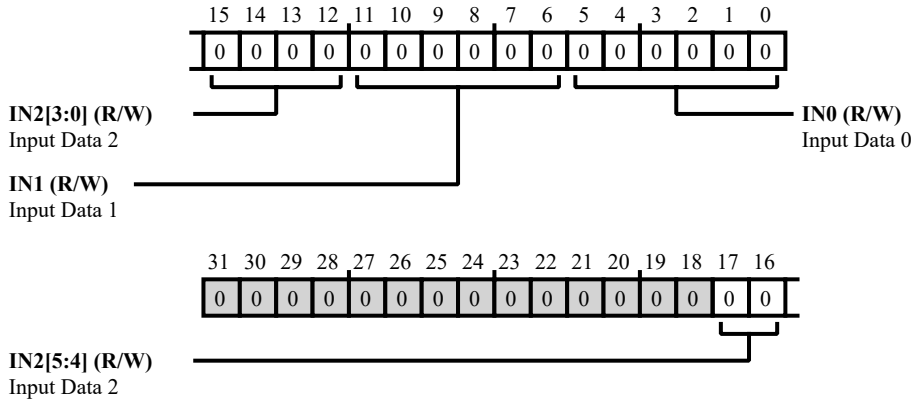


Figure 29-33: DAI_DAT8 Register Diagram

Table 29-40: DAI_DAT8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:12 (R/W)	IN2	Input Data 2. DAI_DAT8.IN2 holds the source signal assignment that will be routed to the DAI_DAT8.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT8.IN1 holds the source signal assignment that will be routed to the DAI_DAT8.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT8.IN0 holds the source signal assignment that will be routed to the DAI_DAT8.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Clock Routing Control Register 0

The DAI_EXTD_CLK0 and DAI_CLK0 registers together provides the extended clock routing connections for the serial ports (SPORTs).

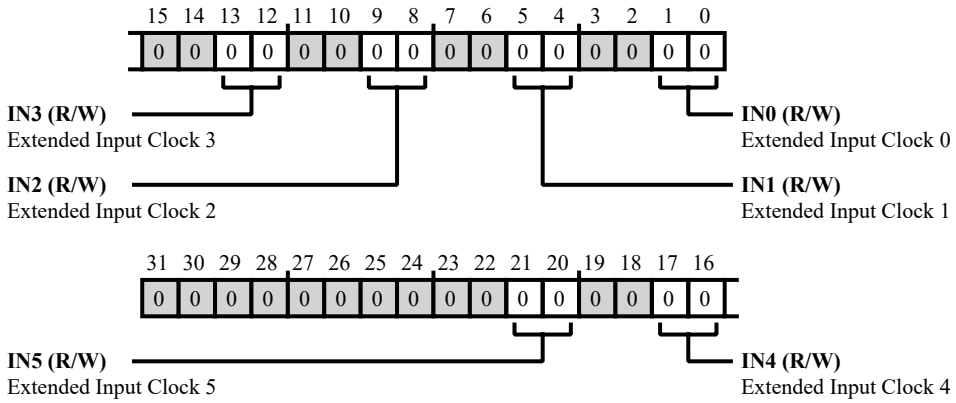


Figure 29-34: DAI_EXTD_CLK0 Register Diagram

Table 29-41: DAI_EXTD_CLK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK0.IN5 and DAI_CLK0.IN5 together hold the extended source signal assignment that will be routed to the DAI_CLK0.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK0.IN4 and DAI_CLK0.IN4 together hold the extended source signal assignment that will be routed to the DAI_CLK0.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK0.IN3 and DAI_CLK0.IN3 together hold the extended source signal assignment that will be routed to the DAI_CLK0.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK0.IN2 and DAI_CLK0.IN2 together hold the extended source signal assignment that will be routed to the DAI_CLK0.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK0.IN1 and DAI_CLK0.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK0.IN1 destination. Refer to the Group A Signals table for source and destination mappings

Table 29-41: DAI_EXTD_CLK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK0.IN0` and DAI_CLK0.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK0.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 1

The DAI_EXTD_CLK1 and DAI_CLK1 registers together provides the extended clock routing connections for the serial ports (SPORTs).

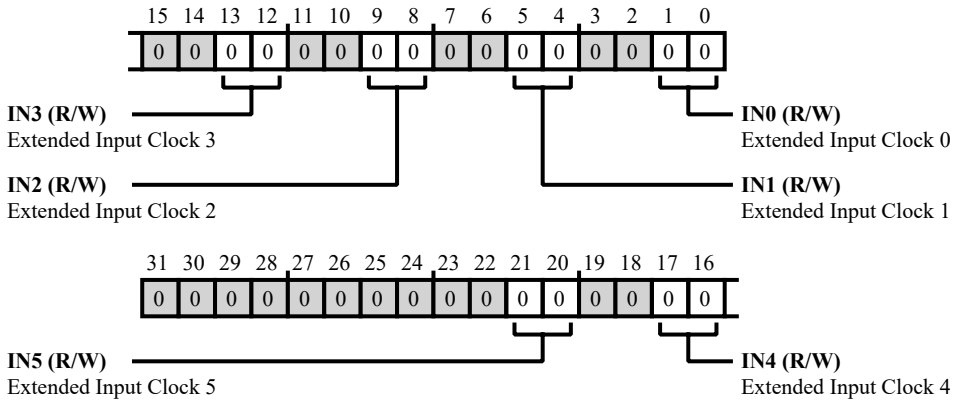


Figure 29-35: DAI_EXTD_CLK1 Register Diagram

Table 29-42: DAI_EXTD_CLK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK1.IN5 and DAI_CLK1.IN5 together hold the extended source signal assignment that will be routed to the DAI_CLK1.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK1.IN4 and DAI_CLK1.IN4 together hold the extended source signal assignment that will be routed to the DAI_CLK1.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK1.IN3 and DAI_CLK1.IN3 together hold the extended source signal assignment that will be routed to the DAI_CLK1.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK1.IN2 and DAI_CLK1.IN2 together hold the extended source signal assignment that will be routed to the DAI_CLK1.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK1.IN1 and DAI_CLK1.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK1.IN1 destination. Refer to the Group A Signals table for source and destination mappings.

Table 29-42: DAI_EXTD_CLK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK1.IN0` and DAI_CLK1.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK1.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 2

The DAI_EXTD_CLK2 and DAI_CLK2 registers together provides the extended clock routing connections for the serial ports (SPORTs).

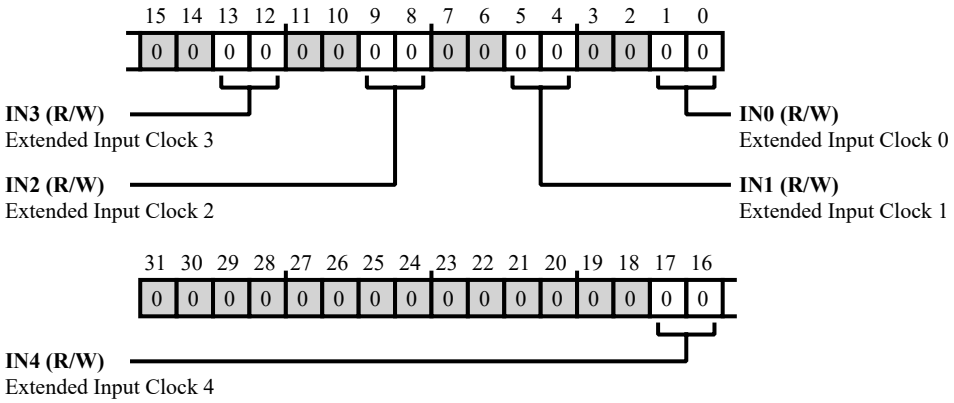


Figure 29-36: DAI_EXTD_CLK2 Register Diagram

Table 29-43: DAI_EXTD_CLK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK2.IN4 and DAI_CLK2.IN4 together hold the extended source signal assignment that will be routed to the DAI_CLK2.IN4 destination. Refer to the Group A Signals table for source and destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK2.IN3 and DAI_CLK2.IN3 together hold the extended source signal assignment that will be routed to the DAI_CLK2.IN3 destination. Refer to the Group A Signals table for source and destination mappings
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK2.IN2 and DAI_CLK2.IN2 together hold the extended source signal assignment that will be routed to the DAI_CLK2.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK2.IN1 and DAI_CLK2.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK2.IN1 destination. Refer to the Group A Signals table for source and destination mappings.
1:0 (R/W)	IN0	Extended Input Clock 0. DAI_EXTD_CLK2.IN0 and DAI_CLK2.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK2.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 3

The DAI_EXTD_CLK3 and DAI_CLK3 registers together provides the extended clock routing connections for the serial ports (SPORTs).

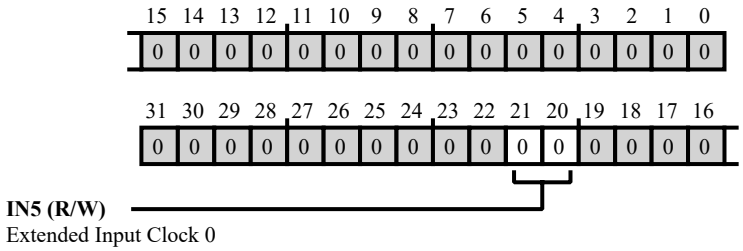


Figure 29-37: DAI_EXTD_CLK3 Register Diagram

Table 29-44: DAI_EXTD_CLK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 0. DAI_EXTD_CLK3.IN5 and DAI_CLK3.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK3.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 4

The DAI_EXTD_CLK4 and DAI_CLK4 registers together provides the extended clock routing connections for the serial ports (SPORTs).

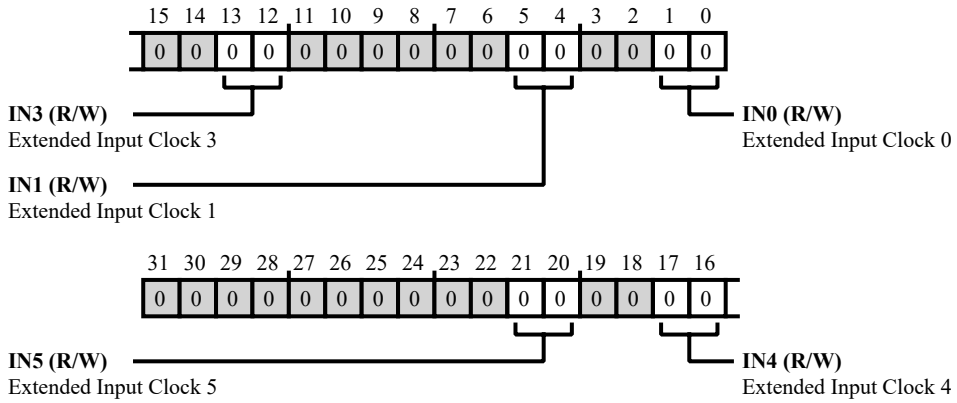


Figure 29-38: DAI_EXTD_CLK4 Register Diagram

Table 29-45: DAI_EXTD_CLK4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK4.IN5 and DAI_CLK4.IN5 together hold the extended source signal assignment that will be routed to the DAI_CLK4.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK4.IN4 and DAI_CLK4.IN4 together hold the extended source signal assignment that will be routed to the DAI_CLK4.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK4.IN3 and DAI_CLK4.IN3 together hold the extended source signal assignment that will be routed to the DAI_CLK4.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK4.IN1 and DAI_CLK4.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK4.IN1 destination. Refer to the Group A Signals table for source and destination mappings.
1:0 (R/W)	IN0	Extended Input Clock 0. DAI_EXTD_CLK4.IN0 and DAI_CLK4.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK4.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 5

The DAI_EXTD_CLK5 and DAI_CLK5 registers together provides the extended clock routing connections for the serial ports (SPORTs).

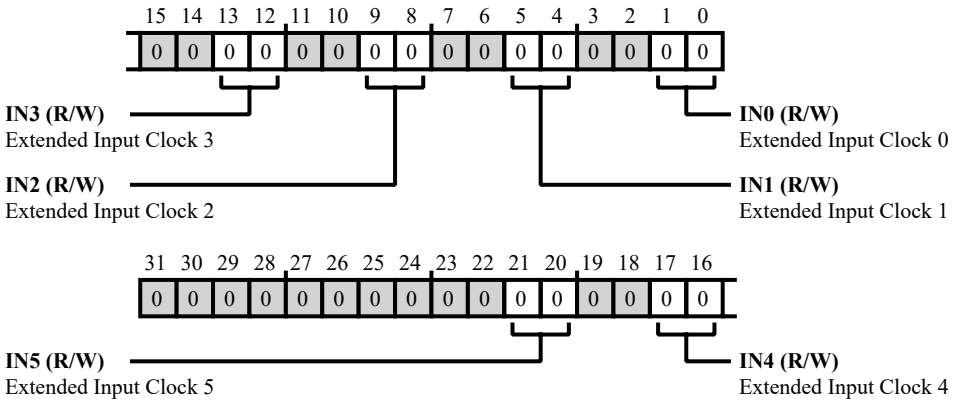


Figure 29-39: DAI_EXTD_CLK5 Register Diagram

Table 29-46: DAI_EXTD_CLK5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK5.IN5 and DAI_CLK5.IN5 together hold the extended source signal assignment that will be routed to the DAI_CLK5.IN5 destination. Refer to the Group A Signals table for source and destination mappings
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK5.IN4 and DAI_CLK5.IN4 together hold the extended source signal assignment that will be routed to the DAI_CLK5.IN4 destination. Refer to the Group A Signals table for source and destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK5.IN3 and DAI_CLK5.IN3 together hold the extended source signal assignment that will be routed to the DAI_CLK5.IN3 destination. Refer to the Group A Signals table for source and destination mappings
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK5.IN2 and DAI_CLK5.IN2 together hold the extended source signal assignment that will be routed to the DAI_CLK5.IN2 destination. Refer to the Group A Signals table for source and destination mappings
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK5.IN1 and DAI_CLK5.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK5.IN1 destination. Refer to the Group A Signals table for source and destination mappings

Table 29-46: DAI_EXTD_CLK5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK5.IN0` and DAI_CLK5.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK5.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 6

The DAI_EXTD_CLK6 and DAI_CLK6 registers together provides the extended clock routing connections for the serial ports (SPORTs).

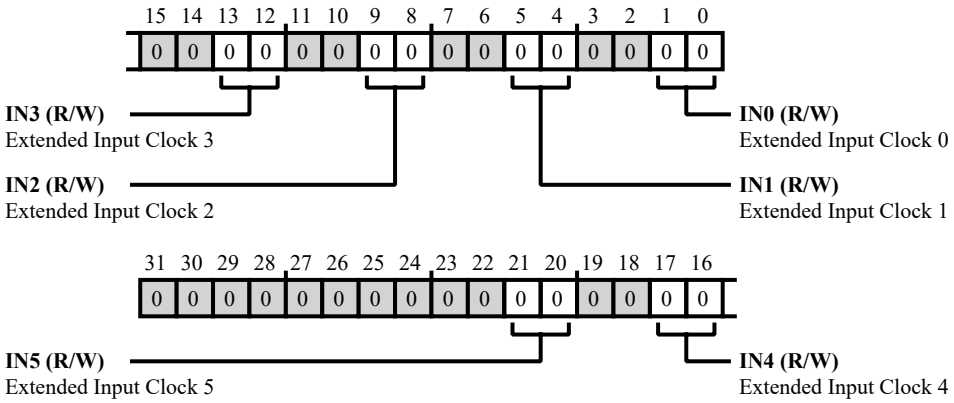


Figure 29-40: DAI_EXTD_CLK6 Register Diagram

Table 29-47: DAI_EXTD_CLK6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK6.IN5 and DAI_CLK6.IN5 together hold the extended source signal assignment that will be routed to the DAI_CLK6.IN5 destination. Refer to the Group A Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK6.IN4 and DAI_CLK6.IN4 together hold the extended source signal assignment that will be routed to the DAI_CLK6.IN4 destination. Refer to the Group A Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK6.IN3 and DAI_CLK6.IN3 together hold the extended source signal assignment that will be routed to the DAI_CLK6.IN3 destination. Refer to the Group A Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK6.IN2 and DAI_CLK6.IN2 together hold the extended source signal assignment that will be routed to the DAI_CLK6.IN2 destination. Refer to the Group A Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK6.IN1 and DAI_CLK6.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK6.IN1 destination. Refer to the Group A Signals table for source and destination mappings.

Table 29-47: DAI_EXTD_CLK6 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK6.IN0` and DAI_CLK6.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK6.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Clock Routing Control Register 7

The DAI_EXTD_CLK7 and DAI_CLK7 registers together provides the extended clock routing connections for the serial ports (SPORTs).

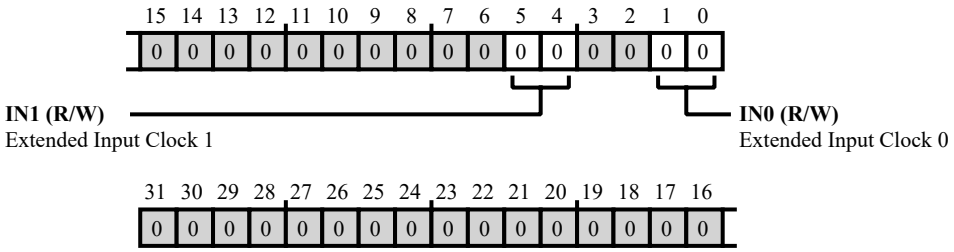


Figure 29-41: DAI_EXTD_CLK7 Register Diagram

Table 29-48: DAI_EXTD_CLK7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK7.IN1 and DAI_CLK7.IN1 together hold the extended source signal assignment that will be routed to the DAI_CLK7.IN1 destination. Refer to the Group A Signals table for source and destination mappings.
1:0 (R/W)	IN0	Extended Input Clock 0. DAI_EXTD_CLK7.IN0 and DAI_CLK7.IN0 together hold the extended source signal assignment that will be routed to the DAI_CLK7.IN0 destination. Refer to the Group A Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 0

The DAI_EXTD_DAT0 and DAI_DAT0 together provides the extended data routing connections to the serial data of serial ports.

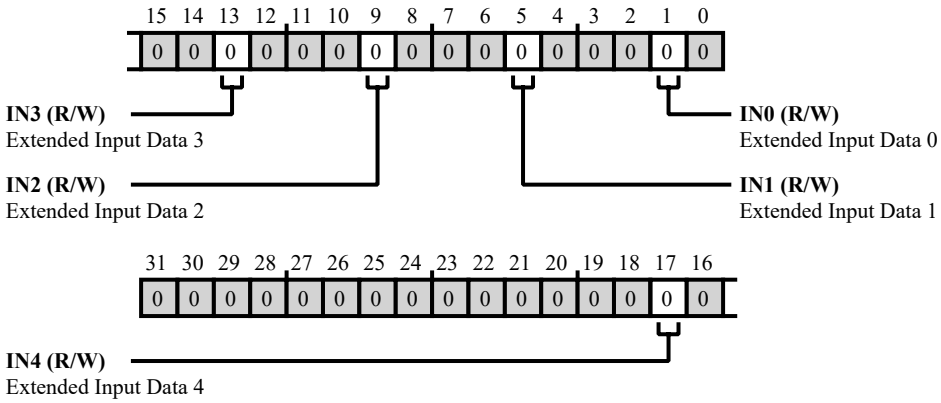


Figure 29-42: DAI_EXTD_DAT0 Register Diagram

Table 29-49: DAI_EXTD_DAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT0.IN4 and DAI_DAT0.IN4 together hold the source signal assignment that will be routed to the DAI_DAT0.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT0.IN3 and DAI_DAT0.IN3 together hold the source signal assignment that will be routed to the DAI_DAT0.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT0.IN2 and DAI_DAT0.IN2 together hold the source signal assignment that will be routed to the DAI_DAT0.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT0.IN1 and DAI_DAT0.IN1 together hold the source signal assignment that will be routed to the DAI_DAT0.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT0.IN0 and DAI_DAT0.IN0 together hold the source signal assignment that will be routed to the DAI_DAT0.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 1

The DAI_EXTD_DAT1 and DAI_DAT1 together provides the extended data routing connections to the serial data of serial ports.

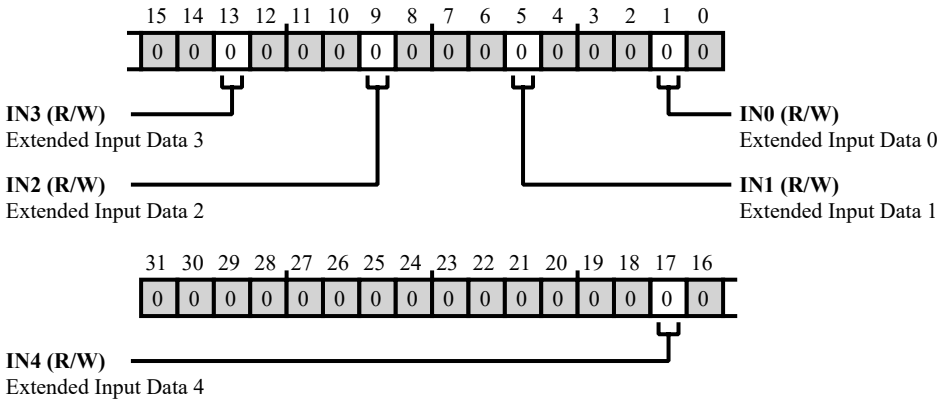


Figure 29-43: DAI_EXTD_DAT1 Register Diagram

Table 29-50: DAI_EXTD_DAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT1.IN4 and DAI_DAT1.IN4 together hold the source signal assignment that will be routed to the DAI_DAT1.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT1.IN3 and DAI_DAT1.IN3 together hold the source signal assignment that will be routed to the DAI_DAT1.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT1.IN2 and DAI_DAT1.IN2 together hold the source signal assignment that will be routed to the DAI_DAT1.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT1.IN1 and DAI_DAT1.IN1 together hold the source signal assignment that will be routed to the DAI_DAT1.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT1.IN0 and DAI_DAT1.IN0 together hold the source signal assignment that will be routed to the DAI_DAT1.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 2

The DAI_EXTD_DAT2 and DAI_DAT2 together provides the extended data routing connections to the serial data of serial ports.

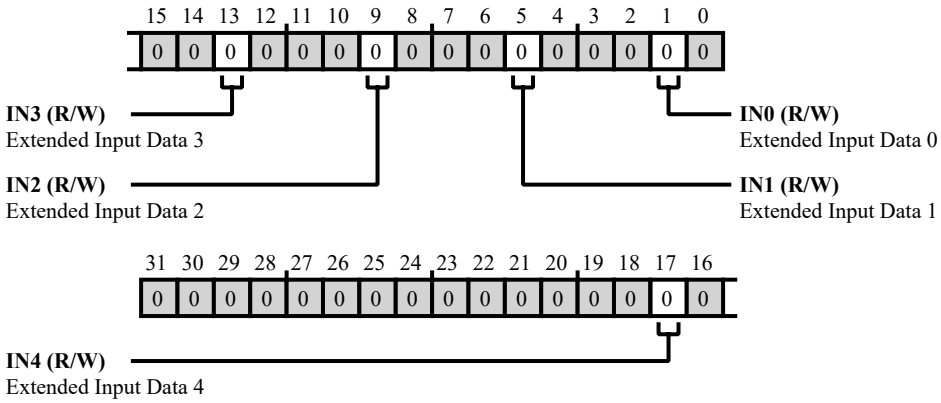


Figure 29-44: DAI_EXTD_DAT2 Register Diagram

Table 29-51: DAI_EXTD_DAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT2.IN4 and DAI_DAT2.IN4 together hold the source signal assignment that will be routed to the DAI_DAT2.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT2.IN3 and DAI_DAT2.IN3 together hold the source signal assignment that will be routed to the DAI_DAT2.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT2.IN2 and DAI_DAT2.IN2 together hold the source signal assignment that will be routed to the DAI_DAT2.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT2.IN1 and DAI_DAT2.IN1 together hold the source signal assignment that will be routed to the DAI_DAT2.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT2.IN0 and DAI_DAT2.IN0 together hold the source signal assignment that will be routed to the DAI_DAT2.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 3

The DAI_EXTD_DAT3 and DAI_DAT3 together provides the extended data routing connections to the serial data of serial ports.

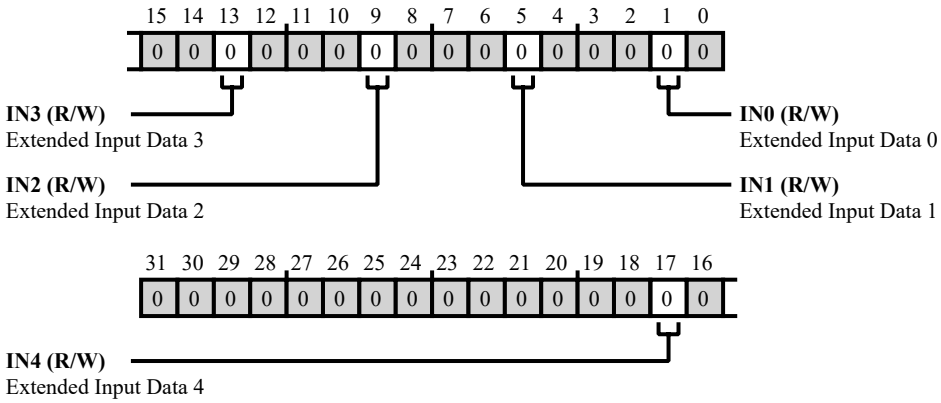


Figure 29-45: DAI_EXTD_DAT3 Register Diagram

Table 29-52: DAI_EXTD_DAT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT3.IN4 and DAI_DAT3.IN4 together hold the source signal assignment that will be routed to the DAI_DAT3.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT3.IN3 and DAI_DAT3.IN3 together hold the source signal assignment that will be routed to the DAI_DAT3.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT3.IN2 and DAI_DAT3.IN2 together hold the source signal assignment that will be routed to the DAI_DAT3.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT3.IN1 and DAI_DAT3.IN1 together hold the source signal assignment that will be routed to the DAI_DAT3.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT3.IN0 and DAI_DAT3.IN0 together hold the source signal assignment that will be routed to the DAI_DAT3.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 4

The DAI_EXTD_DAT4 and DAI_DAT4 together provides the extended data routing connections to the serial data of serial ports.

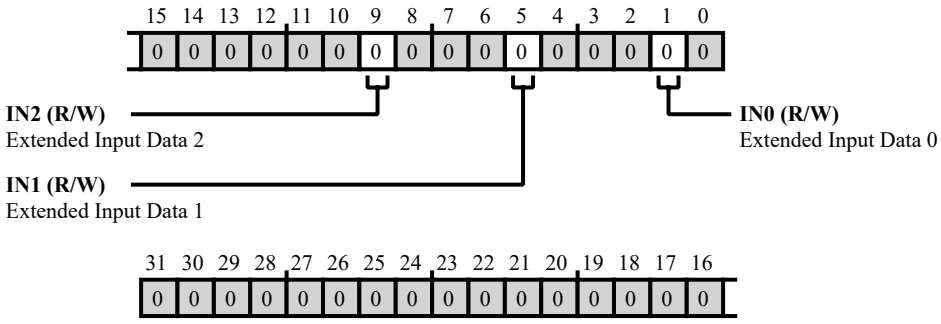


Figure 29-46: DAI_EXTD_DAT4 Register Diagram

Table 29-53: DAI_EXTD_DAT4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT4.IN2 and DAI_DAT4.IN2 together hold the source signal assignment that will be routed to the DAI_DAT4.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT4.IN1 and DAI_DAT4.IN2 together hold the source signal assignment that will be routed to the DAI_DAT4.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT4.IN0 and DAI_DAT4.IN0 together hold the source signal assignment that will be routed to the DAI_DAT4.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 5

The DAI_EXTD_DAT5 and DAI_DAT5 together provides the extended data routing connections to the serial data of serial ports.

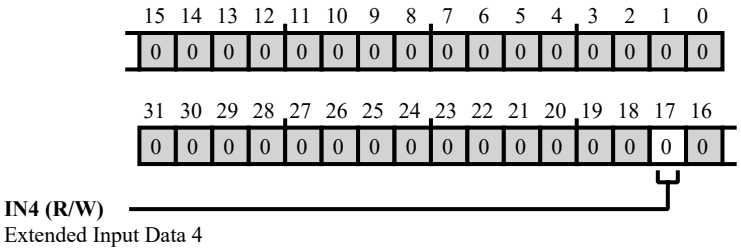


Figure 29-47: DAI_EXTD_DAT5 Register Diagram

Table 29-54: DAI_EXTD_DAT5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT5.IN4 and DAI_DAT5.IN4 together hold the source signal assignment that will be routed to the DAI_DAT5.IN4 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 6

The DAI_EXTD_DAT6 and DAI_DAT6 together provides the extended data routing connections to the serial data of serial ports.

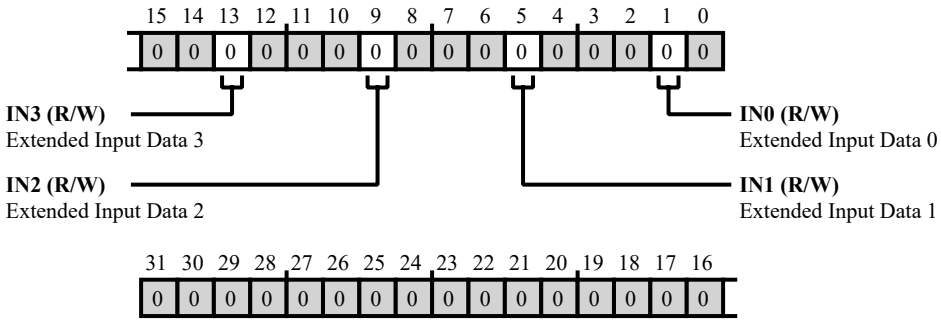


Figure 29-48: DAI_EXTD_DAT6 Register Diagram

Table 29-55: DAI_EXTD_DAT6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT6.IN3 and DAI_DAT6.IN3 together hold the source signal assignment that will be routed to the DAI_DAT6.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT6.IN2 and DAI_DAT6.IN2 together hold the source signal assignment that will be routed to the DAI_DAT6.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT6.IN1 and DAI_DAT6.IN1 together hold the source signal assignment that will be routed to the DAI_DAT6.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT6.IN0 and DAI_DAT6.IN0 together hold the source signal assignment that will be routed to the DAI_DAT6.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 7

The DAI_EXTD_DAT7 and DAI_DAT7 together provides the extended data routing connections to the serial data of serial ports.

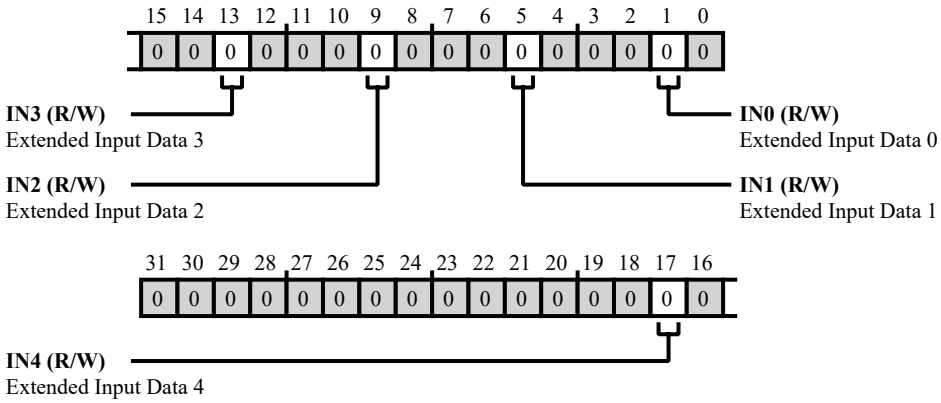


Figure 29-49: DAI_EXTD_DAT7 Register Diagram

Table 29-56: DAI_EXTD_DAT7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT7.IN4 and DAI_DAT7.IN4 together hold the source signal assignment that will be routed to the DAI_DAT7.IN4 destination. Refer to the Group B Signals table for source and destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT7.IN3 and DAI_DAT7.IN3 together hold the source signal assignment that will be routed to the DAI_DAT7.IN3 destination. Refer to the Group B Signals table for source and destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT7.IN2 and DAI_DAT7.IN2 together hold the source signal assignment that will be routed to the DAI_DAT7.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT7.IN1 and DAI_DAT7.IN1 together hold the source signal assignment that will be routed to the DAI_DAT7.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT7.IN0 and DAI_DAT7.IN0 together hold the source signal assignment that will be routed to the DAI_DAT7.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Serial Data Routing Control Register 8

The DAI_EXTD_DAT8 and DAI_DAT8 together provides the extended data routing connections to the serial data of serial ports.

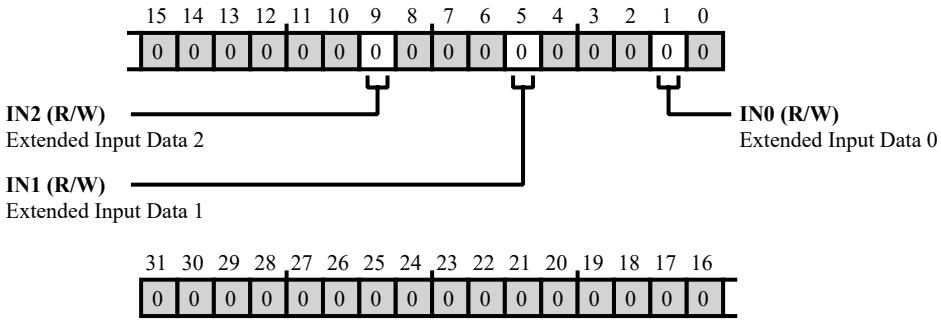


Figure 29-50: DAI_EXTD_DAT8 Register Diagram

Table 29-57: DAI_EXTD_DAT8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT8.IN2 and DAI_DAT8.IN2 together hold the source signal assignment that will be routed to the DAI_DAT8.IN2 destination. Refer to the Group B Signals table for source and destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT8.IN1 and DAI_DAT8.IN1 together hold the source signal assignment that will be routed to the DAI_DAT8.IN1 destination. Refer to the Group B Signals table for source and destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT8.IN0 and DAI_DAT8.IN0 together hold the source signal assignment that will be routed to the DAI_DAT8.IN0 destination. Refer to the Group B Signals table for source and destination mappings.

Extended Frame Sync Routing Control Register 0

The DAI_EXTD_FS0 and DAI_FS0 together provides the extended frame sync routing connections to the frame syncs of serial ports.

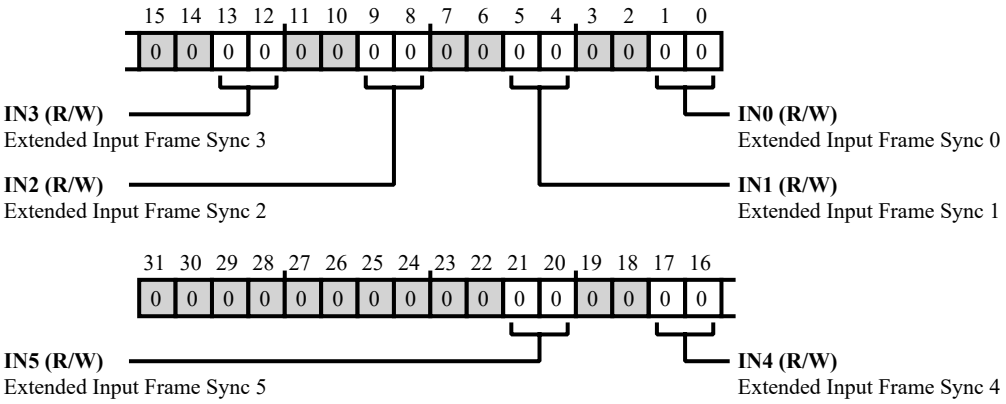


Figure 29-51: DAI_EXTD_FS0 Register Diagram

Table 29-58: DAI_EXTD_FS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Frame Sync 5. DAI_EXTD_FS0.IN5 and DAI_FS0.IN5 together hold the source signal assignment that will be routed to the DAI_FS0.IN5 destination. Refer to the Group C Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Frame Sync 4. DAI_EXTD_FS0.IN4 and DAI_FS0.IN4 together hold the source signal assignment that will be routed to the DAI_FS0.IN4 destination. Refer to the Group C Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS0.IN3 and DAI_FS0.IN3 together hold the source signal assignment that will be routed to the DAI_FS0.IN3 destination. Refer to the Group C Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS0.IN2 and DAI_FS0.IN2 together hold the source signal assignment that will be routed to the DAI_FS0.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS0.IN1 and DAI_FS0.IN1 together hold the source signal assignment that will be routed to the DAI_FS0.IN1 destination. Refer to the Group C Signals table for source and destination mappings.

Table 29-58: DAI_EXTD_FS0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Frame Sync 0. `DAI_EXTD_FS0.IN0` and DAI_FS0.IN0 together hold the source signal assignment that will be routed to the DAI_FS0.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Extended Frame Sync Routing Control Register 1

The DAI_EXTD_FS1 and DAI_FS1 together provides the extended frame sync routing connections to the frame syncs of serial ports.

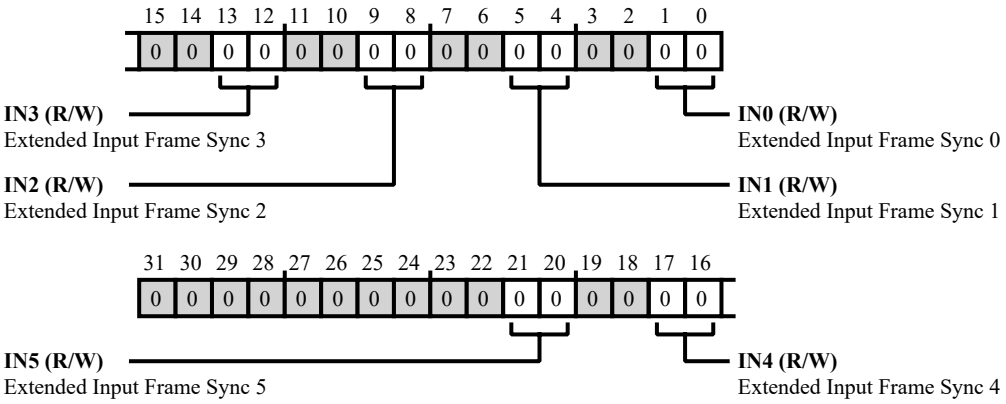


Figure 29-52: DAI_EXTD_FS1 Register Diagram

Table 29-59: DAI_EXTD_FS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Frame Sync 5. DAI_EXTD_FS1.IN5 and DAI_FS1.IN5 together hold the source signal assignment that will be routed to the DAI_FS1.IN5 destination. Refer to the Group C Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Frame Sync 4. DAI_EXTD_FS1.IN4 and DAI_FS1.IN4 together hold the source signal assignment that will be routed to the DAI_FS1.IN4 destination. Refer to the Group C Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS1.IN3 and DAI_FS1.IN3 together hold the source signal assignment that will be routed to the DAI_FS1.IN3 destination. Refer to the Group C Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS1.IN2 and DAI_FS1.IN2 together hold the source signal assignment that will be routed to the DAI_FS1.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS1.IN1 and DAI_FS1.IN1 together hold the source signal assignment that will be routed to the DAI_FS1.IN1 destination. Refer to the Group C Signals table for source and destination mappings.

Table 29-59: DAI_EXTD_FS1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Frame Sync 0. `DAI_EXTD_FS1.IN0` and DAI_FS1.IN0 together hold the source signal assignment that will be routed to the DAI_FS1.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Extended Frame Sync Routing Control Register 2

The DAI_EXTD_FS2 and DAI_FS2 together provides the extended frame sync routing connections to the frame syncs of serial ports.

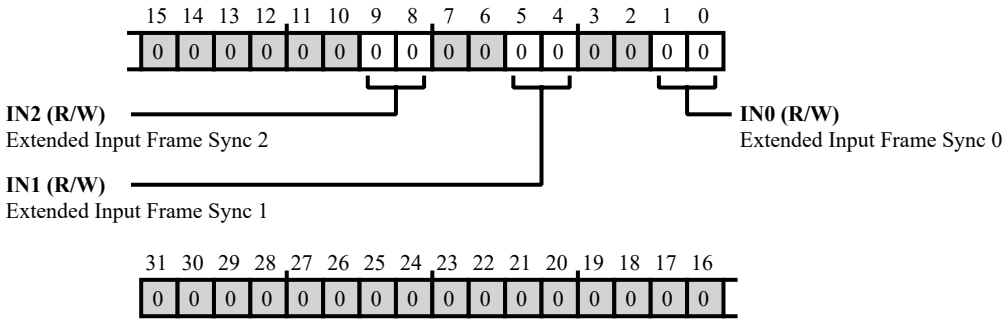


Figure 29-53: DAI_EXTD_FS2 Register Diagram

Table 29-60: DAI_EXTD_FS2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS2.IN2 and DAI_FS2.IN2 together hold the source signal assignment that will be routed to the DAI_FS2.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS2.IN1 and DAI_FS2.IN1 together hold the source signal assignment that will be routed to the DAI_FS2.IN1 destination. Refer to the Group C Signals table for source and destination mappings.
1:0 (R/W)	IN0	Extended Input Frame Sync 0. DAI_EXTD_FS2.IN0 and DAI_FS2.IN0 together hold the source signal assignment that will be routed to the DAI_FS2.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Extended Frame Sync Routing Control Register 4

The DAI_EXTD_FS4 and DAI_FS4 together provides the extended frame sync routing connections to the frame syncs of serial ports.

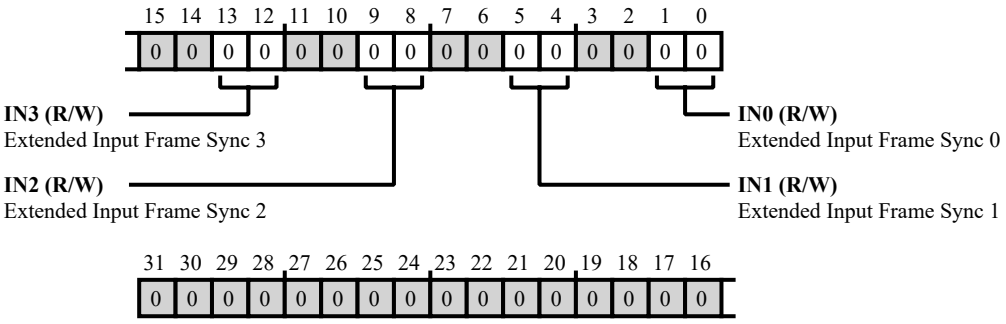


Figure 29-54: DAI_EXTD_FS4 Register Diagram

Table 29-61: DAI_EXTD_FS4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS4.IN3 and DAI_FS4.IN3 together hold the source signal assignment that will be routed to the DAI_FS4.IN3 destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS4.IN2 and DAI_FS4.IN2 together hold the source signal assignment that will be routed to the DAI_FS4.IN2 destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS4.IN1 and DAI_FS4.IN1 together hold the source signal assignment that will be routed to the DAI_FS4.IN1 destination. Refer to the Group C Signals table for source and destination mappings.
1:0 (R/W)	IN0	Extended Input Frame Sync 0. DAI_EXTD_FS4.IN0 and DAI_FS4.IN0 together hold the source signal assignment that will be routed to the DAI_FS4.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Extended Frame Sync Routing Control Register 5

The DAI_EXTD_FS5 and DAI_FS5 together provides the extended frame sync routing connections to the frame syncs of serial ports.

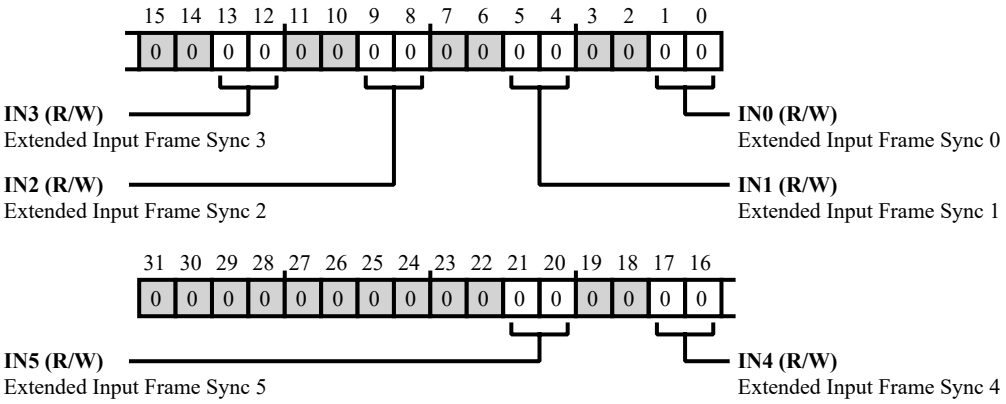


Figure 29-55: DAI_EXTD_FS5 Register Diagram

Table 29-62: DAI_EXTD_FS5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Frame Sync 5. DAI_EXTD_FS5.IN5 and DAI_FS5.IN5 together hold the source signal assignment that will be routed to the DAI_FS5.IN5 destination. Refer to the Group C Signals table for source and destination mappings.
17:16 (R/W)	IN4	Extended Input Frame Sync 4. DAI_EXTD_FS5.IN4 and DAI_FS5.IN4 together hold the source signal assignment that will be routed to the DAI_FS5.IN4 destination. Refer to the Group C Signals table for source and destination mappings.
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS5.IN3 and DAI_FS5.IN3 together hold the source signal assignment that will be routed to the DAI_FS5.IN3 destination. Refer to the Group C Signals table for source and destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS5.IN2 and DAI_FS5.IN2 together hold the source signal assignment that will be routed to the DAI_FS5.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS5.IN1 and DAI_FS5.IN1 together hold the source signal assignment that will be routed to the DAI_FS5.IN1 destination. Refer to the Group C Signals table for source and destination mappings.

Table 29-62: DAI_EXTD_FS5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Frame Sync 0. `DAI_EXTD_FS5.IN0` and DAI_FS5.IN0 together hold the source signal assignment that will be routed to the DAI_FS5.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Extended Frame Sync Routing Control Register 6

The DAI_EXTD_FS6 and DAI_FS6 together provides the extended frame sync routing connections to the frame syncs of serial ports.

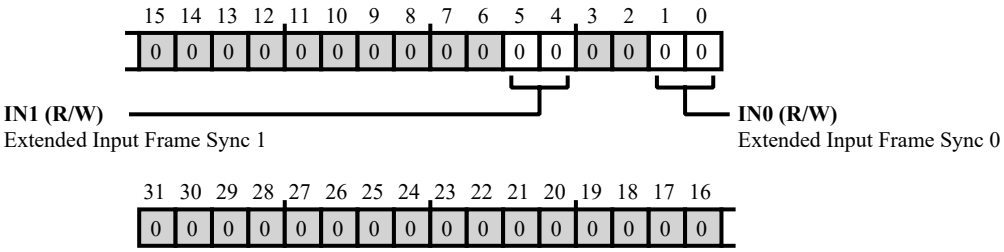


Figure 29-56: DAI_EXTD_FS6 Register Diagram

Table 29-63: DAI_EXTD_FS6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS6.IN1 and DAI_FS6.IN1 together hold the source signal assignment that will be routed to the DAI_FS6.IN1 destination. Refer to the Group C Signals table for source and destination mappings.
1:0 (R/W)	IN0	Extended Input Frame Sync 0. DAI_EXTD_FS6.IN0 and DAI_FS6.IN0 together hold the source signal assignment that will be routed to the DAI_FS6.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Extended Miscellaneous Control Register 0

The DAI_EXTD_MISC0 and DAI_MISC0 register together allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

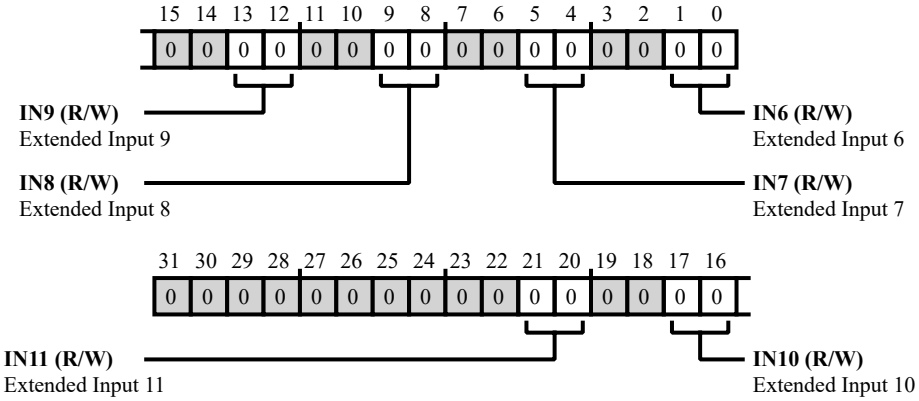


Figure 29-57: DAI_EXTD_MISC0 Register Diagram

Table 29-64: DAI_EXTD_MISC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN11	Extended Input 11. The DAI_EXTD_MISC0.IN11 bit field and DAI_MISC0.IN11 bit field together hold the source signal assignment for miscellaneous A5 input.
17:16 (R/W)	IN10	Extended Input 10. The DAI_EXTD_MISC0.IN10 bit field and DAI_MISC0.IN10 bit field together hold the source signal assignment for miscellaneous A4 input.
13:12 (R/W)	IN9	Extended Input 9. The DAI_EXTD_MISC0.IN9 bit field and DAI_MISC0.IN9 bit field together hold the source signal assignment for miscellaneous A3 input/miscellaneous Interrupt 9(DAI interrupt 31).
9:8 (R/W)	IN8	Extended Input 8. The DAI_EXTD_MISC0.IN8 bit field and DAI_MISC0.IN8 bit field together hold the source signal assignment for miscellaneous A2 input/miscellaneous Interrupt 8(DAI interrupt 30).
5:4 (R/W)	IN7	Extended Input 7. The DAI_EXTD_MISC0.IN7 bit field and DAI_MISC0.IN7 bit field together hold the source signal assignment for miscellaneous A1 input/miscellaneous Interrupt 7(DAI interrupt 29).

Table 29-64: DAI_EXTD_MISC0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN6	Extended Input 6. The `DAI_EXTD_MISC0.IN6` and DAI_MISC0.IN6 bit field together hold the source signal assignment for miscellaneous A0 input/miscellaneous Interrupt 6(DAI interrupt 28).

Extended Miscellaneous Control Register 1

The DAI_EXTD_MISC1 and DAI_MISC1 register together allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

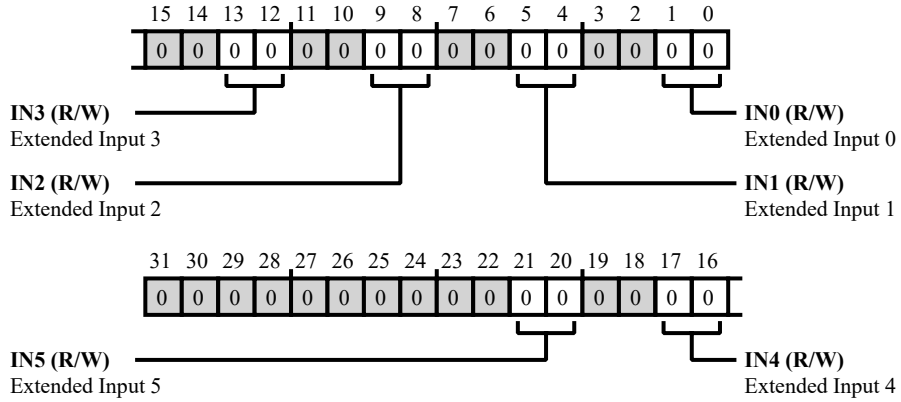


Figure 29-58: DAI_EXTD_MISC1 Register Diagram

Table 29-65: DAI_EXTD_MISC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input 5. The DAI_EXTD_MISC1.IN5 bit field and DAI_MISC1.IN5 bit field together hold the source signal assignment for miscellaneous Interrupt 5(DAI interrupt 27).
17:16 (R/W)	IN4	Extended Input 4. The DAI_EXTD_MISC1.IN4 bit field and DAI_MISC1.IN4 bit field together hold the source signal assignment for miscellaneous Interrupt 4(DAI interrupt 26).
13:12 (R/W)	IN3	Extended Input 3. The DAI_EXTD_MISC1.IN3 bit field and DAI_MISC1.IN3 bit field together hold the source signal assignment for miscellaneous Interrupt 3(DAI interrupt 25).
9:8 (R/W)	IN2	Extended Input 2. The DAI_EXTD_MISC1.IN2 bit field and DAI_MISC1.IN2 bit field together hold the source signal assignment for miscellaneous Interrupt 2(DAI interrupt 24).
5:4 (R/W)	IN1	Extended Input 1. The DAI_EXTD_MISC1.IN1 bit field and DAI_MISC1.IN1 bit field together hold the source signal assignment for miscellaneous Interrupt 1(DAI interrupt 23).
1:0 (R/W)	IN0	Extended Input 0. The DAI_EXTD_MISC1.IN0 bit field and DAI_MISC1.IN0 bit field together hold the source signal assignment for miscellaneous Interrupt 0 (DAI interrupt 22).

Extended Miscellaneous Control Register 2

The DAI_EXTD_MISC2 and DAI_MISC2 register together allows programs to route to the DAI interrupt latch, PCGs hardware completer trigger, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

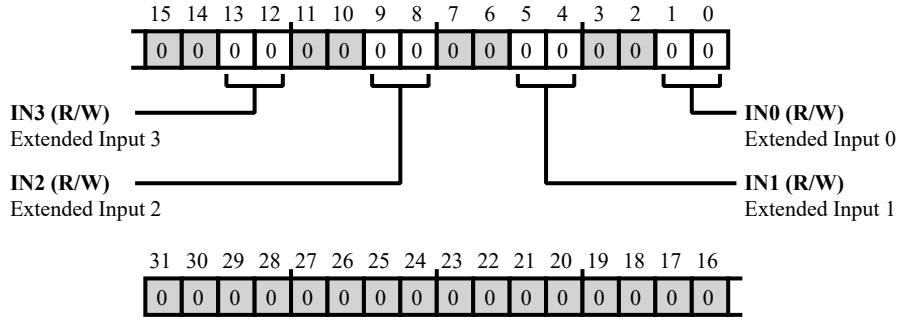


Figure 29-59: DAI_EXTD_MISC2 Register Diagram

Table 29-66: DAI_EXTD_MISC2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:12 (R/W)	IN3	Extended Input 3. The DAI_EXTD_MISC2.IN3 bit field and DAI_MISC1.IN3 bit field together hold the source signal assignment for PCG-F's hardware completer trigger input.
9:8 (R/W)	IN2	Extended Input 2. The DAI_EXTD_MISC2.IN2 bit field and DAI_MISC2.IN2 bit field together hold the source signal assignment for PCG-E's hardware completer trigger input.
5:4 (R/W)	IN1	Extended Input 1. The DAI_EXTD_MISC2.IN1 bit field and DAI_MISC2.IN1 bit field together hold the source signal assignment for PCG-B's hardware completer trigger input.
1:0 (R/W)	IN0	Extended Input 0. The DAI_EXTD_MISC2.IN0 bit field and DAI_MISC2.IN0 bit field together hold the source signal assignment for PCG-A's hardware completer trigger input.

Extended Pin Buffer Enable Register 0

The DAI_EXTD_PBEN0 and DAI_PBEN0 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

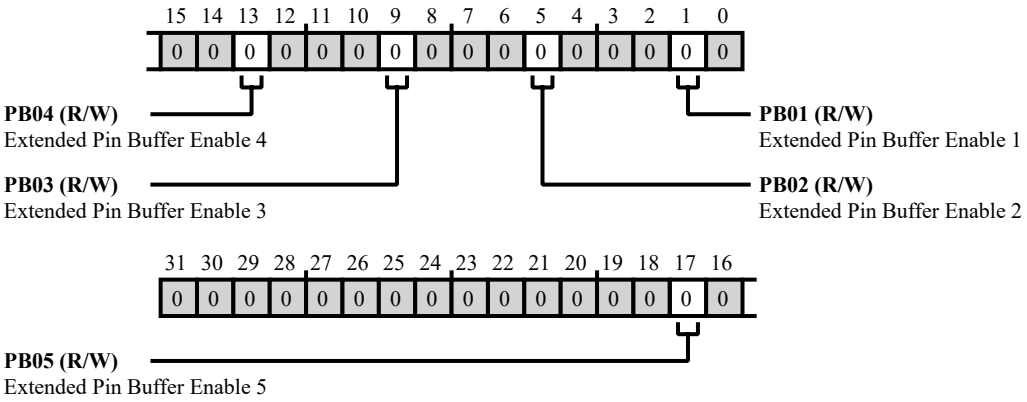


Figure 29-60: DAI_EXTD_PBEN0 Register Diagram

Table 29-67: DAI_EXTD_PBEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB05	Extended Pin Buffer Enable 5. The DAI_EXTD_PBEN0.PB05 bit field and DAI_PBEN0.PB05 bit field together hold the extended source signal assignment that will be routed to the pin buffer enable of DAI port 5.
13 (R/W)	PB04	Extended Pin Buffer Enable 4. The DAI_EXTD_PBEN0.PB04 bit field and DAI_PBEN0.PB04 bit field together hold the extended source signal assignment that will be routed to the pin buffer enable of DAI port 4.
9 (R/W)	PB03	Extended Pin Buffer Enable 3. The DAI_EXTD_PBEN0.PB03 bit field and DAI_PBEN0.PB03 bit field together hold the extended source signal assignment that will be routed to the pin buffer enable of DAI port 3.
5 (R/W)	PB02	Extended Pin Buffer Enable 2. The DAI_EXTD_PBEN0.PB02 bit field and DAI_PBEN0.PB02 bit field together hold the extended source signal assignment that will be routed to the pin buffer enable of DAI port 2.
1 (R/W)	PB01	Extended Pin Buffer Enable 1. The DAI_EXTD_PBEN0.PB01 bit field and DAI_PBEN0.PB01 bit field together hold the extended source signal assignment that will be routed to the pin buffer enable of DAI port 1.

Extended Pin Buffer Enable Register 1

The DAI_EXTD_PBEN1 and DAI_PBEN1 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

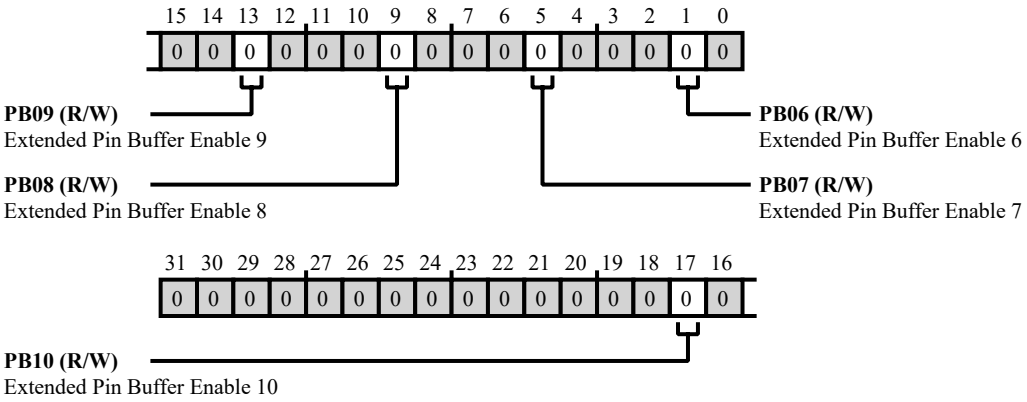


Figure 29-61: DAI_EXTD_PBEN1 Register Diagram

Table 29-68: DAI_EXTD_PBEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB10	Extended Pin Buffer Enable 10. The DAI_EXTD_PBEN1.PB10 bit field and DAI_PBEN0.PB10 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 10.
13 (R/W)	PB09	Extended Pin Buffer Enable 9. The DAI_EXTD_PBEN1.PB09 bit field and DAI_PBEN0.PB09 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 9.
9 (R/W)	PB08	Extended Pin Buffer Enable 8. The DAI_EXTD_PBEN1.PB08 bit field and DAI_PBEN0.PB08 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 8.
5 (R/W)	PB07	Extended Pin Buffer Enable 7. The DAI_EXTD_PBEN1.PB07 bit field and DAI_PBEN0.PB07 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 7.
1 (R/W)	PB06	Extended Pin Buffer Enable 6. The DAI_EXTD_PBEN1.PB06 bit field and DAI_PBEN0.PB06 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 6.

Extended Pin Buffer Enable Register 2

The DAI_EXTD_PBEN2 and DAI_PBEN2 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

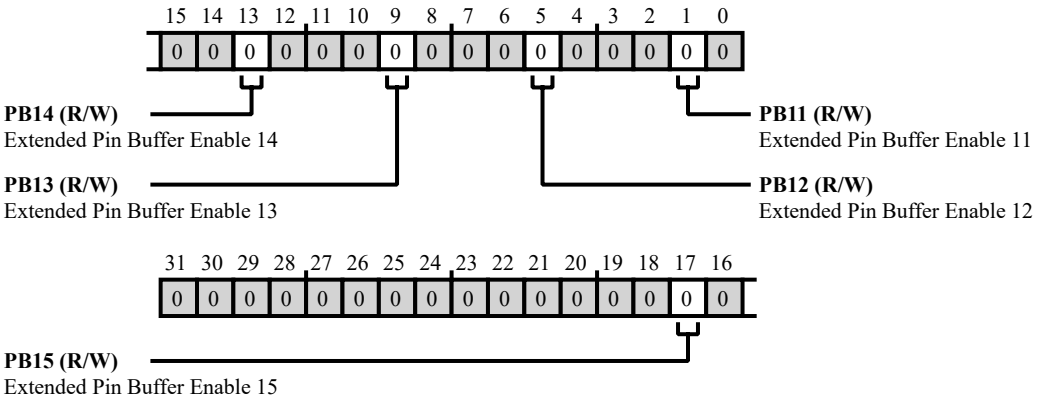


Figure 29-62: DAI_EXTD_PBEN2 Register Diagram

Table 29-69: DAI_EXTD_PBEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB15	Extended Pin Buffer Enable 15. The DAI_EXTD_PBEN2.PB15 bit field and DAI_PBEN0.PB15 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 15.
13 (R/W)	PB14	Extended Pin Buffer Enable 14. The DAI_EXTD_PBEN2.PB14 bit field and DAI_PBEN0.PB14 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 14.
9 (R/W)	PB13	Extended Pin Buffer Enable 13. The DAI_EXTD_PBEN2.PB13 bit field and DAI_PBEN0.PB13 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 13.
5 (R/W)	PB12	Extended Pin Buffer Enable 12. The DAI_EXTD_PBEN2.PB12 bit field and DAI_PBEN0.PB12 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 12.
1 (R/W)	PB11	Extended Pin Buffer Enable 11. The DAI_EXTD_PBEN2.PB11 bit field and DAI_PBEN0.PB11 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 11.

Extended Pin Buffer Enable Register 3

The DAI_EXTD_PBEN3 and DAI_PBEN3 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

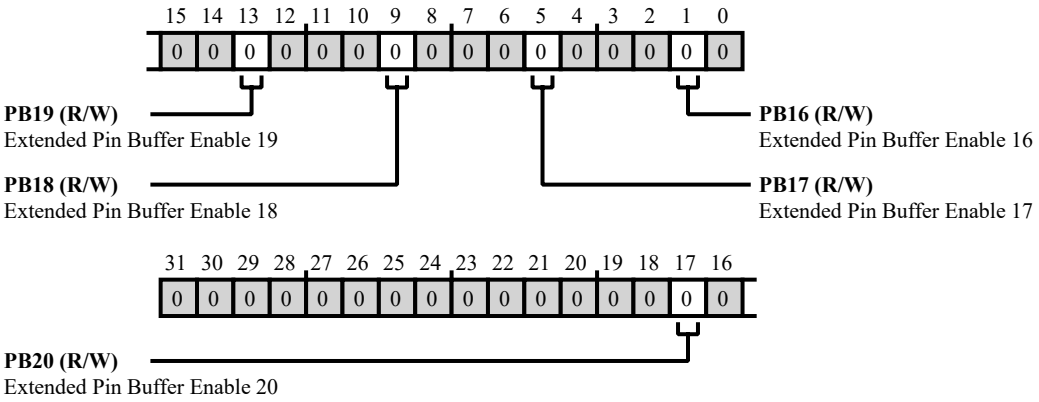


Figure 29-63: DAI_EXTD_PBEN3 Register Diagram

Table 29-70: DAI_EXTD_PBEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB20	Extended Pin Buffer Enable 20. The DAI_EXTD_PBEN3.PB20 bit field and DAI_PBEN0.PB20 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 20.
13 (R/W)	PB19	Extended Pin Buffer Enable 19. The DAI_EXTD_PBEN3.PB19 bit field and DAI_PBEN0.PB19 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 19.
9 (R/W)	PB18	Extended Pin Buffer Enable 18. The DAI_EXTD_PBEN3.PB18 bit field and DAI_PBEN0.PB18 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 18.
5 (R/W)	PB17	Extended Pin Buffer Enable 17. The DAI_EXTD_PBEN3.PB17 bit field and DAI_PBEN0.PB17 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 17.
1 (R/W)	PB16	Extended Pin Buffer Enable 16. The DAI_EXTD_PBEN3.PB16 bit field and DAI_PBEN0.PB16 bit field together hold the extended source signal assignment which will be routed to the pin buffer enable of DAI port 16.

Extended Pin Buffer Assignment Register 0

The DAI_EXTD_PIN0 and DAI_PIN0 register routes extended physical pins that are connected to a bonded pad.

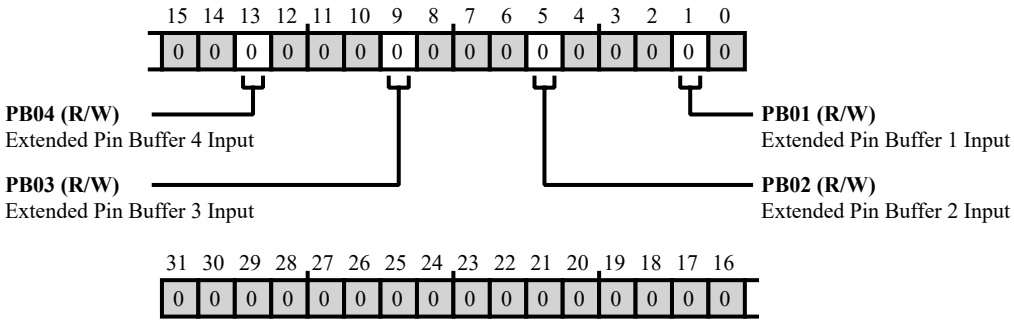


Figure 29-64: DAI_EXTD_PIN0 Register Diagram

Table 29-71: DAI_EXTD_PIN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB04	Extended Pin Buffer 4 Input. DAI_EXTD_PIN0.PB04 and DAI_PIN0.PB04 together hold the source signal assignment that will be routed to the DAI_PIN0.PB04 destination. Refer to the Group D Signals table for source and destination mappings.
9 (R/W)	PB03	Extended Pin Buffer 3 Input. DAI_EXTD_PIN0.PB03 and DAI_PIN.PB03 together hold the source signal assignment that will be routed to the DAI_PIN0.PB03 destination. Refer to the Group D Signals table for source and destination mappings.
5 (R/W)	PB02	Extended Pin Buffer 2 Input. DAI_EXTD_PIN0.PB02 and DAI_PIN0.PB02 together hold the source signal assignment that will be routed to the DAI_PIN0.PB02 destination. Refer to the Group D Signals table for source and destination mappings.
1 (R/W)	PB01	Extended Pin Buffer 1 Input. DAI_EXTD_PIN0.PB01 and DAI_PIN0.PB01 together hold the source signal assignment that will be routed to the DAI_PIN0.PB01 destination. Refer to the Group D Signals table for source and destination mappings.

Extended Pin Buffer Assignment Register 1

The DAI_EXTD_PIN1 and DAI_PIN1 register routes extended physical pins that are connected to a bonded pad.

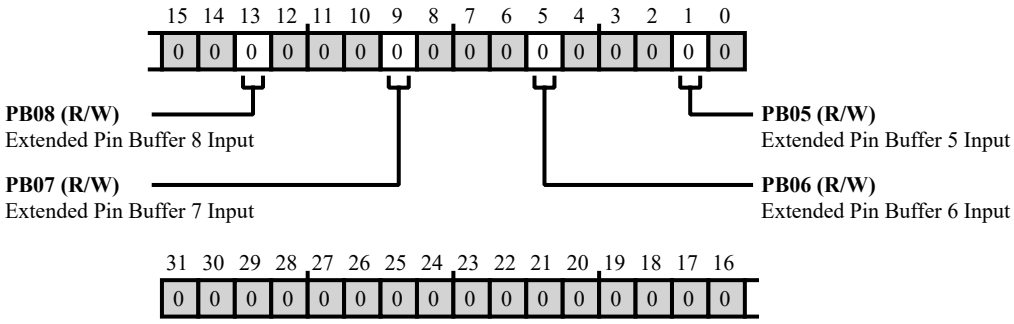


Figure 29-65: DAI_EXTD_PIN1 Register Diagram

Table 29-72: DAI_EXTD_PIN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB08	Extended Pin Buffer 8 Input. DAI_EXTD_PIN1.PB08 and DAI_PIN1.PB08 together hold the source signal assignment that will be routed to the DAI_PIN1.PB08 destination. Refer to the Group D Signals table for source and destination mappings.
9 (R/W)	PB07	Extended Pin Buffer 7 Input. DAI_EXTD_PIN1.PB07 and DAI_PIN1.PB07 together hold the source signal assignment that will be routed to the DAI_PIN1.PB07 destination. Refer to the Group D Signals table for source and destination mappings.
5 (R/W)	PB06	Extended Pin Buffer 6 Input. DAI_EXTD_PIN1.PB06 and DAI_PIN1.PB06 together the source signal assignment that will be routed to the DAI_PIN1.PB06 destination. Refer to the Group D Signals table for source and destination mappings.
1 (R/W)	PB05	Extended Pin Buffer 5 Input. DAI_EXTD_PIN1.PB05 and DAI_PIN1.PB05 together hold the source signal assignment that will be routed to the DAI_PIN1.PB05 destination. Refer to the Group D Signals table for source and destination mappings.

Extended Pin Buffer Assignment Register 2

The DAI_EXTD_PIN2 and DAI_PIN2 register routes extended physical pins that are connected to a bonded pad.

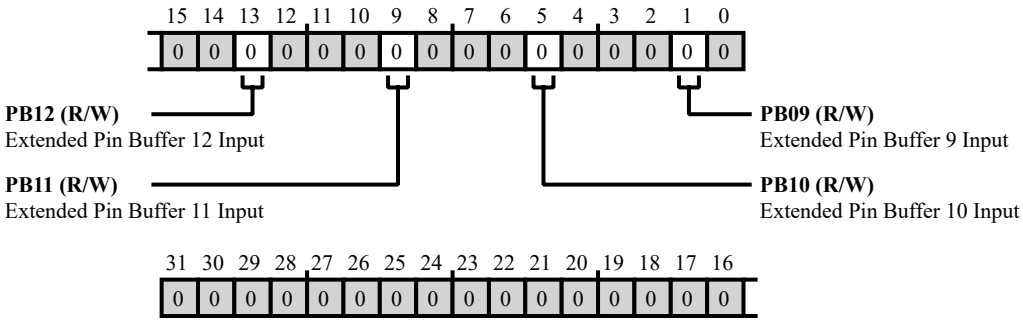


Figure 29-66: DAI_EXTD_PIN2 Register Diagram

Table 29-73: DAI_EXTD_PIN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB12	Extended Pin Buffer 12 Input. DAI_EXTD_PIN2.PB12 and DAI_PIN2.PB12 together hold the source signal assignment that will be routed to the DAI_PIN2.PB12 destination. Refer to the Group D Signals table for source and destination mappings.
9 (R/W)	PB11	Extended Pin Buffer 11 Input. DAI_EXTD_PIN2.PB11 and DAI_PIN2.PB11 together hold the source signal assignment that will be routed to the DAI_PIN2.PB11 destination. Refer to the Group D Signals table for source and destination mappings.
5 (R/W)	PB10	Extended Pin Buffer 10 Input. DAI_EXTD_PIN2.PB10 and DAI_PIN2.PB10 together hold the source signal assignment that will be routed to the DAI_PIN2.PB10 destination. Refer to the Group D Signals table for source and destination mappings.
1 (R/W)	PB09	Extended Pin Buffer 9 Input. DAI_EXTD_PIN2.PB09 and DAI_PIN2.PB09 together hold the source signal assignment that will be routed to the DAI_PIN2.PB09 destination. Refer to the Group D Signals table for source and destination mappings.

Extended Pin Buffer Assignment Register 3

The DAI_EXTD_PIN3 and DAI_PIN3 register routes extended physical pins that are connected to a bonded pad.

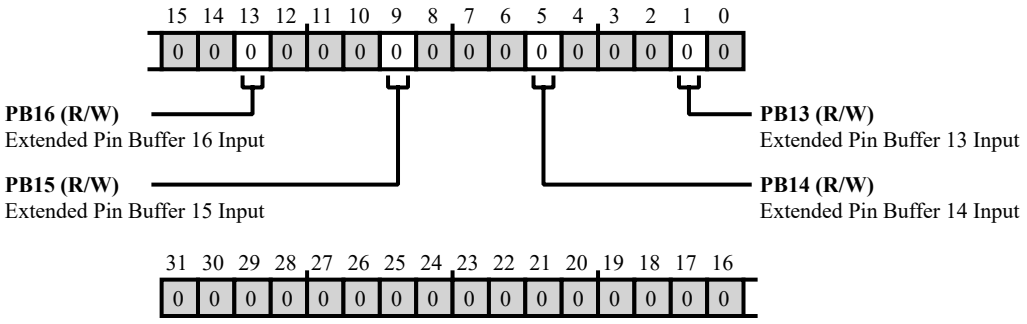


Figure 29-67: DAI_EXTD_PIN3 Register Diagram

Table 29-74: DAI_EXTD_PIN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB16	Extended Pin Buffer 16 Input. DAI_EXTD_PIN3.PB16 and DAI_PIN3.PB16 together hold the source signal assignment that will be routed to the DAI_PIN3.PB16 destination. Refer to the Group D Signals table for source and destination mappings.
9 (R/W)	PB15	Extended Pin Buffer 15 Input. DAI_EXTD_PIN3.PB15 and DAI_PIN3.PB15 together hold the source signal assignment that will be routed to the DAI_PIN3.PB15 destination. Refer to the Group D Signals table for source and destination mappings.
5 (R/W)	PB14	Extended Pin Buffer 14 Input. DAI_EXTD_PIN3.PB14 and DAI_PIN3.PB14 together hold the source signal assignment that will be routed to the DAI_PIN3.PB14 destination. Refer to the Group D Signals table for source and destination mappings.
1 (R/W)	PB13	Extended Pin Buffer 13 Input. DAI_EXTD_PIN3.PB13 and DAI_PIN3.PB13 together hold the source signal assignment that will be routed to the DAI_PIN3.PB13 destination. Refer to the Group D Signals table for source and destination mappings.

Extended Pin Buffer Assignment Register 4

The DAI_EXTD_PIN4 and DAI_PIN4 register routes extended physical pins that are connected to a bonded pad.

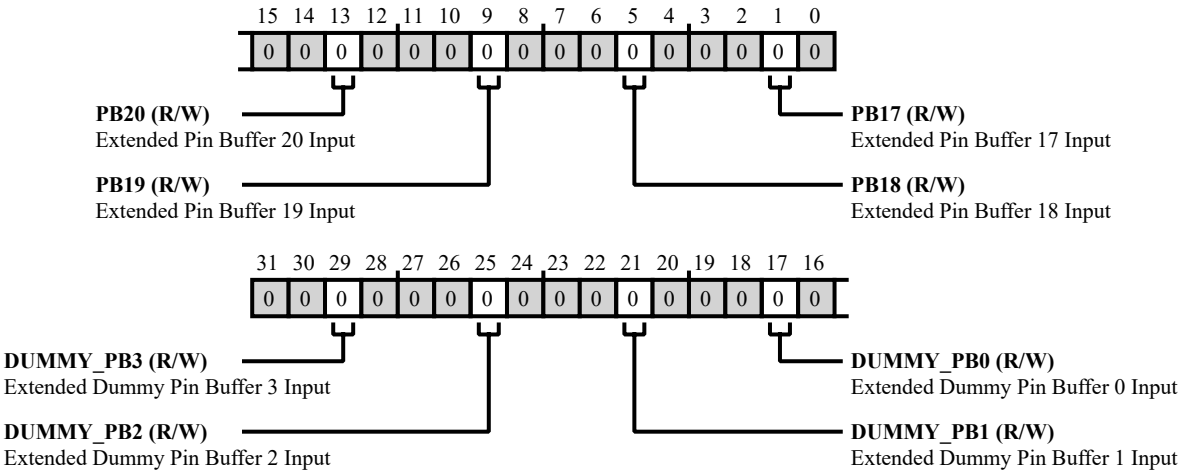


Figure 29-68: DAI_EXTD_PIN4 Register Diagram

Table 29-75: DAI_EXTD_PIN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	DUMMY_PB3	Extended Dummy Pin Buffer 3 Input. DAI_EXTD_PIN4.DUMMY_PB3 and DAI_PIN5.DUMMY_PB3 together hold the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB3 destination. Refer to the Group D Signals table for source and destination mappings.
25 (R/W)	DUMMY_PB2	Extended Dummy Pin Buffer 2 Input. DAI_EXTD_PIN4.DUMMY_PB2 and DAI_PIN5.DUMMY_PB2 together hold the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB2 destination. Refer to the Group D Signals table for source and destination mappings.
21 (R/W)	DUMMY_PB1	Extended Dummy Pin Buffer 1 Input. DAI_EXTD_PIN4.DUMMY_PB1 and DAI_PIN5.DUMMY_PB1 together hold the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB1 destination. Refer to the Group D Signals table for source and destination mappings.
17 (R/W)	DUMMY_PB0	Extended Dummy Pin Buffer 0 Input. DAI_EXTD_PIN4.DUMMY_PB0 and DAI_PIN5.DUMMY_PB0 together hold the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB0 destination. Refer to the Group D Signals table for source and destination mappings.

Table 29-75: DAI_EXTD_PIN4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB20	Extended Pin Buffer 20 Input. `DAI_EXTD_PIN4.PB20` and DAI_PIN4.PB20 together hold the source signal assignment that will be routed to the DAI_PIN4.PB20 destination. Refer to the Group D Signals table for source and destination mappings.
9 (R/W)	PB19	Extended Pin Buffer 19 Input. `DAI_EXTD_PIN4.PB19` and DAI_PIN4.PB19 together hold the source signal assignment that will be routed to the DAI_PIN4.PB19 destination. Refer to the Group D Signals table for source and destination mappings.
5 (R/W)	PB18	Extended Pin Buffer 18 Input. `DAI_EXTD_PIN4.PB18` an DAI_PIN4.PB18 together hold the source signal assignment that will be routed to the DAI_PIN4.PB18 destination. Refer to the Group D Signals table for source and destination mappings.
1 (R/W)	PB17	Extended Pin Buffer 17 Input. `DAI_EXTD_PIN4.PB17` and DAI_PIN4.PB17 together hold the source signal assignment that will be routed to the DAI_PIN4.PB17 destination. Refer to the Group D Signals table for source and destination mappings.

Frame Sync Routing Control Register 0

The DAI_FS0 register routes frame syncs to the serial ports.

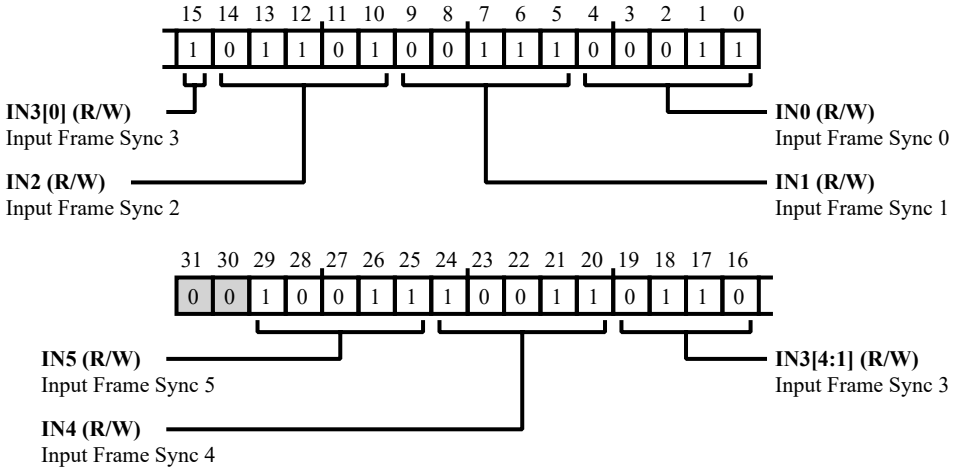


Figure 29-69: DAI_FS0 Register Diagram

Table 29-76: DAI_FS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Frame Sync 5. DAI_FS0.IN5 holds the source signal assignment that will be routed to the DAI_FS0.IN5 destination. Refer to the Group C Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Frame Sync 4. DAI_FS0.IN4 holds the source signal assignment that will be routed to the DAI_FS0.IN4 destination. Refer to the Group C Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS0.IN3 holds the source signal assignment that will be routed to the DAI_FS0.IN3 destination. Refer to the Group C Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS0.IN2 holds the source signal assignment that will be routed to the DAI_FS0.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS0.IN1 holds the source signal assignment that will be routed to the DAI_FS0.IN1 destination. Refer to the Group C Signals table for source and destination mappings.

Table 29-76: DAI_FS0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Frame Sync 0. `DAI_FS0.IN0` holds the source signal assignment that will be routed to the `DAI_FS0.IN0` destination. Refer to the Group C Signals table for source and destination mappings.

Frame Sync Routing Control Register 1

The DAI_FS1 register routes frame syncs to the asynchronous sample rate converter (ASRC).

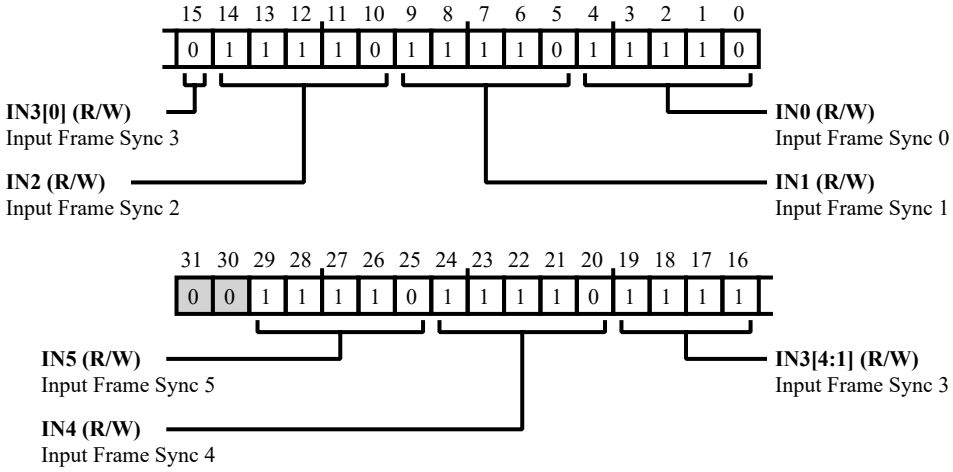


Figure 29-70: DAI_FS1 Register Diagram

Table 29-77: DAI_FS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Frame Sync 5. DAI_FS1.IN5 holds the source signal assignment that will be routed to the DAI_FS1.IN5 destination. Refer to the Group C Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Frame Sync 4. DAI_FS1.IN4 holds the source signal assignment that will be routed to the DAI_FS1.IN4 destination. Refer to the Group C Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS1.IN3 holds the source signal assignment that will be routed to the DAI_FS1.IN3 destination. Refer to the Group C Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS1.IN2 holds the source signal assignment that will be routed to the DAI_FS1.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS1.IN1 holds the source signal assignment that will be routed to the DAI_FS1.IN1 destination. Refer to the Group C Signals table for source and destination mappings.

Table 29-77: DAI_FS1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Frame Sync 0. `DAI_FS1.IN0` holds the source signal assignment that will be routed to the `DAI_FS1.IN0` destination. Refer to the Group C Signals table for source and destination mappings.

Frame Sync Routing Control Register 2

The DAI_FS2 register routes frame syncs to the asynchronous sample rate converter (ASRC).

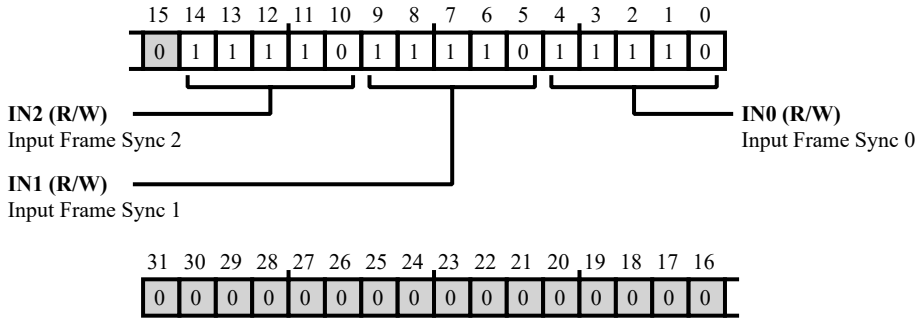


Figure 29-71: DAI_FS2 Register Diagram

Table 29-78: DAI_FS2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS2.IN2 holds the source signal assignment that will be routed to the DAI_FS2.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. The DAI_FS2.IN1 bit field routes the frame sync output to the ASRC3 frame sync input port.
4:0 (R/W)	IN0	Input Frame Sync 0. DAI_FS2.IN0 holds the source signal assignment that will be routed to the DAI_FS2.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Frame Sync Routing Control Register 4

The DAI_FS4 register routes frame syncs to the asynchronous sample rate converter (ASRC).

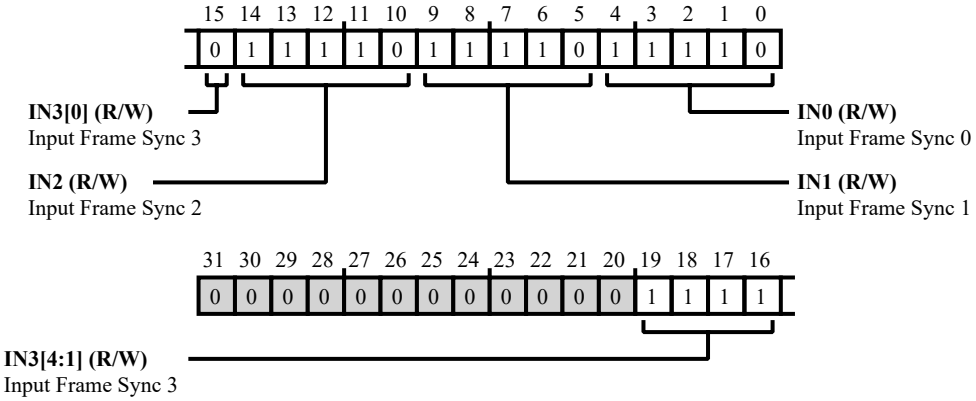


Figure 29-72: DAI_FS4 Register Diagram

Table 29-79: DAI_FS4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS4.IN3 holds the source signal assignment that will be routed to the DAI_FS4.IN3 destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS4.IN2 holds the source signal assignment that will be routed to the DAI_FS4.IN2 destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS4.IN1 holds the source signal assignment that will be routed to the DAI_FS4.IN1 destination. Refer to the Group C Signals table for source and destination mappings.
4:0 (R/W)	IN0	Input Frame Sync 0. DAI_FS4.IN0 holds the source signal assignment that will be routed to the DAI_FS4.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Frame Sync Routing Control Register 5

The DAI_FS5 register routes frame syncs to the asynchronous sample rate converter (ASRC).

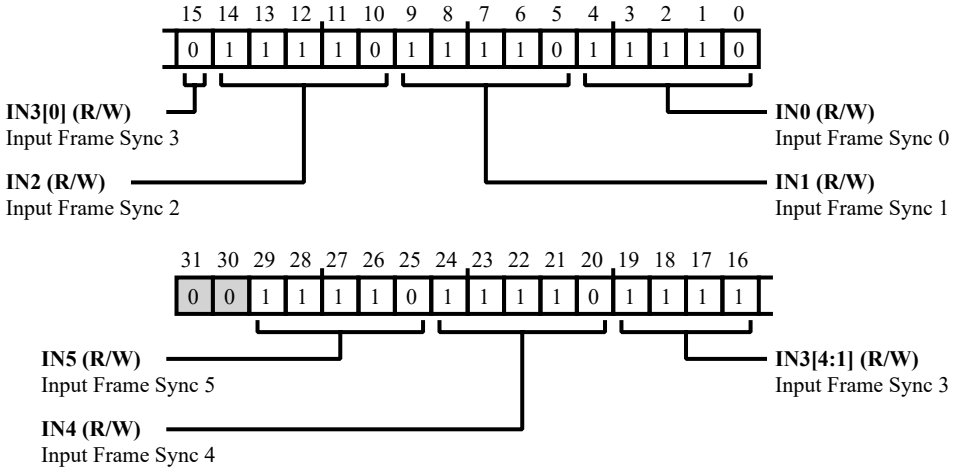


Figure 29-73: DAI_FS5 Register Diagram

Table 29-80: DAI_FS5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Frame Sync 5. DAI_FS5.IN5 holds the source signal assignment that will be routed to the DAI_FS5.IN5 destination. Refer to the Group C Signals table for source and destination mappings.
24:20 (R/W)	IN4	Input Frame Sync 4. DAI_FS5.IN4 holds the source signal assignment that will be routed to the DAI_FS5.IN4 destination. Refer to the Group C Signals table for source and destination mappings.
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS5.IN3 holds the source signal assignment that will be routed to the DAI_FS5.IN3 destination. Refer to the Group C Signals table for source and destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS5.IN2 holds the source signal assignment that will be routed to the DAI_FS5.IN2 destination. Refer to the Group C Signals table for source and destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS5.IN1 holds the source signal assignment that will be routed to the DAI_FS5.IN1 destination. Refer to the Group C Signals table for source and destination mappings.

Table 29-80: DAI_FS5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Frame Sync 0. `DAI_FS5.IN0` holds the source signal assignment that will be routed to the `DAI_FS5.IN0` destination. Refer to the Group C Signals table for source and destination mappings.

Frame Sync Routing Control Register 6

The DAI_FS6 register routes frame syncs to the asynchronous sample rate converter (ASRC).

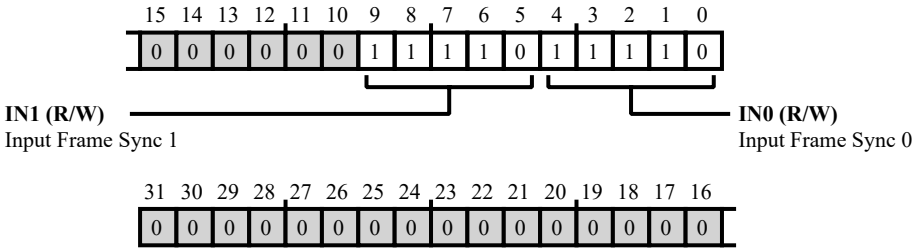


Figure 29-74: DAI_FS6 Register Diagram

Table 29-81: DAI_FS6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS6.IN1 holds the source signal assignment that will be routed to the DAI_FS6.IN1 destination. Refer to the Group C Signals table for source and destination mappings.
4:0 (R/W)	IN0	Input Frame Sync 0. DAI_FS6.IN0 holds the source signal assignment that will be routed to the DAI_FS6.IN0 destination. Refer to the Group C Signals table for source and destination mappings.

Global SPORT Interrupt Grouping Register

The DAI_GBL_INT_EN register contains bits to configure the group interrupts and triggers of the SPORTs when using the SPORT global enable feature.

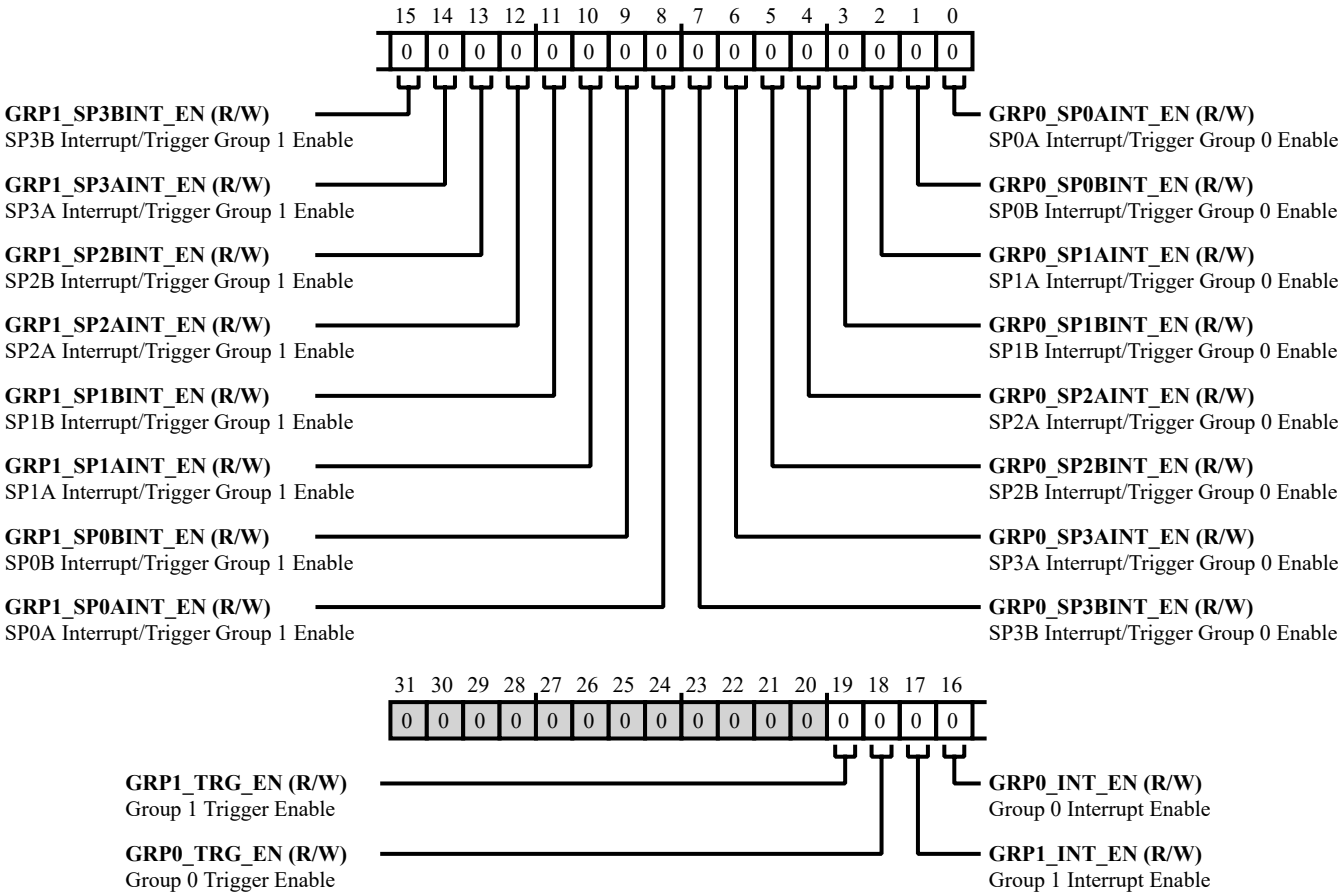


Figure 29-75: DAI_GBL_INT_EN Register Diagram

Table 29-82: DAI_GBL_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	GRP1_TRG_EN	Group 1 Trigger Enable. The DAI_GBL_INT_EN.GRP1_TRG_EN bit enables the trigger-out for group 1.
18 (R/W)	GRP0_TRG_EN	Group 0 Trigger Enable. The DAI_GBL_INT_EN.GRP0_TRG_EN bit enables the trigger-out for group 0.
17 (R/W)	GRP1_INT_EN	Group 1 Interrupt Enable. The DAI_GBL_INT_EN.GRP1_INT_EN bit enables the interrupt for group 1.

Table 29-82: DAI_GBL_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	GRP0_INT_EN	Group 0 Interrupt Enable. The DAI_GBL_INT_EN.GRP0_INT_EN bit enables the interrupt for group 0.
15 (R/W)	GRP1_SP3BINT_EN	SP3B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP3BINT_EN bit enables the interrupt or trigger-out for SPORT3 B when it is part of group 1.
14 (R/W)	GRP1_SP3AINT_EN	SP3A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP3AINT_EN bit enables the interrupt or trigger-out for SPORT3 A when it is part of group 1.
13 (R/W)	GRP1_SP2BINT_EN	SP2B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP2BINT_EN bit enables the interrupt or trigger-out for SPORT2 B when it is part of group 1.
12 (R/W)	GRP1_SP2AINT_EN	SP2A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP2AINT_EN bit enables the interrupt or trigger-out for SPORT2 A when it is part of group 1.
11 (R/W)	GRP1_SP1BINT_EN	SP1B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP1BINT_EN bit enables the interrupt or trigger-out for SPORT1 B when it is part of group 1.
10 (R/W)	GRP1_SP1AINT_EN	SP1A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP1AINT_EN bit enables the interrupt or trigger-out for SPORT1 A when it is part of group 1.
9 (R/W)	GRP1_SP0BINT_EN	SP0B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP0BINT_EN bit enables the interrupt or trigger-out for SPORT0 B when it is part of group 1.
8 (R/W)	GRP1_SP0AINT_EN	SP0A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP0AINT_EN bit enables the interrupt or trigger-out for SPORT0 A when it is part of group 1.
7 (R/W)	GRP0_SP3BINT_EN	SP3B Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP3BINT_EN bit enables the interrupt or trigger-out for SPORT3 B when it is part of group 0.
6 (R/W)	GRP0_SP3AINT_EN	SP3A Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP3AINT_EN bit enables the interrupt or trigger-out for SPORT3 A when it is part of group 0.
5 (R/W)	GRP0_SP2BINT_EN	SP2B Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP2BINT_EN bit enables the interrupt or trigger-out for SPORT2 B when it is part of group 0.

Table 29-82: DAI_GBL_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	GRP0_SP2AINT_EN	SP2A Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP2AINT_EN bit enables the interrupt or trigger-out for SPORT2 A when it is part of group 0.
3 (R/W)	GRP0_SP1BINT_EN	SP1B Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP1BINT_EN bit enables the interrupt or trigger-out for SPORT1 B when it is part of group 0.
2 (R/W)	GRP0_SP1AINT_EN	SP1A Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP1AINT_EN bit enables the interrupt or trigger-out for SPORT1 A when it is part of group 0.
1 (R/W)	GRP0_SP0BINT_EN	SP0B Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP0BINT_EN bit enables the interrupt or trigger-out for SPORT0 B when it is part of group 0.
0 (R/W)	GRP0_SP0AINT_EN	SP0A Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP0AINT_EN bit enables the interrupt or trigger-out for SPORT0 A when it is part of group 0.

Global PCG Enable Control Register

The DAI_GBL_PCG_EN register bits indicate the status (signal high or low) for each pin. The individual bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

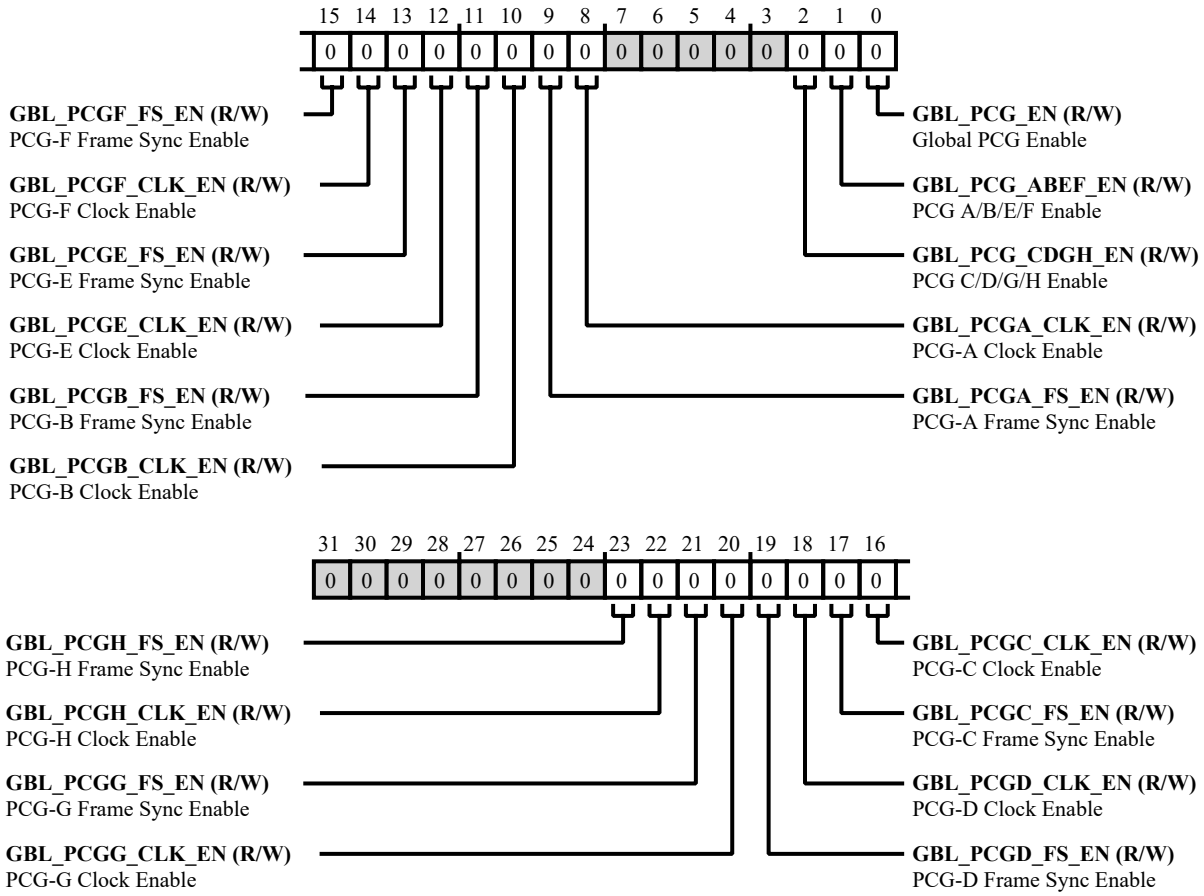


Figure 29-76: DAI_GBL_PCG_EN Register Diagram

Table 29-83: DAI_GBL_PCG_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/W)	GBL_PCGH_FS_EN	PCG-H Frame Sync Enable. The DAI_GBL_PCG_EN.GBL_PCGH_FS_EN bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
22 (R/W)	GBL_PCGH_CLK_EN	PCG-H Clock Enable. The DAI_GBL_PCG_EN.GBL_PCGH_CLK_EN bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.

Table 29-83: DAI_GBL_PCG_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W)	GBL_PCGG_FS_EN	PCG-G Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGG_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
20 (R/W)	GBL_PCGG_CLK_EN	PCG-G Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGG_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
19 (R/W)	GBL_PCGD_FS_EN	PCG-D Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGD_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
18 (R/W)	GBL_PCGD_CLK_EN	PCG-D Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGD_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
17 (R/W)	GBL_PCGC_FS_EN	PCG-C Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGC_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
16 (R/W)	GBL_PCGC_CLK_EN	PCG-C Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGC_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
15 (R/W)	GBL_PCGF_FS_EN	PCG-F Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGF_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
14 (R/W)	GBL_PCGF_CLK_EN	PCG-F Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGF_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
13 (R/W)	GBL_PCGE_FS_EN	PCG-E Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGE_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
12 (R/W)	GBL_PCGE_CLK_EN	PCG-E Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGE_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
11 (R/W)	GBL_PCGB_FS_EN	PCG-B Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGB_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.

Table 29-83: DAI_GBL_PCG_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	GBL_PCGB_CLK_EN	PCG-B Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGB_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
9 (R/W)	GBL_PCGA_FS_EN	PCG-A Frame Sync Enable. The `DAI_GBL_PCG_EN.GBL_PCGA_FS_EN` bit, when set (=1), enables the frame sync generation and when cleared (=0) disables the frame sync generation.
8 (R/W)	GBL_PCGA_CLK_EN	PCG-A Clock Enable. The `DAI_GBL_PCG_EN.GBL_PCGA_CLK_EN` bit, when set (=1), enables the clock generation and when cleared (=0) disables the clock generation.
2 (R/W)	GBL_PCG_CDGH_EN	PCG C/D/G/H Enable. The `DAI_GBL_PCG_EN.GBL_PCG_CDGH_EN` bit, when set (=1), along with bits [23:16] of the same register enables the respective PCGs within a DAI1. This bit when cleared (=0) disables the PCGs within DAI1.
1 (R/W)	GBL_PCG_ABEF_EN	PCG A/B/E/F Enable. The `DAI_GBL_PCG_EN.GBL_PCG_ABEF_EN` bit, when set (=1), along with bits [15:8] of the same register enables the respective PCGs within a DAI0. This bit, when cleared (=0), disables the PCGs within DAI0.
0 (R/W)	GBL_PCG_EN	Global PCG Enable. This bit globally enables the PCGs across DAI0 and DAI1. The `DAI_GBL_PCG_EN.GBL_PCG_EN` bit, when set (=1), along with bit [2:1] and [23:8] enables all the PCGs across DAI0 and DAI1. This bit when cleared (=0) disables all the PCGs across DAI0 and DAI1.

Global SPORT Enable Register

The DAI_GBL_SP_EN register indicate the status (signal high or low) for each pin. The individual bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

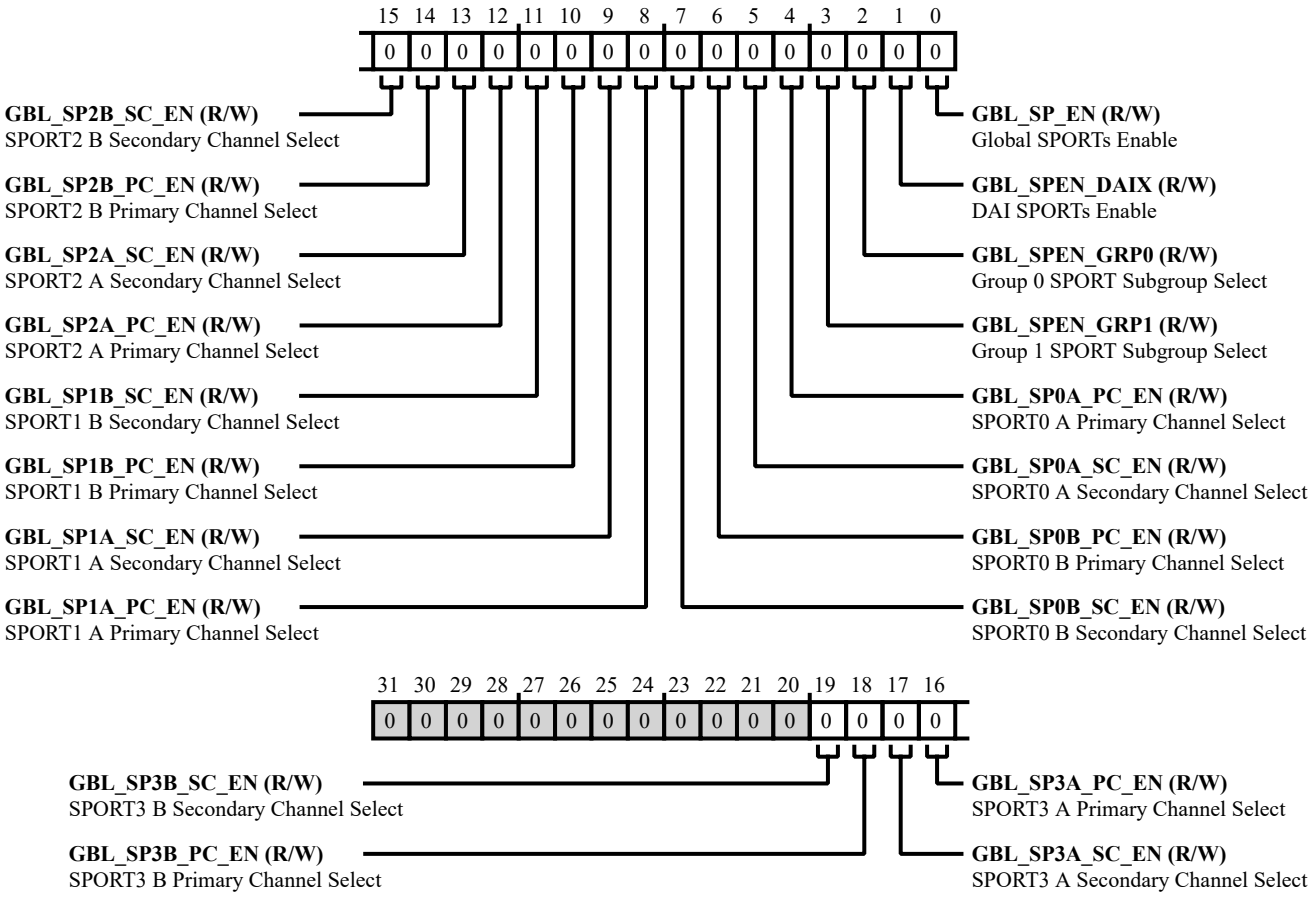


Figure 29-77: DAI_GBL_SP_EN Register Diagram

Table 29-84: DAI_GBL_SP_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	GBL_SP3B_SC_EN	SPORT3 B Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP3B_SC_EN bit selects SPORT3 B as the secondary channel for global enable.
18 (R/W)	GBL_SP3B_PC_EN	SPORT3 B Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP3B_PC_EN bit selects SPORT3 B as the primary channel for global enable.

Table 29-84: DAI_GBL_SP_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	GBL_SP3A_SC_EN	SPORT3 A Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP3A_SC_EN bit selects SPORT3 A as the secondary channel for global enable.
16 (R/W)	GBL_SP3A_PC_EN	SPORT3 A Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP3A_PC_EN bit selects SPORT3 A as the primary channel for global enable.
15 (R/W)	GBL_SP2B_SC_EN	SPORT2 B Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP2B_SC_EN bit selects SPORT2 B as the secondary channel for global enable.
14 (R/W)	GBL_SP2B_PC_EN	SPORT2 B Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP2B_PC_EN bit selects SPORT2 B as the primary channel for global enable.
13 (R/W)	GBL_SP2A_SC_EN	SPORT2 A Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP2A_SC_EN bit selects SPORT2 A as the secondary channel for global enable.
12 (R/W)	GBL_SP2A_PC_EN	SPORT2 A Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP2A_PC_EN bit selects SPORT2 A as the primary channel for global enable.
11 (R/W)	GBL_SP1B_SC_EN	SPORT1 B Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP1B_SC_EN bit selects SPORT1 B as the secondary channel for global enable.
10 (R/W)	GBL_SP1B_PC_EN	SPORT1 B Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP1B_PC_EN bit selects SPORT1 B as the primary channel for global enable.
9 (R/W)	GBL_SP1A_SC_EN	SPORT1 A Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP1A_SC_EN bit selects SPORT1 A as the secondary channel for global enable.
8 (R/W)	GBL_SP1A_PC_EN	SPORT1 A Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP1A_PC_EN bit selects SPORT1 A as the primary channel for global enable.
7 (R/W)	GBL_SP0B_SC_EN	SPORT0 B Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP0B_SC_EN bit selects SPORT0 B as the secondary channel for global enable.
6 (R/W)	GBL_SP0B_PC_EN	SPORT0 B Primary Channel Select. Selects SPORT0 B primary channel for global enable.

Table 29-84: DAI_GBL_SP_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	GBL_SP0A_SC_EN	SPORT0 A Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP0A_SC_EN` bit selects SPORT0 A as the secondary channel for global enable.
4 (R/W)	GBL_SP0A_PC_EN	SPORT0 A Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP0A_PC_EN` bit selects SPORT0 A as the primary channel for global enable.
3 (R/W)	GBL_SPEN_GRP1	Group 1 SPORT Subgroup Select. The `DAI_GBL_SP_EN.GBL_SPEN_GRP1` bit creates a subgroup of selected SPORTs.
2 (R/W)	GBL_SPEN_GRP0	Group 0 SPORT Subgroup Select. The `DAI_GBL_SP_EN.GBL_SPEN_GRP0` bit creates a subgroup of selected SPORTs.
1 (R/W)	GBL_SPEN_DAIX	DAI SPORTs Enable. The `DAI_GBL_SP_EN.GBL_SPEN_DAIX` bit enables the SPORTs selected within a DAI.
0 (R/W)	GBL_SP_EN	Global SPORTs Enable. The `DAI_GBL_SP_EN.GBL_SP_EN` bit enables the selected SPORTs.

Falling-Edge Interrupt Mask Register

The DAI_IMSK_FE register masks and unmasks interrupts generated on the falling edge of a waveform. Note that any of the Group E signals can be mapped to any of the Miscellaneous interrupts (9-0).

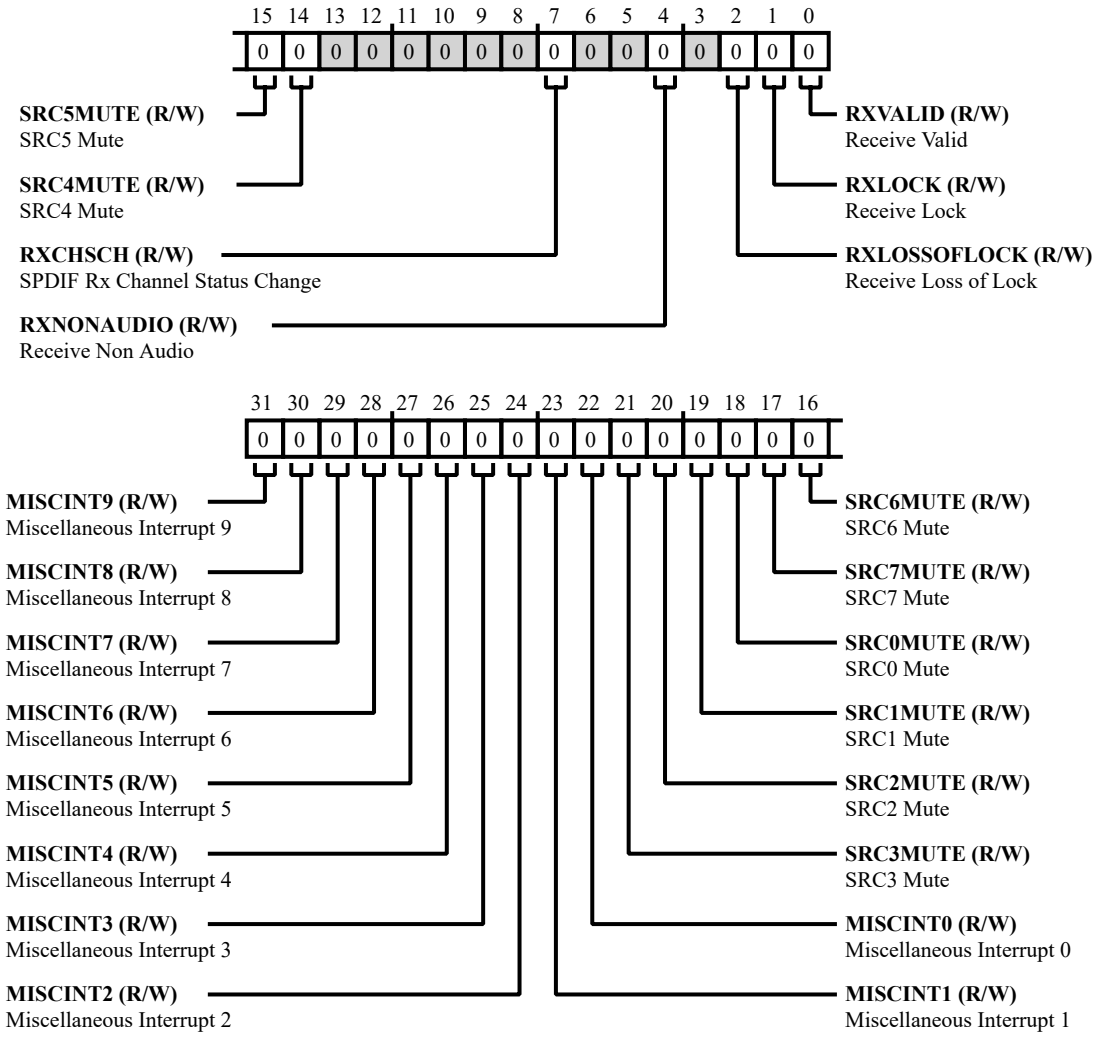


Figure 29-78: DAI_IMSK_FE Register Diagram

Table 29-85: DAI_IMSK_FE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MISCINT9	Miscellaneous Interrupt 9. Setting the DAI_IMSK_FE.MISCINT9 bit unmasks the interrupt for the falling edge of the routed signal.

Table 29-85: DAI_IMSK_FE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	MISCINT8	Miscellaneous Interrupt 8. Setting the DAI_IMSK_FE.MISCINT8 bit unmasks the interrupt for the falling edge of the routed signal.
29 (R/W)	MISCINT7	Miscellaneous Interrupt 7. Setting the DAI_IMSK_FE.MISCINT7 bit unmasks the interrupt for the falling edge of the routed signal.
28 (R/W)	MISCINT6	Miscellaneous Interrupt 6. Setting the DAI_IMSK_FE.MISCINT6 bit unmasks the interrupt for the falling edge of the routed signal.
27 (R/W)	MISCINT5	Miscellaneous Interrupt 5. Setting the DAI_IMSK_FE.MISCINT5 bit unmasks the interrupt for the falling edge of the routed signal.
26 (R/W)	MISCINT4	Miscellaneous Interrupt 4. Setting the DAI_IMSK_FE.MISCINT4 bit unmasks the interrupt for the falling edge of the routed signal.
25 (R/W)	MISCINT3	Miscellaneous Interrupt 3. Setting the DAI_IMSK_FE.MISCINT3 bit unmasks the interrupt for the falling edge of the routed signal.
24 (R/W)	MISCINT2	Miscellaneous Interrupt 2. Setting the DAI_IMSK_FE.MISCINT2 bit unmasks the interrupt for the falling edge of the routed signal.
23 (R/W)	MISCINT1	Miscellaneous Interrupt 1. Setting the DAI_IMSK_FE.MISCINT1 bit unmasks the interrupt for the falling edge of the routed signal.
22 (R/W)	MISCINT0	Miscellaneous Interrupt 0. Setting the DAI_IMSK_FE.MISCINT0 bit unmasks the interrupt for the falling edge of the routed signal.
21 (R/W)	SRC3MUTE	SRC3 Mute. The DAI_IMSK_FE.SRC3MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
20 (R/W)	SRC2MUTE	SRC2 Mute. The DAI_IMSK_FE.SRC2MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.

Table 29-85: DAI_IMSK_FE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	SRC1MUTE	SRC1 Mute. The DAI_IMSK_FE.SRC1MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
18 (R/W)	SRC0MUTE	SRC0 Mute. The DAI_IMSK_FE.SRC0MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
17 (R/W)	SRC7MUTE	SRC7 Mute. The DAI_IMSK_FE.SRC7MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
16 (R/W)	SRC6MUTE	SRC6 Mute. The DAI_IMSK_FE.SRC6MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
15 (R/W)	SRC5MUTE	SRC5 Mute. The DAI_IMSK_FE.SRC5MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
14 (R/W)	SRC4MUTE	SRC4 Mute. The DAI_IMSK_FE.SRC4MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
7 (R/W)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IMSK_FE.RXCHSCH bit masks or unmasks RXCHSCH interrupt for the falling edge of this interrupt. This interrupt is set if there is a change in the SPDIF Rx channel status bits (SPDIF_RX_STAT0/1_A/B).
4 (R/W)	RXNONAUDIO	Receive Non Audio. The DAI_IMSK_FE.RXNONAUDIO bit masks or unmasks the non audio frame mode interrupt for the falling edge of this interrupt. If the channel status indicates non-PCM audio, the NONAUDIO bit flag is set.
2 (R/W)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IMSK_FE.RXLOSSOFLOCK bit masks or unmasks the emphasis loss of lock interrupt for the falling edge of this interrupt. The loss of lock status is set by the S/PDIF receiver if receiver loses the lock of biphase stream.

Table 29-85: DAI_IMSK_FE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	RXLOCK	Receive Lock. The `DAI_IMSK_FE.RXLOCK` bit masks or unmasks the S/PDIF receiver lock for the falling edge of this interrupt. This interrupt occurs when the S/PDIF receiver locks to the S/PDIF stream.
0 (R/W)	RXVALID	Receive Valid. Setting the `DAI_IMSK_FE.RXVALID` bit unmasks the interrupt for the falling edge of the routed signal. This interrupt is set based on whether data received by the S/PDIF is linear PCM or non-linear audio data.

Core Interrupt Priority Assignment Register

The DAI_IMSK_PRI register masks interrupts for DAI high or DAI low interrupt priority.

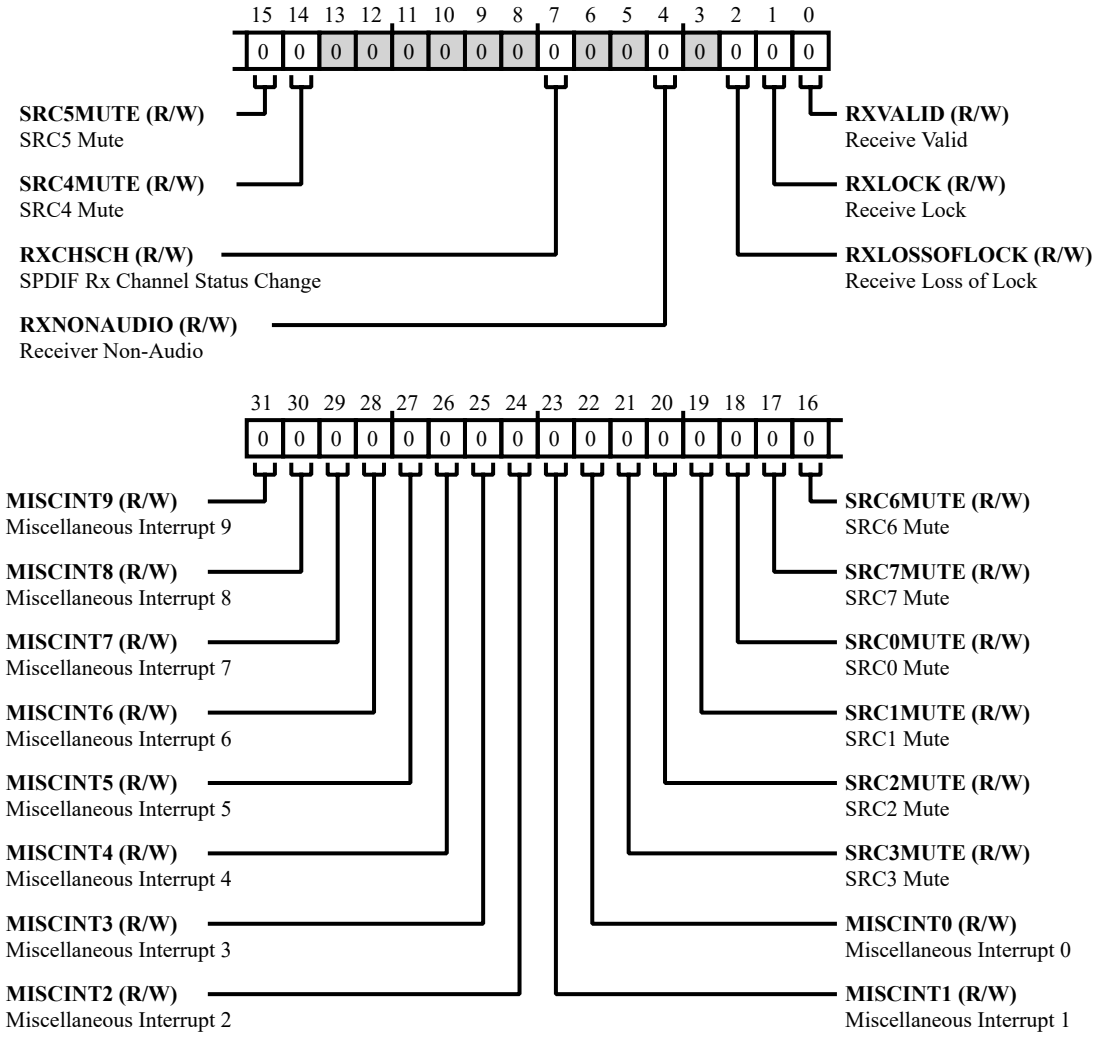


Figure 29-79: DAI_IMSK_PRI Register Diagram

Table 29-86: DAI_IMSK_PRI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MISCINT9	Miscellaneous Interrupt 9. If the DAI_IMSK_PRI.MISCINT9 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT9 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Table 29-86: DAI_IMSK_PRI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	MISCINT8	Miscellaneous Interrupt 8. If the DAI_IMSK_PRI.MISCINT8 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT8 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
29 (R/W)	MISCINT7	Miscellaneous Interrupt 7. If the DAI_IMSK_PRI.MISCINT7 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT7 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
28 (R/W)	MISCINT6	Miscellaneous Interrupt 6. If the DAI_IMSK_PRI.MISCINT6 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT6 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
27 (R/W)	MISCINT5	Miscellaneous Interrupt 5. If the DAI_IMSK_PRI.MISCINT5 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT5 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
26 (R/W)	MISCINT4	Miscellaneous Interrupt 4. If the DAI_IMSK_PRI.MISCINT4 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT4 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
25 (R/W)	MISCINT3	Miscellaneous Interrupt 3. If the DAI_IMSK_PRI.MISCINT3 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT3 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
24 (R/W)	MISCINT2	Miscellaneous Interrupt 2. If the DAI_IMSK_PRI.MISCINT2 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT2 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
23 (R/W)	MISCINT1	Miscellaneous Interrupt 1. If the DAI_IMSK_PRI.MISCINT1 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT1 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
22 (R/W)	MISCINT0	Miscellaneous Interrupt 0. If the DAI_IMSK_PRI.MISCINT0 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT0 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Table 29-86: DAI_IMSK_PRI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W)	SRC3MUTE	SRC3 Mute. If the DAI_IMSK_PRI.SRC3MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC3MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
20 (R/W)	SRC2MUTE	SRC2 Mute. If the DAI_IMSK_PRI.SRC2MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC2MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
19 (R/W)	SRC1MUTE	SRC1 Mute. If the DAI_IMSK_PRI.SRC1MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC1MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
18 (R/W)	SRC0MUTE	SRC0 Mute. If the DAI_IMSK_PRI.SRC0MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC0MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
17 (R/W)	SRC7MUTE	SRC7 Mute. If the DAI_IMSK_PRI.SRC7MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC7MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
16 (R/W)	SRC6MUTE	SRC6 Mute. If the DAI_IMSK_PRI.SRC6MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC6MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
15 (R/W)	SRC5MUTE	SRC5 Mute. If the DAI_IMSK_PRI.SRC5MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC5MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
14 (R/W)	SRC4MUTE	SRC4 Mute. If the DAI_IMSK_PRI.SRC4MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC4MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
7 (R/W)	RXCHSCH	SPDIF Rx Channel Status Change. If the DAI_IMSK_PRI.RXCHSCH bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.RXCHSCH bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Table 29-86: DAI_IMSK_PRI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	RXNONAUDIO	Receiver Non-Audio. If the DAI_IMSK_PRI.RXNONAUDIO bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.RXNONAUDIO bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
2 (R/W)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IMSK_PRI.RXLOSSOFLOCK bit masks or unmasks the emphasis loss of lock interrupt for the falling edge of this interrupt. The loss of lock status is set by the S/PDIF receiver if receiver loses the lock of biphase stream.
1 (R/W)	RXLOCK	Receive Lock. If the DAI_IMSK_PRI.RXLOCK bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.RXLOCK bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
0 (R/W)	RXVALID	Receive Valid. If the DAI_IMSK_PRI.RXVALID bit is cleared (=0), the interrupt is mapped to the INTR_DAI_IRQL from the SEC. If the DAI_IMSK_PRI.RXVALID bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Rising-Edge Interrupt Mask Register

The DAI_IMSK_RE register masks and unmasks interrupts generated on the rising edge of a waveform. Note that any of the Group E signals can be mapped to any of the miscellaneous interrupts (9-0).

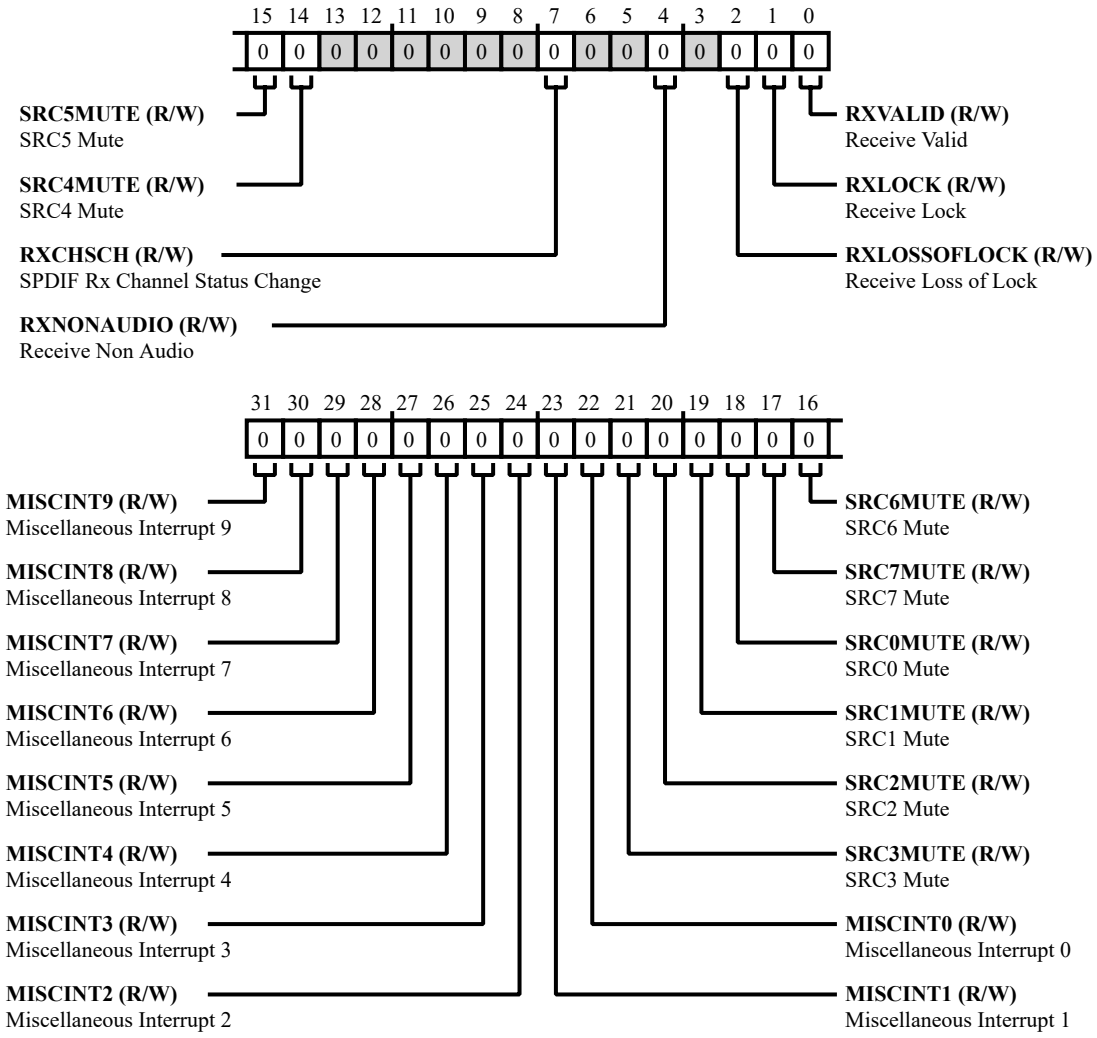


Figure 29-80: DAI_IMSK_RE Register Diagram

Table 29-87: DAI_IMSK_RE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MISCINT9	Miscellaneous Interrupt 9. Setting the DAI_IMSK_RE.MISCINT9 bit unmasks the interrupt for the rising edge of the routed signal.

Table 29-87: DAI_IMSK_RE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	MISCINT8	Miscellaneous Interrupt 8. Setting the DAI_IMSK_RE.MISCINT8 bit unmasks the interrupt for the rising edge of the routed signal.
29 (R/W)	MISCINT7	Miscellaneous Interrupt 7. Setting the DAI_IMSK_RE.MISCINT7 bit unmasks the interrupt for the rising edge of the routed signal.
28 (R/W)	MISCINT6	Miscellaneous Interrupt 6. Setting the DAI_IMSK_RE.MISCINT6 bit unmasks the interrupt for the rising edge of the routed signal.
27 (R/W)	MISCINT5	Miscellaneous Interrupt 5. Setting the DAI_IMSK_RE.MISCINT5 bit unmasks the interrupt for the rising edge of the routed signal.
26 (R/W)	MISCINT4	Miscellaneous Interrupt 4. Setting the DAI_IMSK_RE.MISCINT4 bit unmasks the interrupt for the rising edge of the routed signal.
25 (R/W)	MISCINT3	Miscellaneous Interrupt 3. Setting the DAI_IMSK_RE.MISCINT3 bit unmasks the interrupt for the rising edge of the routed signal.
24 (R/W)	MISCINT2	Miscellaneous Interrupt 2. Setting the DAI_IMSK_RE.MISCINT2 bit unmasks the interrupt for the rising edge of the routed signal.
23 (R/W)	MISCINT1	Miscellaneous Interrupt 1. Setting the DAI_IMSK_RE.MISCINT1 bit unmasks the interrupt for the rising edge of the routed signal.
22 (R/W)	MISCINT0	Miscellaneous Interrupt 0. Setting the DAI_IMSK_RE.MISCINT0 bit unmasks the interrupt for the rising edge of the routed signal.
21 (R/W)	SRC3MUTE	SRC3 Mute. The DAI_IMSK_RE.SRC3MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
20 (R/W)	SRC2MUTE	SRC2 Mute. The DAI_IMSK_RE.SRC2MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.

Table 29-87: DAI_IMSK_RE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	SRC1MUTE	SRC1 Mute. The DAI_IMSK_RE.SRC1MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
18 (R/W)	SRC0MUTE	SRC0 Mute. The DAI_IMSK_RE.SRC0MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
17 (R/W)	SRC7MUTE	SRC7 Mute. The DAI_IMSK_RE.SRC7MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
16 (R/W)	SRC6MUTE	SRC6 Mute. The DAI_IMSK_RE.SRC6MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
15 (R/W)	SRC5MUTE	SRC5 Mute. The DAI_IMSK_RE.SRC5MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
14 (R/W)	SRC4MUTE	SRC4 Mute. The DAI_IMSK_RE.SRC4MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
7 (R/W)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IMSK_RE.RXCHSCH bit masks or unmasks RXCHSCH (Status Change) interrupt for the rising edge of this interrupt. This interrupt is set if there is a change in the SPDIF Rx channel status bits (SPDIF_RX_STAT0/1_A/B).
4 (R/W)	RXNONAUDIO	Receive Non Audio. The DAI_IMSK_RE.RXNONAUDIO bit masks or unmasks the non audio frame mode interrupt for the rising edge of this interrupt. If the channel status indicates non-PCM audio, the NONAUDIO bit flag is set.
2 (R/W)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IMSK_RE.RXLOSSOFLOCK bit masks or unmasks the emphasis loss of lock interrupt for the rising edge of this interrupt. The loss of lock status is set by the S/PDIF receiver if receiver loses the lock of biphase stream.

Table 29-87: DAI_IMSK_RE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	RXLOCK	Receive Lock. The `DAI_IMSK_RE.RXLOCK` bit masks or unmasks the S/PDIF receiver lock for the rising edge of this interrupt. This interrupt occurs when the S/PDIF receiver locks to the S/PDIF stream.
0 (R/W)	RXVALID	Receive Valid. Setting the `DAI_IMSK_RE.RXVALID` bit unmasks the interrupt for the rising edge of the routed signal. This interrupt is set based on whether data received by the S/PDIF is linear PCM or non-linear audio data.

High Priority Interrupt Latch Register

The DAI_IRPTL_H register holds the high priority latched interrupt status for interrupt requests that have been unmasked (enabled) by the DAI_IMSK_FE, DAI_IMSK_RE registers and mapped to the INTR_DAI_IRQH signal in the SEC by DAI_IMSK_PRI registers. If a bit in this register is already set and the corresponding interrupt is masked in the DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI registers, the latch holds its old value, leaving the interrupt asserted until its mask registers (DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI) are reset by software with a W1C operation.

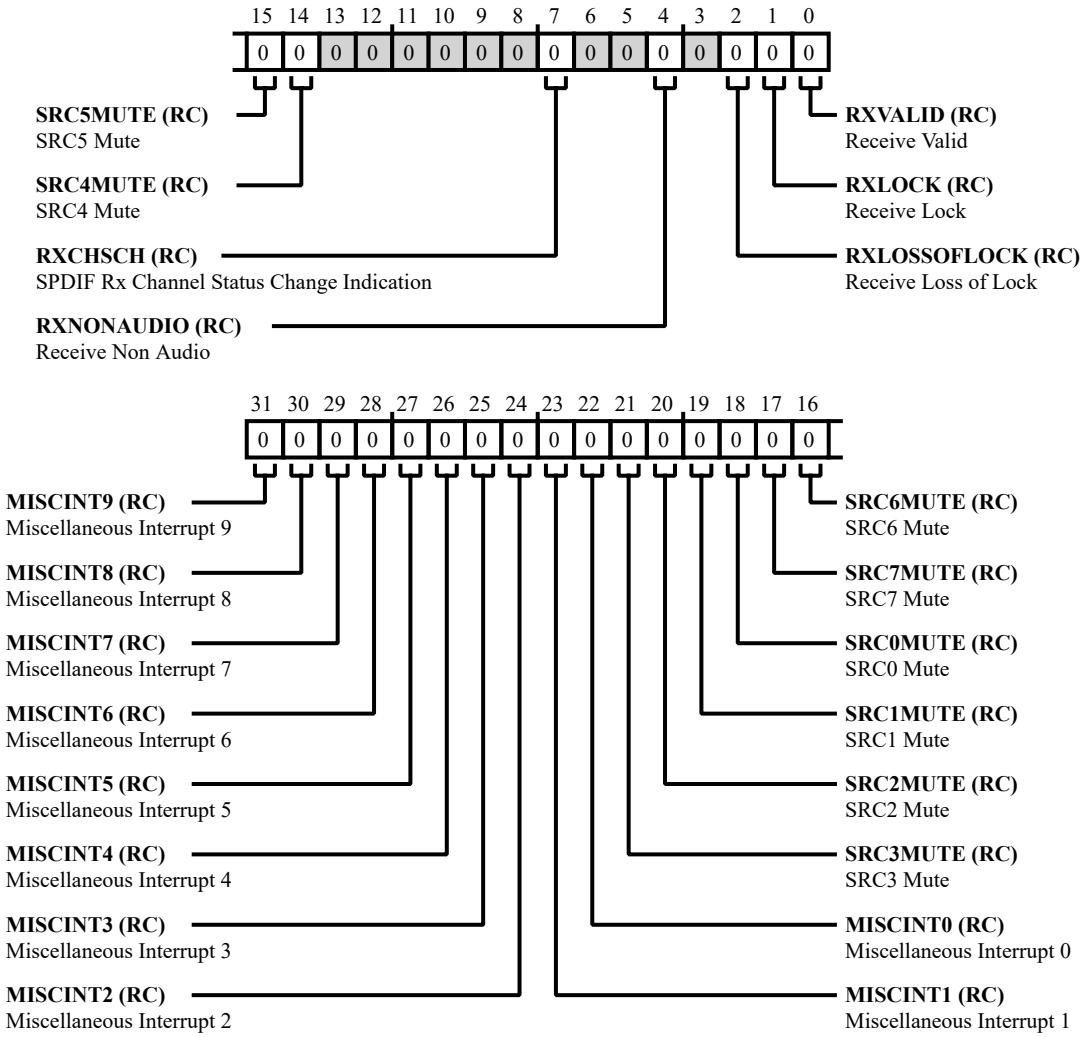


Figure 29-81: DAI_IRPTL_H Register Diagram

Table 29-88: DAI_IRPTL_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (RC/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_H.MISCINT9 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
30 (RC/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_H.MISCINT8 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
29 (RC/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_H.MISCINT7 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
28 (RC/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_H.MISCINT6 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
27 (RC/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_H.MISCINT5 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
26 (RC/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_H.MISCINT4 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
25 (RC/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_H.MISCINT3 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
24 (RC/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_H.MISCINT2 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
23 (RC/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_H.MISCINT1 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
22 (RC/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_H.MISCINT0 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
21 (RC/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_H.SRC3MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.

Table 29-88: DAI_IRPTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20 (RC/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_H.SRC2MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
19 (RC/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_H.SRC1MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
18 (RC/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_H.SRC0MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
17 (RC/NW)	SRC7MUTE	SRC7 Mute. The DAI_IRPTL_H.SRC7MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
16 (RC/NW)	SRC6MUTE	SRC6 Mute. The DAI_IRPTL_H.SRC6MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
15 (RC/NW)	SRC5MUTE	SRC5 Mute. The DAI_IRPTL_H.SRC5MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
14 (RC/NW)	SRC4MUTE	SRC4 Mute. The DAI_IRPTL_H.SRC4MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
7 (RC/NW)	RXCHSCH	SPDIF Rx Channel Status Change Indication. The DAI_IRPTL_H.RXCHSCH bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
4 (RC/NW)	RXNONAUDIO	Receive Non Audio. The DAI_IRPTL_H.RXNONAUDIO bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
2 (RC/NW)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IRPTL_H.RXLOSSOFLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
1 (RC/NW)	RXLOCK	Receive Lock. The DAI_IRPTL_H.RXLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.

Table 29-88: DAI_IRPTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (RC/NW)	RXVALID	Receive Valid. The `DAI_IRPTL_H.RXVALID` bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the `DAI_IMSK_PRI` register.

Shadow High Priority Interrupt Latch Register

The DAI_IRPTL_HS register is the shadow register of the DAI_IRPTL_H register. Its content is the same as the DAI_IRPTL_H register. Reading the DAI_IRPTL_HS register does not affect its contents while reading the contents of the DAI_IRPTL_H registers clears it.

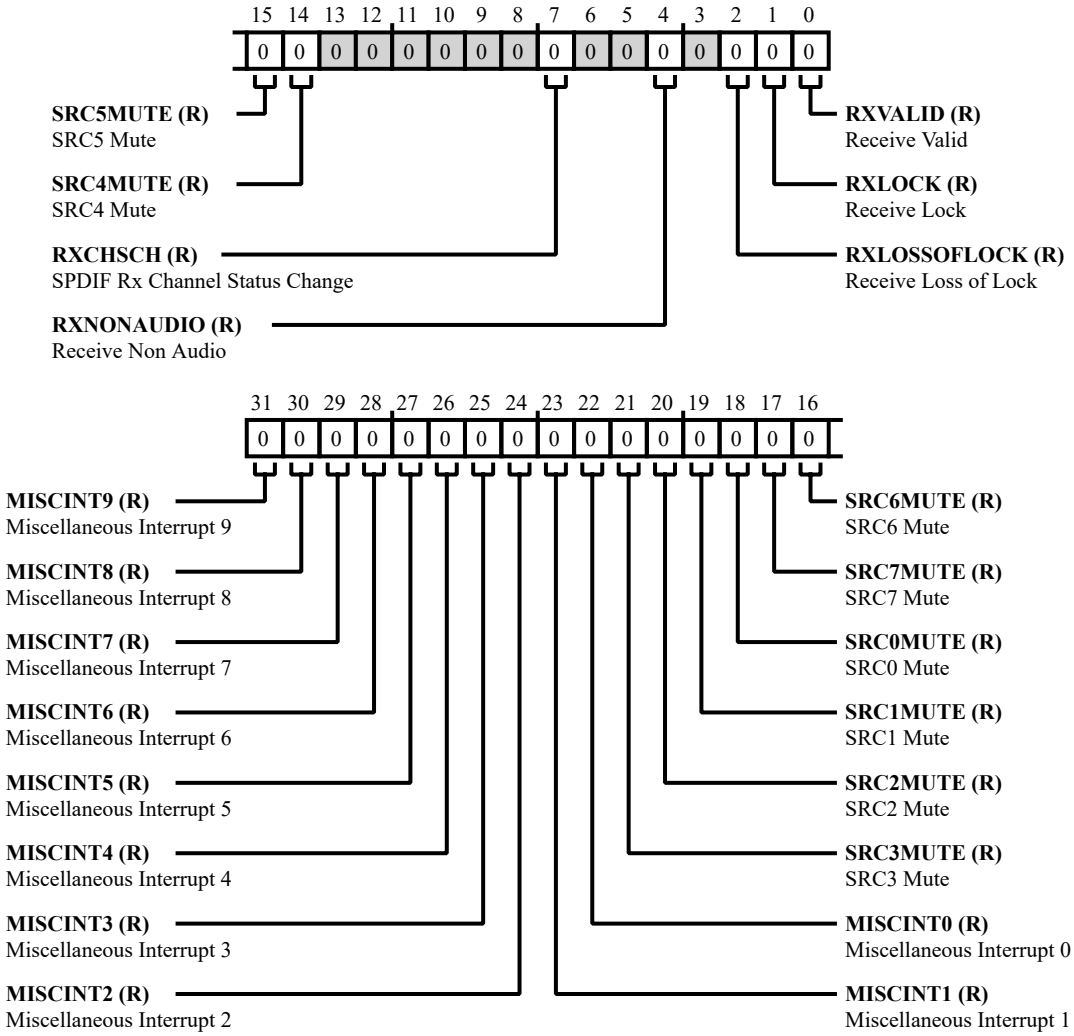


Figure 29-82: DAI_IRPTL_HS Register Diagram

Table 29-89: DAI_IRPTL_HS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_HS.MISCINT9 bit is the shadow of the DAI_IRPTL_H.MISCINT9 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT9 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT9 bit clears it.
30 (R/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_HS.MISCINT8 bit is the shadow of the DAI_IRPTL_H.MISCINT8 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT8 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT8 bit clears it.
29 (R/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_HS.MISCINT7 bit is the shadow of the DAI_IRPTL_H.MISCINT7 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT7 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT7 bit clears it.
28 (R/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_HS.MISCINT6 bit is the shadow of the DAI_IRPTL_H.MISCINT6 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT6 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT6 bit clears it.
27 (R/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_HS.MISCINT5 bit is the shadow of the DAI_IRPTL_H.MISCINT5 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT5 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT5 bit clears it.
26 (R/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_HS.MISCINT4 bit is the shadow of the DAI_IRPTL_H.MISCINT4 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT4 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT4 bit clears it.
25 (R/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_HS.MISCINT3 bit is the shadow of the DAI_IRPTL_H.MISCINT3 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT3 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT3 bit clears it.

Table 29-89: DAI_IRPTL_HS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_HS.MISCINT2 bit is the shadow of the DAI_IRPTL_H.MISCINT2 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT2 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT2 bit clears it.
23 (R/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_HS.MISCINT1 bit is the shadow of the DAI_IRPTL_H.MISCINT1 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT1 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT1 bit clears it.
22 (R/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_HS.MISCINT0 bit is the shadow of the DAI_IRPTL_H.MISCINT0 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT0 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT0 bit clears it.
21 (R/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_HS.SRC3MUTE bit is the shadow of the DAI_IRPTL_H.SRC3MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC3MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC3MUTE bit clears it.
20 (R/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_HS.SRC2MUTE bit is the shadow of the DAI_IRPTL_H.SRC2MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC2MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC2MUTE bit clears it.
19 (R/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_HS.SRC1MUTE bit is the shadow of the DAI_IRPTL_H.SRC1MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC1MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC1MUTE bit clears it.
18 (R/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_HS.SRC0MUTE bit is the shadow of the DAI_IRPTL_H.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC0MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC0MUTE bit clears it.

Table 29-89: DAI_IRPTL_HS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/NW)	SRC7MUTE	SRC7 Mute. The DAI_IRPTL_HS.SRC7MUTE bit is the shadow of the DAI_IRPTL_H.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC7MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC0MUTE bit clears it.
16 (R/NW)	SRC6MUTE	SRC6 Mute. The DAI_IRPTL_HS.SRC6MUTE bit is the shadow of the DAI_IRPTL_H.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC6MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC0MUTE bit clears it.
15 (R/NW)	SRC5MUTE	SRC5 Mute. The DAI_IRPTL_HS.SRC5MUTE bit is the shadow of the DAI_IRPTL_H.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC5MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC0MUTE bit clears it.
14 (R/NW)	SRC4MUTE	SRC4 Mute. The DAI_IRPTL_HS.SRC4MUTE bit is the shadow of the DAI_IRPTL_H.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC4MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC0MUTE bit clears it.
7 (R/NW)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IRPTL_HS.RXCHSCH bit is the shadow of the DAI_IRPTL_H.RXCHSCH bit and contains the same content. Reading the DAI_IRPTL_HS.RXCHSCH bit does not affect its contents while reading the DAI_IRPTL_H.RXCHSCH bit clears it.
4 (R/NW)	RXNONAUDIO	Receive Non Audio. The DAI_IRPTL_HS.RXNONAUDIO bit is the shadow of DAI_IRPTL_H.RXNONAUDIO bit and contains the same content. Reading this bit does not affect its content.
2 (R/NW)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IRPTL_HS.RXLOSSOFLOCK bit is the shadow of DAI_IRPTL_H.RXLOSSOFLOCK bit and contains the same content. Reading this bit does not affect its content.
1 (R/NW)	RXLOCK	Receive Lock. The DAI_IRPTL_HS.RXLOCK bit is the shadow of DAI_IRPTL_H.RXLOCK bit and contains the same content. Reading this bit does not affect its content.

Table 29-89: DAI_IRPTL_HS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	RXVALID	Receive Valid. Setting the `DAI_IRPTL_HS.RXVALID` bit is the shadow of `DAI_IRPTL_H.RXVALID` bit and contains the same content. Reading this bit does not affect its content.

Low Priority Interrupt Latch Register

The DAI_IRPTL_L register holds the low priority latched interrupt status for interrupt requests that have been unmasked (enabled) by the DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI registers. If a bit in DAI_IRPTL_L is already set and the corresponding interrupt is masked in DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI registers, the latch holds its old value, leaving the interrupt asserted until its masked registers (DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI) are reset by software with a W1C operation.

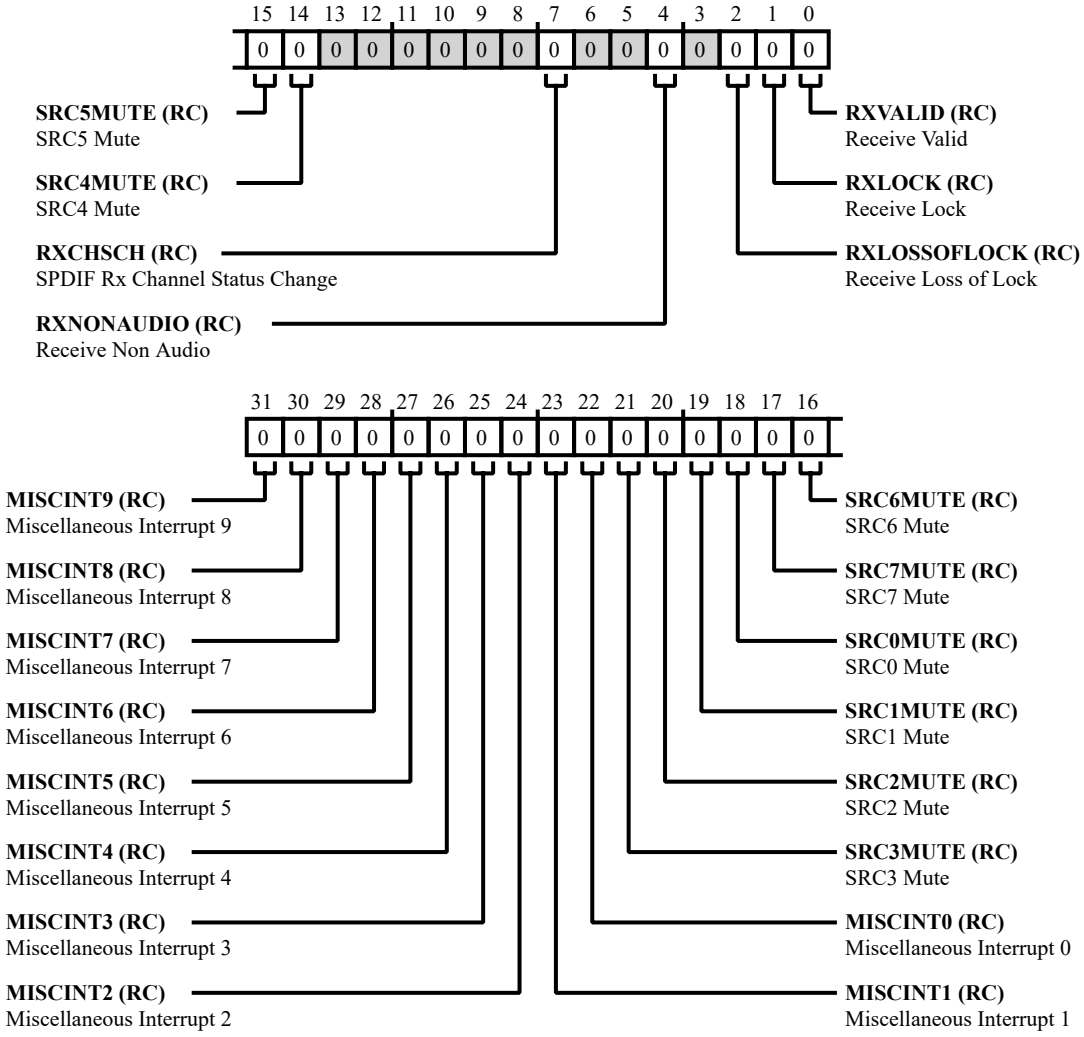


Figure 29-83: DAI_IRPTL_L Register Diagram

Table 29-90: DAI_IRPTL_L Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (RC/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_L.MISCINT9 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
30 (RC/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_L.MISCINT8 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
29 (RC/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_L.MISCINT7 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
28 (RC/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_L.MISCINT6 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
27 (RC/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_L.MISCINT5 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
26 (RC/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_L.MISCINT4 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
25 (RC/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_L.MISCINT3 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
24 (RC/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_L.MISCINT2 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
23 (RC/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_L.MISCINT1 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
22 (RC/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_L.MISCINT0 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
21 (RC/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_L.SRC3MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.

Table 29-90: DAI_IRPTL_L Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20 (RC/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_L.SRC2MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
19 (RC/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_L.SRC1MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
18 (RC/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_L.SRC0MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
17 (RC/NW)	SRC7MUTE	SRC7 Mute. The DAI_IRPTL_L.SRC7MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
16 (RC/NW)	SRC6MUTE	SRC6 Mute. The DAI_IRPTL_L.SRC6MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
15 (RC/NW)	SRC5MUTE	SRC5 Mute. The DAI_IRPTL_L.SRC5MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
14 (RC/NW)	SRC4MUTE	SRC4 Mute. The DAI_IRPTL_L.SRC4MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
7 (RC/NW)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IRPTL_L.RXCHSCH bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
4 (RC/NW)	RXNONAUDIO	Receive Non Audio. The DAI_IRPTL_L.RXNONAUDIO bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
2 (RC/NW)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IRPTL_L.RXLOSSOFLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
1 (RC/NW)	RXLOCK	Receive Lock. The DAI_IRPTL_L.RXLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.

Table 29-90: DAI_IRPTL_L Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (RC/NW)	RXVALID	Receive Valid. The `DAI_IRPTL_L.RXVALID` bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.

Shadow Low Priority Interrupt Latch Register

The DAI_IRPTL_LS register is the shadow register of the DAI_IRPTL_L register. Its content is the same as the DAI_IRPTL_L register. Reading the DAI_IRPTL_LS register does not affect its contents while reading the contents of the DAI_IRPTL_L registers clears it.

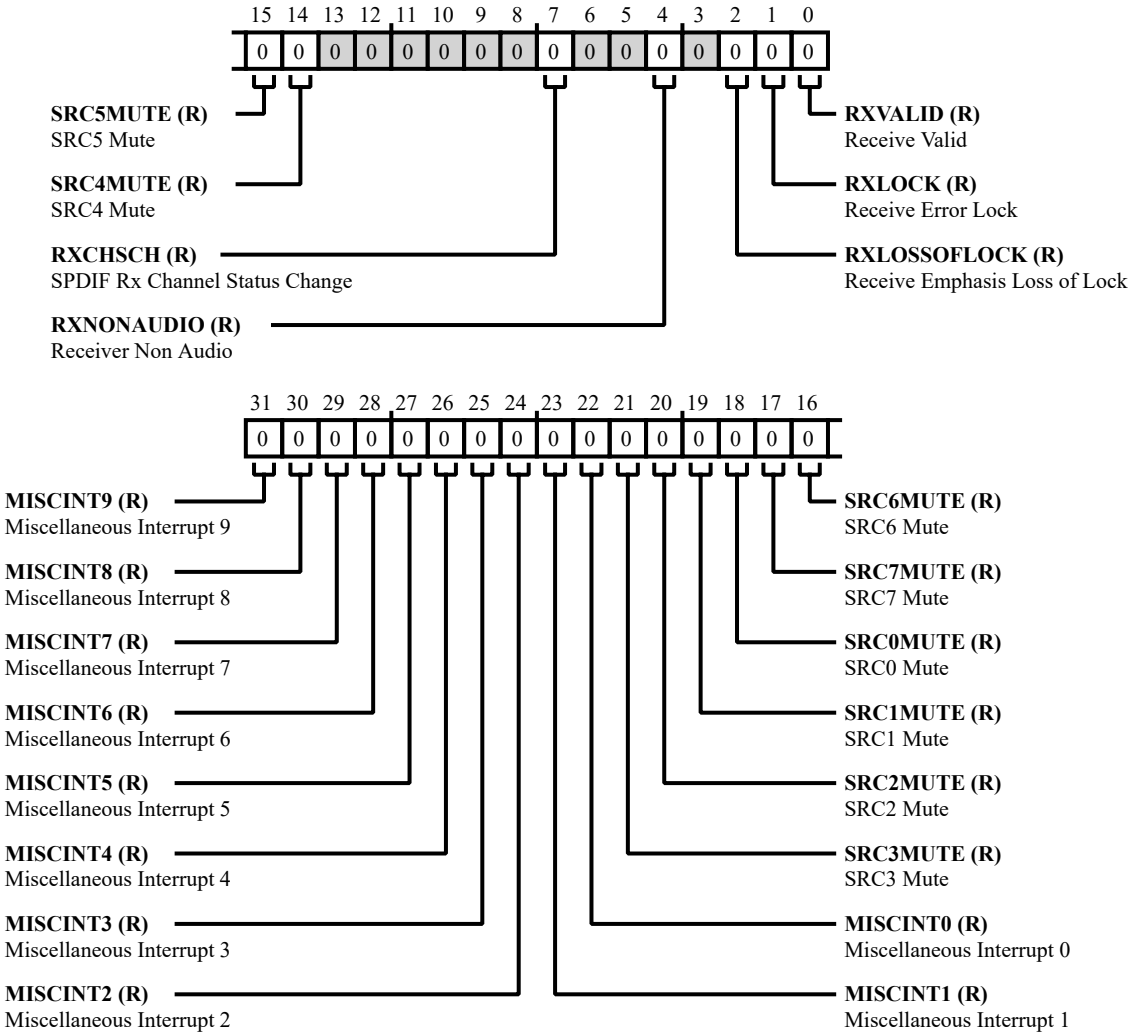


Figure 29-84: DAI_IRPTL_LS Register Diagram

Table 29-91: DAI_IRPTL_LS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_LS.MISCINT9 bit is the shadow of the DAI_IRPTL_L.MISCINT9 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT9 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT9 bit clears it.
30 (R/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_LS.MISCINT8 bit is the shadow of the DAI_IRPTL_L.MISCINT8 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT8 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT8 bit clears it.
29 (R/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_LS.MISCINT7 bit is the shadow of the DAI_IRPTL_L.MISCINT7 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT7 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT7 bit clears it.
28 (R/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_LS.MISCINT6 bit is the shadow of the DAI_IRPTL_L.MISCINT6 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT6 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT6 bit clears it.
27 (R/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_LS.MISCINT5 bit is the shadow of the DAI_IRPTL_L.MISCINT5 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT5 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT5 bit clears it.
26 (R/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_LS.MISCINT4 bit is the shadow of the DAI_IRPTL_L.MISCINT4 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT4 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT4 bit clears it.
25 (R/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_LS.MISCINT3 bit is the shadow of the DAI_IRPTL_L.MISCINT3 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT3 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT3 bit clears it.

Table 29-91: DAI_IRPTL_LS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_LS.MISCINT2 bit is the shadow of the DAI_IRPTL_L.MISCINT2 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT2 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT2 bit clears it.
23 (R/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_LS.MISCINT1 bit is the shadow of the DAI_IRPTL_L.MISCINT1 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT1 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT1 bit clears it.
22 (R/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_LS.MISCINT0 bit is the shadow of the DAI_IRPTL_L.MISCINT0 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT0 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT0 bit clears it.
21 (R/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_LS.SRC3MUTE bit is the shadow of the DAI_IRPTL_L.SRC3MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC3MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC3MUTE bit clears it.
20 (R/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_LS.SRC2MUTE bit is the shadow of the DAI_IRPTL_L.SRC2MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC2MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC2MUTE bit clears it.
19 (R/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_LS.SRC1MUTE bit is the shadow of the DAI_IRPTL_L.SRC1MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC1MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC1MUTE bit clears it.
18 (R/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_LS.SRC0MUTE bit is the shadow of the DAI_IRPTL_L.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC0MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC0MUTE bit clears it.

Table 29-91: DAI_IRPTL_LS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/NW)	SRC7MUTE	SRC7 Mute. The DAI_IRPTL_LS.SRC7MUTE bit is the shadow of the DAI_IRPTL_L.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC7MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC0MUTE bit clears it.
16 (R/NW)	SRC6MUTE	SRC6 Mute. The DAI_IRPTL_LS.SRC6MUTE bit is the shadow of the DAI_IRPTL_L.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC6MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC0MUTE bit clears it.
15 (R/NW)	SRC5MUTE	SRC5 Mute. The DAI_IRPTL_LS.SRC5MUTE bit is the shadow of the DAI_IRPTL_L.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC5MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC0MUTE bit clears it.
14 (R/NW)	SRC4MUTE	SRC4 Mute. The DAI_IRPTL_LS.SRC4MUTE bit is the shadow of the DAI_IRPTL_L.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC4MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC0MUTE bit clears it.
7 (R/NW)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IRPTL_LS.RXCHSCH bit is the shadow of the DAI_IRPTL_L.RXCHSCH bit and contains the same content. Reading the DAI_IRPTL_LS.RXCHSCH bit does not affect its contents while reading the DAI_IRPTL_L.RXCHSCH bit clears it.
4 (R/NW)	RXNONAUDIO	Receiver Non Audio. The DAI_IRPTL_LS.RXNONAUDIO bit is the shadow of the DAI_IRPTL_L.RXNONAUDIO bit and contains the same content. Reading the DAI_IRPTL_LS.RXNONAUDIO bit does not affect its contents while reading the DAI_IRPTL_L.RXNONAUDIO bit clears it.
2 (R/NW)	RXLOSSOFLOCK	Receive Emphasis Loss of Lock. The DAI_IRPTL_LS.RXLOSSOFLOCK bit is the shadow of DAI_IRPTL_L.RXLOSSOFLOCK bit and contains the same content. Reading this bit does not affect its content. Reading the DAI_IRPTL_L.RXLOSSOFLOCK bit clears it.
1 (R/NW)	RXLOCK	Receive Error Lock. The DAI_IRPTL_LS.RXLOCK bit is the shadow of DAI_IRPTL_L.RXLOCK bit and contains the same content. Reading this bit does not affect its content. Reading the DAI_IRPTL_L.RXLOCK bit clears it.

Table 29-91: DAI_IRPTL_LS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	RXVALID	Receive Valid. The `DAI_IRPTL_LS.RXVALID` bit is the shadow of `DAI_IRPTL_L.RXVALID` bit and contains the same content. Reading this bit does not affect its content. Reading the `DAI_IRPTL_L.RXVALID` bit clears it.

Miscellaneous Control Register 0

The DAI_MISC0 register allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

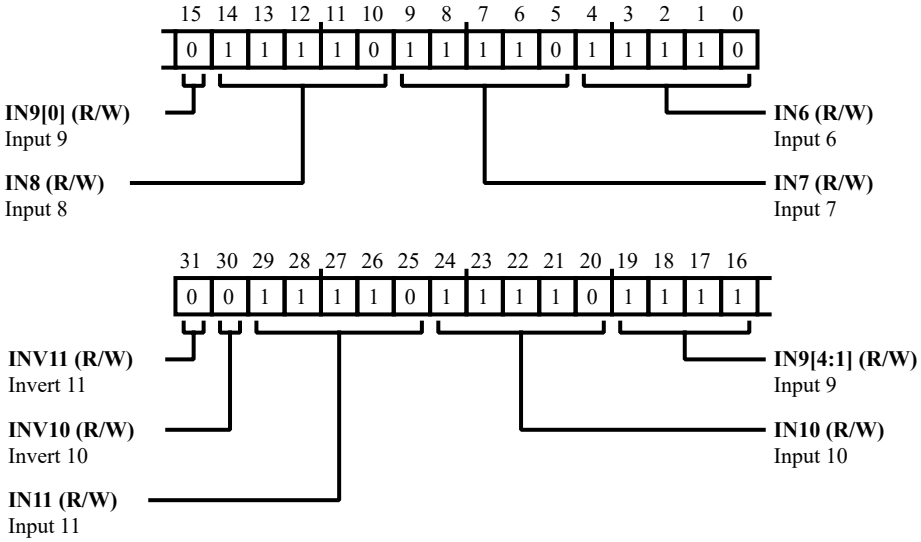


Figure 29-85: DAI_MISC0 Register Diagram

Table 29-92: DAI_MISC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	INV11	Invert 11. The DAI_MISC0.INV11 bit field inverts miscellaneous A5 input.
30 (R/W)	INV10	Invert 10. The DAI_MISC0.INV10 bit field inverts miscellaneous A4 input.
29:25 (R/W)	IN11	Input 11. The DAI_MISC0.IN11 bit field configures miscellaneous A5 input.
24:20 (R/W)	IN10	Input 10. The DAI_MISC0.IN10 bit field configures miscellaneous A4 input.
19:15 (R/W)	IN9	Input 9. The DAI_MISC0.IN9 bit field configures miscellaneous A3 input/miscellaneous Interrupt 9(DAI interrupt 31).
14:10 (R/W)	IN8	Input 8. The DAI_MISC0.IN8 bit field configures miscellaneous A2 input/miscellaneous Interrupt 8(DAI interrupt 30).

Table 29-92: DAI_MISC0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	IN7	Input 7. The `DAI_MISC0.IN7` bit field configures miscellaneous A1 input/miscellaneous Interrupt 7(DAI interrupt 29).
4:0 (R/W)	IN6	Input 6. The `DAI_MISC0.IN6` bit field configures miscellaneous A0 input/miscellaneous Interrupt 6(DAI interrupt 28).

Miscellaneous Control Register 1

The DAI_MISC1 register allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

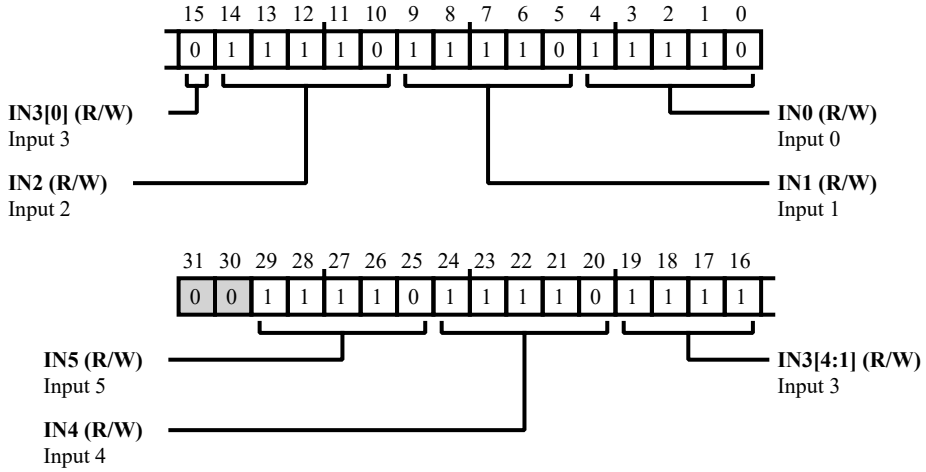


Figure 29-86: DAI_MISC1 Register Diagram

Table 29-93: DAI_MISC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input 5. The DAI_MISC1.IN5 bit field configures miscellaneous Interrupt 5(DAI interrupt 27).
24:20 (R/W)	IN4	Input 4. The DAI_MISC1.IN4 bit field configures miscellaneous Interrupt 4(DAI interrupt 26).
19:15 (R/W)	IN3	Input 3. The DAI_MISC1.IN3 bit field configures miscellaneous Interrupt 3(DAI interrupt 25).
14:10 (R/W)	IN2	Input 2. The DAI_MISC1.IN2 bit field configures miscellaneous Interrupt 2(DAI interrupt 24).
9:5 (R/W)	IN1	Input 1. The DAI_MISC1.IN1 bit field configures miscellaneous Interrupt 1(DAI interrupt 23).

Table 29-93: DAI_MISC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input 0. The `DAI_MISC1.IN0` bit field configures miscellaneous interrupt 0(DAI interrupt 22).

Miscellaneous Control Register 1

The DAI_MISC2 register allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

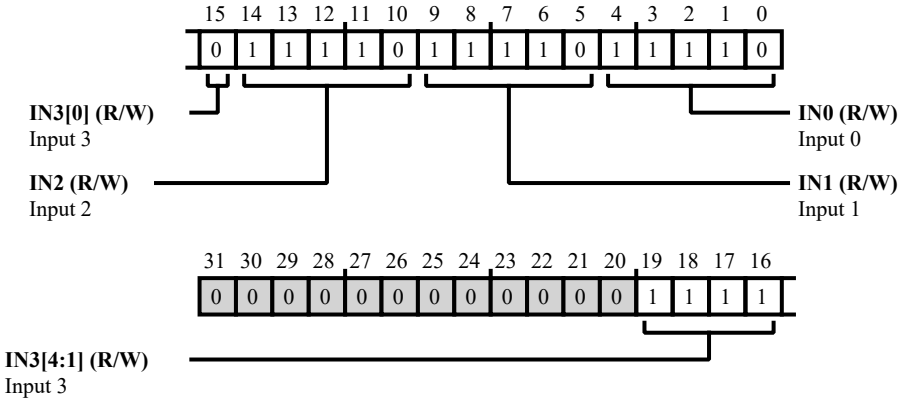


Figure 29-87: DAI_MISC2 Register Diagram

Table 29-94: DAI_MISC2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:15 (R/W)	IN3	Input 3. The DAI_MISC2.IN3 bit field configures PCG-F's hardware completer trigger input.
14:10 (R/W)	IN2	Input 2. The DAI_MISC2.IN2 bit field configures PCG-E's hardware completer trigger input.
9:5 (R/W)	IN1	Input 1. The DAI_MISC2.IN1 bit field configures the PCG-B's hardware completer trigger input.
4:0 (R/W)	IN0	Input 0. The DAI_MISC2.IN0 bit field configures the PCG-A's hardware completer trigger input.

Pin Buffer Enable Register 0

The DAI_PBEN0 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

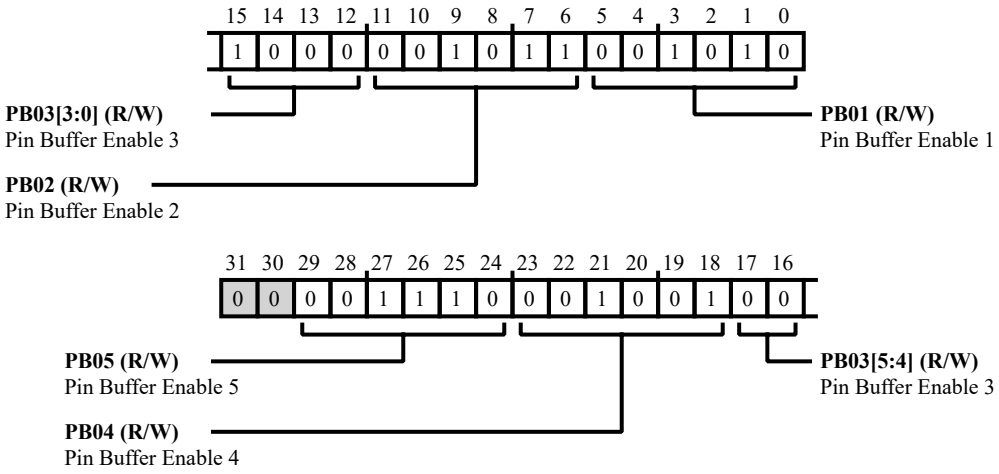


Figure 29-88: DAI_PBEN0 Register Diagram

Table 29-95: DAI_PBEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB05	Pin Buffer Enable 5. The DAI_PBEN0.PB05 bit field is the pin buffer enable for DAI port 5.
23:18 (R/W)	PB04	Pin Buffer Enable 4. The DAI_PBEN0.PB04 bit field is the pin buffer enable for DAI port 4.
17:12 (R/W)	PB03	Pin Buffer Enable 3. The DAI_PBEN0.PB03 bit field is the pin buffer enable for DAI port 3.
11:6 (R/W)	PB02	Pin Buffer Enable 2. The DAI_PBEN0.PB02 bit field is the pin buffer enable for DAI port 2.
5:0 (R/W)	PB01	Pin Buffer Enable 1. The DAI_PBEN0.PB01 bit field is the pin buffer enable for DAI port 1.

Pin Buffer Enable Register 1

The DAI_PBEN1 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

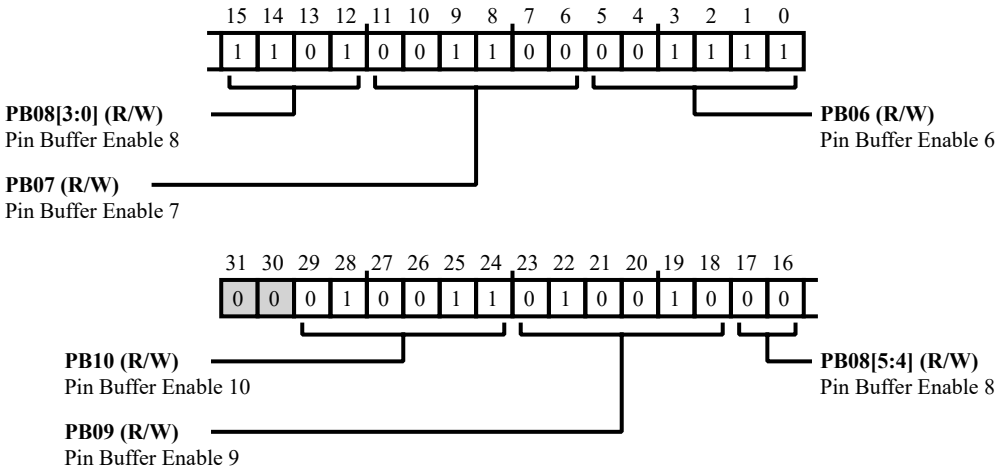


Figure 29-89: DAI_PBEN1 Register Diagram

Table 29-96: DAI_PBEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB10	Pin Buffer Enable 10. The DAI_PBEN1.PB10 bit field is the pin buffer enable for DAI port 10.
23:18 (R/W)	PB09	Pin Buffer Enable 9. The DAI_PBEN1.PB09 bit field is the pin buffer enable for DAI port 9.
17:12 (R/W)	PB08	Pin Buffer Enable 8. The DAI_PBEN1.PB08 bit field is the pin buffer enable for DAI port 8.
11:6 (R/W)	PB07	Pin Buffer Enable 7. The DAI_PBEN1.PB07 bit field is the pin buffer enable for DAI port 7.
5:0 (R/W)	PB06	Pin Buffer Enable 6. The DAI_PBEN1.PB06 bit field is the pin buffer enable for DAI port 6.

Pin Buffer Enable Register 2

The DAI_PBEN2 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

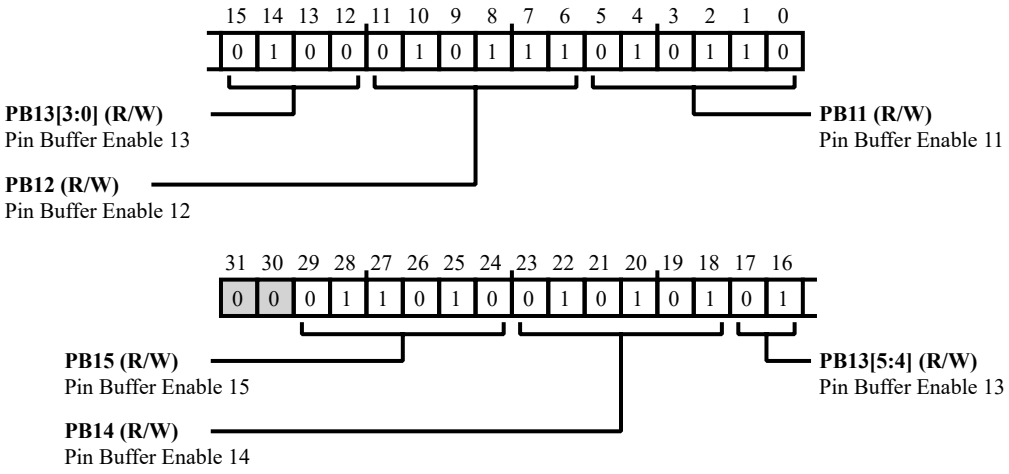


Figure 29-90: DAI_PBEN2 Register Diagram

Table 29-97: DAI_PBEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB15	Pin Buffer Enable 15. The DAI_PBEN2.PB15 bit field is the pin buffer enable for DAI port 15.
23:18 (R/W)	PB14	Pin Buffer Enable 14. The DAI_PBEN2.PB14 bit field is the pin buffer enable for DAI port 14.
17:12 (R/W)	PB13	Pin Buffer Enable 13. The DAI_PBEN2.PB13 bit field is the pin buffer enable for DAI port 13.
11:6 (R/W)	PB12	Pin Buffer Enable 12. The DAI_PBEN2.PB12 bit field is the pin buffer enable for DAI port 12.
5:0 (R/W)	PB11	Pin Buffer Enable 11. The DAI_PBEN2.PB11 bit field is the pin buffer enable for DAI port 11.

Pin Buffer Enable Register 3

The DAI_PBEN3 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

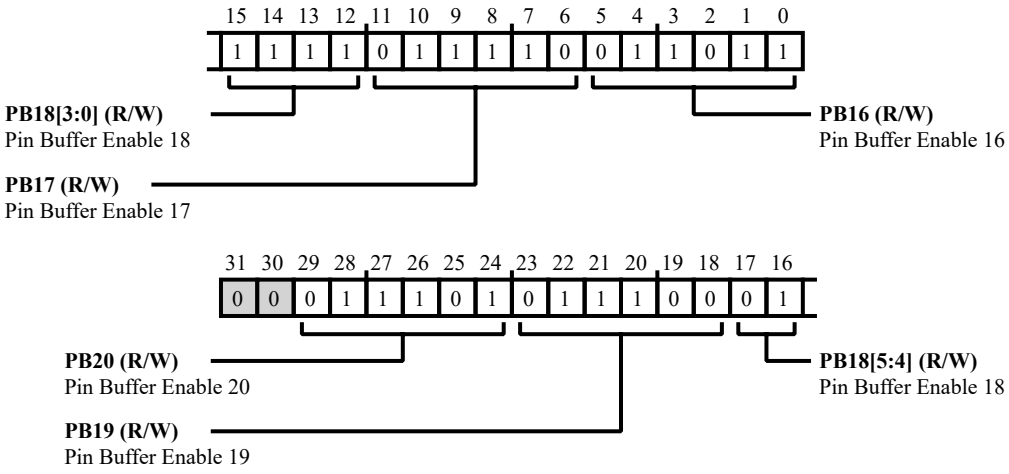


Figure 29-91: DAI_PBEN3 Register Diagram

Table 29-98: DAI_PBEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB20	Pin Buffer Enable 20. The DAI_PBEN3.PB20 bit field is the pin buffer enable for DAI port 20.
23:18 (R/W)	PB19	Pin Buffer Enable 19. The DAI_PBEN3.PB19 bit field is the pin buffer enable for DAI port 19.
17:12 (R/W)	PB18	Pin Buffer Enable 18. The DAI_PBEN3.PB18 bit field is the pin buffer enable for DAI port 18.
11:6 (R/W)	PB17	Pin Buffer Enable 17. The DAI_PBEN3.PB17 bit field is the pin buffer enable for DAI port 17.
5:0 (R/W)	PB16	Pin Buffer Enable 16. The DAI_PBEN3.PB16 bit field is the pin buffer enable for DAI port 16.

Pin Buffer Assignment Register 0

The DAI_PIN0 register routes physical pins that are connected to a bonded pad.

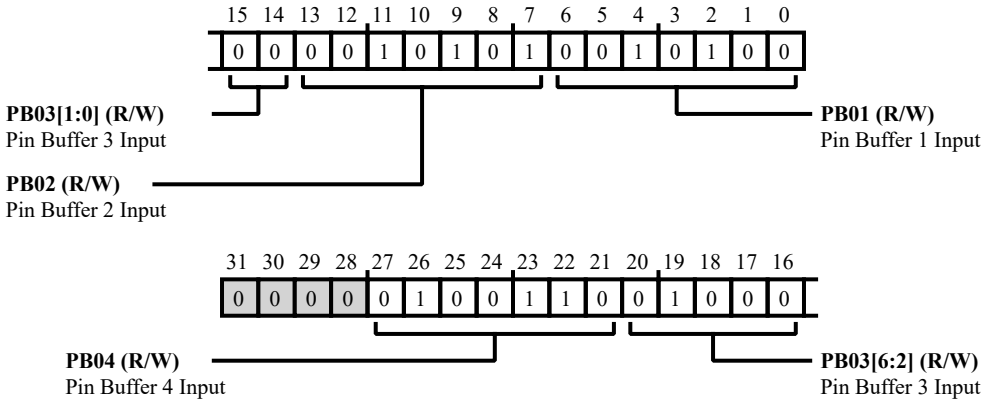


Figure 29-92: DAI_PIN0 Register Diagram

Table 29-99: DAI_PIN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB04	Pin Buffer 4 Input. DAI_PIN0.PB04 holds the source signal assignment that will be routed to the DAI_PIN0.PB04 destination. Refer to the Group D Signals table for source and destination mappings.
20:14 (R/W)	PB03	Pin Buffer 3 Input. DAI_PIN0.PB03 holds the source signal assignment that will be routed to the DAI_PIN0.PB03 destination. Refer to the Group D Signals table for source and destination mappings.
13:7 (R/W)	PB02	Pin Buffer 2 Input. DAI_PIN0.PB02 holds the source signal assignment that will be routed to the DAI_PIN0.PB02 destination. Refer to the Group D Signals table for source and destination mappings.
6:0 (R/W)	PB01	Pin Buffer 1 Input. DAI_PIN0.PB01 holds the source signal assignment that will be routed to the DAI_PIN0.PB01 destination. Refer to the Group D Signals table for source and destination mappings.

Pin Buffer Assignment Register 1

The DAI_PIN1 register routes physical pins that are connected to a bonded pad.

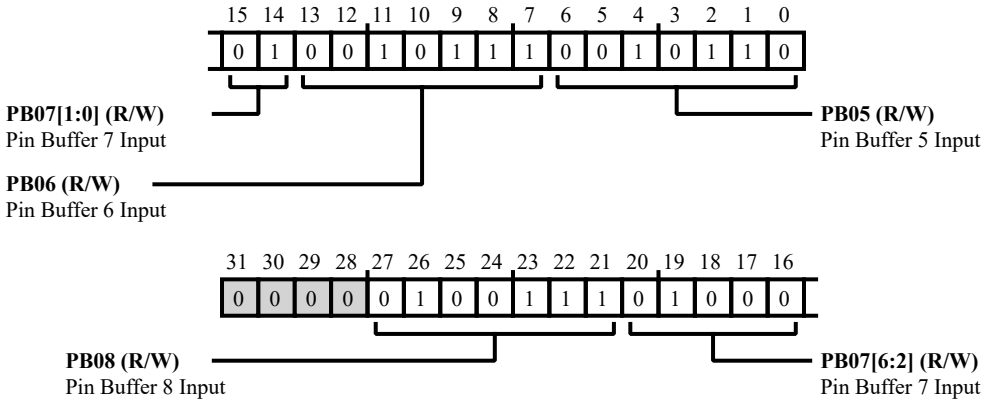


Figure 29-93: DAI_PIN1 Register Diagram

Table 29-100: DAI_PIN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB08	Pin Buffer 8 Input. DAI_PIN1.PB08 holds the source signal assignment that will be routed to the DAI_PIN1.PB08 destination. Refer to the Group D Signals table for source and destination mappings.
20:14 (R/W)	PB07	Pin Buffer 7 Input. DAI_PIN1.PB07 holds the source signal assignment that will be routed to the DAI_PIN1.PB07 destination. Refer to the Group D Signals table for source and destination mappings.
13:7 (R/W)	PB06	Pin Buffer 6 Input. DAI_PIN1.PB06 holds the source signal assignment that will be routed to the DAI_PIN1.PB06 destination. Refer to the Group D Signals table for source and destination mappings.
6:0 (R/W)	PB05	Pin Buffer 5 Input. DAI_PIN1.PB05 holds the source signal assignment that will be routed to the DAI_PIN1.PB05 destination. Refer to the Group D Signals table for source and destination mappings.

Pin Buffer Assignment Register 2

The DAI_PIN2 register routes physical pins that are connected to a bonded pad.

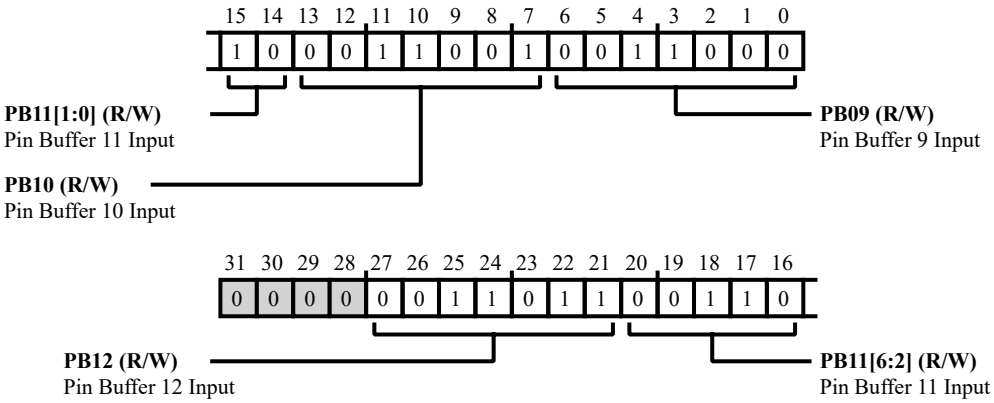


Figure 29-94: DAI_PIN2 Register Diagram

Table 29-101: DAI_PIN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB12	Pin Buffer 12 Input. DAI_PIN2.PB12 holds the source signal assignment that will be routed to the DAI_PIN2.PB12 destination. Refer to the Group D Signals table for source and destination mappings.
20:14 (R/W)	PB11	Pin Buffer 11 Input. DAI_PIN2.PB11 holds the source signal assignment that will be routed to the DAI_PIN2.PB11 destination. Refer to the Group D Signals table for source and destination mappings.
13:7 (R/W)	PB10	Pin Buffer 10 Input. DAI_PIN2.PB10 holds the source signal assignment that will be routed to the DAI_PIN2.PB10 destination. Refer to the Group D Signals table for source and destination mappings.
6:0 (R/W)	PB09	Pin Buffer 9 Input. DAI_PIN2.PB09 holds the source signal assignment that will be routed to the DAI_PIN2.PB09 destination. Refer to the Group D Signals table for source and destination mappings.

Pin Buffer Assignment Register 3

The DAI_PIN3 register routes physical pins that are connected to a bonded pad.

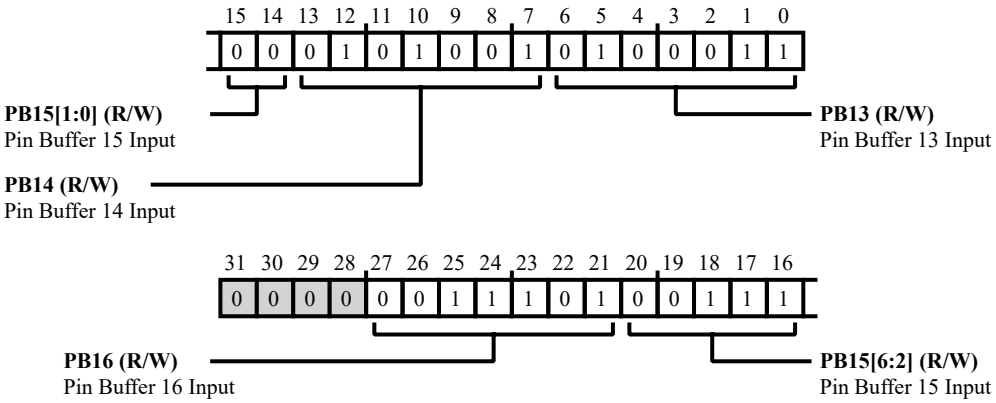


Figure 29-95: DAI_PIN3 Register Diagram

Table 29-102: DAI_PIN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB16	Pin Buffer 16 Input. The DAI_PIN3.PB16 bit field is the pin buffer 16 input.
20:14 (R/W)	PB15	Pin Buffer 15 Input. DAI_PIN3.PB15 holds the source signal assignment that will be routed to the DAI_PIN3.PB15 destination. Refer to the Group D Signals table for source and destination mappings.
13:7 (R/W)	PB14	Pin Buffer 14 Input. DAI_PIN3.PB14 holds the source signal assignment that will be routed to the DAI_PIN3.PB14 destination. Refer to the Group D Signals table for source and destination mappings.
6:0 (R/W)	PB13	Pin Buffer 13 Input. DAI_PIN3.PB13 holds the source signal assignment that will be routed to the DAI_PIN3.PB13 destination. Refer to the Group D Signals table for source and destination mappings.

Pin Buffer Assignment Register 4

The DAI_PIN4 register routes physical pins that are connected to a bonded pad.

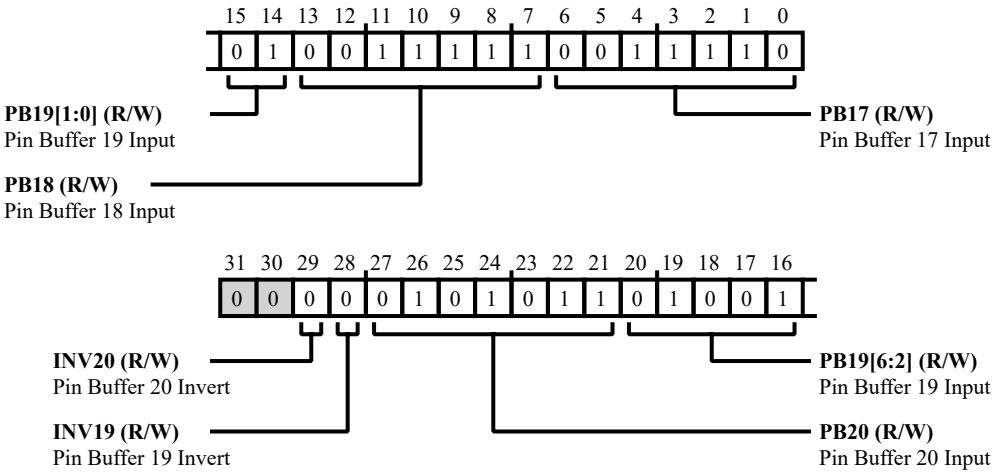


Figure 29-96: DAI_PIN4 Register Diagram

Table 29-103: DAI_PIN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	INV20	Pin Buffer 20 Invert. DAI_PIN4.INV20 holds the source signal assignment that will be routed to the DAI_PIN4.INV20 destination. Refer to the Group D Signals table for source and destination mappings.
28 (R/W)	INV19	Pin Buffer 19 Invert. DAI_PIN4.INV19 holds the source signal assignment that will be routed to the DAI_PIN4.INV19 destination. Refer to the Group D Signals table for source and destination mappings.
27:21 (R/W)	PB20	Pin Buffer 20 Input. DAI_PIN4.PB20 holds the source signal assignment that will be routed to the DAI_PIN4.PB20 destination. Refer to the Group D Signals table for source and destination mappings.
20:14 (R/W)	PB19	Pin Buffer 19 Input. DAI_PIN4.PB19 holds the source signal assignment that will be routed to the DAI_PIN4.PB19 destination. Refer to the Group D Signals table for source and destination mappings.
13:7 (R/W)	PB18	Pin Buffer 18 Input. DAI_PIN4.PB18 holds the source signal assignment that will be routed to the DAI_PIN4.PB18 destination. Refer to the Group D Signals table for source and destination mappings.

Table 29-103: DAI_PIN4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	PB17	Pin Buffer 17 Input. `DAI_PIN4.PB17` holds the source signal assignment that will be routed to the `DAI_PIN4.PB17` destination. Refer to the Group D Signals table for source and destination mappings.

Dummy Pin Buffer Assignment Register

The DAI_PIN5 register routes physical pins that are connected to a bonded pad.

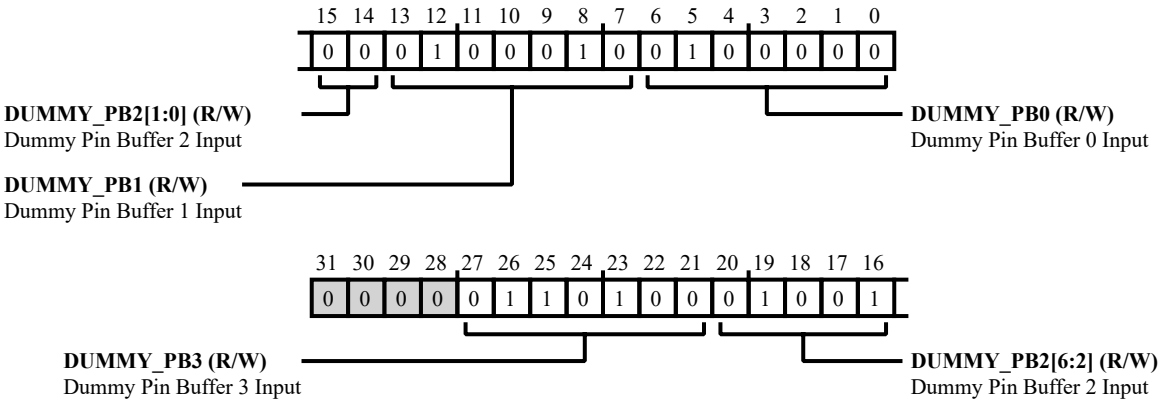


Figure 29-97: DAI_PIN5 Register Diagram

Table 29-104: DAI_PIN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	DUMMY_PB3	Dummy Pin Buffer 3 Input. DAI_PIN5.DUMMY_PB3 holds the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB3 destination. Refer to the Group D Signals table for source and destination mappings.
20:14 (R/W)	DUMMY_PB2	Dummy Pin Buffer 2 Input. DAI_PIN5.DUMMY_PB2 holds the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB2 destination. Refer to the Group D Signals table for source and destination mappings.
13:7 (R/W)	DUMMY_PB1	Dummy Pin Buffer 1 Input. DAI_PIN5.DUMMY_PB1 holds the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB1 destination. Refer to the Group D Signals table for source and destination mappings.
6:0 (R/W)	DUMMY_PB0	Dummy Pin Buffer 0 Input. DAI_PIN5.DUMMY_PB0 holds the source signal assignment that will be routed to the DAI_PIN5.DUMMY_PB0 destination. Refer to the Group D Signals table for source and destination mappings.

Pin Status Register

The DAI_PIN_STAT register bits indicate the status (signal high or low) for each pin. The individual bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

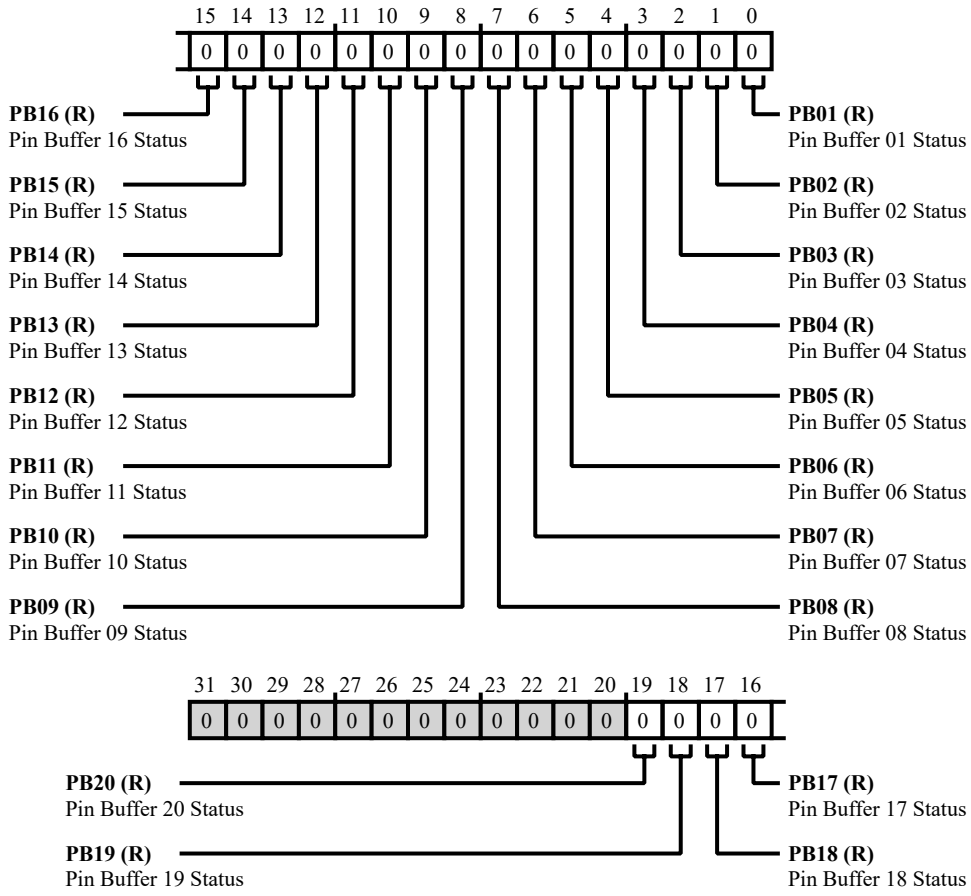


Figure 29-98: DAI_PIN_STAT Register Diagram

Table 29-105: DAI_PIN_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/NW)	PB20	Pin Buffer 20 Status. The DAI_PIN_STAT.PB20 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
18 (R/NW)	PB19	Pin Buffer 19 Status. The DAI_PIN_STAT.PB19 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

Table 29-105: DAI_PIN_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/NW)	PB18	Pin Buffer 18 Status. The DAI_PIN_STAT.PB18 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
16 (R/NW)	PB17	Pin Buffer 17 Status. The DAI_PIN_STAT.PB17 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
15 (R/NW)	PB16	Pin Buffer 16 Status. The DAI_PIN_STAT.PB16 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
14 (R/NW)	PB15	Pin Buffer 15 Status. The DAI_PIN_STAT.PB15 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
13 (R/NW)	PB14	Pin Buffer 14 Status. The DAI_PIN_STAT.PB14 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
12 (R/NW)	PB13	Pin Buffer 13 Status. The DAI_PIN_STAT.PB13 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
11 (R/NW)	PB12	Pin Buffer 12 Status. The DAI_PIN_STAT.PB12 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
10 (R/NW)	PB11	Pin Buffer 11 Status. The DAI_PIN_STAT.PB11 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
9 (R/NW)	PB10	Pin Buffer 10 Status. The DAI_PIN_STAT.PB10 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
8 (R/NW)	PB09	Pin Buffer 09 Status. The DAI_PIN_STAT.PB09 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
7 (R/NW)	PB08	Pin Buffer 08 Status. The DAI_PIN_STAT.PB08 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

Table 29-105: DAI_PIN_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/NW)	PB07	Pin Buffer 07 Status. The DAI_PIN_STAT.PB07 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
5 (R/NW)	PB06	Pin Buffer 06 Status. The DAI_PIN_STAT.PB06 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
4 (R/NW)	PB05	Pin Buffer 05 Status. The DAI_PIN_STAT.PB05 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
3 (R/NW)	PB04	Pin Buffer 04 Status. The DAI_PIN_STAT.PB04 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
2 (R/NW)	PB03	Pin Buffer 03 Status. The DAI_PIN_STAT.PB03 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
1 (R/NW)	PB02	Pin Buffer 02 Status. The DAI_PIN_STAT.PB02 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
0 (R/NW)	PB01	Pin Buffer 01 Status. The DAI_PIN_STAT.PB01 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

30 Serial Port (SPORT)

The programmable serial ports (SPORTs) support various protocols for serial data communication and provide a glueless hardware interface to many industry-standard data converters and codecs. They have high data rates and dual half-duplex data paths and are ideal for establishing a direct serial connection among two or more processors in a multiprocessor system, as many processors provide compatible serial interfaces.

The SPORT top module consists of two half SPORTs with identical functionality and programming requirements. Each half SPORT can be independently configured as either a transmitter or receiver and can be coupled with the other half SPORT within the same SPORT top module. Further, each half SPORT provides two synchronous half-duplex data lines to double the total supported throughput. As such, a single SPORT top module can be used to provide up to four unidirectional or up to two full-duplex data streams. Further channels are possible as well, but utilization of multiple SPORT top modules is required, thus requiring external connections to provide a common time base.

Features

An individual SPORT top module consists of two independently configurable SPORT halves with identical functionality. These SPORT halves offer the following features:

- Up to two bidirectional data lines - each half SPORT supports up to two transmit or receive channels, thus allowing two unidirectional streams into or out of each half SPORT and providing greater flexibility for serial communications. If full-duplex functionality is desired, two SPORT halves can be combined to enable dual-stream bidirectional communication.
- Six operating modes:
    1. Standard DSP serial mode
    2. I^2S mode
    3. Left-Justified mode
    4. Right-Justified mode
    5. Multichannel (TDM) mode
    6. Packed mode

- Supports internally or externally generated clock.
- Support for both even and odd SCLK to SPORT clock (SPORT_CLK) ratios. If both data lines of a half SPORT are active, the maximum throughput is 2 x SPORT_CLK bps.
- Configurable rising or falling edge of the SPORT_CLK for driving and sampling data and frame syncs.
- Gated clock mode support for internally or externally generated clocks in DSP serial mode and stereo modes (left-justified and I^2S mode).
- Supports frameless operation.
- Supports internally or externally generated frame sync signals.
- Programmable frame sync polarity.
- Programmable frame sync timing (synchronous to data or 1 SPORT clock in advance of it).
- Detection of prematurely received external frame syncs (with optional interrupt request generation).
- Programmable level-/edge-sensitivity for external frame syncs.
- Programmable (4–32-bit) data length, either in most significant bit (MSB) first or in least significant bit (LSB) first format, with optional sign-extension on received data.
- Optional 16-bit to 32-bit word packing (as receiver) and 32-bit to 16-bit word unpacking (as transmitter).
- Support for A-law and μ-law compression/decompression hardware companding, according to the G.711 specification, on transmitted/received words in all operating modes.
- Transmit underrun and receive overflow detection (with optional interrupt request generation).
- TDM mode transfers data on 128 contiguous channels from a stream of up to 1024 total channels (useful for H.100/H.110 and other telephony interfaces as a network communication scheme for multiple processors).
- Performs interrupt-driven, single word core transfers to and from on-chip or off-chip memory.
- Dedicated DMA channel for each SPORT half supporting autobuffer (for a repeated, identical range of transfers) and numerous descriptor-based (individual or repeated ranges of transfers with differing DMA parameters) modes.
- Generator and receiver trigger functionality.
- Unique transfer finish interrupt (TFI) signaling when the last transmit word is fully out of the transmit shift register.
- Multiplexer to internally connect critical timing signals between SPORT halves.
- Support for Global Sport Enable functionality, where enabling multiple SPORTs need to be synchronized so that all of them start and/or end at the same time.
- Support for late reception of SPORT data and frame sync.

Signal Descriptions

Each half SPORT module has five dedicated signals, as described in the ***SPORT Signal Descriptions*** table. The actual pin name varies with different SPORT halves. Individual SPORT halves do not share any of its signals across the pair that comprises the SPORT top module; however, it is possible to connect the clock and frame sync signals between the SPORT half pair, as explained in the Multiplexer Logic section.

All of the SPORT signals are multiplexed on the PORT pins, possibly sharing functionality with other peripherals on the device. By default, the PORT pins are in GPIO mode and must be reconfigured for SPORT functionality by setting the appropriate bits in the PORT_FER and PORT_MUX registers. Consult the processor data sheet for details regarding which ports the SPORT signals are available and to ensure to configure the PORT_MUX register before the PORT_FER register.

Table 30-1: SPORT Signal Descriptions

Internal Node	Direction	Description
SPORTx_CLK	I/O	Transmit or receive serial clock. Data and frame syncs are driven or sampled on this clock's edges. This signal can be either internally or externally generated.
SPORTx_FS	I/O	Transmit or receive frame sync. The frame sync pulse initiates shifting of serial data. This signal is either internally or externally generated.
SPORTx_D0	I/O	Primary transmit or receive data channel. This signal can be configured as an output to transmit serial data or as an input to receive serial data.
SPORTx_D1	I/O	Secondary transmit or receive data channel. This signal can be configured as an output to transmit serial data or as an input to receive serial data.
SPORTx_TDV	O	Multichannel transmit data valid. This signal is only active in multichannel transmit mode and is asserted during enabled slots, as defined by the channel selection registers (SPORT_CS0_A through SPORT_CS3_B).

The data channel signals are transmit signals when the serial port is configured in transmit mode (SPORT_CTL_A.SPTRAN = 1). They are receive signals when the serial port is configured in receive mode (SPORT_CTL_A.SPTRAN = 0). The following sections further describe the SPORT signals.

NOTE: These sections explicitly refer to the registers associated with half SPORT A, but the same concepts also apply to half SPORT B.

Serial Clock

The serial port clock (SPORT_ACLK) is either a receive serial clock or a transmit serial clock, depending on the transfer direction (SPORT_CTL_A.SPTRAN), governing when the data bits are serially shifted into or out of the SPORT and when the frame sync signal is driven (in internal frame sync mode) or sampled (in external frame sync mode). It can be internally generated from the processor's system clock (SCLK) or externally provided. If the half SPORT is configured in internal clock mode (SPORT_CTL_A.ICLK= 1), then the SPORT_DIV_A.CLKDIV field specifies the divisor applied to SCLK to generate the SPORT clock. As it is a 16-bit divisor, a wide range of serial clock rates is possible. Use the following equation to calculate the serial clock frequency:

SPORT_ACLK = [SCLK ÷ (SPORT_DIV_A.CLKDIV + 1)]

From this, the following equation can be used to determine the value of SPORT_DIV_A.CLKDIV, given the SCLK frequency and the desired frequency of the SPORT clock:

SPORT_DIV_A.CLKDIV= [(SCLK ÷ SPORT_ACLK) - 1]

The half SPORT also supports a 1:1 SPORT_ACLK to ratio (per the equations above, program the clock divisor field to zero). In this case, the resulting SPORT clock frequency is equal to SCLK.

NOTE: Be careful not to exceed the maximum SPORT_ACLK frequency specified in the processor data sheet.

In certain operating modes, the SPORT can be configured to generate a gated clock, which is active only during valid data. In some applications, a SPORT uses it to generate a general-purpose clock in the system. In this case, enable the SPORT with the appropriate SPORT_DIV_A.CLKDIV divisor field in internal clock mode.

If a SPORT is configured in external clock mode (SPORT_CTL_A.ICLK= 0), the serial clock is an input signal, thus making the SPORT operate in target mode. In this mode, the SPORT_DIV_A.CLKDIV is irrelevant and is ignored. The optional loopback capability provided by the internal SPORT multiplexer (SPMUX) block allows the target SPORT to use the serial clock from the neighboring SPORT half in the same SPORT top module.

An externally-supplied serial clock does not need to be synchronous with the processor clocks. Further, the external clock can be a gated clock, but it must comply with the requirements described in the Gated Clock Mode section.

Refer to the product data sheet for exact AC timing specifications.

Frame Sync

The SPORT frame sync (SPORT_AFS) signal is either a receive frame sync or a transmit frame sync, depending on the transfer direction (SPORT_CTL_A.SPTRAN), which is used to determine the start of a new word or frame. When this signal goes active, the serial port starts serially shifting data into or out of the SPORT. The frame sync signal can be internally generated based on its serial clock (SPORT_ACLK) or externally provided, as configured by the SPORT_CTL_A.IFS bit.

If the half SPORT is configured to generate frame syncs (SPORT_CTL_A.IFS= 1), then the SPORT_DIV_A.FSDIV field specifies the divisor used to generate the periodic SPORT_AFS signal from the SPORT clock. As this is a 16-bit divisor, a wide range of frame sync rates to initiate periodic transfers is possible. Whether the serial clock is internally or externally generated, this divisor is a count of SPORT clock cycles between frame sync pulses, the formula for which is:

Number of serial clocks between frame syncs = (SPORT_DIV_A.FSDIV + 1)

From this, the following equation can be used to determine the value of SPORT_DIV_A.FSDIV, given the serial clock frequency and the desired frame sync frequency:

SPORT_DIV_A.FSDIV = [(SPORT_ACLK ÷ SPORT_AFS) – 1]

The frame sync is continuously active when SPORT_DIV_A.FSDIV= 0. The value of SPORT_DIV_A.FSDIV cannot be less than the serial word length minus one (the value of the SPORT_CTL_A.SLEN bit field). Failure

to adhere to this guideline can cause an external device to abort the current operation or cause other unpredictable results.

NOTE: After enabling the SPORT, the first internal frame sync appears after a delay of SPORT_DIV_A.FSDIV+ 3 SPORT clock cycles.

If a SPORT is configured for external frame syncs (SPORT_CTL_A.IFS = 0), then SPORT_AFS is an input signal and the SPORT_DIV_A.FSDIV field of the SPORT_DIV_A register is irrelevant and ignored. By default, this external signal is level-sensitive, but it can be configured as an edge-sensitive signal by setting the SPORT_CTL_A.FSED bit. The frame sync is expected to be synchronous with the serial clock. If not, it must meet the timing requirements that appear in the data sheet.

The SPORT can be used as a counter for dividing an external clock to generate periodic pulses or periodic interrupts. To do so, enable the SPORT with the appropriate SPORT_DIV_A.FSDIV divisor field with the SPORT configured for an external clock and internal data-independent frame syncs.

In some of the operating modes, the SPORT can be programmed to treat the frame sync signal as an optional signal by clearing the SPORT_CTL_A.FSR bit. Even with this bit cleared, the SPORT requires a single frame sync assertion to start the continuous transfers, after which it is ignored (for externally supplied frame syncs) or not generated (for internally-generated frame syncs). Characteristics of the frame sync depend on the settings in the SPORT control registers and the operating mode of the SPORT. For more information, refer to the SPORT_CTL_A register.

Data Signals

Each half SPORT has two bidirectional data lines known as the primary (SPORT_AD0) and secondary (SPORT_AD1) data channels. Both data lines can be configured as either transmitters or receivers using the SPORT_CTL_A.SPTRAN bit, thus permitting dual unidirectional data streams to increase the data throughput of the SPORT.

NOTE: Configuring one transmit data channel and one receive data channel on a single half SPORT is not supported.

The primary and secondary data lines can be individually enabled or disabled using the SPORT_CTL_A.SPENPRI and the SPORT_CTL_A.SPENSEC bits, respectively. However, if using both, enable or disable them concurrently. These data lines operate in a synchronous manner (sharing a clock and frame sync) but have separate data paths with unique data buffers, shift registers and optional companding logic. All of the SPORT control settings are common for both channels, but the single DMA channel per half SPORT serves both the primary and secondary data channels.

When a SPORT is configured in multichannel transmit mode, the data pins three-state during inactive channel slots, thus allowing multiple transmitters to operate on the same bus with different active channels.

See the Architectural Concepts section for more details about data transfer operation.

Transmit Data Valid Signal

The transmit data valid (SPORT_ATDV) signal is available only in transmit multichannel modes (including packed mode). It is driven active during enabled multichannel slots, and it is driven inactive during the disabled channels. In other words, the SPORT_ATDV signal is active when data is being driven to the data pins and inactive when the data pins are being three-stated. As such, the SPORT_ATDV signal can serve as an output-enable signal for the data transmit pin.

SRU Programming

The SPORT uses the SRU (signal routing unit) to connect the SPORT data, serial clock, frame sync, and external sync (if external synchronization is required). Inputs also must be routed through the SRU. Program the corresponding SRU registers to connect the outputs to the required destinations. For details of the routing, see DAI Routing Capabilities in the *Digital Audio Interface (DAI)* chapter.

SRU SPORT Receive Controller

If the SPORT is receiver, it must feed its controller output clock back to its input clock. This is required to trigger the SPORT's state machine. Using SPORT 0A as an example receiver, programs should route SPT0_ACLK_O to SPT0_ACLK_I. This is not required if the SPORT is operating as a transmitter in controller mode.

Functional Description

The following sections provide general information about the functionality of the processor's serial ports:

- Architectural Concepts
- Data Types and Companding
- Transmit Path
- Receive Path

ADSP-2184x SPORT Register List

The Serial Port (SPORT) controller, with its range of clock and frame synchronization options, supports a variety of serial communication protocols and provides a glueless hardware interface to many industry-standard data converters and CODECs. Each SPORT has two independent halves (A and B), and each half contains two channels (primary and secondary). A set of registers governs SPORT operations. For more information on SPORT functionality, see the SPORT register descriptions.

Table 30-2: ADSP-2184x SPORT Register List

Name	Description
SPORT_CS0_A	Half SPORT 'A' Multichannel 0-31 Select Register
SPORT_CS0_B	Half SPORT 'B' Multichannel 0-31 Select Register

Table 30-2: ADSP-2184x SPORT Register List (Continued)

Name	Description
SPORT_CS1_A	Half SPORT 'A' Multichannel 32-63 Select Register
SPORT_CS1_B	Half SPORT 'B' Multichannel 32-63 Select Register
SPORT_CS2_A	Half SPORT 'A' Multichannel 64-95 Select Register
SPORT_CS2_B	Half SPORT 'B' Multichannel 64-95 Select Register
SPORT_CS3_A	Half SPORT 'A' Multichannel 96-127 Select Register
SPORT_CS3_B	Half SPORT 'B' Multichannel 96-127 Select Register
SPORT_CTL2_A	Half SPORT 'A' Control 2 Register
SPORT_CTL2_B	Half SPORT 'B' Control 2 Register
SPORT_CTL_A	Half SPORT 'A' Control Register
SPORT_CTL_B	Half SPORT 'B' Control Register
SPORT_DIV_A	Half SPORT 'A' Divisor Register
SPORT_DIV_B	Half SPORT 'B' Divisor Register
SPORT_ERR_A	Half SPORT 'A' Error Register
SPORT_ERR_B	Half SPORT 'B' Error Register
SPORT_MCTL_A	Half SPORT 'A' Multichannel Control Register
SPORT_MCTL_B	Half SPORT 'B' Multichannel Control Register
SPORT_MSTAT_A	Half SPORT 'A' Multichannel Status Register
SPORT_MSTAT_B	Half SPORT 'B' Multichannel Status Register
SPORT_RXPRI_A	Half SPORT 'A' Rx Buffer (Primary) Register
SPORT_RXPRI_B	Half SPORT 'B' Rx Buffer (Primary) Register
SPORT_RXSEC_A	Half SPORT 'A' Rx Buffer (Secondary) Register
SPORT_RXSEC_B	Half SPORT 'B' Rx Buffer (Secondary) Register
SPORT_TXPRI_A	Half SPORT 'A' Tx Buffer (Primary) Register
SPORT_TXPRI_B	Half SPORT 'B' Tx Buffer (Primary) Register
SPORT_TXSEC_A	Half SPORT 'A' Tx Buffer (Secondary) Register
SPORT_TXSEC_B	Half SPORT 'B' Tx Buffer (Secondary) Register

ADSP-2184x SPORT Interrupt List

Table 30-3: ADSP-2184x SPORT Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
254	SPORT0_A_DMA	SPORT0 Channel A DMA	Level	0
255	SPORT0_A_STAT	SPORT0 Channel A Status	Level	
256	SPORT0_B_DMA	SPORT0 Channel B DMA	Level	1
257	SPORT0_B_STAT	SPORT0 Channel B Status	Level	
258	SPORT1_A_DMA	SPORT1 Channel A DMA	Level	2
259	SPORT1_A_STAT	SPORT1 Channel A Status	Level	
260	SPORT1_B_DMA	SPORT1 Channel B DMA	Level	3
261	SPORT1_B_STAT	SPORT1 Channel B Status	Level	
262	SPORT2_A_DMA	SPORT2 Channel A DMA	Level	4
263	SPORT2_A_STAT	SPORT2 Channel A Status	Level	
264	SPORT2_B_DMA	SPORT2 Channel B DMA	Level	5
265	SPORT2_B_STAT	SPORT2 Channel B Status	Level	
266	SPORT3_A_DMA	SPORT3 Channel A DMA	Level	6
267	SPORT3_A_STAT	SPORT3 Channel A Status	Level	
268	SPORT3_B_DMA	SPORT3 Channel B DMA	Level	7
269	SPORT3_B_STAT	SPORT3 Channel B Status	Level	
270	SPORT4_A_DMA	SPORT4 Channel A DMA	Level	10
271	SPORT4_A_STAT	SPORT4 Channel A Status	Level	
272	SPORT4_B_DMA	SPORT4 Channel B DMA	Level	11
273	SPORT4_B_STAT	SPORT4 Channel B Status	Level	
274	SPORT5_A_DMA	SPORT5 Channel A DMA	Level	12
275	SPORT5_A_STAT	SPORT5 Channel A Status	Level	
276	SPORT5_B_DMA	SPORT5 Channel B DMA	Level	13
277	SPORT5_B_STAT	SPORT5 Channel B Status	Level	
278	SPORT6_A_DMA	SPORT6 Channel A DMA	Level	14
279	SPORT6_A_STAT	SPORT6 Channel A Status	Level	
280	SPORT6_B_DMA	SPORT6 Channel B DMA	Level	15
281	SPORT6_B_STAT	SPORT6 Channel B Status	Level	
282	SPORT7_A_DMA	SPORT7 Channel A DMA	Level	16
283	SPORT7_A_STAT	SPORT7 Channel A Status	Level	
284	SPORT7_B_DMA	SPORT7 Channel B DMA	Level	17

Table 30-3: ADSP-2184x SPORT Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
285	SPORT7_B_STAT	SPORT7 Channel B Status	Level	
290	SPORT0_A_DMA_ERR	SPORT0 Channel A DMA Error	Level	
291	SPORT0_B_DMA_ERR	SPORT0 Channel B DMA Error	Level	
292	SPORT1_A_DMA_ERR	SPORT1 Channel A DMA Error	Level	
293	SPORT1_B_DMA_ERR	SPORT1 Channel B DMA Error	Level	
294	SPORT2_A_DMA_ERR	SPORT2 Channel A DMA Error	Level	
295	SPORT2_B_DMA_ERR	SPORT2 Channel B DMA Error	Level	
296	SPORT3_A_DMA_ERR	SPORT3 Channel A DMA Error	Level	
297	SPORT3_B_DMA_ERR	SPORT3 Channel B DMA Error	Level	
298	SPORT4_A_DMA_ERR	SPORT4 Channel A DMA Error	Level	
299	SPORT4_B_DMA_ERR	SPORT4 Channel B DMA Error	Level	
300	SPORT5_A_DMA_ERR	SPORT5 Channel A DMA Error	Level	
301	SPORT5_B_DMA_ERR	SPORT5 Channel B DMA Error	Level	
302	SPORT6_A_DMA_ERR	SPORT6 Channel A DMA Error	Level	
303	SPORT6_B_DMA_ERR	SPORT6 Channel B DMA Error	Level	
304	SPORT7_A_DMA_ERR	SPORT7 Channel A DMA Error	Level	
305	SPORT7_B_DMA_ERR	SPORT7 Channel B DMA Error	Level	

ADSP-2184x SPORT Trigger List

Table 30-4: ADSP-2184x SPORT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
125	SPORT0_A_DMA	SPORT0 Channel A DMA	Edge
126	SPORT0_B_DMA	SPORT0 Channel B DMA	Edge
127	SPORT1_A_DMA	SPORT1 Channel A DMA	Edge
128	SPORT1_B_DMA	SPORT1 Channel B DMA	Edge
129	SPORT2_A_DMA	SPORT2 Channel A DMA	Edge
130	SPORT2_B_DMA	SPORT2 Channel B DMA	Edge
131	SPORT3_A_DMA	SPORT3 Channel A DMA	Edge
132	SPORT3_B_DMA	SPORT3 Channel B DMA	Edge
133	SPORT4_A_DMA	SPORT4 Channel A DMA	Edge

Table 30-4: ADSP-2184x SPORT Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
134	SPORT4_B_DMA	SPORT4 Channel B DMA	Edge
135	SPORT5_A_DMA	SPORT5 Channel A DMA	Edge
136	SPORT5_B_DMA	SPORT5 Channel B DMA	Edge
137	SPORT6_A_DMA	SPORT6 Channel A DMA	Edge
138	SPORT6_B_DMA	SPORT6 Channel B DMA	Edge
139	SPORT7_A_DMA	SPORT7 Channel A DMA	Edge
140	SPORT7_B_DMA	SPORT7 Channel B DMA	Edge

Table 30-5: ADSP-2184x SPORT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
157	SPORT0_A_DMA	SPORT0 Channel A DMA	Pulse
158	SPORT0_B_DMA	SPORT0 Channel B DMA	Pulse
159	SPORT1_A_DMA	SPORT1 Channel A DMA	Pulse
160	SPORT1_B_DMA	SPORT1 Channel B DMA	Pulse
161	SPORT2_A_DMA	SPORT2 Channel A DMA	Pulse
162	SPORT2_B_DMA	SPORT2 Channel B DMA	Pulse
163	SPORT3_A_DMA	SPORT3 Channel A DMA	Pulse
164	SPORT3_B_DMA	SPORT3 Channel B DMA	Pulse
165	SPORT4_A_DMA	SPORT4 Channel A DMA	Pulse
166	SPORT4_B_DMA	SPORT4 Channel B DMA	Pulse
167	SPORT5_A_DMA	SPORT5 Channel A DMA	Pulse
168	SPORT5_B_DMA	SPORT5 Channel B DMA	Pulse
169	SPORT6_A_DMA	SPORT6 Channel A DMA	Pulse
170	SPORT6_B_DMA	SPORT6 Channel B DMA	Pulse
171	SPORT7_A_DMA	SPORT7 Channel A DMA	Pulse
172	SPORT7_B_DMA	SPORT7 Channel B DMA	Pulse

ADSP-2184x SPORT DMA Channel List

Table 30-6: ADSP-2184x SPORT DMA Channel List

DMA ID	DMA Channel Name	Description
DMA0	SPORT0_A_DMA	SPORT0 Channel A DMA

Table 30-6: ADSP-2184x SPORT DMA Channel List (Continued)

DMA ID	DMA Channel Name	Description
DMA1	SPORT0_B_DMA	SPORT0 Channel B DMA
DMA2	SPORT1_A_DMA	SPORT1 Channel A DMA
DMA3	SPORT1_B_DMA	SPORT1 Channel B DMA
DMA4	SPORT2_A_DMA	SPORT2 Channel A DMA
DMA5	SPORT2_B_DMA	SPORT2 Channel B DMA
DMA6	SPORT3_A_DMA	SPORT3 Channel A DMA
DMA7	SPORT3_B_DMA	SPORT3 Channel B DMA
DMA10	SPORT4_A_DMA	SPORT4 Channel A DMA
DMA11	SPORT4_B_DMA	SPORT4 Channel B DMA
DMA12	SPORT5_A_DMA	SPORT5 Channel A DMA
DMA13	SPORT5_B_DMA	SPORT5 Channel B DMA
DMA14	SPORT6_A_DMA	SPORT6 Channel A DMA
DMA15	SPORT6_B_DMA	SPORT6 Channel B DMA
DMA16	SPORT7_A_DMA	SPORT7 Channel A DMA
DMA17	SPORT7_B_DMA	SPORT7 Channel B DMA

Block Diagram

Each SPORT top module consists of two separate blocks, known as half SPORTs (HSPORT) A and B, each with identical functionality and programming models. The ***Half Serial Port Block Diagram*** shows a detailed block diagram of a half SPORT.

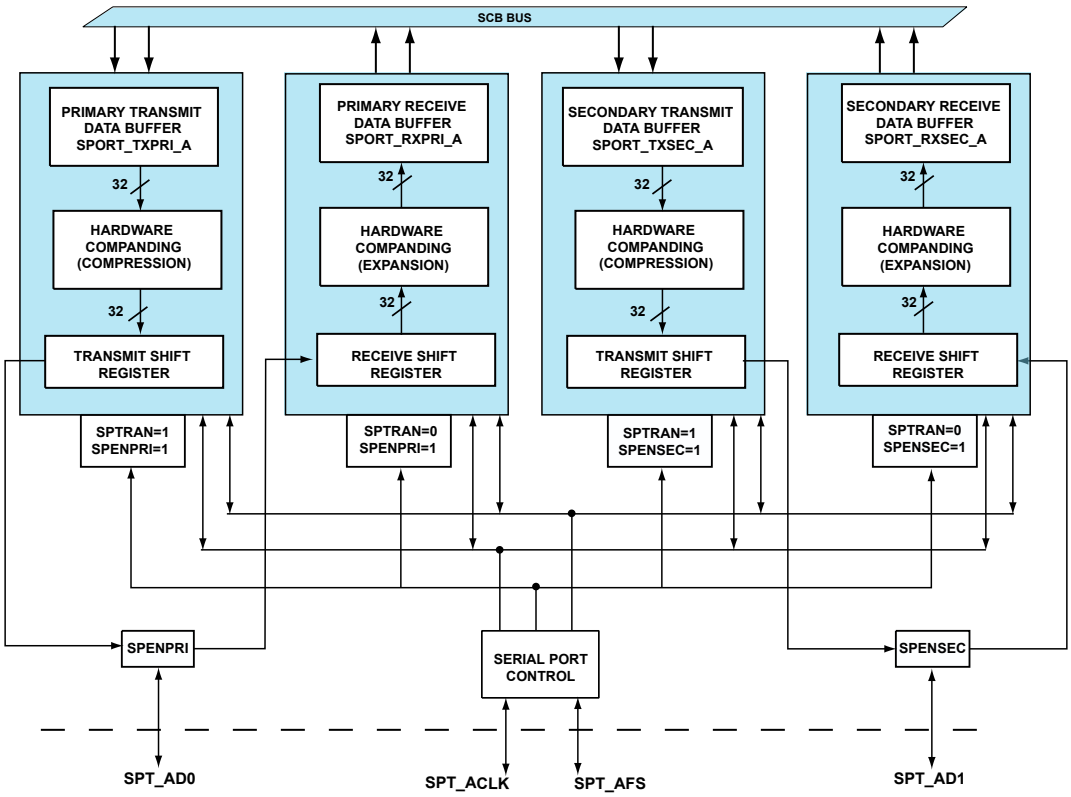


Figure 30-1: Half Serial Port Block Diagram

Architectural Concepts

Each half SPORT (HSPORT) block has its own set of control registers and data buffers, grouped per SPORT module. The HSPORT A and B blocks can be independently configured as either a transmitter or a receiver, with the option to be coupled together internally within the single SPORT top module. Each HSPORT also supports two synchronous bidirectional datapaths, referred to as the primary (`D0`) and secondary (`D1`) data lines, as shown in the *Top-Level SPORT Diagram* figure.

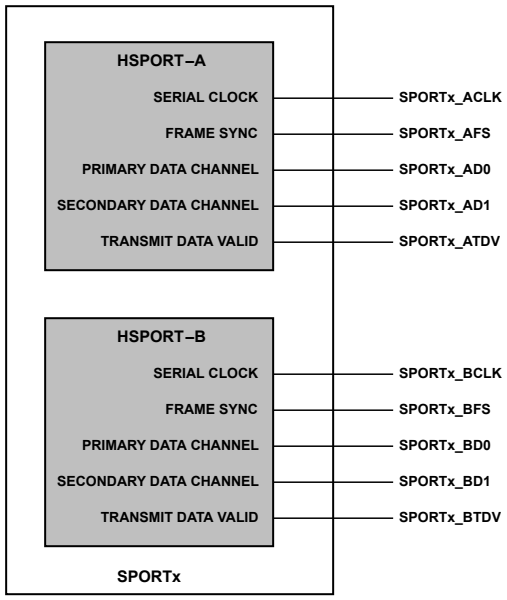


Figure 30-2: Top-Level SPORT Diagram

The SPORT_CTL_A.SPTRAN bit controls the direction for both datapaths of the HSPORT. Depending on whether the HSPORT is a transmitter or a receiver, the pair of data signals respectfully transmit or receive data bits synchronously. The dual data signals of each HSPORT cannot transmit and receive the data simultaneously in support of full-duplex operation, however, two HSPORTs can be combined to achieve this.

Serial communications are synchronized to the serial clock signal, where a valid clock pulse must accompany each data bit. Each HSPORT can take its clock from an external source or internally generate it from the processor's system clock using the SPORT_DIV_A.CLKDIV clock divisor bit field. Both primary and secondary data channels shift data based on the SPORT_CLK rate and the clock polarity defined by the SPORT_CTL_A.CKRE bit.

In addition to the serial clock signal, a frame synchronization signal is used to signify the beginning of an individual data word or a multichannel data stream (block of words). Each SPORT can take the frame sync signal from an external source or generate it (SPORT_FS), depending on the SPORT_CTL_A.IFS bit. An internally generated frame sync is derived from the SPORT clock using the SPORT_DIV_A.FSDIV divisor field. Both primary and secondary datapaths start shifting data either synchronous to or one serial clock in advance of detecting/generating a valid frame sync signal, as determined by the SPORT_CTL_A.LAFS bit. Various communication protocols for serial data can be emulated according to the frame sync format, and all frame sync options are available whether the signal is generated internally or externally.

NOTE: These SPORTs are not UARTs and cannot communicate with an RS-232 device or any other asynchronous communications protocol.

Multiplexer Logic

The SPORT multiplexing block (SPMUX) is situated between the SPORT hardware block and the processor's pin multiplexing logic. It allows the flexibility to route and share the clock and frame sync signals between the HSPORT A and B halves within each SPORT top module, which can double the data throughput (if both SPORT halves are transmitters or both are receivers) or provide full-duplex capabilities (if one HSPORT is a receiver and the other a

transmitter) without the need to allocate pins for the peripheral or make physical connections outside the processor. The SPORT_CTL2_A register is used to configure this loopback feature.

NOTE: Throughout this section, HSPORT A is used as a reference, but all concepts also apply to HSPORT B.

The multiplexing depends on the configuration of the SPORT_CTL_A.IFS and SPORT_CTL_A.ICLK bits, and the SPORT_CTL2_A.CKMUXSEL and SPORT_CTL2_A.FSMUXSEL bit settings control the multiplexing. The ***Frame Sync Combinations*** and ***Clock Combinations*** tables show the valid combinations for the bit settings.

NOTE: All other settings are illegal. However, hardware does not check or prevent the illegal settings. Ensure that programs use only legal combinations.

The column headers in the ***Frame Sync Combinations*** table are defined as follows:

- FS Combination = Frame sync combination, referenced in the notes that follow the ***Clock Combinations*** table.
- HSA_IFS = the setting of the SPORT_CTL_A.IFS configuration bit.
- HSB_IFS = the setting of the SPORT_CTL_B.IFS configuration bit.
- FSAMUX = the setting of the SPORT_CTL2_A.FSMUXSEL configuration bit.
- FSBMUX = the setting of the SPORT_CTL2_B.FSMUXSEL configuration bit.

The Routing column in the ***Frame Sync Combinations*** table defines how the signals are used between the SPORT halves and which pin is used for the frame sync (whether it is an input or an output). Within the column, the inequality characters (≤ and ≥) are used to show the direction of the signal, and the following abbreviations are used (where x = A or B):

- HSx_FI = Frame sync input signal, provided by an external device or the complementing HSPORT.
- HSx_FO = Frame sync output signal.
- SPx_FS = HSPORT's frame sync pin, where the signal is either:
  - Provided by an external source and distributed to both HSPORT frame sync signals.
  - Internally generated by one HSPORT and routed to both the pin and to the complementary HSPORT frame sync signal.

Table 30-7: Frame Sync Combinations

FS Combination	HSA_IFS	HSB_IFS	FSAMUX	FSBMUX	Routing
1	0	0	0	0	Native FS Operation
2	0	1	0	0	Native FS Operation
3	1	0	0	0	Native FS Operation
4	1	1	0	0	Native FS Operation

Table 30-7: Frame Sync Combinations (Continued)

FS Combination	HSA_IFS	HSB_IFS	FSAMUX	FSBMUX	Routing
5	0	0	1	0	HSA_FI ≤ SPB_FS; HSB_FI ≤ SPB_FS
6	0	1	1	0	HSA_FI ≤ HSB_FO ≥ SPB_FS
7	0	0	0	1	HSB_FI ≤ SPA_FS; HSA_FI ≤ SPA_FS
8	1	0	0	1	HSB_FI ≤ HSA_FO ≥ SPA_FS

The column headers in the ***Clock Combinations*** table are defined as follows:

- CLK Combination = Clock combination, referenced in the notes that follow the table.
- HSA_ICLK = the setting of the `SPORT_CTL_A.ICLK` configuration bit.
- HSB_ICLK = the setting of the `SPORT_CTL_B.ICLK` configuration bit.
- CKAMUX = the setting of the `SPORT_CTL2_A.CKMUXSEL` configuration bit.
- CKBMUX = the setting of the `SPORT_CTL2_B.CKMUXSEL` configuration bit.

The Routing column in the ***Clock Combinations*** table defines how the signals are used between the SPORT halves and which pin is used for the serial clock (whether it is an input or an output). Within the column, the inequality characters (≤ and ≥) are used to show the direction of the signal, and the following abbreviations are used (x = A or B):

- HSx_CI = Serial clock input signal, provided by an external device or the complementing HSPORT.
- HSx_CO = Serial clock output signal.
- SPx_CLK = HSPORT's serial clock pin, where the signal is either:
  - provided by an external source and distributed to both HSPORT serial clock signals, or
  - internally generated by one HSPORT and routed to both the pin and to the complementary HSPORT serial clock signal.

Table 30-8: Clock Combinations

CLK Combination	HSA_ICLK	HSB_ICLK	CKAMUX	CKBMUX	Routing
9	0	0	0	0	Native CLK Operation
10	0	1	0	0	Native CLK Operation
11	1	0	0	0	Native CLK Operation
12	1	1	0	0	Native CLK Operation

Table 30-8: Clock Combinations (Continued)

CLK Combination	HSA_ICLK	HSB_ICLK	CKAMUX	CKBMUX	Routing
13	0	0	1	0	HSA_CI ≤ SPB_CLK; HSB_CI ≤ SPB_CLK
14	0	1	1	0	HSA_CI ≤ HSB_CO ≥ SPB_CLK
15	0	0	0	1	HSB_CI ≤ SPA_CLK; HSA_CI ≤ SPA_CLK
16	1	0	0	1	HSB_CI ≤ HSA_CO ≥ SPA_CLK

The following is a comprehensive list of the legal combinations for the above described frame sync and clock multiplexing configurations:

- FS Combinations 1–4 are compatible with all CLK Combinations (9–16)
- CLK Combinations 9–12 are compatible with all FS Combinations (1–8)
- FS Combination 5 is only compatible with CLK Combination 13 (and vice versa)
- FS Combination 6 is only compatible with CLK Combination 14 (and vice versa)
- FS Combination 7 is only compatible with CLK Combination 15 (and vice versa)
- FS Combination 8 is only compatible with CLK Combination 16 (and vice versa)

NOTE: Program only the `SPORT_CTL2` register of the HSPORT that is accepting the signal from the other HSPORT. However, be sure to set the `SPORT_CTL_A.CKRE` and `SPORT_CTL_A.LFS` polarity bits to have identical settings between the HSPORTs when making internal connections via the SPMUX block.

Data Types and Companding

The SPORT uses the data type select field `SPORT_CTL_A.DTYPE` bit to specify one of the four data formats supported by serial ports. These formats apply to any of the operating modes of serial port.

Table 30-9: Data Type Bit Field Settings

DTYPE field	SPORT Receiver	SPORT Transmitter
00	Right-justify, zero-fill unused most significant bits	Normal operation
01	Right-justify, sign-extend unused most significant bits	Reserved
10	Expand using µ-law	Compress using µ-law
11	Expand using A-law	Compress using A-law

These formats apply to data words loaded into the SPORT transmit or receive data buffers. The first two data formats (`00` and `01` values of `SPORT_CTL_A.DTYPE`) are applicable only when SPORT is configured as receiver.

When SPORT is configured as transmitter, only the significant bits are transmitted (per the field defined in control register). Therefore, the transmit data buffers are not actually zero-filled or sign-extended.

The other two data formats enable the companding logic on the transmit or receive path. Companding (compressing or expanding) is the process of logarithmically encoding and decoding data to minimize the number of bits sent. The SPORTs of the processor support the two most widely used companding algorithms, A-law, and µ-law. The algorithms are performed according to the CCITT G.711 specification.

If selected, companding applies to both the enabled data channels. When enabled as SPORT transmitter, writes to transmit buffer make the content compressed to 8 bits according to algorithm selected. (The content is zero filled to the width of the transmit word.) Similarly, if configured in receive mode, the 8 bits in the receive data buffers expand in right-justified, zero fill format per the algorithm selected. If companding is enabled in multichannel mode, it applies to all the active channels.

The compression for transmit data requires a minimum word length of 8 for proper function. If `SPORT_CTL_A.SLEN` is less than 7, then expansion does not work correctly. Also, if the data value is greater than 13-bit A-law or 14-bit µ-law maximum, it automatically compresses to the maximum value.

NOTE: The processor companding logic supports in-place companding feature. So, companding can be used for debug without enabling SPORT.

Companding as a Function

Since the values in the transmit and receive buffers are companded in place, the SPORT can use the companding hardware without transmitting or receiving data, which can be useful during testing or debugging. For companding to execute properly, program the SPORT registers prior to loading data values into the SPORT buffers.

To compress data in place without transmitting, use the following procedure:

1. Set the SPORT as a transmitter (`SPORT_CTL_A.SPTRAN` = 1) with both primary and secondary data channels disabled (`SPORT_CTL_A.SPENPRI` = `SPORT_CTL_A.SPENSEC`= 0).
2. Enable companding in the `SPORT_CTL_A.DTYPE` field.
3. Write a 32-bit data word to the transmit buffer.
4. Wait two system clock cycles to allow the SPORT companding hardware to reload the transmit buffer with the companded value. Any instructions that do not access the transmit buffer can be used to cause this delay.
5. Read the 8-bit compressed value from the transmit buffer.

To expand data in place, use the same sequence of operations with the receive buffer instead of the transmit buffer.

Transmit Path

When the `SPORT_CTL_A.SPTRAN` control bit is set, the HSPORT is in transmit mode. Primary and secondary transmit data paths are then enabled using the `SPORT_CTL_A.SPENPRI` and `SPORT_CTL_A.SPENSEC` bits, respectively. The primary and secondary data paths are unique and identical, each including its own transmit data buffer, optional companding logic, and transmit shift register.

The data buffer on the primary transmit data path is SPORT_TXPRI_A, and the data buffer on the secondary transmit data path is SPORT_TXSEC_A. The transmit data buffer and output shift register form a FIFO type of structure. When packing is disabled (SPORT_CTL_A.PACK = 0), the SPORT can hold as many as three data words. If packing is enabled (SPORT_CTL_A.PACK = 1), the serial port can hold two packed data words at any time.

The transmit data for primary and secondary channels is written to the SPORT_TXPRI_A and SPORT_TXSEC_A transmit data buffers, respectively. The transmit data buffers can be accessed in core mode through the peripheral bus or in DMA mode through the DMA bus. When a SPORT is configured in transmit mode, the receive paths are deactivated and do not respond to serial clock or frame sync signals. Because the receive data buffers and receive shift registers are also deactivated, reading from an empty and inactive receive data buffer can cause the core to hang indefinitely.

NOTE: Be sure to avoid accesses to inactive data buffers. Such accesses can cause unpredictable SPORT behavior or a hang condition and are not reported in any status register.

This data can optionally be compressed in hardware according to the selected algorithm (μ-law or A-law) and then automatically transferred to the transmit shift register. The shift register, clocked by the SPORT_ACLK signal, then serially outputs this data on the SPORT_AD0 and/or SPORT_AD1 pins (if both are enabled, these output data bits are transmitted synchronously). If the SPORT uses a framing signal, the SPORT_AFS signal indicates the start of the serial word transmission.

When using DMA mode, a single DMA feeds the data buffers of the enabled channels (primary and/or secondary). When using both channels, interleave the data of these channels starting with the primary channel in the transmit buffer.

When the SPORT is configured in non-multichannel mode as a transmitter, the enabled SPORT data pins (SPORT_AD0 and/or SPORT_AD1) are always driven. When a SPORT channel is enabled, data from the transmit data buffer is loaded into the transmit shift register. The shift register then immediately latches the first bit of data (either the LSB or MSB, depending on the SPORT_CTL_A.LSBF configuration bit) and drives it to the respective data pin such that it is ready when the frame sync signal asserts. Similarly, if the frame sync period exceeds the serial word length, then the data pins are driven with the first bit of the next word for transmission immediately after the active word completes, and the outputs are held during the inactive serial clock cycles (clock cycles between frame sync pulses).

When the SPORT is configured in multichannel mode, the data pins are driven only during active transmit channels and are always three-stated during inactive channel slots.

The SPORT provides status of transmit data buffers and also error detection logic for transmit errors such as an underrun condition. See the Error Detection (Status) Interrupt section for more details.

Receive Path

When the SPORT_CTL_A.SPTRAN bit is cleared, the SPORT is in receive mode. Primary and/or secondary receive data paths can be enabled by setting the SPORT_CTL_A.SPENPRI and SPORT_CTL_A.SPENSEC

configuration bits, respectively. These data paths are unique but identical, each with a receive shift register, optional companding logic, and a receive data buffer.

The data buffer on the primary receive path is SPORT_RXPRI_A, and the data buffer on the secondary receive path is SPORT_RXSEC_A. The receive data paths act like a three-deep (32-bit words) FIFO because they have two data registers plus an input shift register.

Upon enabling the SPORT data channels, the input shift register shifts in data bits on the SPORT_AD0 and/or SPORT_AD1 pins, synchronous to the SPORT clock signal. If the SPORT uses a framing signal, the SPORT_AFS signal indicates the beginning of the serial word (or frame) to be received. When an entire word is shifted into the primary and secondary channels, the data can be optionally expanded in hardware according to a selected algorithm (µ-law or A-law) and then automatically transferred to the SPORT_RXPRI_A and/or SPORT_RXSEC_A data buffers.

The receive data buffers can be read in core mode through the peripheral bus or in DMA mode through the DMA bus. When the SPORT uses DMA mode, a single DMA reads the data buffers of the enabled channels (primary and/or secondary) and interleaves them in memory beginning with the primary channel when both channels are enabled. When using both channels, software must de-interleave the data of these channels.

The SPORT provides the status of receive data buffers and also error detection logic for receive errors such as overflow. See the Error Detection (Status) Interrupt section for more details.

When a SPORT is configured in receive mode, the transmit paths are deactivated and do not respond to serial clock or frame sync signals. As the transmit data buffers and transmit shift registers in the data paths are also deactivated, programs must not try to access them.

NOTE: Be sure to avoid accesses to inactive data buffers. Such accesses can cause unpredictable SPORT behavior or a hang condition and are not reported in any status register.

Operating Modes and Options

The SPORT has a number of operating modes:

- Standard DSP Serial mode
- I^2S mode
- Left-Justified mode
- Right-Justified mode
- Multichannel (TDM) mode
- Packed I^2S mode

The SPORT halves within a SPORT top module can be independently configured in any of these operating modes unless they are coupled together using SPMUX logic, in which case they must be configured identically. Each half SPORT has its own set of control and data registers and is programmed similarly.

The ***Control Bits for SPORT Operating Modes*** table lists all the programmable configuration bits in the SPORT_CTL_A control register, which combine to determine the overall function and operating mode of the SPORT. The columns are arranged according to the setting of the SPORT_CTL_A.OPMODE bit that selects between standard DSP/multichannel modes and the various I²S modes, and the cell contents are defined as follows:

- Yes – bit is programmable for this mode of operation and may be written
- Reserved – bit is not programmable for this mode of operation and must not be written
- = value – bit must be set to this value to enable this mode of operation
- FUNCTION – indicates alternate function for this bit in this mode

NOTE: When changing operating modes, first clear the SPORT_CTL_A register before again writing the register with the new configuration settings.

Table 30-10: Control Bits for SPORT Operating Modes

Name (Bit #)	Standard DSP Serial Mode	I²S and Left-Justified Mode	Right-Justified Mode	Multichannel (TDM) Mode	Packed I²S Mode
SPENPRI (0)	Valid				
DTYPE (2:1)	Valid	Reserved		Valid	
LSBF (3)	Valid	Reserved		Valid	
SLEN (8:4)	Valid				
PACK (9)	Valid				
ICLK (10)	Valid				
OPMODE (11)	= 0	= 1	= 1	= 0	= 1
CKRE (12)	Valid				
FSR (13)	Valid	Reserved			
IFS (14)	Valid				
DIFS (15)	Valid			Reserved	
LFS (16)	Valid	L_FIRST/PLFS		Valid	L_FIRST/PLFS
LAFS (17)	Valid	OPMODE2	Valid	Reserved	
RJUST (18)	Reserved		= 1	Reserved	
FSED (19)	Valid	Reserved		Valid	Reserved
TFIEN (20)	Valid				
GCLKEN (21)	Valid		Reserved		
SPENSEC (24)	Valid				
SPTRAN (25)	Valid				

Serial Word Length

The SPORT uses the SPORT_CTL_A.SLEN field to determine the word length of the serial data to transmit or receive. Each half SPORT can independently handle word lengths up to 32 bits, and the value that must be programmed to the SPORT_CTL_A.SLEN field is obtained from:

SLEN = Desired SPORT word length - 1

The minimal word length depends on the selected operating mode. Words smaller than 32 bits are right justified in the transmit or receive buffers; however, data can be shifted in or out in MSB or LSB first format, as configured by the SPORT_CTL_A.LSBF bit. The received word can also be sign-extended or zero-filled when storing the data to processor memory, as governed by the SPORT_CTL_A.DTYPE bit.

The ***Data Lengths for SPORT Operating Modes*** table shows the range of valid word lengths for each of the supported SPORT operating modes.

Table 30-11: Data Lengths for SPORT Operating Modes

Mode	SPORT Word Length (SLEN+1)
Standard DSP Serial	4–32
I^2S	5–32
Left-Justified	5–32
Right-Justified	5–32
Multichannel (TDM)	5–32
Packed I^2S	5–32

NOTE: When the companding feature is enabled on the data path, it limits the word length settings. See Data Types and Companding for more details about word lengths required for companding. When more than 32 bits per frame sync are required to transmit or receive, use the multichannel mode to spread the data across numerous continuous channels.

Clock Sample and Drive Edges

The SPORT uses two control signals to sample or drive the serial data:

1. Serial clock (SPORT_ACLK) - bit clock for the serial data.
2. Frame sync (SPORT_AFS) - divides the incoming data stream into frames.

These control signals can be internally generated or externally provided, as determined by the SPORT_CTL_A.ICLK and SPORT_CTL_A.IFS bit settings, respectively.

Data and frame syncs can be sampled on the rising or falling edges of the SPORT clock signal, as determined by the SPORT_CTL_A.CKRE bit. By default, the SPORT_CTL_A.CKRE = 0 setting configures the falling edge of the SPORT_ACLK signal as the sampling edge for receive data and externally supplied frame syncs. The receive data and frame syncs can be sampled on the rising edges of SPORT_ACLK when SPORT_CTL_A.CKRE = 1.

NOTE: The SPORT drives transmit data and internal frame sync signals on the opposite serial clock edge of the sampling edge. Be sure to select the same value for SPORT_CTL_A.CKRE for transmit and receive functions for any two HSPORTs that are connected, and always verify the correct polarity for any external device connected to the SPORT.

The ***Frame Sync and Data Driven on Rising Edge*** figure provides an example of the drive and sample edges when two HSPORTs are connected , each with SPORT_CTL_A.CKRE = 0. In this example, the HSPORT that is configured as the transmitter drives the serial clock and frame sync signals, and both HSPORTs are configured for early, active high frame syncs and a word length of eight bits.

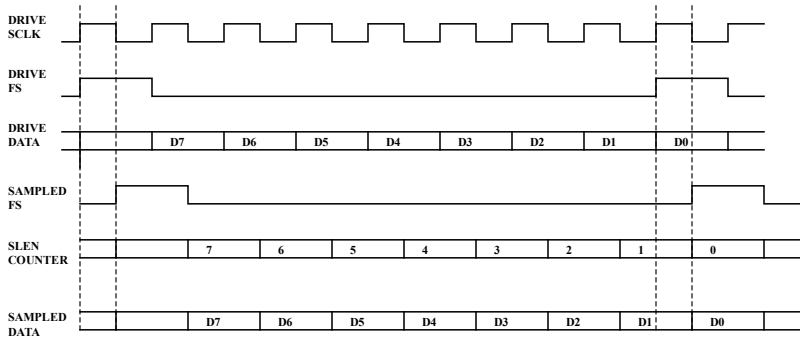


Figure 30-3: Frame Sync and Data Driven on Rising Edge

NOTE: The SCLK in the ***Frame Sync and Data Driven on Rising Edge*** figure is SCLK.

As shown, the transmitting HSPORT provides the clock and generates the frame sync. Because the HSPORTs are configured for early frame mode, the first bit of data is driven one serial clock later, with subsequent bits being driven on the following rising clock edges in the signal train. When the receiving HSPORT samples the frame sync signal (as indicated in the SAMPLED FS waveform), the SPORT_CTL_A.SLEN bit counter is loaded with the SPORT_CTL_A.SLEN setting, after which each SPORT_ACLK decrements the SPORT_CTL_A.SLEN counter until the full word is received. In this figure, the DRIVE FS and SAMPLED FS waveforms show the frame sync required for the next word in a continuous data stream. Note that it is legal for this frame sync to be sampled synchronous to the last bit of the previous data being sampled, as the early frame mode means that the data lags the frame sync by one serial clock cycle. If the frame sync were sampled as asserted before the D0 bit is sampled, the frame sync error is logged in the receiver's status register.

Since the transmitter drives the internally-generated frame sync and data on the rising edge of the serial clock, the receiver must use the falling edge to sample the externally-supplied frame sync and data.

Frame Sync Options

The following sections provide details regarding the programmable aspects of the SPORT frame sync signal. See the specific operating mode sections for more information regarding frame sync requirements and behavior for each specific operating mode.

Data-Dependent versus Data-Independent Frame Syncs

By default, the generation of a frame sync signal is data-dependent:

- When the SPORT is configured as a transmitter (SPORT_CTL_A.SPTRAN = 1), an internally generated transmit frame sync is output when a new data word has been loaded into the channel transmit buffer of the SPORT (by either the core or the DMA engine).
- When the SPORT is configured as a receiver (SPORT_CTL_A.SPTRAN = 0), an internally-generated receive frame sync is output only when the receive data buffer is not full.

The data-independent frame sync option, enabled by setting the SPORT_CTL_A.DIFS bit, allows for the generation of a periodic framing signal, regardless of the status of the data buffers. When this bit is set, the frame sync output will be continuous and periodic, according to the setting of the SPORT_DIV_A.FSDIV field.

Support for Edge-Detected and Level-Sensitive Frame Syncs

The level-sensitive nature of frame sync signals operates well in a noise-free environment. However, if noise corrupts the signals coming into the SPORT, the internal logic can lose synchronization. For example, excessive noise on the frame sync signal may cause the frame sync to be sampled as inactive on the clock edge that it is intended to be synchronous to, but then be sampled at the correct active level one cycle later. Similarly, a noisy clock signal can cause an unintended clock edge, resulting in potential premature sampling of the frame sync signal being applied to the pin.

The ***Level-Sensitive Frame Sync versus Edge Sensitive Frame Sync*** figure describes a scenario where an external frame sync signal is corrupted due to noise, causing the receiving SPORT module to incorrectly sample the signal. If the frame sync is driven on the rising edge of the serial clock at t_A, the SPORT would normally sample the signal on the falling edge of the serial clock at t_B. Due to the noise, however, the SPORT misses the first edge of the frame sync and instead samples it at t_C.

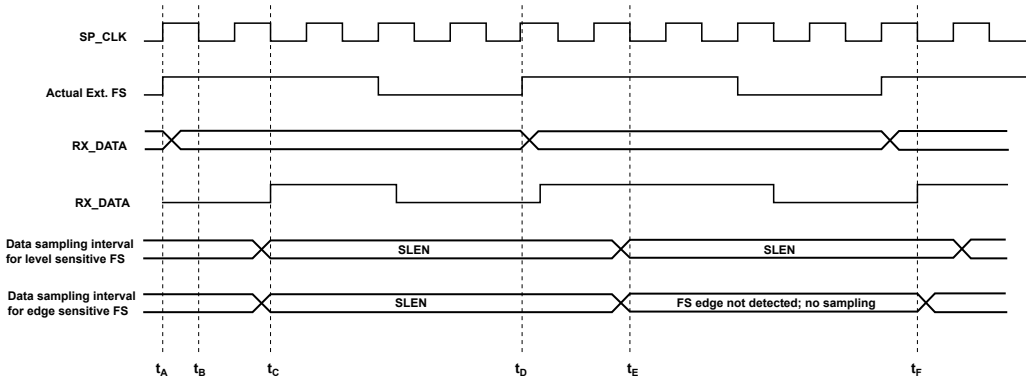


Figure 30-4: Level-Sensitive Frame Sync versus Edge Sensitive Frame Sync

When the above occurs, the internal word length counter runs for a period equal to the SPORT_CTL_A.SLEN field of the control register, but it erroneously expires at t_E rather than at the appropriate point at t_D, thus receiving incorrect data. Further, if a new level-sensitive frame sync edge arrives at time t_D, the SPORT samples this framing signal again at t_E. As such, the frame sync sampling continues to be misaligned with the external data.

To help address this, the SPORT module provides an option to configure the frame sync signal to instead be edge-sensitive via the SPORT_CTL_A.FSED configuration bit. When this bit is set with active high frame syncs

enabled (SPORT_CTL_A.LFS = 0), the rising edge of the frame sync is valid. Conversely, when the frame sync is active low (SPORT_CTL_A.LFS= 1), the falling edge is defined to be valid.

NOTE: SPORT_CTL_A.FSED is valid only in external frame sync mode. In internal frame sync mode, the setting of this bit is irrelevant and ignored.

In the above example, an edge-sensitive frame sync signal is not detected at t_E because the edge of the framing signal already occurred in the previous cycle (t_D) and there is no new edge to detect at t_E. As a result, the internal word length counter remains idle for this frame, thus ignoring the incorrect data, and the counter correctly resumes operation at t_F when a new frame sync edge is detected.

This activity sets the SPORT_ERR_A.FSERRSTAT bit and optionally generates a premature frame sync error interrupt.

Frame sync edge detection is used by default for stereo modes. MCM mode and DSP serial mode choose between edge detection and normal mode of FS detection.

NOTE: When the SPORT is first enabled, an already active externally applied frame sync will not commence operation. The SPORT will wait for a valid change in the frame sync's state from inactive to active before operation begins.

Early versus Late Frame Syncs

Frame sync signals can occur in the same serial clock cycle as the first bit of the data word (late) or one serial clock cycle before the first bit (early), as controlled by the SPORT_CTL_A.LAFS bit.

By default, the frame sync signal is configured to be early (SPORT_CTL_A.LAFS = 0). The first bit of the transmit data word will be driven one serial clock cycle after the frame sync is asserted (whether sensed externally or internally provided), and the first bit of the receive data word is expected to lag the frame sync by one serial clock cycle. The frame sync is not checked again until the entire word has been transferred.

If data transmission is continuous in early framing mode, then an internally-generated frame sync signal will be asserted (pulsed active for one serial clock cycle) synchronous to the last data bit of the current transfer, as the first bit of the next transfer will be immediately driven in the next serial clock cycle (no clocks are wasted). This event is not a premature frame sync error, so the SPORT_ERR_A.FSERRSTAT bit is not set.

The frame sync can alternatively be configured as late (SPORT_CTL_A.LAFS = 1), in which case the first bit of the transmit data word is available in the same serial clock cycle that the frame sync is asserted (whether sensed externally or internally provided), and the first bit of the receive data word is also latched in the same cycle. Serial clock edges latch the receive data bits, but the frame sync signal is checked only during the first bit of each word. Internally generated frame syncs remain asserted for the entire length of the data word in late framing mode.

The ***Normal Framing (Early Frame Sync) Versus Alternate Framing (Late Frame Sync)*** figure illustrates these concepts.

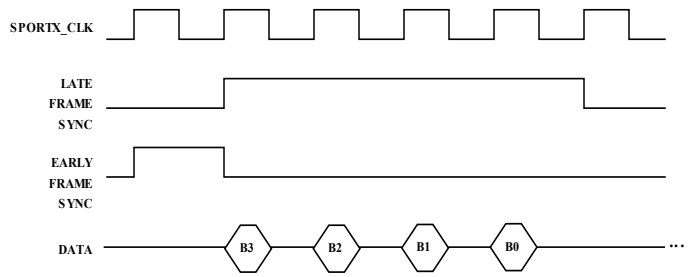


Figure 30-5: Normal Framing (Early Frame Sync) Versus Alternate Framing (Late Frame Sync)

Framed versus Unframed Frame Syncs

The use of a frame sync signal is optional for SPORT operation, as controlled by the `SPORT_CTL_A.FSR` bit. When the frame sync is configured to be required (`SPORT_CTL_A.FSR` = 1), the data is defined to be framed (a frame sync signal must accompany every data word). To allow continuous transmission from the processor, ensure that a new data word is loaded into the transmit buffer before the ongoing transfer is completed (this is automatically cared for when DMA is used to transmit blocks of data).

Data words can be transferred continuously in what is referred to as unframed data mode, which is appropriate for continuous reception, by setting `SPORT_CTL_A.FSR` = 0. In this configuration, a single frame sync is still required to initiate communication, but it is subsequently not required once the communication begins. From that point onward, externally provided frame syncs are ignored and internally generated frame syncs are not driven. The ***Framed versus Unframed Data Stream*** figure shows the differences in SPORT operation between framed and unframed data modes with the frame sync configured to be early (`SPORT_CTL_A.LAFS` = 0).

NOTE: When DMA is enabled in a mode where frame syncs are not required, chaining can delay DMA requests. DMA requests are not always serviced frequently enough to guarantee continuous unframed data flow. Monitor status bits or check for a SPORT error interrupt to detect underflow or overflow of data.

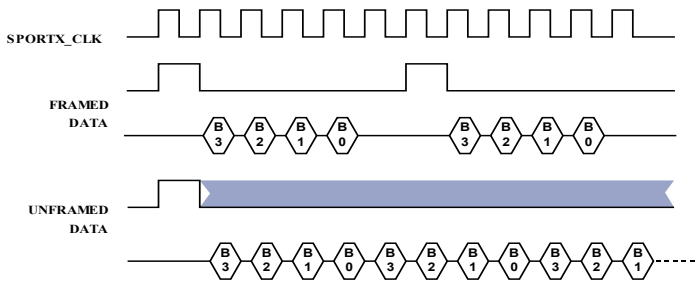


Figure 30-6: Framed versus Unframed Data Stream

Frame Sync Polarity

The framing signals can be active high or active low, as governed by the `SPORT_CTL_A.LFS` bit:

- When `SPORT_CTL_A.LFS` = 0, the corresponding frame sync signal is active high.
- When `SPORT_CTL_A.LFS` = 1, the corresponding frame sync signal is active low.

Active high is the default polarity of the frame sync signal.

Premature Frame Sync Error Detection

A SPORT framing signal is used to synchronize transmit or receive data. In external frame sync mode, any frame sync received during an active frame is premature and invalid. When this occurs, the SPORT_ERR_A.FSERRSTAT bit is set to indicate the framing error, and an optional error interrupt request can be generated for this event by setting the SPORT_ERR_A.FSERRMSK bit.

NOTE: The SPORT_ERR_A.FSERRSTAT bit is not set in the presence of uncleared underflow or overflow errors.

Refer to the ***Frame Sync Error Detection*** figure. The frame sync error bit gets set when an unexpected frame sync occurs during the ongoing data transfer (transmission or reception).

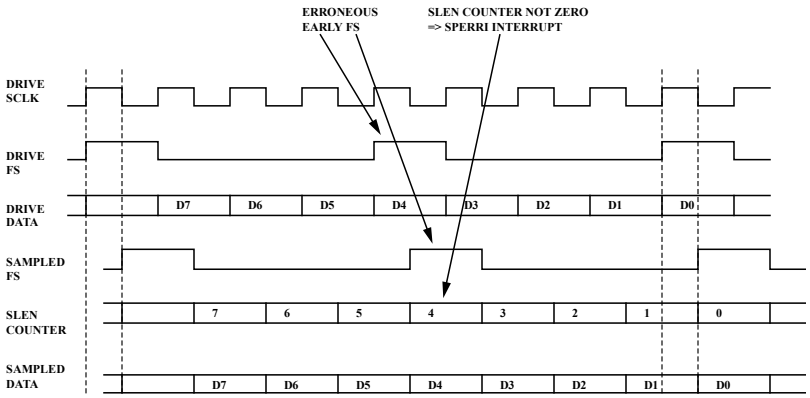


Figure 30-7: Frame Sync Error Detection

NOTE: SCLK in the ***Frame Sync Error Detection*** figure is SCLK.

Whether a SPORT is receiving or transmitting, its bit count is set to the programmed serial word length when the frame sync is sampled, which is then decremented every subsequent serial clock cycle until the transfer has completed. At this point, the bit count reaches zero and will be reset to the programmed serial length when the next frame sync is sampled. As such, the bit count value is non-zero during an active transfer, and the frame sync error is asserted if a frame sync is sampled when this count is non-zero.

Mode Selection

The SPORT's operating mode is configured in the SPORT_CTL_A and SPORT_MCTL_A registers. The ***SPORT Operating Modes*** table provides specific guidance to properly program these SPORT control registers for the desired mode of operation.

Table 30-12: SPORT Operating Modes

Operating Modes	SPORT_CTL_A.OPMODE	SPORT_CTL_A.LAFS	SPORT_CTL_A.RJUST	SPORT_MCTL_A.MCE
Standard DSP Serial	0	Programmable	Reserved	0
I^2S	1	0	Reserved	0

Table 30-12: SPORT Operating Modes (Continued)

Operating Modes	SPORT_CTL_A.OPMODE	SPORT_CTL_A.LAFS	SPORT_CTL_A.RJUST	SPORT_MCTL_A.MCE
Left-Justified	1	1	Reserved	0
Right-Justified	1	1	1	0
Multichannel	0	Reserved	Reserved	1
Packed I^2S mode	1	Reserved	Reserved	1

The following sections provide detailed information for each of the supported SPORT modes of operation.

Standard DSP Serial Mode

The SPORT can be configured in standard DSP serial mode by clearing the SPORT_CTL_A.OPMODE and SPORT_MCTL_A.MCE bits. This mode provides great flexibility in terms of programmable options to configure the SPORTs to communicate with various serial devices such as serial data converters and audio codecs. In order to properly connect to such devices, various clocking, framing, and data formatting options are available.

Timing Control Bits

Several bits in the SPORT_CTL_A control register define the configuration of the SPORT in standard DSP serial mode:

- SLEN: serial word length (4–32 bits)
- LSBF: shift LSB or MSB first
- ICLK: internally generated or externally provided serial clock
- CKRE: sample on rising or falling edge of the serial clock
- IFS: internally generated or externally provided frame sync
- FSR: framed or continuous operation
- DIFS: data-dependent or data-independent frame sync
- LFS: active high or active low frame sync
- LAFS: frame sync synchronous to data or one clock cycle before it
- PACK: 16-bit to 32-bit packing option
- GCLKEN: free-running or gated clock

Clocking Options

In standard DSP serial mode, the SPORTs can either accept an external serial clock or generate one internally, as controlled by the SPORT_CTL_A.ICLK bit. For internally generated serial clocks (SPORT_CTL_A.ICLK = 1), the SPORT_DIV_A.CLKDIV field configures the serial clock rate from the system clock.

The SPORT clock can also be gated, where it is only valid during an active transfer, as controlled by the SPORT_CTL_A.GCLKEN bit.

The SPORT clock edge used for driving and sampling of serial data and frame syncs is configured using the SPORT_CTL_A.CKRE bit:

- If SPORT_CTL_A.CKRE = 0, input data and frame sync signals are sampled on the falling edge of the serial clock, and output data and frame sync signals are driven on the rising edge.
- If SPORT_CTL_A.CKRE = 1, input data and frame sync signals are sampled on the rising edge of the serial clock, and output data and frame sync signals are driven on the falling edge.

Stereo Modes

The SPORTs support three widely used stereo modes of operation:

- I^2S mode
- Left-Justified mode
- Right-Justified mode

In these modes, the serial data stream consists of left and right channels. The following sections describe these modes in more detail.

Channel Order

The active low frame sync (SPORT_CTL_A.LFS) bit is used to determine the polarity of the frame sync signal in the non-stereo modes of operation. For the stereo modes of operation, it instead controls whether the right or left channel is first in the data transfer. The ***Channel Order Bit Settings*** table shows which word is transmitted or received first, based on the setting of the SPORT_CTL_A.LFS bit.

Table 30-13: Channel Order Bit Settings

Mode	**SPORT_CTL_A.LFS=0**	**SPORT_CTL_A.LFS=1**
Left-Justified or Right-Justified	Left channel first	Right channel first
I^2S or Packed I^2S	Right channel first	Left channel first

I^2S Mode

I^2S mode is a commonly used stereo mode, where left and right channel data words are interleaved in the serial data stream and each transition of the frame sync signal is associated with one of the channels. The left channel data is transferred during the low segment of the frame sync signal, and the right channel data is transferred during the high segment of the frame sync signal. As such, the frame sync signal is treated as a left-right (L/R) clock in this mode.

To set up the SPORT in I^2S mode, the following configuration is required:

- SPORT_CTL_A.OPMODE = 1

- SPORT_CTL_A.LFS = 0
- SPORT_MCTL_A.MCE = 0

Protocol Configuration Options

Several bits in the SPORT_CTL_A control register must be configured to be compliant with the I²S standard, but they can be otherwise configured to support non-standard operations

- SLEN–programmable (allowable word lengths are 5–32 bits)
- LSBF–set to 0 (MSB first)
- ICLK–programmable (serial bit clock can be internally generated or externally provided)
- IFS–programmable (serial L/R clock source must match serial bit clock source)
- LFS–set to 1 (left channel first)
- CKRE–set to 1 (sample L/R clock and data on rising edge of bit clock)

Serial Bit Clock and L/R Clock Rates

When the SPORT is configured to generate the bit clock and the L/R clock (SPORT_CTL_A.ICLK = SPORT_CTL_A.IFS = 1), set the serial bit clock rate using the SPORT_DIV_A.CLKDIV bit field and the L/R clock rate using the SPORT_DIV_A.FSDIV bit field.

The ***Word Select Timing in I²S Mode*** figure shows the SPORT timing in I²S mode. The data lags the L/R clock transition by one SCLK cycle, and the transfer begins with the left channel data word first.

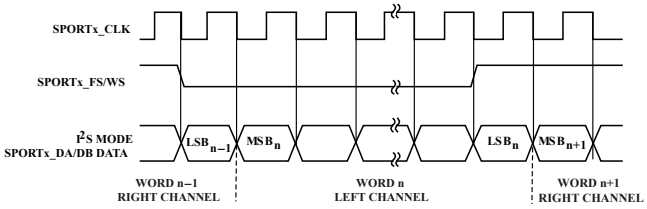


Figure 30-8: Word Select Timing in I²S Mode

Left-Justified Mode

Left-justified mode is a stereo mode subset of the I²S standard. As in I²S mode, the frame sync signal acts as a left-right clock (L/R clock), where left and right data samples are transferred each L/R clock period. The left channel is associated with the high segment of the frame sync, and the right channel aligns with the low segment of the frame sync. The difference between left-justified mode and standard I²S mode is that the channel data is driven in the same bit clock cycle as the L/R clock transition (rather than one bit clock cycle later), such that the MSB is synchronous with the leading edge of the frame sync transition.

To set the SPORT up in left-justified mode, the following configuration is required:

- SPORT_CTL_A.OPMODE = 1

- SPORT_CTL_A.LAFS = 1
- SPORT_MCTL_A.MCE = 0

Protocol Configuration Options

Several bits in the SPORT_CTL_A control register must be configured to operate the SPORT in left-justified mode, but they can be otherwise configured as well:

- SLEN: programmable (allowable word lengths are 5–32 bits)
- LSBF: set to 0 (MSB first)
- ICLK: programmable (serial bit clock can be internally generated or externally provided)
- IFS: programmable (serial L/R clock source must match serial bit clock source)
- LFS: set to 0 (left channel first)
- CKRE: set to 1 (sample L/R clock and data on rising edge of bit clock)

Serial Bit Clock and L/R Clock Rates

If the SPORT is configured to generate the bit clock and the L/R clock (SPORT_CTL_A.ICLK = SPORT_CTL_A.IFS = 1), set the serial bit clock rate using the SPORT_DIV_A.CLKDIV bit field and the L/R clock rate using the SPORT_DIV_A.FSDIV bit field.

The ***Word Select Timing in Left-Justified Mode*** figure shows the SPORT timing in left-justified mode. The start of a data sample is synchronous to the L/R clock transition, and the transfer begins with the left channel data word first.

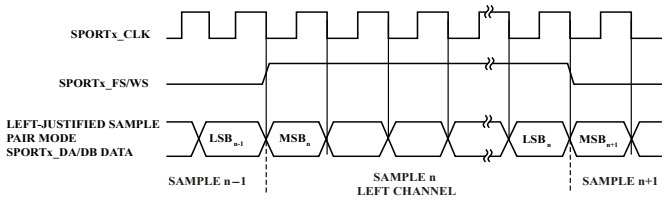


Figure 30-9: Word Select Timing in Left-Justified Mode

Right-Justified Mode

Right-justified mode is a stereo mode subset of the I^{2}S standard. As in I^{2}S mode and left-justified mode, the frame sync signal acts as a left-right clock (L/R clock), where left and right data samples are transferred each L/R clock period. The left channel is associated with the high segment of the frame sync, and the right channel aligns with the low segment of the frame sync. The difference between right-justified mode and standard I^{2}S mode is that the LSB of the channel data ends at the point that the L/R clock transitions to frame the next sample (rather than one bit clock cycle after the L/R clock transition).

To set the SPORT up in right-justified mode, the following configuration is required:

- SPORT_CTL_A.OPMODE = 1

- SPORT_CTL_A.RJUST = 1
- SPORT_MCTL_A.MCE = 0

Timing Control Bits

Several bits in the SPORT_CTL_A control register must be configured to operate the SPORT in right-justified mode, but they can be otherwise configured as well:

- SLEN: programmable (allowable word lengths are 5–32 bits)
- LSBF: set to 0 (MSB first)
- ICLK: programmable (serial bit clock can be internally generated or externally provided)
- IFS: programmable (serial L/R clock source must match serial bit clock source)
- LFS: set to 0 (left channel first)
- CKRE: set to 1 (sample L/R clock and data on rising edge of bit clock)

Serial Bit Clock and L/R Clock Rates

If the SPORT is configured to generate the bit clock and the L/R clock (SPORT_CTL_A.ICLK = SPORT_CTL_A.IFS = 1), set the serial bit clock rate using the SPORT_DIV_A.CLKDIV bit field and the L/R clock rate using the SPORT_DIV_A.FSDIV bit field.

The ***Word Select Timing in Right-Justified Mode*** figure shows the SPORT timing in right-justified mode. The transmitter aligns the transmit data such that the last bit of the serial word is sent in the last clock cycle of the L/R clock (frame sync) signal marking the channels.

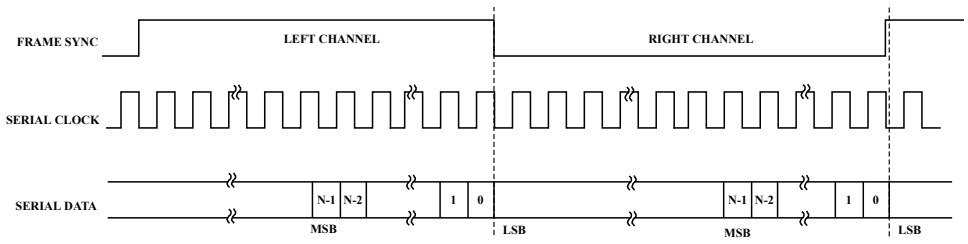


Figure 30-10: Word Select Timing in Right-Justified Mode

NOTE: For some SPORT-compatible ADCs or DACs such as the AD1871, right-justified mode is limited to commonly used ratios such as 64 FS and 128 FS. FS is the sampling frequency of ADCs and DACs, referred to as the SPORT's L/R clock (frame sync) signal.

Consider the SPORT timing for right-justified mode, as shown in the ***Timing Comparison Between Different Stereo Modes*** figure. The frame sync width is limited to 32 SPORT clock periods (or 32 bits per channel) if:

- the SPORT's frame sync (L/R clock) runs at the FS rate, and
- the SPORT's serial bit clock runs at the 64 FS rate

The limitation applies to the frame sync width of either channel. If the data is confined to 24 bits, the SPORT introduces a 32–24=8-bit clock delay before it starts to transmit or capture data.

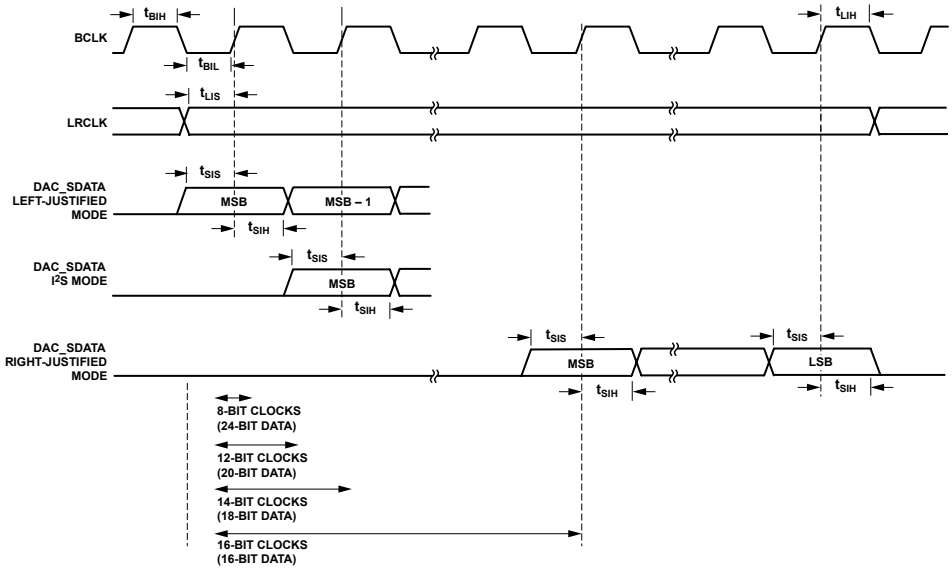


Figure 30-11: Timing Comparison Between Different Stereo Modes

Similarly, to support the 128 FS bit clock frequency, the frame sync width becomes 64 serial bit clock periods per channel. The delay can be a maximum of 59 bit clocks (64 – 5, which is the minimum serial data length in right-justified mode).

The starting point of the first bit is delayed so that the LSB of the serial data aligns properly with the end of the channel. A 6-bit counter is added for this purpose in the stereo mode counter, which is programmed by writing the least significant six bits of the `SPORT_MCTL_A.WOFFSET` field. Though this is a multichannel mode configuration register, the SPORT uses these bits in right-justified mode to configure the offset from the transition of the L/R clock to where the first data bit must be driven to have the end of the last bit align properly with the next L/R clock transition. The software must program this register with the appropriate delay.

Multichannel (TDM) Mode

The multichannel mode of SPORT operation allows the SPORT to communicate as part of a time division multiplexed (TDM) serial system. In TDM communications, a large frame of streamed serial data words is defined to be a particular length. It consists of a specific number of channels, and each channel contains one serial data word of the defined data length. For example, a 24-word block of 24-bit data can be defined to be a window within a frame, having a duration of 576 bit clocks and comprised of 24 continuous channels. The SPORT is configured to transfer on specific channels within a defined window in this frame.

To set the SPORT up in multichannel mode, the following configuration is required:

- `SPORT_CTL_A.OPMODE` = 0
- `SPORT_MCTL_A.MCE` = 1

In multichannel mode, the SPORT can selectively transfer data on up to a maximum window size of 128 continuous channels out of a maximum 1024-channel frame while ignoring all the disabled channels within the window and all the channels outside the window. The SPORT can do any of the following on each channel:

- Transmit data (SPORT_CTL_A.SPTRAN = 1)
- Receive data (SPORT_CTL_A.SPTRAN = 0)
- Do nothing (during inactive channels)

Channel selection is configured in the half SPORT multichannel select registers (SPORT_CS0_A - SPORT_CS3_A) before enabling SPORT operation for multichannel mode. Programming of these registers is especially important in DMA data unpacked mode, since the SPORT data buffers begin operation immediately after the SPORT data lines are enabled. Be sure to enable multichannel operation (set the SPORT_MCTL_A.MCE bit) prior to enabling the SPORT itself.

Clocking Options

In multichannel mode, the SPORTs can either accept an external serial clock or generate one internally, as governed by the SPORT_CTL_A.ICLK bit. For an internally-generated serial clock (SPORT_CTL_A.ICLK = 1), the SPORT_DIV_A.CLKDIV bit field is used to configure the serial clock rate, as derived from the system clock.

The serial clock edges used to drive and sample data and frame syncs are also configurable using the SPORT_CTL_A.CKRE bit:

- If SPORT_CTL_A.CKRE= 0, input data and frame sync signals are sampled on the falling edge of the serial clock, and output data and frame sync signals are driven on the rising edge.
- If SPORT_CTL_A.CKRE = 1, input data and frame sync signals are sampled on the rising edge of the serial clock, and output data and frame sync signals are driven on the falling edge.

Frame Sync Options

The frame sync signal synchronizes the channels and restarts each multichannel sequence, starting with the channel 0 data word. For internally-generated frame syncs (SPORT_CTL_A.IFS = 1), the frame sync period in multichannel mode is defined as:

FS period = [(SPORT_CTL_A.SLEN + 1) × number of channels] - 1

The active level for the frame sync signal is also configurable by programming the SPORT_CTL_A.LFS bit. Set this bit to make the frame sync an active low signal, and clear it to make it active high.

In multichannel mode, frame sync timing resembles late framing mode (although the SPORT_CTL_A.LAFS bit is reserved in this mode). The first bit of the transmit data word is driven and the first bit of the receive data word is sampled in the same serial clock cycle as the frame sync, provided there is no programmed frame delay (SPORT_MCTL_A.MFD = 0).

Once the frame sync signal is asserted, word transfers are performed continuously for the duration of the active window, and no further frame syncs are required for different channels within the window. As such, internally-generated frame syncs are always data-independent, and the SPORT_CTL_A.DIFS bit is reserved.

Transmit Data Valid (TDV)

Each SPORT features a transmit data valid signal (SPORT_ATDV), which is driven high during enabled transmit channels. Because the SPORT output data signals are three-stated during inactive channels, the SPORT_ATDV signal signifies when the processor is actively driving the SPORT data outputs, thus serving as an output-enable signal for the data transmit pin(s).

Active Channel Selection Registers

In multichannel mode, the SPORT supports a window size of up to 128 channels for transmitting or receiving data, where it can selectively receive or transmit data in any of these 128 channels. Each channel can be individually enabled or disabled using the multichannel selection registers (SPORT_CS0_A to SPORT_CS3_A) to select the channels in which to transfer data during a multichannel communication stream. Data words associated with enabled channels are transmitted or received in the respective channels, while disabled channels cause a transmit SPORT to three-state the data output pins and a receive SPORT to ignore the data.

The four 32-bit multichannel selection registers combine to form up to a 128-bit meta-register to accommodate the maximum window size of 128 channels. Setting any bit within these registers enables the associated channel. The 128 channels are sequentially numbered from bit 0 in the SPORT_CS0_A register (corresponding to channel 0 of the window) to bit 31 of the SPORT_CS3_A register (corresponding to channel 127 of the window). For example, setting bit 13 of the SPORT_CS1_A register enables channel number 45 (add 32 for the channels in the SPORT_CS0_A). Likewise, setting bit 5 of the SPORT_CS3_A register enables channel number 101 (add 96 for the 32 channels in each of the SPORT_CS0_A, SPORT_CS1_A, and SPORT_CS2_A registers).

Multichannel Frame Delay (MFD)

The multichannel frame delay (SPORT_MCTL_A.MFD) field specifies the delay in serial bit clocks between the frame sync pulse and the first data bit in the frame. This configurability allows the processor to work with different types of telephony interface devices.

As SPORT_MCTL_A.MFD is a 4-bit field, the maximum value allowed for the frame delay is 15 serial clock cycles. When set to 0, the frame sync is concurrent with the first data bit. If SPORT_MCTL_A.MFD>0, a new frame sync can occur during the last channel(s) of a previous frame and still be valid (does not cause a frame sync error).

NOTE: When the required frame delay exceeds 15 serial clocks, use the window offset field (SPORT_MCTL_A.WOFFSET) to delay the start of channel 0 in increments of the serial word length, and then adjust SPORT_MCTL_A.MFD accordingly. For example, when the serial word length is 12 bits and the desired frame delay is 16 serial clock cycles, set the SPORT_MCTL_A.WOFFSET field to 1 to insert a 12-bit delay after the frame sync to where the channel 0 data begins, and then program SPORT_MCTL_A.MFD to four (that is, 16 - 12).

NOTE: When the ASRC module is used with SPORTs, note the following:

- In ASRC TDM mode, the delay between the frame sync (FS) and the data is one clock cycle.
- For the SPORT MFD feature with respect to ASRC TDM mode, select the MFD (SPORT_MCTL_A.MFD / SPORT_MCTL_B.MFD) bit as follows:
  - The ASRC input is expecting one clock cycle between the FS and the data. Configure the MFD transmit bit to 1 (the SPORT which transmits the data to ASRC serial in)
  - The ASRC output has one clock cycle delay between the FS and the data. Configure the MFD receive bit to 1 (for the SPORT which receives the ASRC serial out data).

Window Size (WSIZE)

Select the number of channels used in multichannel operation by programming the 7-bit SPORT_MCTL_A.WSIZE field. This field must be set to the actual number of channels minus one (SPORT_MCTL_A.WSIZE = Number of channels -1).

The 10-bit SPORT_MSTAT_A.CURCHAN field holds the channel number currently being serviced during multi-channel operation.

Window Offset (WOFFSET)

The window offset (SPORT_MCTL_A.WOFFSET) field specifies where in the 1024-channel frame to place the start of the active window (up to 128 channels long). A value of 0 specifies no channel offset from the frame sync (channel 0 immediately follows it). Any non-zero value indicates the number of channels that come between the frame sync and the start of channel 0 of the active frame, with 896 (for example, 1024–128) being the largest value that permits using all 128 channels.

As an example, a program could define an active window comprised of eight channels (SPORT_MCTL_A.WSIZE = 7) with a window offset of 93 (SPORT_MCTL_A.WOFFSET = 93). If configured in this fashion, the 8-channel window that the SPORT will transfer within resides in the channel range from 93 to 100 in the up-to-1024-channel frame.

Do not change the window offset or the number of multichannel slots (SPORT_MCTL_A.WSIZE) while the SPORT is enabled. If the combination of the window size and offset place any portion of the window out-of-range relative to the channel counter, none of the channels are enabled.

Companding Selection

Like the other operating modes, companding logic can optionally be applied to serial data (compression logic for transmit mode or expansion logic for receive mode). The two widely used companding algorithms, A-law and μ-law, are selectable using the SPORT_CTL_A.DTYPE field.

If companding is enabled, the companding algorithm is applied to both the primary and secondary data paths. In multichannel mode, companding can be applied to either all or none of the enabled channels (companding cannot be selected on a per-channel basis).

Multichannel DMA Data Packing (MCPDE)

Multichannel DMA data packing and unpacking are enabled using the `SPORT_MCTL_A.MCPDE` bit.

When set, data is packed, and the SPORT expects the data in the DMA buffer to correspond only with enabled SPORT channels. For example, if only channels 1 and 9 are enabled in a 10-channel window (`SPORT_MCTL_A.WSIZE` = 9), the SPORT expects the buffer to be exactly two words in length, where channel 1 is associated with the first element in the buffer and channel 9 is associated with the second.

When cleared, data is unpacked, and the SPORT expects the DMA buffer to have a word for each of the channels in the active window, whether the channel is enabled or not. As such, the DMA buffer size must be exactly the size of the window. Using the same example as the packed case above, if only channels 1 and 9 are enabled in a 10-channel window (`SPORT_MCTL_A.WSIZE` = 9), then the DMA buffer size is ten words. The data at offsets 1 and 9 within the buffer are associated with the data transfers of channels 1 and 9, respectively. The rest of the words in the buffer are unused.

Packed I²S Mode

The SPORT supports a packed I²S mode, which can be used for audio codec communications using multiple channels. This mode allows applications to send more than the standard 32 bits per channel available through standard I²S mode. Packed mode is implemented using standard multichannel mode (and is therefore programmed similarly to multichannel mode).

To set the SPORT up in packed I²S mode, the following configuration is required:

- `SPORT_CTL_A.OPMODE` = 1
- `SPORT_MCTL_A.MCE` = 1

Like multichannel mode, packed I²S mode also supports a maximum of 128 channels, where up to 128 channels of data can be transferred for every transition of the frame sync signal acting as an L/R clock (for example, up to 128 left-channel words transfer during the high portion of the L/R clock, and up to 128 right-channel words transfer during the low portion).

As shown in the ***Packed I²S Mode 128 Operation*** figure, the packed waveforms are the same as those waveforms used in multichannel mode, except the frame sync is toggled for every frame and emulates I²S mode.

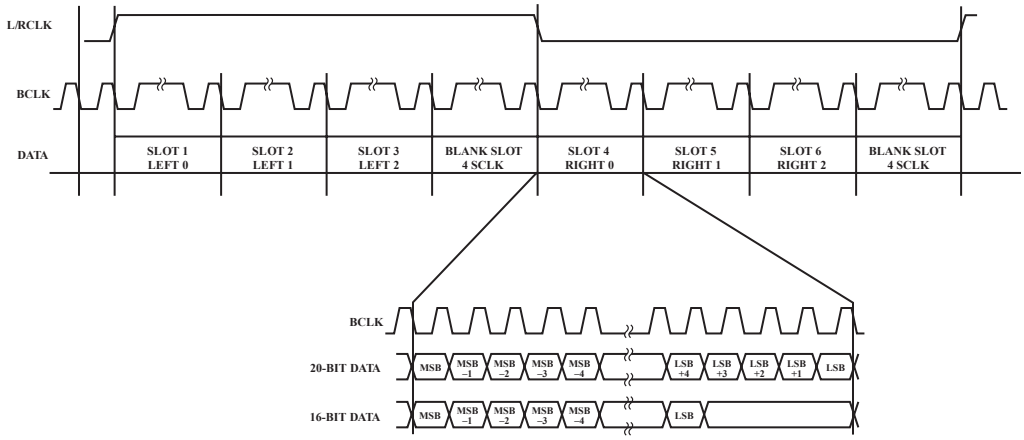


Figure 30-12: Packed I²S Mode 128 Operation

Serial Bit Clock Options

In packed I²S mode, the SPORTs can either accept an external serial bit clock or generate one internally, as governed by the `SPORT_CTL_A.ICLK` configuration bit. For an internally-generated serial bit clock (`SPORT_CTL_A.ICLK` = 1), use the `SPORT_DIV_A.CLKDIV` bit field to configure the serial bit clock rate from the system clock.

The serial bit clock edge that is used for sampling or driving data and frame syncs is programmable using the `SPORT_CTL_A.CKRE` bit.

L/R Clock (Frame Sync) Options

The frame sync period in packed I²S mode is defined as:

FS period = [(`SPORT_CTL_A.SLEN` + 1) × number of channels] - 1.

The L/R clock can be supplied externally or internally generated depending on the `SPORT_CTL_A.IFS` bit setting. The logic level of the L/R clock associated with the left and right channel data can be changed using the `SPORT_CTL_A.LFS` configuration bit.

Gated Clock Mode

Some system components such as ADCs and DACs utilize a SPI-compatible protocol for the interface. To communicate with such devices, the SPORT must support a gated clock, where the data valid information is embedded in the clock (for example, the clock only toggles when data is valid). This gated clock feature is enabled using the `SPORT_CTL_A.GCLKEN` bit.

To enable the gated clock mode of operation, program the SPORT to comply with the following requirements.

- Do not enable gated clock functionality in right-justified or multichannel mode
- Gated clock mode has the following requirements for other control bits:

  - The serial clock and frame sync signals must have the same source (SPORT_CTL_A.ICLK = SPORT_CTL_A.IFS)
  - Unframed mode is not supported (SPORT_CTL_A.FSR must be set)
  - Clear the SPORT_CTL_A.DIFS bit in transmit mode; set it in receive mode
- Satisfy the following necessary conditions when gated clock mode is enabled:
  - Seven serial clock cycles are required between enabling the SPORT and the first frame sync. If this requirement is not met, the SPORT can drop the first data (for subsequent data, this requirement is not applicable)
  - For externally-provided clock and frame sync, the frame sync must be inactive during clock synchronization after the SPORT has been enabled
  - For an edge-detected frame sync (SPORT_CTL_A.FSED = 1), the frame sync must transition back to the inactive state before the current word transfer is complete (or when the clock is still running). If this requirement is not met, the SPORT does not recognize the next valid frame sync and skips the channel. The SPORT continues to skip the frame syncs until the frame sync transitions back to an inactive state while the clock is active.

Data Transfers and Interrupts

SPORT data can be transferred to or from internal or external memory by two methods:

- Core-driven, single-word transfers
- DMA-driven, multiple-word transfers (optionally with multiple work units)

Core-driven transfers use SPORT interrupts to signal the processor core to perform MMR-based single-word transfers to or from the SPORT data buffers. DMA can be set up to automatically transfer a configurable number of serial words between the SPORT transmit/receive data buffers and memory, and then generate a data completion interrupt request when a work unit or a series of work units completes, thus signaling by the SEC to the processor core that a block of data has been transferred.

The following sections provide information on core-driven and DMA-driven data transfers.

Data Buffers

When programming the serial port data channels (primary or secondary) as a transmitter by setting SPORT_CTL_A.SPTRAN =1, only the corresponding transmit data buffers (SPORT_TXPRI_A and SPORT_TXSEC_A) become active. The receive data buffers (SPORT_RXPRI_A and SPORT_RXSEC_A) remain inactive. Similarly, when the SPORT data channels are programmed for receive operation (SPORT_CTL_A.SPTRAN =0), then only corresponding receive data buffers (SPORT_RXPRI_A and SPORT_RXSEC_A) are active. Do not attempt to read or write inactive data buffers. When the processor operates on the inactive transmit or receive buffers while the SPORT is enabled, unpredictable results can occur.

Each of these buffers is 32-bit wide (corresponds to maximum serial data word length). When using word lengths less than 32 bits for SPORT operation, the data in these buffers is automatically right justified. (The LSB bit of data is at the bit 0 location of the buffer). The upper unused bits can be zero-filled or sign-extended depending on SPORT_CTL_A.DTYPE field.

Transmit Data Buffers (SPORT_TXPRI_A and SPORT_TXSEC_A)

When enabled as a transmitter (SPORT_CTL_A.SPTRAN =1), each SPORT half has its own set of transmit data buffers. The primary (0) and secondary (1) data paths of each SPORT half have separate data buffers, referred to as SPORT_TXPRI_A and SPORT_TXSEC_A, respectively.

These transmit data buffers are 32 bits wide. Load these buffers with the data for transmission on the primary and secondary data channels. The DMA controller loads the data automatically. Or, the program running on the processor core loads the data manually.

Together with the output shift register, transmit data buffers act like a two-location FIFO. If data packing is disabled (SPORT_CTL_A.PACK =0), the transmit path can hold as many as three data words. If data packing is enabled (SPORT_CTL_A.PACK =1), it can hold two packed data words at any given time.

When the transmit shift register becomes empty (transfer out all the bits of previous word), data in the transmit data buffer is automatically loaded into it. An interrupt occurs when the output transmit shift register has been loaded, signifying that the transmit data buffer is empty and ready to accept the next word. This interrupt does not occur when serial port is operating in DMA mode or when the corresponding interrupt enable mask bit is set.

If only the primary data path of a SPORT half is enabled, programs must not write to the inactive secondary transmit data buffer and conversely. If the core keeps writing to the inactive buffer, the status of that transmit buffer becomes full. This state can cause the core to hang indefinitely, since data is never transmitted to the output shift register.

Receive Data Buffers (SPORT_RXPRI_A and SPORT_RXSEC_A)

When enabled as receiver (SPORT_CTL_A.SPTRAN =0), each SPORT half has its own set of receive data buffers. The primary (0) and secondary (1) data paths of each SPORT half have separate data buffers, referred as SPORT_RXPRI_A and SPORT_RXSEC_A, respectively. Together with input shift register, the receive data buffers act like a three-location FIFO, as the receive path has two data registers.

These receive data buffers are the 32 bits wide. These buffers are automatically loaded from the receive shift register when a complete word has been received into it. An interrupt occurs when the receive data buffer is loaded, signifying that new data is available in the receive data buffer and is ready to read. This interrupt does not occur when the serial port is operating in DMA mode or when the corresponding interrupt enable mask bit is set.

If only the primary data path of a SPORT half is enabled, programs must not read from the inactive secondary receive data buffer and conversely. If the core keeps reading from the inactive buffer, the status of that receive buffer becomes empty. This state can cause the core to hang indefinitely since new data is never received through the input shift register.

Data Buffer Status

The SPORT provides status information about its primary and secondary data buffers through the SPORT_CTL_A.DXSPRI and SPORT_CTL_A.DXSSEC bits, respectively. It also provides error status information through the corresponding SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC bits, respectively. Depending on the SPORT_CTL_A.SPTRAN bit setting, these bits reflect the status of either the pair of transmit (SPORT_TXPRI_A and SPORT_TXSEC_A) or receive (SPORT_RXPRI_A and SPORT_RXSEC_A) buffers, indicating whether the buffer is full, partially full, or empty.

When attempting to read from an empty receive buffer or write to a full transmit buffer, the SPORT delays access until the buffer is ready, potentially resulting in excessive MMR bus response times. To avoid this when doing core-driven transfers, always check the buffer status to determine if the access can be made. The SPORT updates the status bits in the SPORT_CTL_A register during reads and writes by the core processor.

NOTE: These status bits are updated during reads and writes from the core processor even when the SPORT is disabled.

Two complete 32-bit words can be stored in the receive buffer while a third word shifts in. Therefore, almost three complete words can be received without the receive buffer being read before an overflow occurs. After receiving the third word completely, the shift register contents overwrite the second word, which will occur if the first word has not yet been read by the processor core or the DMA controller. This receive overflow condition is flagged through the error status bits of the SPORT_CTL_A register on the last bit of the third word.

Data Buffer Packing

When the SPORT is configured as a receiver with a serial data word length of 16 or less, the received data words can be packed into a 32-bit word. Similarly, if the SPORT is configured as a transmitter with a serial data word length of 16 or less, then 32-bit words being transmitted can be unpacked into 16-bit words. The SPORT_CTL_A.PACK bit is used to select this packing or unpacking feature.

When SPORT_CTL_A.PACK = 1, two consecutive received words are packed into a single 32-bit word, or each 32-bit word is unpacked and transmitted as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15–0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31–16. This packing method applies to both receive (packing) and transmit (unpacking) operations. In this case, the transmit and receive interrupt requests are generated for the 32-bit packed words, not for each 16-bit word.

NOTE: When 16-bit received data is packed into 32-bit words and stored in normal word space in the processor's internal memory, the 16-bit words can be read or written using short word space addressing.

Single-Word (Core) Transfers

The SPORTs can transmit or receive individual data words with interrupt requests occurring as each data word is transferred. When a SPORT is enabled with the corresponding DMA channel disabled, interrupt requests are generated when:

- a complete word has been received in the receive data buffer or

- the transmit data buffer is not full

When performing core transfers, be sure to access only those buffers that are associated with enabled data paths, as governed by the transfer direction (SPORT_CTL_A.SPTRAN) bit and the primary/secondary (SPORT_CTL_A.SPENPRI/SPORT_CTL_A.SPENSEC) data enable bits. If inactive SPORT data buffers are read from or written to by the core while the SPORT is enabled, the core can hang. For example, if a half SPORT is programmed to be a transmitter and the core reads from one of the receive buffers associated with that half SPORT, the core can hang as if it were reading an empty buffer that is active and awaiting new data to arrive. Because this is a transmitting HSPORT, that data will never arrive, thus locking the core up until the SPORT is reset. To avoid such a situation, be sure to check the status of the appropriate data buffer before attempting a core access to it by interrogating the SPORT_CTL_A.DXSPRI or SPORT_CTL_A.DXSSEC status bits.

DMA Transfers

Direct memory access (DMA) provides a mechanism for transferring an entire block of serial data before an interrupt is generated. The processor's on-chip DMA controller automatically handles the DMA transfer, thus allowing the processor core to run in parallel until the entire block of data is transferred. When the interrupt request occurs, a service routine can then process the entire block of data (rather than react to single words), thus significantly reducing overhead.

Each half SPORT has a dedicated DMA channel that serves both the primary and secondary data paths. When configured as a transmitter (SPORT_CTL_A.SPTRAN = 1) with both the primary and secondary data paths enabled (SPORT_CTL_A.SPENPRI = SPORT_CTL_A.SPENSEC = 1), the DMA channel requires that the source DMA buffer interleave the data beginning with the primary channel, as it will alternately load to the primary and secondary transmit data buffers once it is enabled. The complementary operation is true in receive mode (SPORT_CTL_A.SPTRAN = 0) when both data paths are enabled, as the DMA channel alternately reads from the primary and secondary receive data buffers and interleaves them in the destination DMA buffer. As such, software must de-interleave the data corresponding to the primary and secondary channels from the receive DMA buffer.

If the SPORT is configured in stereo mode, the same DMA channel handles both the left and right channels of both data paths (primary and/or secondary). Therefore, for a transmit DMA with only one data path enabled, the source buffer must be populated such that the left- and right-channel data is interleaved. If both data paths are enabled, the DMA channel alternately loads to the primary and secondary transmit data buffers once it is enabled. As such, the interleaving requirement is for the primary left-channel data to be followed by the secondary left-channel data, then the primary right-channel data, and finally the secondary right-channel data. The complementary operation is true in receive mode, where the DMA channel alternately reads from the primary and secondary receive data buffers and interleave them in the destination DMA buffer. For the stereo modes of operation, the destination DMA buffer is interleaved as left-right data for a single data input. If both data paths are enabled, the destination DMA buffer is written with the primary and secondary left-channel data followed by the primary and secondary right-channel data. As such, software must de-interleave the primary and secondary left- and right-channel data from the receive DMA buffer, as defined by this scheme.

Since both the primary and secondary data paths share the single DMA channel, each half SPORT has a single interrupt request for data completion, as well as an error interrupt request. The DMA controller can generate an

interrupt request at the end of a chain of DMA work units (when using multiple descriptors) or at the end of individual DMA work unit.

The SPORT DMA channels are assigned a higher priority than all the other DMA channels (for example, the SPI port). Having higher priority causes the SPORT DMA transfers to execute first when multiple DMA requests occur in the same cycle. The SPORT DMA channels are numbered and prioritized in the DMA channel list table in the DMA chapter.

Although the most efficient DMA transfers execute with 32-bit words, the SPORTs can handle word sizes from 4 to 32 bits (as defined by `SPORT_CTL_A.SLEN` field). If the serial data length is 16 bits or smaller, two pieces of data can be packed into 32-bit words for each DMA transfer, as selected by setting the `SPORT_CTL_A.PACK` bit. When this bit is set, the SPORT generates the transmit and receive interrupts for the 32-bit packed words, not for each 16-bit word. For more information, see the Data Buffer Status section.

NOTE: The SPORT DMA channel can access both internal memory and external memory of the processor without any core overhead.

Data Transfer Interrupt

Each half SPORT features a data transfer interrupt request that is shared by both the primary and secondary data channels in both transmit and receive modes. To determine the source of the data transfer interrupt request, applications can check the primary and secondary data buffer status bits (`SPORT_CTL_A.DXSPRI` and , `SPORT_CTL_A.DXSSEC` , respectively).

When using core-driven transfers, this interrupt's meaning depends on the direction of the SPORT:

- As transmitter (`SPORT_CTL_A.SPTRAN` = 1) - the transmit data buffer is empty
- As receiver (`SPORT_CTL_A.SPTRAN` = 0) - new data is available in the receive data buffer

NOTE: When data packing is enabled (`SPORT_CTL_A.PACK` = 1), the core-driven transmit and receive interrupt requests are generated for 32-bit packed words, not for each 16-bit word.

In both cases, the interrupt request can be used to signal the core that an individual transfer has completed. For transmit operations, it indicates that the transmit data buffer can be safely loaded (either the buffer is already empty, or the last data has moved from the data buffer to the shift register). For receive operations, it indicates that new data has arrived and can be read (or must be read before a subsequent word overwrites it).

When the SPORT is configured to use DMA to move data between memory and the peripheral (the most generic way to use dedicated DMA for sport data transfers), the same data transfer interrupt request instead indicates the completion of the transfer of a block of serial data (rather than a single word). When DMA is used, the DMA count register must be initialized to specify the number of words to transfer. This count decrements after each DMA transfer on the channel, and the data transfer interrupt request signal is asserted when the word count reaches zero (for example, a DMA work unit has finished).

For transmit DMA, the interrupt request is raised when the last word in the DMA work unit is loaded from the source memory to the HSPORT FIFO. This interrupt request can signal to the core that a new DMA work unit can be configured or that other software threads can now run. The transmit interrupt request can optionally be deferred

until the last word of the work unit has fully shifted out of the shift register (see the Transfer Finish Interrupt (TFI) section for details).

For receive DMA, the interrupt request is raised when the last word is loaded to the destination memory. In addition to that described for transmit DMA, this interrupt request also serves as an indication to the core that there is a buffer of newly acquired data that is ready to be processed.

See the DMA chapter for further details regarding enabling of the DMA interrupt requests associated with the various modes of DMA operation.

NOTE: As a single DMA channel services both the primary and secondary data paths associated with the SPORT, there is a single DMA completion interrupt request.

Transfer Finish Interrupt (TFI)

When configured for transmit DMA (`SPORT_CTL_A.SPTRAN` =1), the data transfer interrupt request gets generated by the DMA engine itself when it decrements its count register upon loading the last element from memory to the HSPORT hardware. Alternately, the SPORT can use a Transmit Finish Interrupt (TFI) to signal the actual end of the transmission (for example, when the last bit of the last data word of the buffer has shifted out of the SPORT to the system) by setting the `SPORT_CTL_A.TFIEN` bit. When this bit is set, then DMA signal that would normally assert the data transfer interrupt request instead signals the SPORT that the DMA work unit is complete. The SPORT then waits until all the data in the FIFO is shifted out (including the transmit shift register) and asserts the TFI interrupt request upon completion.

NOTE: To enable this functionality in the DMA engine, be sure to configure the interrupt type field in the DMA configuration register for Peripheral interrupt. See the DMA chapter for further details.

Error Detection (Status) Interrupt

In addition to the dedicated data transfer interrupt request, each half SPORT also features an optional error status interrupt request that can be triggered when error conditions occur relative to data or frame syncs associated with the half SPORT.

Data-related errors depend on the direction of the SPORT and reflect overflow or underflow conditions, which are depicted in the `SPORT_CTL_A` control register as read-only sticky bits `SPORT_CTL_A.DERRPRI` and `SPORT_CTL_A.DERRSEC` (for the primary and secondary channels, respectively).

- When the SPORT is configured as a transmitter, these bits provide transmit data buffer underflow status. When the frame sync signal occurs when the transmit data buffer is empty, the underflow bit corresponding with the offending transmit data buffer is set, as the SPORT will transmit data whenever it detects a valid frame sync signal, whether new data is present or not.
- When the SPORT is configured as a receiver, these bits provide receive overflow status. When a channel receives new data while the receive buffer is already full, the new data overwrites the existing data, thus causing an overflow. When this occurs, the overflow bit corresponding with the offending receive data buffer is set, as the SPORT receives data whenever it detects a valid frame sync signal, whether there is room in the receive buffer or not.

Each half SPORT also features an error register (SPORT_ERR_A), which is the source for the assertion of the described data-related error status bits. When a data-related error occurs on the primary or secondary data paths, the error is logged in the SPORT_ERR_A.DERRPSTAT or SPORT_ERR_A.DERRSSTAT bits, respectively. To enable these status bits to generate the HSPORT status interrupt request in the SEC, the corresponding SPORT_ERR_A.DERRPMSK and SPORT_ERR_A.DERRSMSK bits must be set (for the primary and secondary data paths, respectively).

The SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC channel error status bits are sticky read-only bits that can be cleared in two ways:

- Reset the error detection logic by disabling the channel associated with the error condition (clear the SPORT_CTL_A.SPENPRI or SPORT_CTL_A.SPENSEC control bit).
- Clear the source of the interrupt by writing-1-to-clear the SPORT_ERR_A.FSERRSTAT, SPORT_ERR_A.DERRPSTAT, or SPORT_ERR_A.DERRSSTAT status bits.

In addition to data-related errors, SPORT_ERR_A also tracks frame sync errors in the SPORT_ERR_A.FSERRSTAT status bit. Similar to the data-related errors, the frame sync error can be enabled as a source for raising the error status interrupt request via the SEC by setting the SPORT_ERR_A.FSERRMSK bit. A frame sync error occurs when the frame sync is detected prematurely, as explained in the Premature Frame Sync Error Detection section.

A frame sync error is not detected in the following cases:

- When there is no active transmit or receive data, and the frame sync pulse occurs due to noise on the input signal–when there is no active transfer, a noise-induced frame sync pulse is valid.
- When there is an active underflow or overflow error–frame sync errors cannot be detected because the SPORT error logic does not run after one of the data errors has occurred and remains unserviced.
- When the frame sync pulse does not the meet minimum timing requirements–when the frame sync pulse is shorter than a SPORT clock period, there is no guarantee that it gets sampled and may go unnoticed.

Grouping of SPORTs

For some applications, enabling multiple SPORTs need to be synchronized so that all of them start and/or end at the same time. It may also be required to generate/receive a single interrupt/trigger for the DMA transfer of all the SPORTs in the same group.

The DAI_GBL_SP_EN register is used to control enabling/disabling. The DAI_GBL_INT_EN register can be used to control interrupt/trigger generation/reception by multiple SPORTs in a group.

NOTE: All references to SPORT 0-3 in the context of DAI1 must be read as SPORT4-7.

SPORT Enable Grouping

The figure shows the block diagram of possible group enabling options for the SPORTs in DAI0 and DA1.

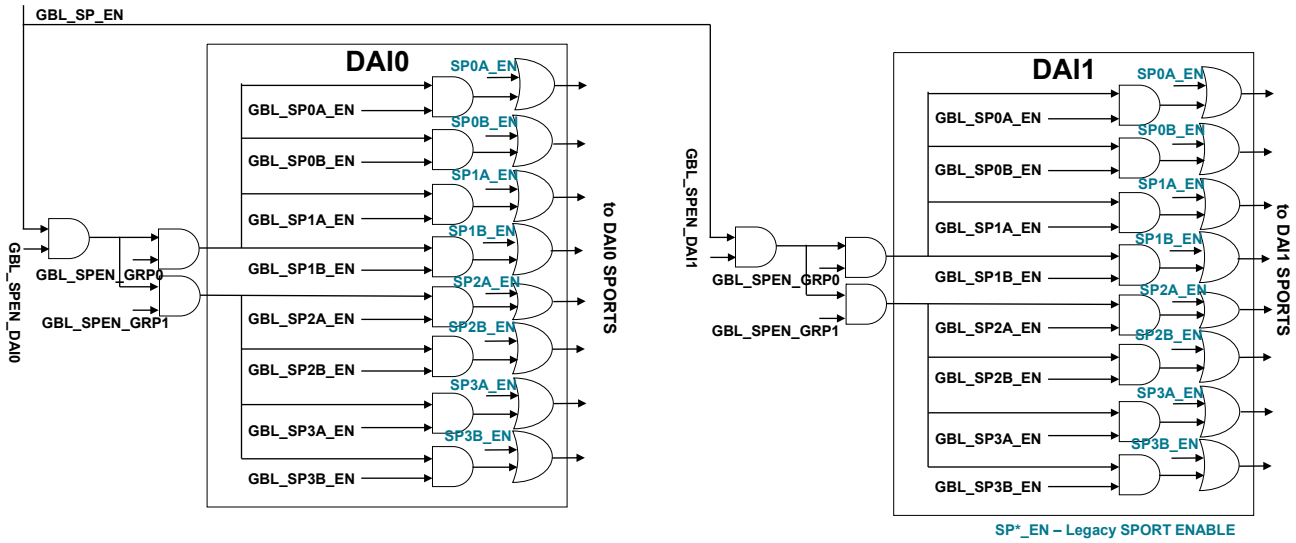


Figure 30-13: Group Enabling Options in DAI0 and DA1

It shows one of the channels in each half sport, same is applicable for other channel as well. For example, GBL_SP0A_EN means GBL_SP0A_PC_EN or GBL_SP0A_SC_EN.

Each DAI has a DAI_GBL_SP_EN register. To enable the SPORTs grouped within the same DAI at the same time, write to the corresponding DAI_GBL_SP_EN register is required. When a group of SPORTs in different DAIs must be enabled simultaneously, writing to the DAI_GBL_SP_EN.GBL_SP_EN field is required. This bit is reserved for the DAI1.

The following group enable options are possible:

- Up to 16 channels in 8 half SPORTs in DAI0
- Up to 16 channels in 8 half SPORTs in DAI1
- Up to 8 channels in 4 half SPORTs in each DAI
- Up to 32 channels in 16 half SPORTs (8 half SPORTs in DAI0 and 8 half SPORTs in DAI1)

NOTE: While grouping SPORTs across DAIs, two interrupts (one per DAI) are generated and managed by software. Grouping can be done only once as the DAI_GBL_SP_EN.GBL_SP_EN field is used.

NOTE: The grouping of SPORTs and its interrupts/triggers are two separate entities. When the interrupts and triggers must be grouped similar to the SPORT grouping, use the group interrupt/trigger enable register.

SPORT Interrupt/Trigger Grouping

Each DAI has a DAI_GBL_INT_EN register. Each of these registers has two group interrupt enable bits DAI_GBL_INT_EN.GRP0_INT_EN and DAI_GBL_INT_EN.GRP1_INT_EN, two group trigger (controller) enable bits DAI_GBL_INT_EN.GRP0_TRG_EN and DAI_GBL_INT_EN.GRP1_TRG_EN. There are individual GRPx_SPyINT_EN bits for each group to enable both interrupt/trigger for up to 8 channels (4 half SPORTs). Enable these bits as per the grouping in the DAI_GBL_SP_EN register.

NOTE: To enable or to disable the SPORT interrupt/trigger grouping feature, the individual sport interrupt/trigger enable bits must be programmed along with the respective group interrupt/trigger enable bits in the DAI_GBL_INT_EN register.

Regarding grouping of SPORT interrupts/triggers:

- All the interrupts/triggers in a single group are ANDed and a single interrupt/trigger goes to SEC/TRU.
- To generate group interrupt/trigger, all the SPORTs in the corresponding group should be configured to generate interrupt (DMA_CFG.INT)/trigger (DMA_CFG.TRIG).
- The interrupts/triggers generated by the SPORTs in a group can either be used as a single group interrupt/trigger (enabled by the DAI_GBL_INT_EN register) or the individual SPORT DMA done interrupts/triggers.
- As there are no separate status registers for global interrupts, to clear the global interrupts, clear (W1C) the individual SPORT DMA_STAT.IRQDONE bits. Theoretically, the global interrupt gets cleared by clearing the DMA_STAT.IRQDONE bit of at least one SPORT in the group. However, it is recommended to clear the DMA_STAT.IRQDONE bits of all the SPORTs in the group to achieve a clean state for the next global interrupt generation.
- The DAI_GBL_INT_EN.GRP1_TRG_EN/DAI_GBL_INT_EN.GRP0_TRG_EN bits enables group controller trigger functionality for SPORTs in a group.

SPORT Programming Model

The following sections provide programming guidance for setting up the SPORTs for use in an application:

- Initializing Core-Driven (Non-MCM) Transfers
- Initializing Multichannel Transfers
- Using DMA for SPORT Transfers
- Using Companding as a Function

Initializing Core-Driven (Non-MCM) Transfers

The following programming model applies to all of Standard DSP Serial Mode, I^2S Mode, Left-Justified Mode, and Right-Justified Mode for core-driven transfers. More steps are required to properly initialize the SEC to service the SPORT interrupts (see the SEC chapter for details).

NOTE: This example uses half SPORT A registers. With appropriate changes to register names, this example also applies to half SPORT B.

1. Clear the SPORT_CTL_A and SPORT_MCTL_A configuration registers.

   ADDITIONAL INFORMATION: Clearing these registers ensures that the SPORT logic (including the multichannel logic) is fully reset before attempting to reprogram it.

2. Optionally program the SPORT_DIV_A clock divisor register.

   ADDITIONAL INFORMATION: This step is only required for internally-generated timing signals. Configure the serial bit clock and/or frame sync (or L/R clock, for stereo modes) rates according to the guidance in the Serial Clock and Frame Sync sections.

3. Program the SPORT_CTL_A primary configuration register.

   ADDITIONAL INFORMATION: Set the SPORT operating mode along with the configurable clock, frame sync, word length, direction, and data format options (see the Operating Modes and Options section for details). Do not set the SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC buffer enable bits in this step.

4. Optionally program the SPORT_CTL2_A secondary configuration register.

   ADDITIONAL INFORMATION: This step is required only if internal multiplexing logic must be enabled to share clock and frame sync signals between a top SPORT module's A and B halves (see the Multiplexer Logic section for details).

5. Optionally program the SPORT_ERR_A error register.

   ADDITIONAL INFORMATION: This step is required only if a separate SPORT error interrupt is desired (see the Error Detection (Status) Interrupt section for details).

6. For Right-Justified mode only, program the SPORT_MCTL_A.WOFFSET field.

   ADDITIONAL INFORMATION: In Right-Justified mode, this field serves as the delay count (DCNT) required to align the LSB of each stereo channel with the L/R clock transition and must be programmed manually (see the Right-Justified Mode section for details).

7. Enable the primary/secondary data path(s) in the SPORT_CTL_A register.

   ADDITIONAL INFORMATION: This should be performed in a read-modify-write operation setting the SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC bits, as appropriate.

8. Write data to be transmitted to the transmit buffer (SPORT_TXPRI_A and/or SPORT_TXSEC_A) *or* read data that has been received from the receive buffer (SPORT_RXPRI_A and/or SPORT_RXSEC_A).

   ADDITIONAL INFORMATION: These accesses are typically performed in the context of an interrupt service routine. See the SEC chapter for further information. Do not attempt to read or write inactive data buffers. If the core attempts to access inactive transmit or receive buffers while the SPORT is enabled, unpredictable results may occur.

Initializing Multichannel Transfers

When in Multichannel (TDM) Mode or Packed I^2S Mode, the SPORT is in a multichannel operational mode. Follow the steps below to properly initialize the SPORT for multichannel modes of operation. More steps are required to properly (see the SEC chapter for details).

NOTE: This example uses half SPORT A registers. With appropriate register changes, this example also applies to half SPORT B.

1. Clear the SPORT_CTL_A and SPORT_MCTL_A registers.

   ADDITIONAL INFORMATION: Clearing these registers ensures that the SPORT logic (including the multichannel logic) is fully reset before attempting to reprogram it.

2. Optionally program the SPORT_DIV_A clock divisor register.

   ADDITIONAL INFORMATION: This step is only required for internally-generated timing signals. Configure the serial bit clock and/or frame sync (or L/R clock, for stereo modes) rates according to the guidance in the Serial Clock and SPORT_CTL2_A sections.

3. Program the SPORT_CS0_A - SPORT_CS3_A channel select registers.

4. Program the SPORT_MCTL_A multichannel configuration register.

   ADDITIONAL INFORMATION: The SPORT supports many multichannel options. For more information, see the Multichannel (TDM) Mode section. Do not set the SPORT_MCTL_A.MCE enable bit in this step.

5. Program the SPORT_CTL_A primary configuration register.

   ADDITIONAL INFORMATION: Set the SPORT operating mode along with the configurable clock, frame sync, word length, direction, and data format options (see the Operating Modes and Options section for details). Do not set the SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC buffer enable bits in this step.

6. Optionally program the SPORT_CTL2_A secondary configuration register.

   ADDITIONAL INFORMATION: This step is required only if internal multiplexing logic must be enabled to share clock and frame sync signals between a top SPORT module's A and B halves (see the Multiplexer Logic section for details).

7. Optionally program the SPORT_ERR_A error register.

   ADDITIONAL INFORMATION: This step is required only if a separate SPORT error interrupt request is desired (see the Error Detection (Status) Interrupt section for details).

8. Set the SPORT_MCTL_A.MCE bit to enable multichannel mode.

9. Enable the primary/secondary data path(s) in the SPORT_CTL_A register.

   ADDITIONAL INFORMATION: This should be performed in a read-modify-write operation setting the SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC bits, as appropriate. DMA mode is recommended for multichannel modes of operation. For more information, see the Using DMA for SPORT Transfers programming model.

Using DMA for SPORT Transfers

DMA is supported in all SPORT operating modes (Standard DSP Serial Mode, I²S Mode, Left-Justified Mode, Right-Justified Mode, Multichannel (TDM) Mode or Packed I²S Mode). To enable DMA operation with the SPORT, execute the steps described in this section after initializing and enabling the SPORT. Instead of using the single word read or write operations described in the referenced programming models, the DMA engine automates accesses to the enabled SPORT data buffers.

NOTE: This example uses half SPORT A registers. With appropriate changes to register names, it also applies to half SPORT B.

1. Follow the guidance in the multichannel (Initializing Multichannel Transfers) or non-multichannel (Initializing Core-Driven (Non-MCM) Transfers) programming models to properly initialize and enable the SPORT hardware.
2. Prepare the data buffers in memory.

   ADDITIONAL INFORMATION: Ensure that the DMA buffer is defined according to the DMA Transfers section. For the multichannel modes of operation, be sure to also consider the setting of the `SPORT_MCTL_A.MCPDE` bit, as described in the Multichannel DMA Data Packing (MCPDE) section.
3. Initialize and enable the DMA channel allocated for the SPORT, as described in the Direct Memory Access (DMA) chapter.

Using Companding as a Function

The data in the transmit and receive buffers are companded in place. The following programming model can be used to exercise the companding hardware without transferring data, which is useful for test and debug purposes.

NOTE: This example uses half SPORT A registers. With appropriate changes to register names, this example also applies to half SPORT B.

1. Configure the SPORT as a transmitter (`SPORT_CTL_A.SPTRAN=1`) with both the primary and secondary data channels disabled (`SPORT_CTL_A.SPENPRI=0` and `SPORT_CTL_A.SPENSEC=0`).
2. Enable the desired companding scheme in the `SPORT_CTL_A.DTYPE` field.
3. Write a 32-bit word to one of the transmit buffers.
4. Wait two system clock cycles.

   ADDITIONAL INFORMATION: This delay is required to allow the SPORT companding hardware to reload the transmit buffer with the companded result. Any instructions that do not access the transmit buffer can be used to cause this delay.
5. Read the 8-bit compressed value from the transmit buffer written above.

To expand data in place, use the same sequence of operations with the receive buffer instead of the transmit buffer. When expanding data in this way, set the appropriate serial word length (SPORT_CTL_A.SLEN).

Programming Global SPORT Groups

Complete the following steps to program global SPORT groups.

NOTE: To complete all of the SPORT group transfers simultaneously, all of the SPORTs in the group must have a similar data/clock/frame sync configuration.

1. For each SPORT, program the following registers for the desired SPORT mode of operation.
   - SPORT_DIV_A/SPORT_DIV_B
   - SPORT_CTL2_A/SPORT_CTL2_B
   - SPORT_CTL_A/SPORT_CTL_B
   - SPORT_MCTL_A/SPORT_MCTL_B
   - SPORT_CS0_A/SPORT_CS0_B
   - SPORT_CS1_ASPORT_CS1_B
   - SPORT_CS2_ASPORT_CS2_B
   - SPORT_CS3_ASPORT_CS3_B

   ADDITIONAL INFORMATION: The sport enable control bits (in SPORT_CTL2_A/SPORT_CTL2_B registers) must be cleared (disabled).
2. For a global group trigger as the controller, program each SPORT DMA in the group as the controller trigger.
3. Program the sport enable and group enable bits in the DAI_GBL_SP_EN register for the desired global sport grouping.

   ADDITIONAL INFORMATION: The grouping of SPORTs (enable and disable) and the associated interrupts/triggers are separate activities. If the interrupts and triggers must be grouped similar to the SPORTs (enable and disable) grouping, use the DAI_GBL_INT_EN register.
4. Program the DAI_GBL_INT_EN register to enable the grouped interrupt or trigger generation for the SPORT group.

   ADDITIONAL INFORMATION: To enable the SPORT grouped interrupt and trigger feature, each sport interrupt or trigger enable bit must be programmed with the respective group interrupt and trigger enable bits in the DAI_GBL_INT_EN register.
5. Program the DAI_GBL_SP_EN.GBL_SP_EN bit to enable the global sport groups at the same instance.

Disabling Global SPORT Groups

Complete the following steps to disable global SPORT groups.

1. Use the individual SPORT DMA status register to clear the group interrupt. There is no specific status register for global interrupts.
2. Disable the global SPORT group interrupt and trigger bits for the group in the DAI_GBL_INT_EN register.

   ADDITIONAL INFORMATION: To disable the SPORT grouped interrupt and trigger feature, each sport interrupt and trigger enable bit must be programmed with the respective group interrupt and trigger enable bits in the DAI_GBL_INT_EN register.
3. Disable the global individual SPORT enable and the global SPORT group enable bits in the DAI_GBL_SP_EN register.
4. Disable the DMAs for the SPORTs when the transfer is complete.

ADSP-2184x SPORT Register Descriptions

Serial Port (SPORT) contains the following registers.

Table 30-14: ADSP-2184x SPORT Register List

Name	Description
SPORT_CS0_A	Half SPORT 'A' Multichannel 0-31 Select Register
SPORT_CS0_B	Half SPORT 'B' Multichannel 0-31 Select Register
SPORT_CS1_A	Half SPORT 'A' Multichannel 32-63 Select Register
SPORT_CS1_B	Half SPORT 'B' Multichannel 32-63 Select Register
SPORT_CS2_A	Half SPORT 'A' Multichannel 64-95 Select Register
SPORT_CS2_B	Half SPORT 'B' Multichannel 64-95 Select Register
SPORT_CS3_A	Half SPORT 'A' Multichannel 96-127 Select Register
SPORT_CS3_B	Half SPORT 'B' Multichannel 96-127 Select Register
SPORT_CTL2_A	Half SPORT 'A' Control 2 Register
SPORT_CTL2_B	Half SPORT 'B' Control 2 Register
SPORT_CTL_A	Half SPORT 'A' Control Register
SPORT_CTL_B	Half SPORT 'B' Control Register
SPORT_DIV_A	Half SPORT 'A' Divisor Register
SPORT_DIV_B	Half SPORT 'B' Divisor Register
SPORT_ERR_A	Half SPORT 'A' Error Register
SPORT_ERR_B	Half SPORT 'B' Error Register

Table 30-14: ADSP-2184x SPORT Register List (Continued)

Name	Description
SPORT_MCTL_A	Half SPORT 'A' Multichannel Control Register
SPORT_MCTL_B	Half SPORT 'B' Multichannel Control Register
SPORT_MSTAT_A	Half SPORT 'A' Multichannel Status Register
SPORT_MSTAT_B	Half SPORT 'B' Multichannel Status Register
SPORT_RXPRI_A	Half SPORT 'A' Rx Buffer (Primary) Register
SPORT_RXPRI_B	Half SPORT 'B' Rx Buffer (Primary) Register
SPORT_RXSEC_A	Half SPORT 'A' Rx Buffer (Secondary) Register
SPORT_RXSEC_B	Half SPORT 'B' Rx Buffer (Secondary) Register
SPORT_TXPRI_A	Half SPORT 'A' Tx Buffer (Primary) Register
SPORT_TXPRI_B	Half SPORT 'B' Tx Buffer (Primary) Register
SPORT_TXSEC_A	Half SPORT 'A' Tx Buffer (Secondary) Register
SPORT_TXSEC_B	Half SPORT 'B' Tx Buffer (Secondary) Register

Half SPORT 'A' Multichannel 0-31 Select Register

Each of the bits (when set, =1) of the SPORT_CS0_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

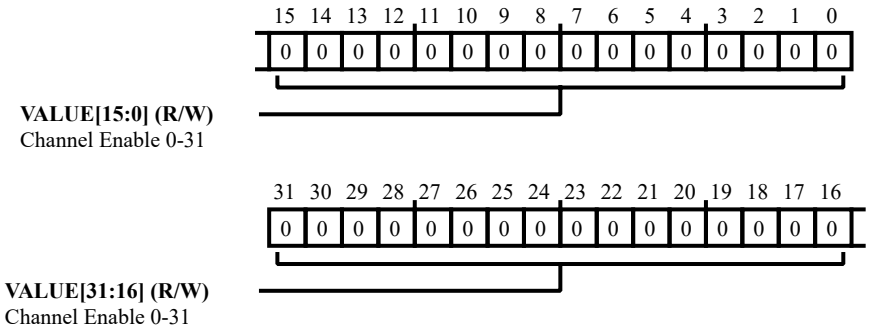


Figure 30-14: SPORT_CS0_A Register Diagram

Table 30-15: SPORT_CS0_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 0-31.

Half SPORT 'B' Multichannel 0-31 Select Register

Each of the bits (when set, =1) of the SPORT_CS0_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

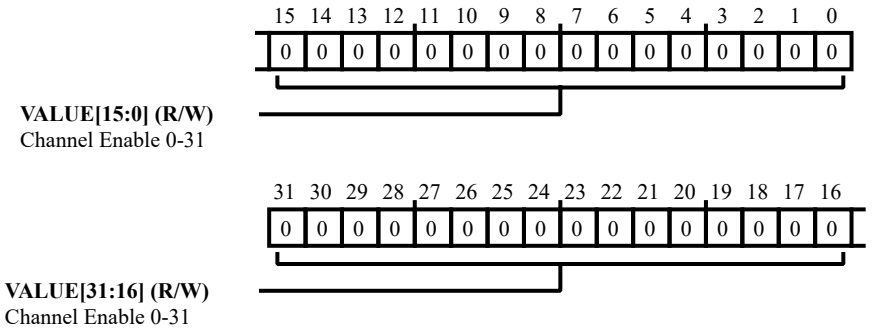


Figure 30-15: SPORT_CS0_B Register Diagram

Table 30-16: SPORT_CS0_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 0-31.

Half SPORT 'A' Multichannel 32-63 Select Register

Each of the bits (when set, =1) of the SPORT_CS1_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

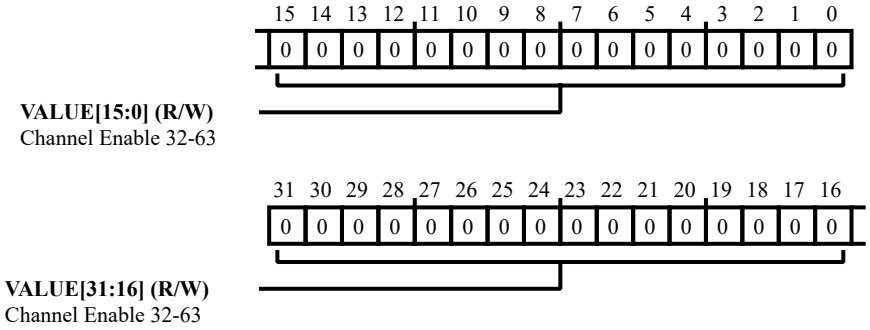


Figure 30-16: SPORT_CS1_A Register Diagram

Table 30-17: SPORT_CS1_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 32-63.

Half SPORT 'B' Multichannel 32-63 Select Register

Each of the bits (when set, =1) of the SPORT_CS1_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

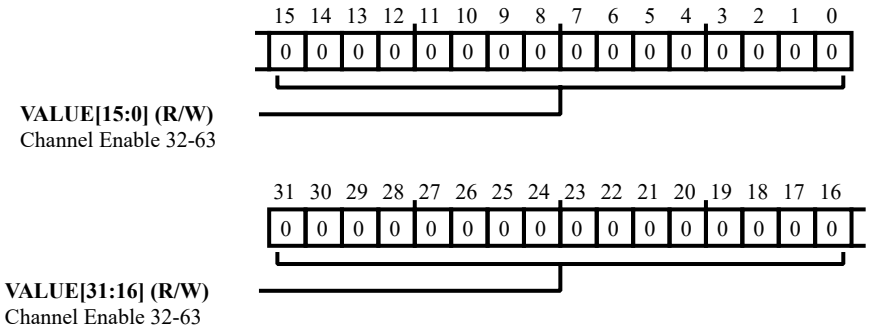


Figure 30-17: SPORT_CS1_B Register Diagram

Table 30-18: SPORT_CS1_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 32-63.

Half SPORT 'A' Multichannel 64-95 Select Register

Each of the bits (when set, =1) of the SPORT_CS2_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

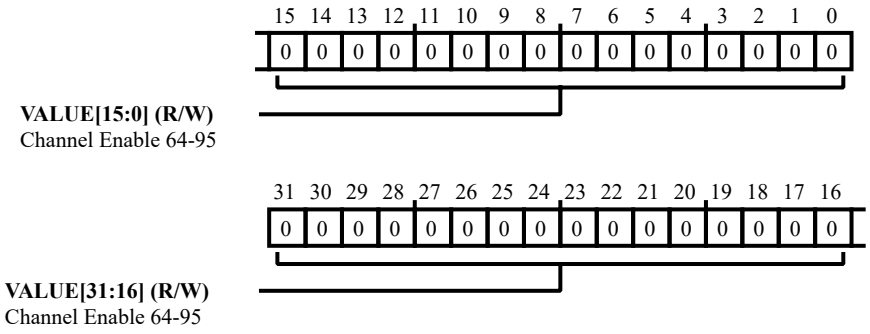


Figure 30-18: SPORT_CS2_A Register Diagram

Table 30-19: SPORT_CS2_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 64-95.

Half SPORT 'B' Multichannel 64-95 Select Register

Each of the bits (when set, =1) of the SPORT_CS2_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

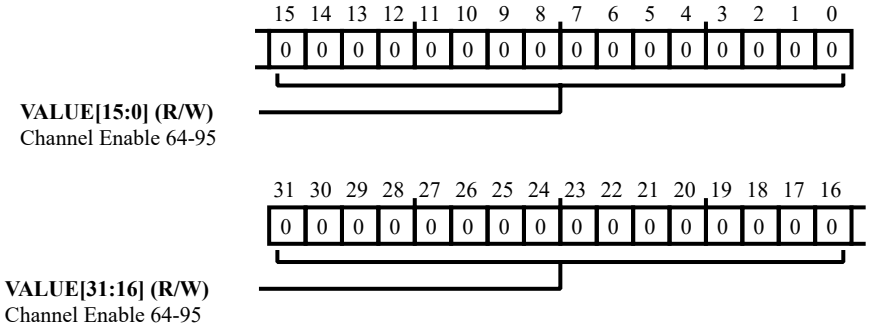


Figure 30-19: SPORT_CS2_B Register Diagram

Table 30-20: SPORT_CS2_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 64-95.

Half SPORT 'A' Multichannel 96-127 Select Register

Each of the bits (when set, =1) of the SPORT_CS3_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

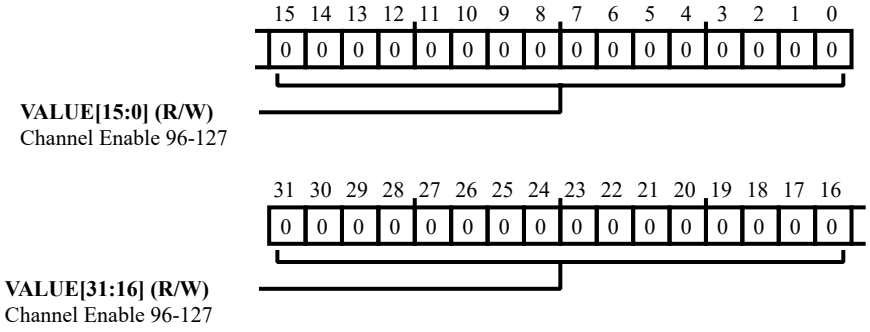


Figure 30-20: SPORT_CS3_A Register Diagram

Table 30-21: SPORT_CS3_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 96-127.

Half SPORT 'B' Multichannel 96-127 Select Register

Each of the bits (when set, =1) of the SPORT_CS3_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

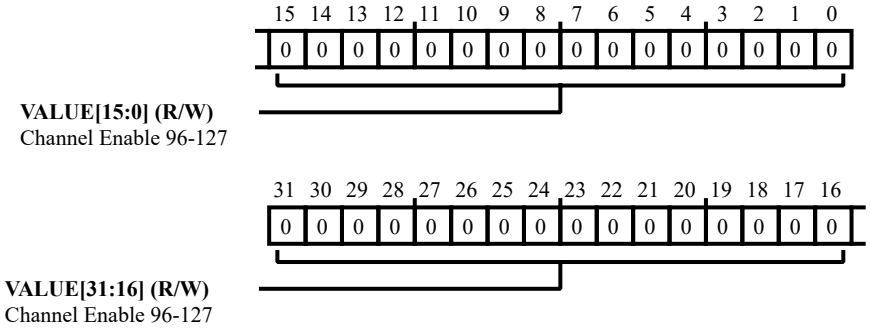


Figure 30-21: SPORT_CS3_B Register Diagram

Table 30-22: SPORT_CS3_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 96-127.

Half SPORT 'A' Control 2 Register

The SPORT_CTL2_A register controls multiplexing options for sharing serial clock and frame sync signals across the related half SPORTs.

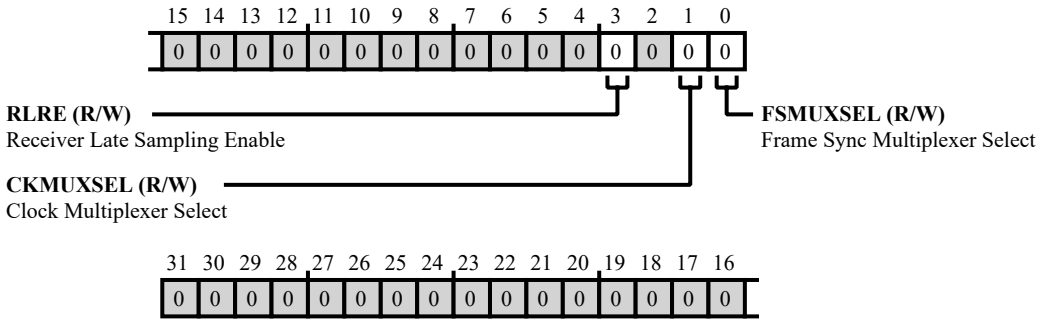


Figure 30-22: SPORT_CTL2_A Register Diagram

Table 30-23: SPORT_CTL2_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RLRE	Receiver Late Sampling Enable. The SPORT_CTL2_A.RLRE bit enables the half SPORT's late sampling of received data and frame sync by half of the operating clock cycle. Note: This mode of operation is applicable when Clock/FS are internally generated, and data is externally generated (i.e data and control are in opposite direction).	
1 (R/W)	CKMUXSEL	Clock Multiplexer Select. The SPORT_CTL2_A.CKMUXSEL bit enables multiplexing of the half SPORT's serial clock. In this mode, the serial clock of the related half SPORT is used instead of the half SPORT's own serial clock. For example, if the SPORT_CTL2_A.CKMUXSEL bit is enabled, half SPORT 'A' uses SPORT_BCLK instead of SPORT_ACLK.	
		0	Disable serial clock multiplexing
		1	Enable serial clock multiplexing
0 (R/W)	FSMUXSEL	Frame Sync Multiplexer Select. The SPORT_CTL2_A.FSMUXSEL bit enables multiplexing of the half SPORT's frame sync. In this mode, the frame sync of the related half SPORT is used instead of the half SPORT's own frame sync. For example, if the SPORT_CTL2_A.FSMUXSEL bit is enabled, half SPORT 'A' uses SPORT_BFS instead of SPORT_AFS.	
		0	Disable frame sync multiplexing
		1	Enable frame sync multiplexing

Half SPORT 'B' Control 2 Register

The SPORT_CTL2_B register controls multiplexing options for sharing serial clock and frame sync signals across the related half SPORTs.

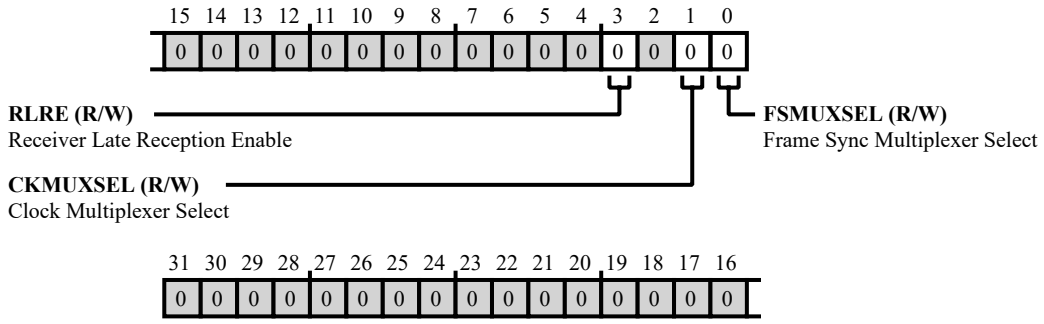


Figure 30-23: SPORT_CTL2_B Register Diagram

Table 30-24: SPORT_CTL2_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RLRE	Receiver Late Reception Enable. The SPORT_CTL2_B.RLRE bit enables the half SPORT's late sampling of received data and frame sync by half of the operating clock cycle.	
1 (R/W)	CKMUXSEL	Clock Multiplexer Select. The SPORT_CTL2_B.CKMUXSEL bit enables multiplexing of the half SPORT's serial clock. In this mode, the serial clock of the related half SPORT is used instead of the half SPORT's own serial clock. For example, if the SPORT_CTL2_B.CKMUXSEL bit is enabled, half SPORT 'B' uses SPORT_ACLK instead of SPORT_BCLK.	
		0	Disable serial clock multiplexing
		1	Enable serial clock multiplexing
0 (R/W)	FSMUXSEL	Frame Sync Multiplexer Select. The SPORT_CTL2_B.FSMUXSEL bit enables multiplexing of the half SPORT's frame sync. In this mode, the frame sync of the related half SPORT is used instead of the half SPORT's own frame sync. For example, if the SPORT_CTL2_B.FSMUXSEL bit is enabled, half SPORT 'B' uses SPORT_AFS instead of SPORT_BFS.	
		0	Disable frame sync multiplexing
		1	Enable frame sync multiplexing

Half SPORT 'A' Control Register

The SPORT_CTL_A register contains transmit and receive control bits for SPORT half 'A', including serial port mode selection for the half SPORT's primary and secondary channels. The function of some bits in the SPORT_CTL_A register vary depending on the SPORT's operating mode. For more information, see the SPORT operating modes description. If reading reserved bits, the read value is the last written value to these bits or is the reset value of these bits.

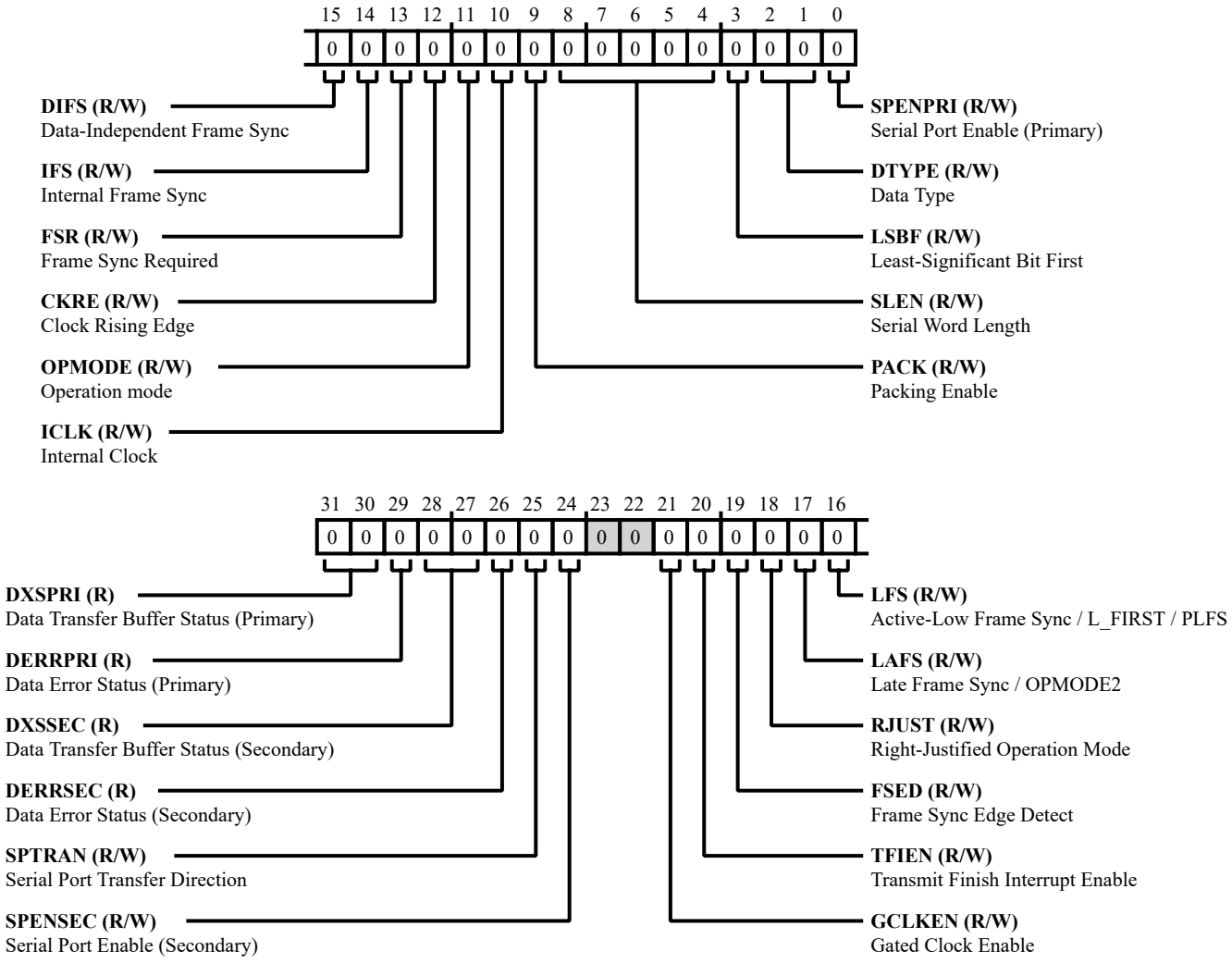


Figure 30-24: SPORT_CTL_A Register Diagram

Table 30-25: SPORT_CTL_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	DXSPRI	Data Transfer Buffer Status (Primary). The SPORT_CTL_A.DXSPRI bit field indicates the status of the half SPORT's primary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full
29 (R/NW)	DERRPRI	Data Error Status (Primary). The SPORT_CTL_A.DERRPRI bit reports the half SPORT's primary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_A.FSR bit =1, the SPORT_CTL_A.DERRPRI bit indicates whether the SPORT_AFS signal (from an internal or external source) occurred while the SPORT_TXPRI_A data buffer was empty (during transmit) or the SPORT_RXPRI_A data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_AFS signal. It is important to note that, as a receiver, the SPORT_CTL_A.DERRPRI bit indicates when the channel has received new data while the SPORT_RXPRI_A receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_A.FSR bit =0, the SPORT_CTL_A.DERRPRI bit is set whenever the SPORT is required to transmit while the SPORT_TXPRI_A transmit buffer is empty. It is also set whenever the SPORT is required to receive while the SPORT_RXPRI_A receive buffer is full. The SPORT clears the SPORT_CTL_A.DERRPRI bit if the SPORT_ERR_A.DERRPSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
28:27 (R/NW)	DXSSEC	Data Transfer Buffer Status (Secondary). The SPORT_CTL_A.DXSSEC bit field indicates the status of the half SPORT's secondary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full

Table 30-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	DERRSEC	Data Error Status (Secondary). The SPORT_CTL_A.DERRSEC bit reports the half SPORT's secondary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_A.FSR bit =1, the SPORT_CTL_A.DERRSEC bit indicates whether the SPORT_AFS signal (from an internal or external source) occurred while the SPORT_TXSEC_A data buffer was empty (during transmit) or the SPORT_RXSEC_A data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_AFS signal. It is important to note that, as a receiver, the SPORT_CTL_A.DERRSEC bit indicates when the channel has received new data while the SPORT_RXSEC_A receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_A.FSR bit =0, the SPORT_CTL_A.DERRSEC bit is set whenever the SPORT is required to transmit while the SPORT_TXSEC_A transmit buffer is empty. It is also set whenever the SPORT is required to receive while the SPORT_RXSEC_A receive buffer is full. The SPORT clears the SPORT_CTL_A.DERRSEC bit if the SPORT_ERR_A.DERRSSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
25 (R/W)	SPTRAN	Serial Port Transfer Direction. The SPORT_CTL_A.SPTRAN bit selects the transfer direction (receive or transmit) for the half SPORT's primary and secondary channels. When the direction is receive, the half SPORT activates the receive buffers, and the SPORT_ACLK and SPORT_AFS pins control the receive buffers. The transmit buffers are inactive when the half SPORT's transfer direction is receive. When the direction is transmit, the half SPORT activates the transmit buffers, and the SPORT_ACLK and SPORT_AFS pins control the transmit shift registers. The receive buffers are inactive when the half SPORT's transfer direction is transmit.	
		0	Receive
		1	Transmit
24 (R/W)	SPENSEC	Serial Port Enable (Secondary). The SPORT_CTL_A.SPENSEC bit enables the half SPORT's secondary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Table 30-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	GCLKEN	Gated Clock Enable. The SPORT_CTL_A.GCLKEN bit enables gated clock operation for the half SPORT when in DSP serial mode or left-justified stereo modes (SPORT_CTL_A.OPMODE = 0 or 1). This bit is ignored when the half SPORT is in right-justified mode (SPORT_CTL_A.RJUST =1) or multichannel mode (SPORT_MCTL_A.MCE =1). When the SPORT_CTL_A.GCLKEN bit is enabled, the SPORT clock is active when the SPORT is transferring data or when the frame sync changes (transitions to active state).	
		0	Disable
		1	Enable
20 (R/W)	TFIEN	Transmit Finish Interrupt Enable. The SPORT_CTL_A.TFIEN bit selects when the half SPORT issues its transmission complete interrupt request, if a DMA complete interrupt request is enabled by the DMA_CFG.INT configuration. When enabled (SPORT_CTL_A.TFIEN =1), the DMA complete peripheral interrupt request is generated when the last bit of last word in the DMA is shifted out. When disabled (SPORT_CTL_A.TFIEN =0), the DMA interrupt request is generated when the DMA counter expires (the last word goes to the transmit buffer).	
		0	Last word sent (DMA count done) interrupt
		1	Last bit sent (Tx buffer done) interrupt
19 (R/W)	FSED	Frame Sync Edge Detect. The SPORT_CTL_A.FSED bit enables the half SPORT to start transmitting or receiving after detecting an active edge of an external frame sync. The SPORT_CTL_A.FSED bit may be enabled even during an active frame sync, and the half SPORT starts the transfer on the next valid rising or falling edge of external frame sync. If disabled (SPORT_CTL_A.FSED =0), the half SPORT operates in the standard level-sensitive detection mode for external frame sync.	
		0	Level detect frame sync
		1	Edge detect frame sync

Table 30-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	RJUST	Right-Justified Operation Mode. The SPORT_CTL_A.RJUST bit enables the half SPORT (if SPORT_CTL_A.OPMODE =1) to transfer data in right-justified operation mode. In this mode, the half SPORT aligns data to the end of the frame sync, rather than the start of the frame sync. When using right-justified mode, systems should program an appropriate delay count to introduce a clock delay before the half SPORT state machine starts to capture data. This value is set in the DCNT field (right-justified mode usage of the SPORT_MCTL_A.WOFFSET field). For information about appropriate delay selections, see the SPORT operating modes section.	
		0	Disable
		1	Enable
17 (R/W)	LAFS	Late Frame Sync / OPMODE2. When the half SPORT is in DSP standard mode (SPORT_CTL_A.OPMODE =0) or in right-justified mode (SPORT_CTL_A.RJUST =1), the SPORT_CTL_A.LAFS bit selects whether the half SPORT generates a late frame sync (SPORT_AFS during first data bit) or generates an early frame sync signal (SPORT_AFS during serial clock cycle before first data bit). When the half SPORT is in I^2S / left-justified mode (SPORT_CTL_A.OPMODE =1), the SPORT_CTL_A.LAFS bit acts as OPMODE2, selecting whether the half SPORT is in left-justified mode or I^2S mode. When the half SPORT is in multichannel mode (SPORT_MCTL_A.MCE =1), the SPORT_CTL_A.LAFS bit is reserved.	
		0	Early frame sync (or I^2S mode)
		1	Late frame sync (or left-justified mode)
16 (R/W)	LFS	Active-Low Frame Sync / L_FIRST / PLFS. When the half SPORT is in DSP standard mode and multichannel mode (SPORT_CTL_A.OPMODE =0), the SPORT_CTL_A.LFS bit selects whether the half SPORT uses active low or active high frame sync. When the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1), the SPORT_CTL_A.LFS bit acts as L_FIRST, selecting whether the half SPORT transfers data first for the left or right channel.	
		0	Active high frame sync (DSP standard mode) or rising edge frame sync (multichannel mode) or right channel first (I^2S/packed mode) or left channel first (left-justified mode)
		1	Active low frame sync (DSP standard mode) or falling edge frame sync (multichannel mode) or left channel first (I^2S/packed mode) or right channel first (left-justified mode)

Table 30-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	DIFS	Data-Independent Frame Sync. The SPORT_CTL_A.DIFS bit selects whether the half SPORT uses a data-independent or data-dependent frame sync. When using a data-independent frame sync, the half SPORT generates the sync at the interval selected by the SPORT_DIV_A.FSDIV bit. When using a data-dependent frame sync, the half SPORT generates the sync on the selected interval when the transmit buffer is not empty or when the receive buffer is not full. Note that the SPORT_CTL_A.DIFS bit is automatically set when the half SPORT is in packed or multichannel modes.	
		0	Data-dependent frame sync
		1	Data-independent frame sync
14 (R/W)	IFS	Internal Frame Sync. The SPORT_CTL_A.IFS bit selects whether the half SPORT uses an internal frame sync or uses an external frame sync. Note that the externally-generated frame sync does not need to be synchronous with the processor's system clock.	
		0	External frame sync
		1	Internal frame sync
13 (R/W)	FSR	Frame Sync Required. The SPORT_CTL_A.FSR bit selects whether or not the half SPORT requires frame sync for data transfer. This bit is automatically set when the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1) or is in multichannel mode (SPORT_MCTL_A.MCE =1).	
		0	No frame sync required
		1	Frame sync required
12 (R/W)	CKRE	Clock Rising Edge. The SPORT_CTL_A.CKRE bit selects the rising or falling edge of the SPORT_ACLK clock for the half SPORT to sample receive data and frame sync. Note that the half SPORT changes the state of transmit data and frame sync signals on the non-selected edge of the SPORT_ACLK. Also, note that the transmit and receive related SPORT halves (A and B) should be programmed with the same value for the SPORT_CTL_A.CKRE bit. This programming drives the internally-generated signals on one edge of SPORT_ACLK and samples the received signals on the opposite edge.	
		0	Clock falling edge
		1	Clock rising edge

Table 30-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	OPMODE	Operation mode. The SPORT_CTL_A.OPMODE bit selects whether the half SPORT operates in DSP standard/multichannel mode or operates in I^2S/packed/left-justified mode. The mode selection affects the operation of the SPORT_CTL_A.LAFS and SPORT_CTL_A.LFS bits. Also, the SPORT_CTL_A.OPMODE bit enables or disables operation of the SPORT_CTL_A.GCLKEN, SPORT_CTL_A.FSED, SPORT_CTL_A.RJUST, SPORT_CTL_A.DIFS, SPORT_CTL_A.FSR, and SPORT_CTL_A.CKRE bits.	
		0	DSP standard/multichannel mode
		1	I^2S/packed/left-justified mode
10 (R/W)	ICLK	Internal Clock. When the half SPORT is in DSP standard mode (SPORT_CTL_A.OPMODE =0), the SPORT_CTL_A.ICLK bit selects whether the half SPORT uses an internal or external clock. For internal clock enabled, the half SPORT generates the SPORT_ACLK clock signal, and SPORT_ACLK is an output. The SPORT_DIV_A.CLKDIV serial clock divisor value determines the clock frequency. For internal clock disabled, the SPORT_ACLK clock signal is an input, and the serial clock divisor is ignored. Note that the externally-generated serial clock does not need to be synchronous with the processor's system clock.	
		0	External clock
		1	Internal clock
9 (R/W)	PACK	Packing Enable. The SPORT_CTL_A.PACK bit enables the half SPORT to perform 16- to 32-bit packing on received data and to perform 32- to 16-bit unpacking on transmitted data. The receive packing operation packs two successive received words into a single 32-bit word. The transmit unpacking operation unpacks each 32-bit word and transmits it as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15:0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31:16. This format applies to both receive (packing) and transmit (unpacking) operations. Companding may be used with word packing or unpacking. The half SPORT generates data transfer related interrupts when packing is enabled. The transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.	
		0	Disable
		1	Enable

Table 30-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:4 (R/W)	SLEN	Serial Word Length. The SPORT_CTL_A.SLEN bits selects word length in bits for the half SPORT's data transfers. Word may be from 4- to 32-bits in length. The formula for selecting the word length in bits is: SPORT_CTL_A.SLEN = (serial word length in bits) - 1 For DSP standard mode (SPORT_CTL_A.OPMODE =0), use SPORT_CTL_A.SLEN of 3 to 31 bits. For I^2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1), use SPORT_CTL_A.SLEN of 4 to 31 bits.	
3 (R/W)	LSBF	Least-Significant Bit First. The SPORT_CTL_A.LSBF bit selects whether the half SPORT transmits or receives data LSB first or MSB first.	
		0	MSB first sent/received (big endian)
		1	LSB first sent/received (little endian)
2:1 (R/W)	DTYPE	Data Type. The SPORT_CTL_A.DTYPE bits selects the data type formatting for the half SPORT's data transfers in DSP standard mode (SPORT_CTL_A.OPMODE =0).	
		0	Right-justify data, zero-fill unused MSBs
		1	Right-justify data, sign-extend unused MSBs
		2	u-law compand data
		3	A-law compand data
0 (R/W)	SPENPRI	Serial Port Enable (Primary). The SPORT_CTL_A.SPENPRI bit enables the half SPORT's primary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Half SPORT 'B' Control Register

The SPORT_CTL_B register contains transmit and receive control bits for SPORT half 'B', including serial port mode selection for the half SPORT's primary and secondary channels. The function of some bits in the SPORT_CTL_B register vary, depending on the SPORT's operating mode. For more information, see the SPORT operating modes description. If reading reserved bits, the read value is the last written value to these bits or is the reset value of these bits.

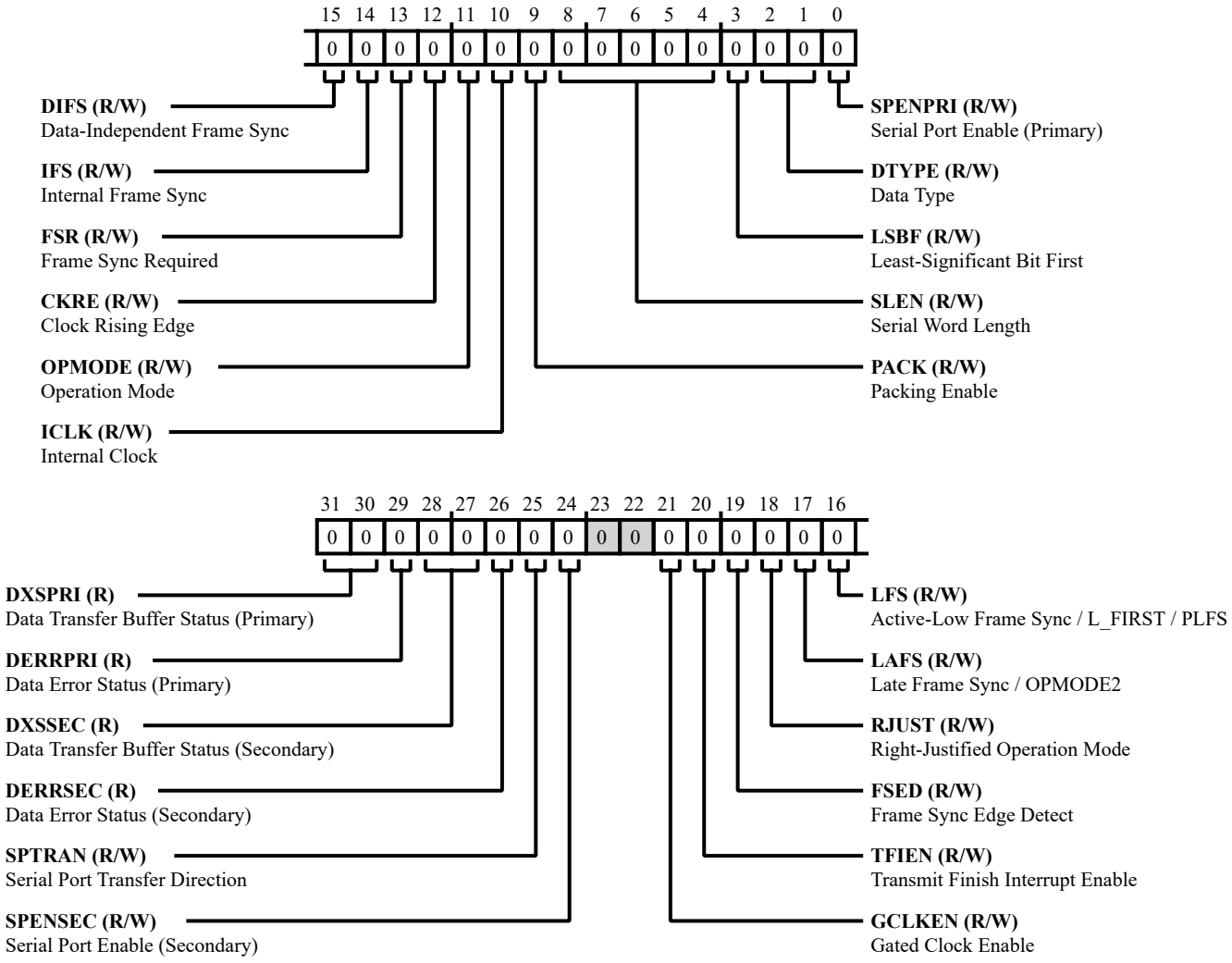


Figure 30-25: SPORT_CTL_B Register Diagram

Table 30-26: SPORT_CTL_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	DXSPRI	Data Transfer Buffer Status (Primary). The SPORT_CTL_B.DXSPRI bit field indicates the status of the half SPORT's primary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full
29 (R/NW)	DERRPRI	Data Error Status (Primary). The SPORT_CTL_B.DERRPRI bit reports the half SPORT's primary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_B.FSR bit =1, the SPORT_CTL_B.DERRPRI bit indicates whether the SPORT_BFS signal (from an internal or external source) occurred while the SPORT_TXPRI_B data buffer was empty (during transmit) or the SPORT_RXPRI_B data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_BFS signal. It is important to note that, as a receiver, the SPORT_CTL_B.DERRPRI bit indicates when the channel has received new data while the SPORT_RXPRI_B receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_B.FSR bit =0, the SPORT_CTL_B.DERRPRI bit is set whenever the SPORT is required to transmit while the SPORT_TXPRI_B transmit buffer is empty and is set whenever the SPORT is required to receive while the SPORT_RXPRI_B receive buffer is full. The SPORT clears the SPORT_CTL_B.DERRPRI bit if the SPORT_ERR_B.DERRPSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
28:27 (R/NW)	DXSSEC	Data Transfer Buffer Status (Secondary). The SPORT_CTL_B.DXSSEC bit field indicates the status of the half SPORT's secondary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	DERRSEC	Data Error Status (Secondary). The SPORT_CTL_B.DERRSEC bit reports the half SPORT's secondary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_B.FSR bit =1, the SPORT_CTL_B.DERRSEC bit indicates whether the SPORT_BFS signal (from an internal or external source) occurred while the SPORT_TXSEC_B data buffer was empty (during transmit) or the SPORT_RXSEC_B data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_BFS signal. It is important to note that, as a receiver, the SPORT_CTL_B.DERRSEC bit indicates when the channel has received new data while the SPORT_RXSEC_B receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_B.FSR bit =0, the SPORT_CTL_B.DERRSEC bit is set whenever the SPORT is required to transmit while the SPORT_TXSEC_B transmit buffer is empty. It is also set whenever the SPORT is required to receive while the SPORT_RXSEC_B receive buffer is full. The SPORT clears the SPORT_CTL_B.DERRSEC bit if the SPORT_ERR_B.DERRSSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
25 (R/W)	SPTRAN	Serial Port Transfer Direction. The SPORT_CTL_B.SPTRAN bit selects the transfer direction (receive or transmit) for the half SPORT's primary and secondary channels. When the direction is receive, the half SPORT activates the receive buffers, and the SPORT_BCLK and SPORT_BFS pins control the receive buffers. The transmit buffers are inactive when the half SPORT's transfer direction is receive. When the direction is transmit, the half SPORT activates the transmit buffers, and the SPORT_BCLK and SPORT_BFS pins control the transmit shift registers. The receive buffers are inactive when the half SPORT's transfer direction is transmit.	
		0	Receive
		1	Transmit
24 (R/W)	SPENSEC	Serial Port Enable (Secondary). The SPORT_CTL_B.SPENSEC bit enables the half SPORT's secondary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	GCLKEN	Gated Clock Enable. The SPORT_CTL_B.GCLKEN bit enables gated clock operation for the half SPORT when in DSP serial mode or left-justified stereo modes (SPORT_CTL_B.OPMODE = 0 or 1). This bit is ignored when the half SPORT is in right-justified mode (SPORT_CTL_B.RJUST =1) or multichannel mode (SPORT_MCTL_B.MCE =1). When SPORT_CTL_B.GCLKEN is enabled, the SPORT clock is active when the SPORT is transferring data or when the frame sync changes (transitions to active state).	
		0	Disable
		1	Enable
20 (R/W)	TFIEN	Transmit Finish Interrupt Enable. The SPORT_CTL_B.TFIEN bit selects when the half SPORT issues its transmission complete interrupt request, if a DMA complete interrupt request is enabled by the DMA_CFG.INT configuration. When enabled (SPORT_CTL_B.TFIEN =1), the DMA complete peripheral interrupt request is generated when the last bit of last word in the DMA is shifted out. When disabled (SPORT_CTL_B.TFIEN =0), the DMA interrupt request is generated when the DMA counter expires (the last word goes to the transmit buffer).	
		0	Last word sent (DMA count done) interrupt
		1	Last bit sent (Tx buffer done) interrupt
19 (R/W)	FSED	Frame Sync Edge Detect. The SPORT_CTL_B.FSED bit enables the half SPORT to start transmitting or receiving after detecting an active edge of an external frame sync. The SPORT_CTL_B.FSED may be enabled even during an active frame sync, and the half SPORT starts the transfer on the next valid rising or falling edge of external frame sync. If disabled (SPORT_CTL_B.FSED =0), the half SPORT operates in the standard level-sensitive detection mode for external frame sync.	
		0	Level detect frame sync
		1	Edge detect frame sync

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	RJUST	Right-Justified Operation Mode. The SPORT_CTL_B.RJUST bit enables the half SPORT (if SPORT_CTL_B.OPMODE =1) to transfer data in right-justified operation mode. In this mode, the half SPORT aligns data to the end of the frame sync, rather than the start of the frame sync. When using right-justified mode, systems should program an appropriate delay count to introduce a clock delay before the half SPORT state machine starts to capture data. This value is set in the DCNT field (right-justified mode usage of the SPORT_MCTL_B.WOFFSET field). For information about appropriate delay selections, see the SPORT operating modes section.	
		0	Disable
		1	Enable
17 (R/W)	LAFS	Late Frame Sync / OPMODE2. When the half SPORT is in DSP standard mode (SPORT_CTL_B.OPMODE =0) or in right-justified mode (SPORT_CTL_B.RJUST =1), the SPORT_CTL_B.LAFS bit selects whether the half SPORT generates a late frame sync (SPORT_BFS during first data bit) or generates an early frame sync signal (SPORT_BFS during serial clock cycle before first data bit). When the half SPORT is in I^2S / left-justified mode (SPORT_CTL_B.OPMODE =1), the SPORT_CTL_B.LAFS bit acts as OPMODE2, selecting whether the half SPORT is in left-justified mode or I^2S mode. When the half SPORT is in multichannel mode (SPORT_MCTL_B.MCE =1), the SPORT_CTL_B.LAFS bit is reserved.	
		0	Early frame sync (or I^2S mode)
		1	Late frame sync (or left-justified mode)

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	LFS	Active-Low Frame Sync / L_FIRST / PLFS. When the half SPORT is in DSP standard mode and multichannel mode (SPORT_CTL_B.OPMODE =0), the SPORT_CTL_B.LFS bit selects whether the half SPORT uses active low or active high frame sync. When the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1), the SPORT_CTL_B.LFS bit acts as L_FIRST, selecting whether the half SPORT transfers data first for the left or right channel.	
		0	Active high frame sync (DSP standard mode) or rising edge frame sync (multichannel mode) or right channel first (I^2S/packed mode) or left channel first (left-justified mode)
		1	Active low frame sync (DSP standard mode) or falling edge frame sync (multichannel mode) or left channel first (I^2S/packed mode) or right channel first (left-justified mode)
15 (R/W)	DIFS	Data-Independent Frame Sync. The SPORT_CTL_B.DIFS bit selects whether the half SPORT uses a data-independent or data-dependent frame sync. When using a data-independent frame sync, the half SPORT generates the sync at the interval selected by SPORT_DIV_B.FSDIV. When using a data-dependent frame sync, the half SPORT generates the sync on the selected interval when the transmit buffer is not empty or when the receive buffer is not full. Note that the SPORT_CTL_B.DIFS bit is automatically set when the half SPORT is in packed or multichannel modes.	
		0	Data-dependent frame sync
		1	Data-independent frame sync
14 (R/W)	IFS	Internal Frame Sync. The SPORT_CTL_B.IFS bit selects whether the half SPORT uses an internal frame sync or uses an external frame sync. Note that the externally-generated frame sync does not need to be synchronous with the processor's system clock.	
		0	External frame sync
		1	Internal frame sync

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	FSR	Frame Sync Required. The SPORT_CTL_B.FSR selects whether or not the half SPORT requires frame sync for data transfer. This bit is automatically set when the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1) or is in multichannel mode (SPORT_MCTL_B.MCE =1).	
		0	No frame sync required
		1	Frame sync required
12 (R/W)	CKRE	Clock Rising Edge. The SPORT_CTL_B.CKRE selects the rising or falling edge of the SPORT_BCLK clock for the half SPORT to sample receive data and frame sync. Note that the half SPORT changes the state of transmit data and frame sync signals on the non-selected edge of the SPORT_BCLK. Also note that the transmit and receive related SPORT halves (A and B) should be programmed with the same value for SPORT_CTL_B.CKRE. This programming drives the internally-generated signals on one edge of SPORT_BCLK and samples the received signals on the opposite edge.	
		0	Clock falling edge
		1	Clock rising edge
11 (R/W)	OPMODE	Operation Mode. The SPORT_CTL_B.OPMODE bit selects whether the half SPORT operates in DSP standard / multichannel mode or operates in I^2S / packed / left-justified mode. The mode selection affects the operation of the SPORT_CTL_B.LAFS and SPORT_CTL_B.LFS bits. Also, the SPORT_CTL_B.OPMODE bit enables or disables operation of the SPORT_CTL_B.GCLKEN, SPORT_CTL_B.FSED, SPORT_CTL_B.RJUST, SPORT_CTL_B.DIFS, SPORT_CTL_B.FSR, and SPORT_CTL_B.CKRE bits.	
		0	DSP standard/multichannel mode
		1	I^2S/packed/left-justified mode
10 (R/W)	ICLK	Internal Clock. When the half SPORT is in DSP standard mode (SPORT_CTL_B.OPMODE =0), the SPORT_CTL_B.ICLK bit selects whether the half SPORT uses an internal or external clock. For internal clock enabled, the half SPORT generates the SPORT_BCLK clock signal, and the SPORT_BCLK is an output. The SPORT_DIV_B.CLKDIV serial clock divisor value determines the clock frequency. For internal clock disabled, the SPORT_BCLK clock signal is an input, and the serial clock divisor is ignored. Note that the externally-generated serial clock does not need to be synchronous with the processor's system clock.	
		0	External clock
		1	Internal clock

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	PACK	Packing Enable. The SPORT_CTL_B.PACK bit enables the half SPORT to perform 16- to 32-bit packing on received data and to perform 32- to 16-bit unpacking on transmitted data. The receive packing operation packs two successive received words into a single 32-bit word. The transmit unpacking operation unpacks each 32-bit word and transmits it as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15:0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31:16. This format applies to both receive (packing) and transmit (unpacking) operations. Companding may be used with word packing or unpacking. The half SPORT generates data transfer related interrupts when packing is enabled. The transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.	
		0	Disable
		1	Enable
8:4 (R/W)	SLEN	Serial Word Length. The SPORT_CTL_B.SLEN bits selects word length in bits for the half SPORT's data transfers. Word may be from 4- to 32-bits in length. The formula for selecting the word length in bits is: SPORT_CTL_B.SLEN = (serial word length in bits) - 1 For DSP standard mode (SPORT_CTL_B.OPMODE =0), use SPORT_CTL_B.SLEN of 3 to 31 bits. For I^2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1), use SPORT_CTL_B.SLEN of 4 to 31 bits.	
3 (R/W)	LSBF	Least-Significant Bit First. The SPORT_CTL_B.LSBF bit selects whether the half SPORT transmits or receives data LSB first or MSB first.	
		0	MSB first sent/received (big endian)
		1	LSB first sent/received (little endian)
2:1 (R/W)	DTYPE	Data Type. The SPORT_CTL_B.DTYPE bits selects the data type formatting for the half SPORT's data transfers in DSP standard mode (SPORT_CTL_B.OPMODE =0).	
		0	Right-justify data, zero-fill unused MSBs
		1	Right-justify data, sign-extend unused MSBs
		2	u-law compand data
		3	A-law compand data

Table 30-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	SPENPRI	Serial Port Enable (Primary). The `SPORT_CTL_B.SPENPRI` bit enables the half SPORT's primary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Half SPORT 'A' Divisor Register

The SPORT_DIV_A register contains divisor values that determine frequencies of internally-generated clocks and frame syncs for half SPORT 'A'.

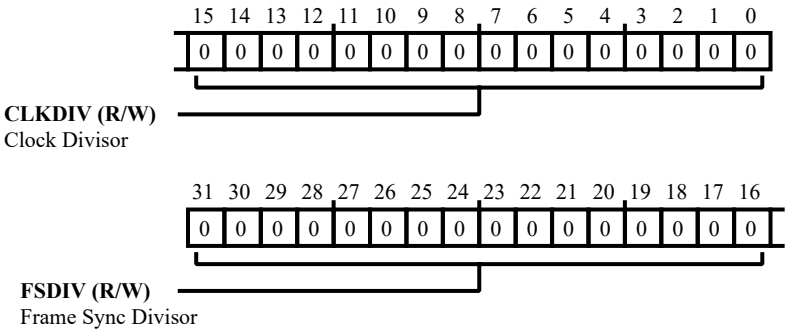


Figure 30-26: SPORT_DIV_A Register Diagram

Table 30-27: SPORT_DIV_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	FSDIV	Frame Sync Divisor. The SPORT_DIV_A.FSDIV bits select the number of transmit or receive clock cycles that the half SPORT counts before generating a frame sync pulse. The half SPORT counts serial clock cycles whether these are from an internally- or an externally-generated serial clock. The formula relating SPORT_DIV_A.FSDIV to the number of cycles between frame sync pulses is: SPORT_DIV_A.FSDIV = (number of serial clocks between frame syncs) - 1 Use the following equation to determine the value of SPORT_DIV_A.FSDIV, given the serial clock frequency and desired frame sync frequency: FSDIV = (SPORT_ACLK / SPORT_AFS) - 1 Note that the frame sync is continuously active when SPORT_DIV_A.FSDIV = 0. The value of SPORT_DIV_A.FSDIV should not be less than the serial word length (SPORT_CTL_A.SLEN), as this may cause an external device to abort the current operation or cause other unpredictable results.

Table 30-27: SPORT_DIV_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	CLKDIV	Clock Divisor. The SPORT_DIV_A.CLKDIV bits select the divisor that the half SPORT uses to calculate the serial clock (SPORT_ACLK) from the processor system clock (SCLK). The divisor is a 16-bit value, allowing a wide range of serial clock rates. When configured for internal clock (SPORT_CTL_A.ICLK =1), legal SPORT_DIV_A.CLKDIV values are 0 to 65535. Given the processor system clock frequency and desired serial clock frequency, use the following formula to calculate the value of SPORT_DIV_A.CLKDIV: CLKDIV = (SCLK / SPORT_ACLK) - 1 For the maximum serial clock frequency, see the processor data sheet.

Half SPORT 'B' Divisor Register

The SPORT_DIV_B contains divisor values that determine frequencies of internally-generated clocks and frame syncs for SPORT half 'B'.

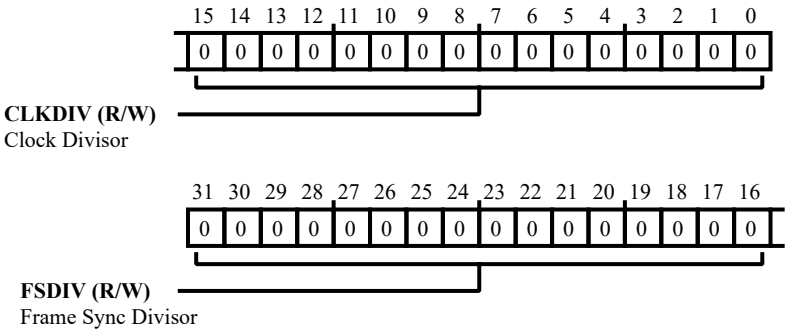


Figure 30-27: SPORT_DIV_B Register Diagram

Table 30-28: SPORT_DIV_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	FSDIV	Frame Sync Divisor. The SPORT_DIV_B.FSDIV bits select the number of transmit or receive clock cycles that the half SPORT counts before generating a frame sync pulse. The half SPORT counts serial clock cycles whether these are from an internally- or an externally-generated serial clock. The formula relating SPORT_DIV_B.FSDIV to the number of cycles between frame sync pulses is: SPORT_DIV_B.FSDIV = (number of serial clocks between frame syncs) - 1 Use the following equation to determine the value of SPORT_DIV_B.FSDIV, given the serial clock frequency and desired frame sync frequency: FSDIV = (SPORT_BCLK / SPORT_BFS) - 1 Note that the frame sync is continuously active when SPORT_DIV_B.FSDIV = 0. The value of SPORT_DIV_B.FSDIV should not be less than the serial word length (SPORT_CTL_B.SLEN), as this may cause an external device to abort the current operation or cause other unpredictable results.

Table 30-28: SPORT_DIV_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	CLKDIV	Clock Divisor. The SPORT_DIV_B.CLKDIV bits select the divisor that the half SPORT uses to calculate the serial clock (SPORT_BCLK) from the processor system clock (SCLK). The divisor is a 16-bit value, allowing a wide range of serial clock rates. When configured for internal clock (SPORT_CTL_B.ICLK =1), legal SPORT_DIV_B.CLKDIV values are 0 to 65535. Given the processor system clock frequency and desired serial clock frequency, use the following formula to calculate the value of SPORT_DIV_B.CLKDIV: CLKDIV = (SCLK / SPORT_BCLK) - 1 For the maximum serial clock frequency, see the processor data sheet.

Half SPORT 'A' Error Register

The SPORT_ERR_A register contains error status and error interrupt mask bits for SPORT half 'A', including error handling bits for the half SPORT's primary and secondary channels and frame sync. Detected errors are frame sync violations or buffer over/underflow conditions.

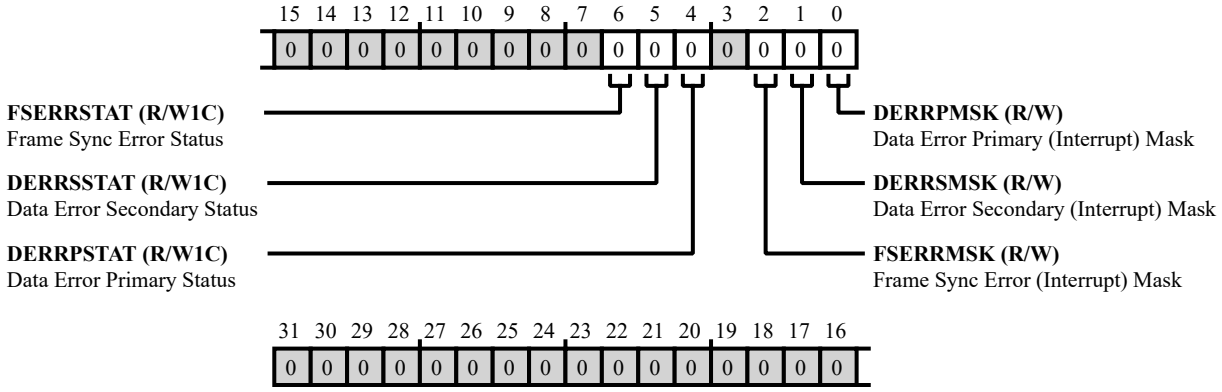


Figure 30-28: SPORT_ERR_A Register Diagram

Table 30-29: SPORT_ERR_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	FSERRSTAT	Frame Sync Error Status. The SPORT_ERR_A.FSERRSTAT bit indicates that the half SPORT has detected a frame sync when the bit count (bits remaining in the frame) is non-zero. When a half SPORT is receiving or transmitting, its bit count is set to a word length (for example, SPORT_CTL_A.SLEN = 31). After each serial clock edge, the half SPORT decrements the transfer's bit count. After the word is received or transmitted, the transfer's bit count reaches zero, and the half SPORT resets it (for example, to 32) on next frame sync. Normal SPORT data transfers always have a non-zero bit count value when active transmission or reception is occurring. Normal SPORT frame syncs occur after the bit count becomes zero.	
		0	No error
		1	Error (non-zero bit count at frame sync)

Table 30-29: SPORT_ERR_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1C)	DERRSSTAT	Data Error Secondary Status. The SPORT_ERR_A.DERRSSTAT bit indicates the error status for the half SPORT's secondary channel data buffers. During transmit (SPORT_CTL_A.SPTRAN =1), the SPORT_ERR_A.DERRSSTAT bit indicates the transmit underflow status. During receive (SPORT_CTL_A.SPTRAN =0), the SPORT_ERR_A.DERRSSTAT bit indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a secondary data error. This bit can also be used to clear the read-only SPORT_CTL_A.DERRSEC status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
4 (R/W1C)	DERRPSTAT	Data Error Primary Status. The SPORT_ERR_A.DERRPSTAT bit indicates the error status for the half SPORT's primary channel data buffers. During transmit (SPORT_CTL_A.SPTRAN =1), the SPORT_ERR_A.DERRPSTAT bit indicates the transmit underflow status. During receive (SPORT_CTL_A.SPTRAN =0), the SPORT_ERR_A.DERRPSTAT bit indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a primary data error. This bit can also be used to clear the read-only SPORT_CTL_A.DERRPRI status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
2 (R/W)	FSERRMSK	Frame Sync Error (Interrupt) Mask. The SPORT_ERR_A.FSERRMSK unmasks (enables) the half SPORT to generate the frame sync error interrupt request.	
		0	Mask (disable)
		1	Unmask (enable)
1 (R/W)	DERRSMSK	Data Error Secondary (Interrupt) Mask. The SPORT_ERR_A.DERRSMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the secondary channel.	
		0	Mask (disable)
		1	Unmask (enable)
0 (R/W)	DERRPMSK	Data Error Primary (Interrupt) Mask. The SPORT_ERR_A.DERRPMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the primary channel.	
		0	Mask (disable)
		1	Unmask (enable)

Half SPORT 'B' Error Register

The SPORT_ERR_B register contains error status and error interrupt mask bits for SPORT half 'B', including error handling bits for the half SPORT's primary and secondary channels and frame sync. Detected errors are frame sync violations or buffer over/underflow conditions.

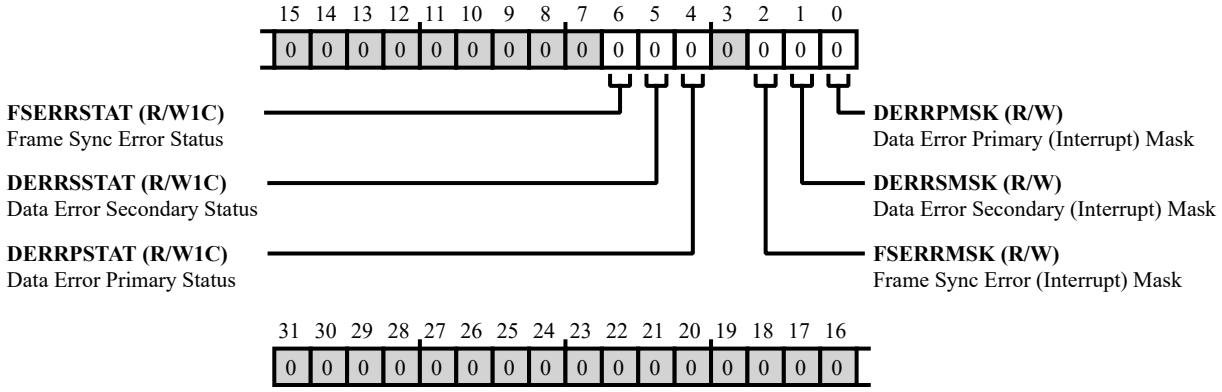


Figure 30-29: SPORT_ERR_B Register Diagram

Table 30-30: SPORT_ERR_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	FSERRSTAT	Frame Sync Error Status. The SPORT_ERR_B.FSERRSTAT bit indicates that the half SPORT has detected a frame sync when the bit count (bits remaining in the frame) is non-zero. When a half SPORT is receiving or transmitting, its bit count is set to a word length (for example, SPORT_CTL_B.SLEN = 31). After each serial clock edge, the half SPORT decrements the transfer's bit count. After the word is received or transmitted, the transfer's bit count reaches zero, and the half SPORT resets it (for example, to 32) on next frame sync. Normal SPORT data transfers always have a non-zero bit count value when active transmission or reception is occurring. Normal SPORT frame syncs occur after the bit count becomes zero.	
		0	No error
		1	Error (non-zero bit count at frame sync)

Table 30-30: SPORT_ERR_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1C)	DERRSSTAT	Data Error Secondary Status. The SPORT_ERR_B.DERRSSTAT bit indicates the error status for the half SPORT's secondary channel data buffers. During transmit (SPORT_CTL_B.SPTRAN =1), SPORT_ERR_B.DERRSSTAT indicates the transmit underflow status. During receive (SPORT_CTL_B.SPTRAN =0), SPORT_ERR_B.DERRSSTAT indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a secondary data error. This bit can also be used to clear the read-only SPORT_CTL_B.DERRSEC status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
4 (R/W1C)	DERRPSTAT	Data Error Primary Status. The SPORT_ERR_B.DERRPSTAT bit indicates the error status for the half SPORT's primary channel data buffers. During transmit (SPORT_CTL_B.SPTRAN =1), the SPORT_ERR_B.DERRPSTAT bit indicates the transmit underflow status. During receive (SPORT_CTL_B.SPTRAN =0), the SPORT_ERR_B.DERRPSTAT bit indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a primary data error. This bit can also be used to clear the read-only SPORT_CTL_B.DERRPRI status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
2 (R/W)	FSERRMSK	Frame Sync Error (Interrupt) Mask. The SPORT_ERR_B.FSERRMSK unmasks (enables) the half SPORT to generate the frame sync error interrupt request.	
		0	Mask (disable)
		1	Unmask (enable)
1 (R/W)	DERRSMSK	Data Error Secondary (Interrupt) Mask. The SPORT_ERR_B.DERRSMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the secondary channel.	
		0	Mask (disable)
		1	Unmask (enable)
0 (R/W)	DERRPMSK	Data Error Primary (Interrupt) Mask. The SPORT_ERR_B.DERRPMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the primary channel.	
		0	Mask (disable)
		1	Unmask (enable)

Half SPORT 'A' Multichannel Control Register

The SPORT_MCTL_A register controls the half SPORT's multichannel operations. This register enables multichannel operation, enables multichannel data packing, selects the multichannel frame delay, selects the number of multichannel slots, and selects the multichannel window offset size.

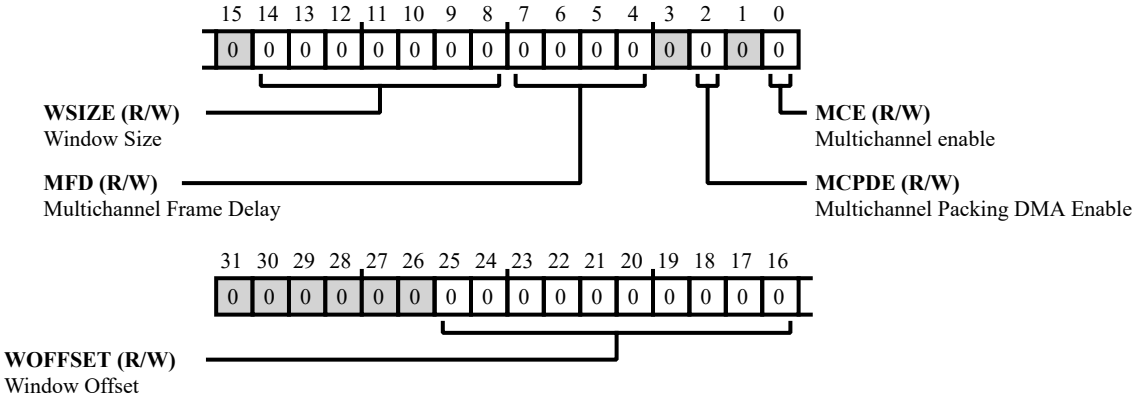


Figure 30-30: SPORT_MCTL_A Register Diagram

Table 30-31: SPORT_MCTL_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/W)	WOFFSET	Window Offset. The SPORT_MCTL_A.WOFFSET bits select the start location for the half SPORT's active window of channels within the 1024-channel range. A value of 0 specifies no offset and 896 is the largest value that permits using all 128 channels. When multichannel mode is disabled (SPORT_MCTL_A.MCE =0) and the right-justified mode is enabled (SPORT_CTL_A.RJUST =1), the least significant 6 bits of SPORT_MCTL_A.WOFFSET serve as the delay count (DCNT) field. These bits introduce a clock delay before the half SPORT state machine starts to capture data. For information about appropriate delay selections, see the SPORT operating modes section.
14:8 (R/W)	WSIZE	Window Size. The SPORT_MCTL_A.WSIZE bits select the window size for the half SPORT's active window of channels. Use the following formula to calculate the window size value: SPORT_MCTL_A.WSIZE = (number of channel slots) -1
7:4 (R/W)	MFD	Multichannel Frame Delay. The SPORT_MCTL_A.MFD bits select the delay (in serial clock cycles) between the half SPORT's multichannel frame sync pulse and channel 0. The 4-bit field allows selecting multichannel frame delay of 0-15 serial clocks.

Table 30-31: SPORT_MCTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	MCPDE	Multichannel Packing DMA Enable. The SPORT_MCTL_A.MCPDE bit enables DMA data packing for transmit and enables DMA data unpacking for the half SPORT's multichannel data transfers.	
		0	Disable
		1	Enable
0 (R/W)	MCE	Multichannel enable. The SPORT_MCTL_A.MCE bit enables multichannel operations for the half SPORT. The half SPORT is configured in normal multichannel mode if SPORT_CTL_A.OPMODE=0; while it is configured in packed mode if SPORT_CTL_A.OPMODE=1. When configuring in these modes, the multi-channel enable bit (SPORT_MCTL_A.MCE) should be set before enabling the SPORT data channel enable bits (SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC). When these channel bits transition from 1 to 0, note that the half SPORT's data transfer buffers are cleared, and the SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC bits are cleared.	
		0	Disable
		1	Enable

Half SPORT 'B' Multichannel Control Register

The SPORT_MCTL_B register controls the half SPORT's multichannel operations. This register enables multichannel operation, enables multichannel data packing, selects the multichannel frame delay, selects the number of multichannel slots, and selects the multichannel window offset size.

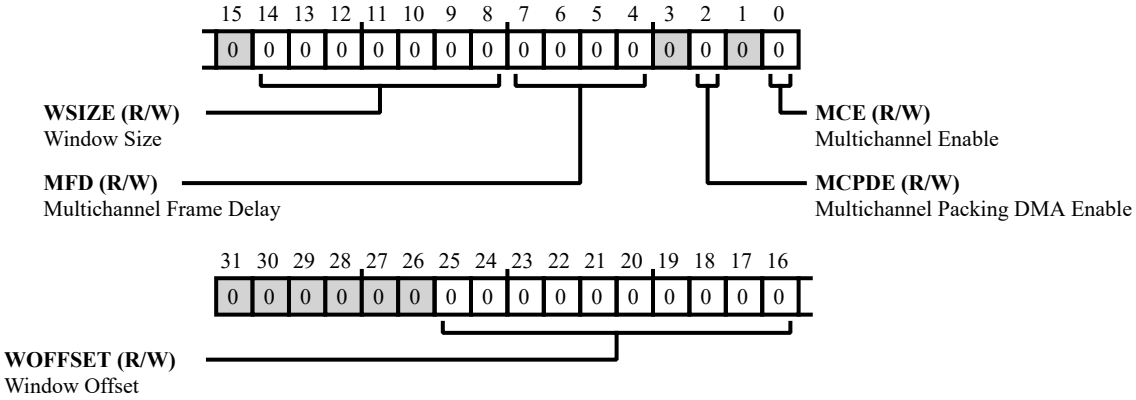


Figure 30-31: SPORT_MCTL_B Register Diagram

Table 30-32: SPORT_MCTL_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/W)	WOFFSET	Window Offset. The SPORT_MCTL_B.WOFFSET bits select the start location for the half SPORT's active window of channels within the 1024-channel range. A value of 0 specifies no offset and 896 is the largest value that permits using all 128 channels. When multichannel mode is disabled (SPORT_MCTL_B.MCE =0) and right-justified mode is enabled (SPORT_CTL_B.RJUST =1), the least significant 6 bits of SPORT_MCTL_B.WOFFSET serve as the delay count (DCNT) field. These bits introduce a clock delay before the half SPORT state machine starts to capture data. For information about appropriate delay selections, see the SPORT operating modes section.
14:8 (R/W)	WSIZE	Window Size. The SPORT_MCTL_B.WSIZE bits select the window size for the half SPORT's active window of channels. Use the following formula to calculate the window size value: SPORT_MCTL_B.WSIZE = (number of channel slots) -1
7:4 (R/W)	MFD	Multichannel Frame Delay. The SPORT_MCTL_B.MFD bits select the delay (in serial clock cycles) between the half SPORT's multichannel frame sync pulse and channel 0. The 4-bit field allows selecting a multichannel frame delay of 0-15 serial clocks.

Table 30-32: SPORT_MCTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	MCPDE	Multichannel Packing DMA Enable. The SPORT_MCTL_B.MCPDE bit enables DMA data packing for transmit and enables DMA data unpacking for the half SPORT's multichannel data transfers.	
		0	Disable
		1	Enable
0 (R/W)	MCE	Multichannel Enable. The SPORT_MCTL_B.MCE bit enables multichannel operations for the half SPORT. The half SPORT is configured in normal multichannel mode if SPORT_CTL_B.OPMODE=0; while it is configured in packed mode if SPORT_CTL_B.OPMODE=1. When configuring in these modes, the multichannel enable bit (SPORT_MCTL_B.MCE) should be set before enabling SPORT data channel enable bits (SPORT_CTL_B.SPENPRI and/or SPORT_CTL_B.SPENSEC). When these channel bits transition from 1 to 0, note that the half SPORT's data transfer buffers are cleared, and the SPORT_CTL_B.DERRPRI and SPORT_CTL_B.DERRSEC bits are cleared.	
		0	Disable
		1	Enable

Half SPORT 'A' Multichannel Status Register

The SPORT_MSTAT_A register indicates the current multichannel being serviced among the half SPORT's active channels in multichannel mode. The half SPORT increments the value by one in this register as each channel is serviced. The value in the SPORT_MSTAT_A register restarts at 0 at each frame sync.

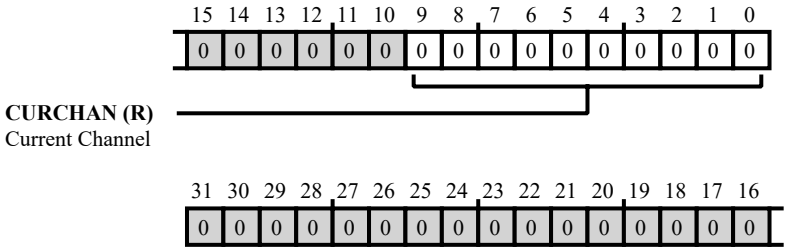


Figure 30-32: SPORT_MSTAT_A Register Diagram

Table 30-33: SPORT_MSTAT_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	CURCHAN	Current Channel. The SPORT_MSTAT_A.CURCHAN bits indicate the half SPORT's current channel being serviced in multichannel mode.

Half SPORT 'B' Multichannel Status Register

The SPORT_MSTAT_B register indicates the current multichannel being serviced among the half SPORT's active channels in multichannel mode. The half SPORT increments the value by one in this register as each channel is serviced. The value in the SPORT_MSTAT_B register restarts at 0 at each frame sync.

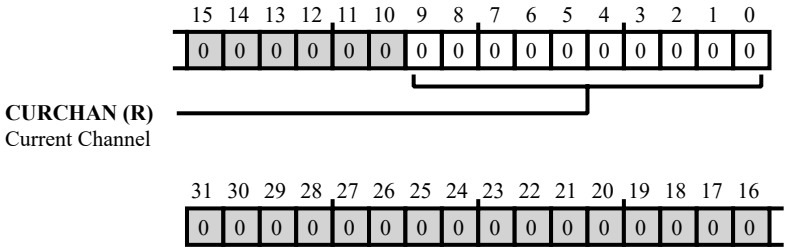


Figure 30-33: SPORT_MSTAT_B Register Diagram

Table 30-34: SPORT_MSTAT_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	CURCHAN	Current Channel. The SPORT_MSTAT_B.CURCHAN bits indicate the half SPORT's current channel being serviced in multichannel mode.

Half SPORT 'A' Rx Buffer (Primary) Register

The SPORT_RXPRI_A register buffers the half SPORT's primary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the primary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXPRI_A register. This data can be read in core mode (in interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXPRI_A register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

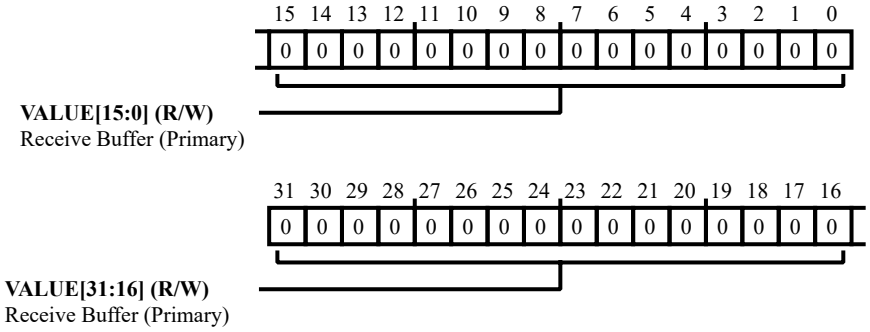


Figure 30-34: SPORT_RXPRI_A Register Diagram

Table 30-35: SPORT_RXPRI_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Primary). The SPORT_RXPRI_A.VALUE bits hold the half SPORT's primary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Rx Buffer (Primary) Register

The SPORT_RXPRI_B register buffers the half SPORT's primary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the primary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXPRI_B register. This data can be read in core mode (in interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXPRI_B register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

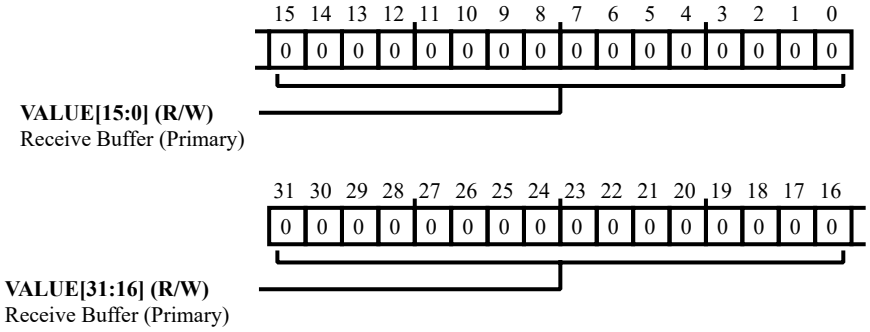


Figure 30-35: SPORT_RXPRI_B Register Diagram

Table 30-36: SPORT_RXPRI_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Primary). The SPORT_RXPRI_B.VALUE bits hold the half SPORT's primary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

Half SPORT 'A' Rx Buffer (Secondary) Register

The SPORT_RXSEC_A register buffers the half SPORT's secondary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the secondary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXSEC_A register. This data can be read in core mode (in interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXSEC_A register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

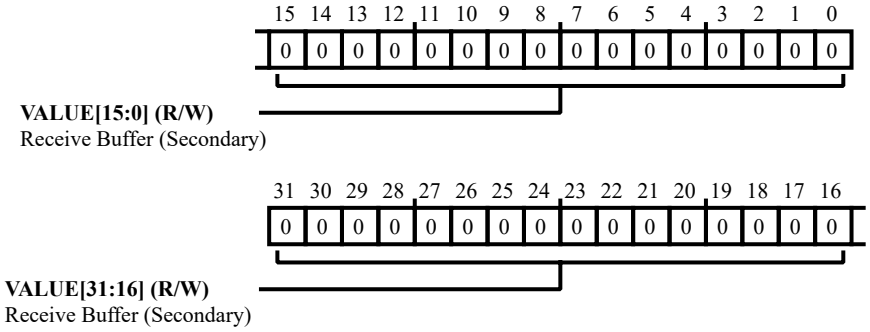


Figure 30-36: SPORT_RXSEC_A Register Diagram

Table 30-37: SPORT_RXSEC_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Secondary). The SPORT_RXSEC_A.VALUE bits hold the half SPORT's secondary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Rx Buffer (Secondary) Register

The SPORT_RXSEC_B register buffers the half SPORT's secondary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the secondary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXSEC_B register. This data can be read in core mode (in interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXSEC_B register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

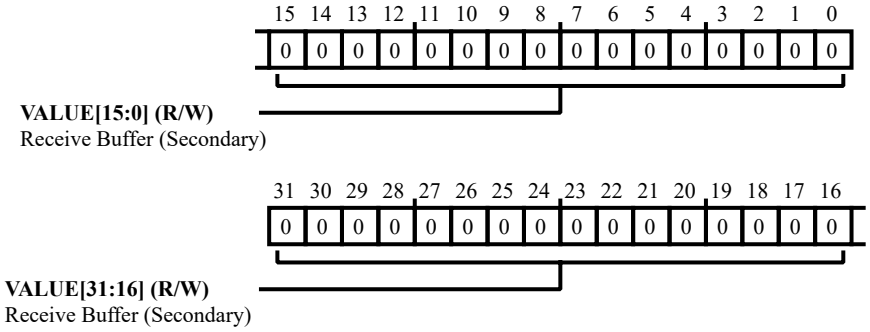


Figure 30-37: SPORT_RXSEC_B Register Diagram

Table 30-38: SPORT_RXSEC_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Secondary). The SPORT_RXSEC_B.VALUE bits hold the half SPORT's secondary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

Half SPORT 'A' Tx Buffer (Primary) Register

The SPORT_TXPRI_A register buffers the half SPORT's primary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the primary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXPRI_A register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_A.PACK =0); while it acts as a two-location buffer when packing is enabled (SPORT_CTL_A.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXPRI_A register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

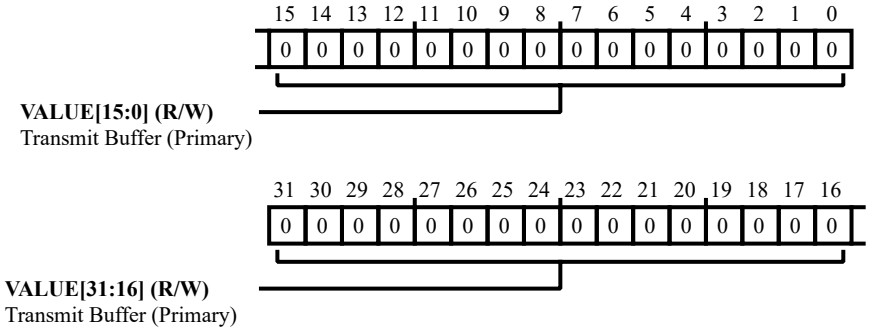


Figure 30-38: SPORT_TXPRI_A Register Diagram

Table 30-39: SPORT_TXPRI_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Primary). The SPORT_TXPRI_A.VALUE bits hold the half SPORT's primary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Tx Buffer (Primary) Register

The SPORT_TXPRI_B register buffers the half SPORT's primary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the primary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXPRI_B register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_B.PACK =0); while it acts as a two-location buffer when packing is enabled (SPORT_CTL_B.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXPRI_B register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

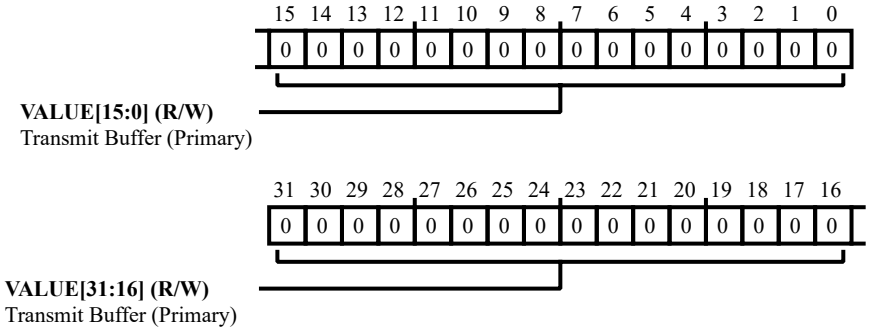


Figure 30-39: SPORT_TXPRI_B Register Diagram

Table 30-40: SPORT_TXPRI_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Primary). The SPORT_TXPRI_B.VALUE bits hold the half SPORT's primary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

Half SPORT 'A' Tx Buffer (Secondary) Register

The SPORT_TXSEC_A register buffers the half SPORT's secondary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the secondary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXSEC_A register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_A.PACK =0); while it acts as a two-location buffer when packing is enabled (SPORT_CTL_A.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXSEC_A register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

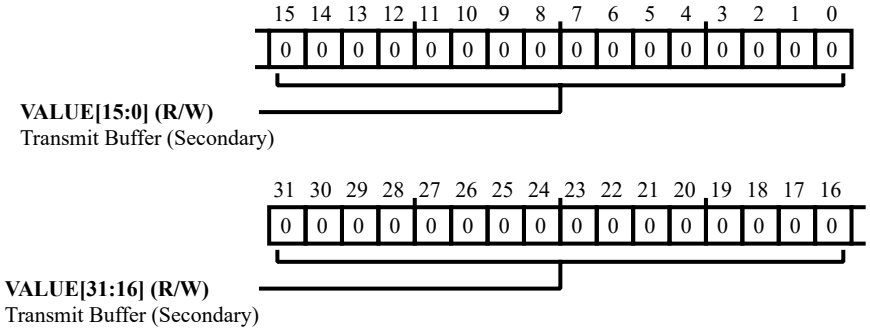


Figure 30-40: SPORT_TXSEC_A Register Diagram

Table 30-41: SPORT_TXSEC_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Secondary). The SPORT_TXSEC_A.VALUE bits hold the half SPORT's secondary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Tx Buffer (Secondary) Register

The SPORT_TXSEC_B register buffers the half SPORT's secondary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the secondary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXSEC_B register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_B.PACK =0); while it acts as two-location buffer when packing is enabled (SPORT_CTL_B.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXSEC_B register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

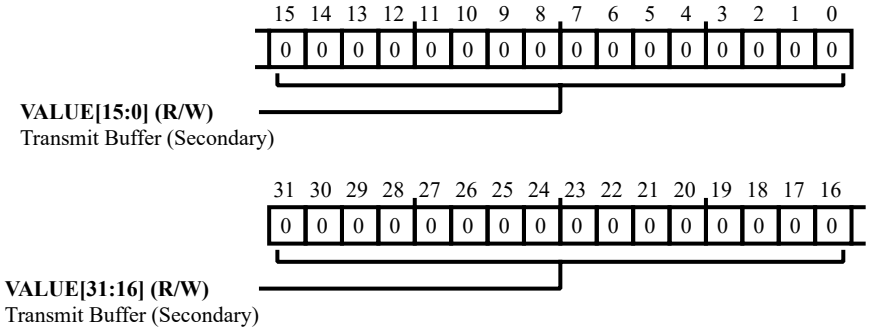


Figure 30-41: SPORT_TXSEC_B Register Diagram

Table 30-42: SPORT_TXSEC_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Secondary). The SPORT_TXSEC_B.VALUE bits hold the half SPORT's secondary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

31 Precision Clock Generators (PCG)

The precision clock generators are used to produce a pair of signals from a clock input signal. The two signals generated are normally used as a serial bit clock and frame sync pair. The PCG is part of the DAI. There are four PCG units per DAI. *PCGA*, *PCGB*, *PCGE*, and *PCGF* belong to DAI0; *PCGC*, *PCGD*, *PCGG*, and *PCGH* belong to DAI1.

Each of these four units (A, B, E, and F) generates one clock (CLKA_O, CLKB_O, CLKE_O, and CLKF_O) and one frame sync (FSA_O, FSB_O, FSE_O, and FSF_O) output. In addition to clock and frame sync outputs, each of these four units generates one inverted version of clock output (INV_CLKA_O, INV_CLKB_O, INV_CLKE_O, and INV_CLKF_O) and one inverted version of frame sync output (INV_FSA_O, INV_FSB_O, INV_FSE_O, and INV_FSF_O).

Features

The following list describes the features of the precision clock generators.

- SRU allows the routing of all of the PCG signals in one DAI (two PCG units in DAI)
- FracNPLL output clock that can be used as input clock for all PCGs.
- Input clock selection: SYS_CLKIN0 and SYS_CLKIN1 can be used as the input clock for PCG, when configured to use CLKIN as clock source
- Provides four different clock dividers for serial clock, frame sync, phase (20-bit), and pulse width (16-bit)
- Phase shift allows adjustment of the frame sync relative to the serial clock and can be shifted the full period and wrapped around
- Provides pulse width control for arbitrary frame sync signal generation
- Bypass mode for external frame sync manipulation
- External trigger mode starts PCG operation
- No additional jitter is introduced when using off-chip clocks
- Global enable. (See the DAI chapter for details)

Functional Description

The *PCG Block Diagram* shows the blocks within the module and its connection to the DAI. The following sections provide information on the function of these blocks.

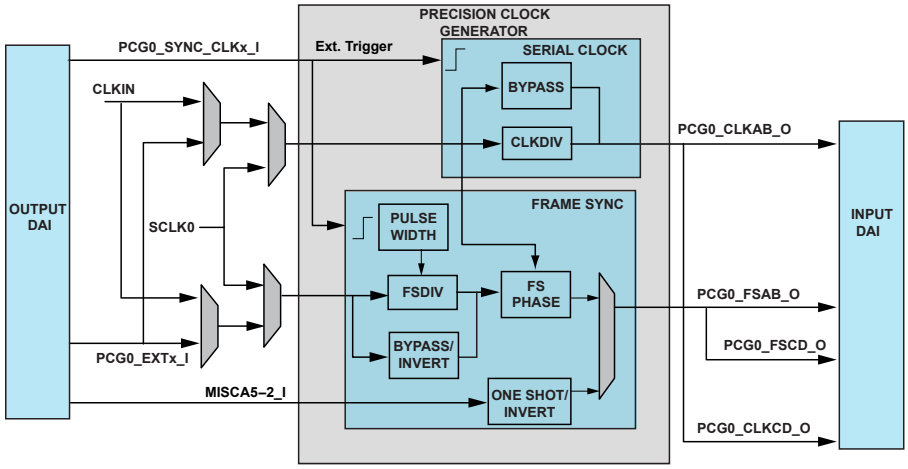


Figure 31-1: PCG Block Diagram

NOTE: In the *PCG Block Diagram*, the CLKINSEL bit field determines whether SYS_CLKIN0 or SYS_CLKIN1 is input to DAI.

ADSP-2184x PCG Register List

Precision Clock Generator

Table 31-1: ADSP-2184x PCG Register List

Name	Description
PCG_CTLA0	Precision Clock A Control 0 Register
PCG_CTLA1	Precision Clock A Control 1 Register
PCG_CTLB0	Precision Clock B Control 0 Register
PCG_CTLB1	Precision Clock B Control 1 Register
PCG_CTLC0	Precision Clock C Control 0 Register
PCG_CTLC1	Precision Clock C Control 1 Register
PCG_CTLD0	Precision Clock D Control 0 Register
PCG_CTLD1	Precision Clock D Control 1 Register
PCG_CTLE0	Precision Clock E Control 0 Register
PCG_CTLE1	Precision Clock E Control 1 Register
PCG_CTLF0	Precision Clock F Control 0 Register

Table 31-1: ADSP-2184x PCG Register List (Continued)

Name	Description
PCG_CTLF1	Precision Clock F Control 1 Register
PCG_CTLG0	Precision Clock G Control 0 Register
PCG_CTLG1	Precision Clock G Control 1 Register
PCG_CTLH0	Precision Clock H Control 0 Register
PCG_CTLH1	Precision Clock H Control 1 Register
PCG_PW1	Precision Clock Pulse Width Control 1 Register
PCG_PW2	Precision Clock Pulse Width Control 2 Register
PCG_PW3	Precision Clock Pulse Width Control 3 Register
PCG_PW4	Precision Clock Pulse Width Control 4 Register
PCG_SYNC1	Precision Clock Frame Sync Synchronization 1 Register
PCG_SYNC2	Precision Clock Frame Sync Synchronization 2 Register
PCG_SYNC3	Precision Clock Frame Sync Synchronization 3 Register
PCG_SYNC4	Precision Clock Frame Sync Synchronization 4 Register

ADSP-2184x PCG Trigger List

Table 31-2: ADSP-2184x PCG Trigger List Generators

Trigger ID	Name	Description	Sensitivity
100		Reserved	
101		Reserved	
102		Reserved	
103		Reserved	
104		Reserved	
105		Reserved	
106		Reserved	
107		Reserved	

Table 31-3: ADSP-2184x PCG Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
128	PCG0_HWA	PCG0 PCG-A Hardware trigger control	Pulse
129	PCG0_HWB	PCG0 PCG-B Hardware trigger control	Pulse
130	PCG0_HWC	PCG0 PCG-C Hardware trigger control	Pulse

Table 31-3: ADSP-2184x PCG Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
131	PCG0_HWD	PCG0 PCG-D Hardware trigger control	Pulse
132	PCG0_HWE	PCG0 PCG-E Hardware trigger control	Pulse
133	PCG0_HWF	PCG0 PCG-F Hardware trigger control	Pulse
134	PCG0_HWG	PCG0 PCG-G Hardware trigger control	Pulse
135	PCG0_HWH	PCG0 PCG-H Hardware trigger control	Pulse

Internal Interface

The fundamental clock of the PCG is *SCLK0*. The clock to this module can be shut off for power savings.

Serial Clock

Each of the four units (A, B, C, and D) produces a clock output. Serial clock generation from a unit is independently enabled and controlled. Sources for the serial clock generation can be either from the CLKIN, SCLK0, or a DAI pin source.

When CLKIN is chosen as input clock in PCG, the clock source bits (PCG_SYNC1.CLKA_CLKINSEL, PCG_SYNC1.CLKB_CLKINSEL) determine whether the clock source is SYS_CLKIN0 .

Note that the divider is working in normal mode for PCG0_CTLx1.CLKDIV > 1. For PCG_CTLA1.CLKDIV/ PCG_CTLB1.CLKDIV = 0 or 1, the divider operates in bypass mode (input clock is fed directly to its output). In bypass mode, the clock at the output can theoretically run at up to the SCLK0 frequency. Check the data sheet for the specified maximum operation speed of the DAI pin buffers.

Note that the clock output is always set (as closely as possible) to a 50% duty cycle. If the clock divisor is even, the duty cycle of the clock output is exactly 50%. If the clock divisor is odd, then the duty cycle is slightly less than 50%. The low period of the output clock is one input clock period more than the high period of the output clock. For higher values of an odd divisor, the duty cycle is close to 50%.

NOTE: A PCG clock output cannot be fed to its own input.

Frame Sync

The following sections describe the use of frame syncs in the PCGs.

Frame Sync Output

Each of the four units (A through D) produces a synchronization signal for framing serial data. The frame sync outputs are much more flexible since they must accommodate the wide variety of serial protocols used by peripherals.

Frame sync generation from a unit is independently enabled and controlled. Sources for the frame sync generation can be either from the crystal buffer output, SCLK0, or an external pin source. There is only one external source pin for both frame sync and clock output for a unit.

If an external source is selected for both frame sync and clock output for a unit, then they operate on the same input signal. Apart from enable and source select control bits, a 20-bit divisor controls frame sync generation.

Divider Mode Selection

If a frame sync divisor is greater than 1, the PCG frame sync output frequency is equal to the input clock frequency, divided by a 20-bit integer. This integer is specified in the `PCG_CTLA0.FSDIV`/`PCG_CTLB0.FSDIV` bit field (bits 19:0).

However, if the frame sync divisor is 0 or 1, the PCG's frame sync clock generation unit is bypassed, and the frame sync input is connected directly to the frame sync out put. For `PCG_CTLB0.FSDIV` =0, 1 the `PCG_PW1` - `PCG_PW4` registers function differently than in normal mode.

Phase Shift

Phase shift is a frame sync parameter that defines the phase shift of the frame sync relative to the input clock of the same unit. This feature allows the shifting of the frame sync signal in time relative to the clock input signal. Frame sync phase shifting is often required by peripherals that need a frame sync signal to lead or lag a clock signal.

For example, the I^2S protocol specifies that the frame sync transition from high-to-low occurs one clock cycle before the beginning of a frame. Since an I^2S frame is 64 clock cycles long, delaying the frame sync by 63 cycles produces the required framing.

Phase shifting is represented as a full 20-bit value. Even when the frame sync is divided by the maximum amount, the phase can be shifted to the full range, from zero to one input clock short of the period.

NOTE: Phase shifting is specified as a 2 x 10-bit divider value in the `PCG_CTLA0.FSPHASEHI` bit field (bits 29:20) and in the `PCG_CTLA1.FSPHASELO` bit field (bits 29:20).

A single 20-bit value spans these two-bit fields. The upper half of the word (bits 19:10) is in the `PCG_CTLA0` register, and the lower half (bits 9:0) is in the `PCG_CTLA1` register.

The phase shift between clock and frame sync outputs can be programmed using the `PCG_PW1` - `PCG_PW4` registers function registers and all of the control registers under these conditions:

- The input clock source for the clock generator output and the frame sync generator output is the same.
- The clock and frame sync are enabled at the same time using a single atomic instruction.
- The frame sync divisor is an integral multiple of the clock divisor.

NOTE: When using a clock and frame sync as a synchronous pair, the units must be enabled in a single atomic instruction before their parameters are modified. Both units must also be disabled in a single atomic instruction.

NOTE: If the phase shift is 0 (see the ***Phase and Pulse Width Settings*** figure), the clock and frame sync outputs rise at the same time.

If the phase shift is 1, the frame sync output transitions one input clock period ahead of the clock transition.

If the phase shift is *divisor* – 1, the frame sync transitions *divisor* – 1 input clock periods ahead of the clock transitions.

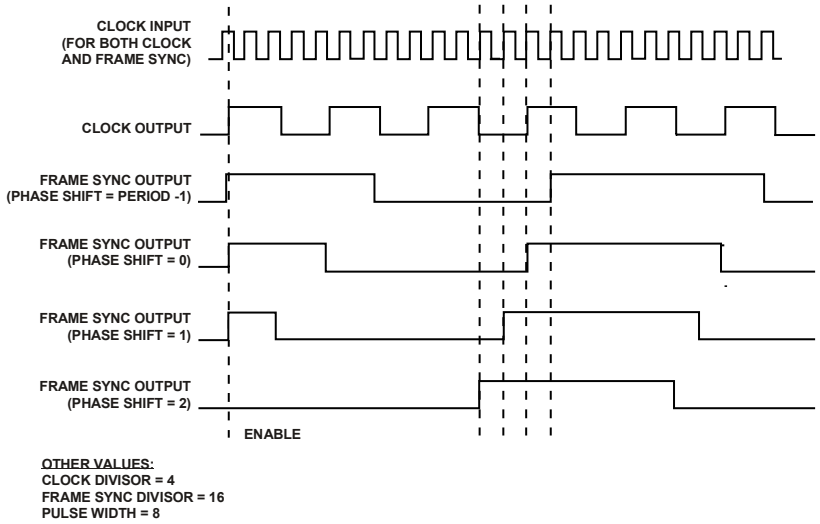


Figure 31-2: Phase and Pulse Width Settings

NOTE: When generating single frame sync pulses (the length of one SPORT clock cycle), take care with respect to the drive and sampling edges. If the rules are violated, for example if the SPORT is not driving data, the module cannot detect a valid sample edge.

Pulse Width

Pulse width is the number of input clock periods for which the frame sync output is high.

A 16-bit value determines the width of the framing pulse. Settings for pulse width can range from zero to `DIV` - 1. The pulse width should be less than the divisor of the frame sync. The pulse width of frame sync is specified in the `PCG_PW1.FSA`, `PCG_PW1.FSB`, `PCG_PW2.FSC`, and `PCG_PW2.FSD` bits (15-0) and (31-16). .

Default Pulse Width

If the pulse width count is equal to 0 and if the `PCG_CTLA0.FSDIV`/`PCG_CTLB0.FSDIV` bit field is even, then the actual pulse width of the frame sync output is equal to:

For even divisors: frame sync divisor/2

If the pulse width count is equal to 0 and if the `PCG_CTLA0.FSDIV`/`PCG_CTLB0.FSDIV` bit field is odd, then the actual pulse width of the frame sync output is equal to:

For odd divisors: frame sync divisor – 1/2

Input Clock Source Considerations

The core Phase-Locked Loop (PLL) has been designed to provide clocking for the processor core. The performance specifications of this PLL are appropriate for the core. But they have not been optimized or specified for precision data converters where jitter directly translates into time quantization errors and distortion.

The PCG allows the routing of external clock sources which are independent of the core PLL.

Timing Example for I²S Mode

For I²S mode, the frame sync should be driven at the falling edge of SPORT clock. In other words, the frame sync edge must coincide with the falling edge of the SPORT clock. To satisfy this requirement, program the phase of the frame sync accordingly in the PCG control registers.

For example, assume that the input clock source for both clock and frame sync are the same and both the clock and frame sync are enabled at the same time. Also, assume that the clock divisor value for generating the required SPORT clock is `PCG_CTLA1.CLKDIV` = 4. Then, for a 32-bit word length, the frame sync divisor value is:

`PCG_CTLA0.FSDIV` = 256, `PCG_CTLA1.CLKDIV` = 4.

By default, for phase = 0, the rising edge of both SPORT clock and frame sync coincide. To make sure that the frame sync edges coincide with the falling edge of the SPORT clock, program the phase value as:

`PCG_CTLA1.CLKDIV/2` = 2.

Cross Mode Connections

The symmetric dual DAI architecture allows cross connections between both PCGs (A,B) and (C,D) to the other DAI. Each PCG (A through D) supports an alternative input clock (PCG0_EXTx_I)(see Figure 31-1 PCG Block Diagram) which can be sourced via a DAI pin buffer from the other DAI. Note however if routing a source (clock or FS) only DAI pin buffer 2 to 20 can be used (DAI pin buffer 1 is no longer available and is replaced by the DAI CRS buffer for the other DAI).

See DAI Routing Capabilities for more information.

Operating Modes

The following sections provide information on the operating modes of the precision clock generator.

Normal Mode

When the frame sync divisor is set to any value other than zero or one, the PCGs operate in normal mode. In normal mode, the divisor determines the frequency of the frame sync output where:

Frequency of Frame Sync Output = Input Frequency/Divisor

The value of the pulse width control determines the high period of the frame sync output. The value of the pulse width control must be less than the value of the divisor.

The value of the phase control determines the phase of the frame sync output. When the phase is zero, then the positive edges of the clock and frame sync coincide when:

- the clock and frame sync dividers are enabled at the same time using an atomic instruction
- the divisors of the clock and frame sync are the same
- the source for the clock and frame sync is the same

The number of input clock cycles that have already elapsed before the frame sync is enabled is equal to the difference between the divisor and the phase values. When the phase is a small fraction of the divisor, then the frame sync leads the clock. When the phase is only slightly less than the frame sync divisor, then the frame sync lags the clock. The frame sync phase must not be greater than the divisor.

Bypass Mode

When the frame sync divisor for the frame sync has a value of zero or one, the frame sync is in bypass mode, and the `PCG_PW1` - `PCG_PW4` registers are registers have different functionality than in normal mode.

NOTE: In normal mode, bits 15:0 and 31:16 of the `PCG_PW1` - `PCG_PW4` registers are used to program the pulse width count. In bypass mode, bits 15:2 and 31:18 are ignored. Bits 1:0 and 17:16 are renamed to `PCG_PW1.STROBEA` and `PCG_PW1.INVFSA`, respectively. This functionality is described in more detail as follows.

If the `PCG_PW1.STROBEA` register is cleared, then the input is directly passed (see the ***Bypass and Inverted Bypass*** figure) to the frame sync output either inverted or not inverted, depending on the `PCG_PW1.INVFSA` bits.

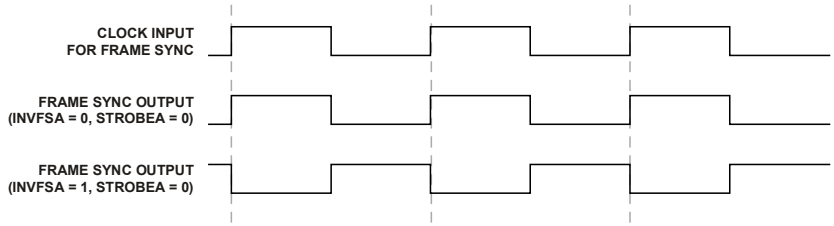


Figure 31-3: Bypass and Inverted Bypass

One-Shot Mode

In the one-shot mode operation shown in the ***One Shot Mode PCG A (MISCA2_I Input)*** figure, the PCG produces a series of periods but does not run continuously.

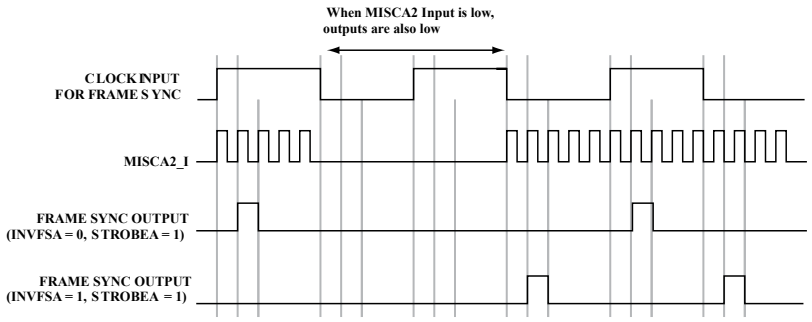


Figure 31-4: One Shot Mode PCG A (MISCA2_I Input)

Bypass mode also enables the generation of a strobe pulse (one-shot frame sync). Strobe usage ignores the divider counters and looks to the SRU to provide the input signal.

In the bypass mode, if the `PCG_PW1.STROBEA` bit =1, then a one-shot pulse is generated. This one-shot pulse has the duration equal to the period of `DAI_MISCAx_I` for the PCGx unit. This pulse is generated either at the falling or rising edge of the input clock, depending on the value of the `PCG_PW1.INVFSA` bit. The output pulse width is equal to the period of the SRU source signal `DAI_MISCAx_I`. The pulse begins at the second rising edge of `MISCAx_I` following a rising edge of the clock input. When the `PCG_PW1.INVFSA` bit is set, the pulse begins at the second rising edge of `DAI_MISCAx_I` coinciding with or following a falling edge of the clock input.

NOTE: A strobe period is defined to be the period of the FS input clock signal as specified by the `PCG_CTLA1.FSSRC` bit.

Audio System Example

The ***PCG Setup for I2S or Left-Justified DAI*** figure shows an example of the interconnections between the S/PDIF receiver, ASRC, and the PCGs. The interconnections are made by programming the signal routing unit. It shows how to set up two precision clock generators using the S/PDIF receiver and an asynchronous sample rate converter (ASRC) to interface to an external audio DAC. The PCG is configured to provide a fixed ASRC/DAC output sample rate of 65.098 kHz. The input to the S/PDIF receiver is typically 44.1 kHz if supplied by a CD player, but can also be from other source at any nominal sample rates.

Similarly, the phase shift for frame syncs B, C, and D is specified in the corresponding control registers (`PCG_CTLA0` through `PCG_CTLB1`).

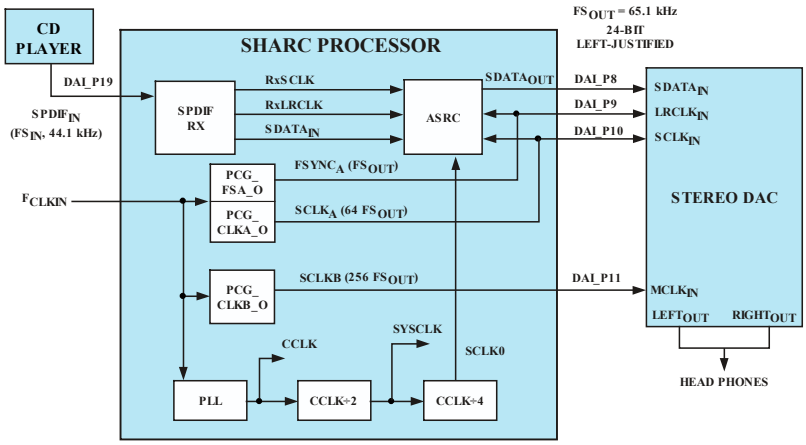


Figure 31-5: PCG Setup for I²S or Left-Justified DAI

Three synchronous clocks are required in audio systems:

1. Frame sync (FS)
2. Serial bit clock (64 FS)
3. Controller DAC clock (256 FS)

Since each PCG has only two outputs, this example requires two PCGs. Furthermore, because the digital audio interface requires a fixed-phase relation between serial clock and FS, these two outputs should come from one PCG (PCG A). The controller clock comes from the second (PCG B).

The combined PCGs can provide a selection of synchronous clock frequencies to support alternate sample rates for the ASRCs and external DACs. However, the range of choices is limited by CLKIN and the ratio of `PCG_CLKx_O`: serial clock:*FS*. The ratio is normally fixed at 256:64:1 to support digital audio left-justified, I²S and right-justified interface modes.

Many DACs also support 384, 512, and 786x FS for `PCG_CLKx_O`, which allows some additional flexibility in choosing serial clock.

Note that the falling edge of serial clock must always be synchronous with both edges of FS. This condition requires that the phase of the serial clock and FS signals for a common PCG (PCG A) be adjustable.

While the frequency of the controller DAC clock (`PCG_CLKx_O`) must be synchronous with the sample rate supplied to the external DAC, there is no fixed-phase requirement.

Configure the clock divisor and source and low-phase word first, followed by the control register enable bits, which are set together. When the `PCG_PW1` - `PCG_PW4` registers are configured to zero (default), the FS pulse width is (divisor /2) for even divisors and (divisor-1)/2 for odd divisors. Alternatively, the `PCG_PW1`-`PCG_PW4` registers can be set high for exactly one-half the period of CLKIN cycles for a 50% duty cycle, provided the FS divisor is an even number.

Hardware Trigger Control

In addition to the existing enable control to generate the clock and frame sync, this feature provides a flexibility to start the clock and frame sync generation upon receiving a hardware trigger (internal completer trigger). This feature applies to all modes of operation.

For example, when the PCG_SYNC1.HWB_TRIGEN bit is set, it enables the synchronization of clock A and frame sync A to the external LRCLK sync (in external LRCLK synchronization mode) upon receiving a hardware trigger. Refer to the PCG_SYNC1.HWA_TRIGEN/ PCG_SYNC1.HWB_TRIGEN and PCG_SYNC2.HWC_TRIGEN/ PCG_SYNC2.HWD_TRIGEN bit descriptions.

NOTE: To exercise the hardware trigger based synchronization, program the destination registers (DAI_MISC2.IN0, DAI_MISC2.IN1 DAI_MISC2.IN2, and DAI_MISC2.IN3) to route the trigger signals to the PCGs . For example, if PCGA must be enabled for the hardware trigger based synchronization, program the DAI_MISC2.IN0 and DAI_EXTD_MISC2.IN0 bit fields to 5'b00000 and 2'b01, respectively. Each PCG has a dedicated trigger control from the TRU unit, there are four trigger controls for DAI0 and four trigger controls for DAI1.

Refer to TRU Completer Trigger list table for the completer trigger information.

Clock Configuration Examples

For a CLKIN = 33.330 MHz, the two PCGs provide the three synchronous clocks PCGx_CLK, serial clock and *FS* for the SRCs and external DAC. These divisors are stored in the PCG_CTLA1.CLKDIV/PCG_CTLB1.CLKDIV bit fields.

The integer divisors for several sample rates based on 33.330 MHz *CLKIN* are shown in the ***Precision Clock Generator Division Ratios*** table.

Table 31-4: Precision Clock Generator Division Ratios (33.330 CLKIN)

Sample Rate kHz)	PCG Divisors		FSDIV A*[1]
	CLKDIV B	CLKDIV A	
130.195	1	4	256
65.098	***2***	***8***	***512***
43.398	3	12	768
32.549	4	16	1024
26.039	5	20	1280
21.699	6	24	1536
18.599	7	28	1792

*1 The frame sync divisor must be an even integer to produce a 50% duty cycle waveform. See Frame Sync.

Global PCG Enable

For some applications, enabling multiple PCGs (CLK and FS) must be synchronized. This feature provides the flexibility to enable multiple PCGs (CLK and FS) globally. The global PCG enable feature works independently, while the legacy PCG enable (CLK and FS) remains the same. The legacy PCG enable can be used when the PCGs need to be enabled individually.

The DAI_GBL_PCG_EN register controls the enabling and disabling of PCGs globally. The global enable feature can be used in the following scenarios:

- to enable all the eight PCGs (includes both DAI0 and DAI1) at the same time
- to enable all four PCGs within each DAIx at the same time
- to enable multiple PCGs (across DAIs) at the same time

NOTE: See the DAI_GBL_PCG_EN register description for details.

Programming Guidelines

Complete the following steps to globally enable all of the PCGs :

1. Clear the individual (local) PCG enable bits from PCG_CTLx registers. (for example, PCG_CTLA0.CLKEN and PCG_CTLA0.FSEN bits) without modifying any other settings.
2. Wait for n CCLK cycles; n = PCG source clock period/processor clock period.
3. Program the PCG registers (PCG_CTLx/PCG_SYNC1) with the desired configuration without setting the PCG enable bits.
4. Enable all the required PCGs in the DAI_GBL_PCG_EN register that need to be enabled globally.

Complete the following steps to globally disable the PCGs using the DAI_GBL_PCG_EN register:

1. Disable individual PCG bits to disable a particular PCG.
2. Disable DAI0/1 enable bits to disable PCGs within the DAI.
3. Disable the global PCG enable bit in cases where all PCGs must be disabled.

PCG Event Control

The following sections describe the generation and control of PCG events.

External Event Trigger

The trigger with the external clock is enabled by setting bits 0 and 16 of the PCG_SYNC1/PCG_SYNC2/PCG_SYNC3/PCG_SYNC4 registers .

Refer to the ***FS Output Synchronization With External Trigger Input*** figure. Since the rising edge of the external clock is used to synchronize with the frame sync, the frame sync output is not generated until a rising edge of the external clock is sensed.

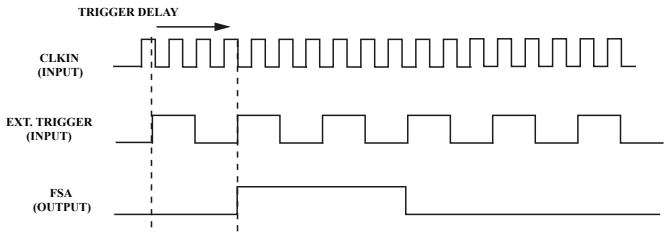


Figure 31-6: FS Output Synchronization With External Trigger Input

External Event Trigger Delay

The time delay between the rising trigger edge and the start of serial clock and frame sync varies between 2.5 to 3.5 input clock periods. If the input clock and the trigger signal are synchronous, the delay is 3 input clock periods. Consider the following cases:

- SCLK0 is the input source. In this case, if the given trigger event is synchronous to SCLK0, the delay is 3 SCLK0 periods. If the trigger signal is asynchronous with SCLK0, the delay varies from 2.5 SCLK0 periods to 3.5 SCLK0 periods. (It depends on whether the trigger edge occurs in the positive half cycle or negative half cycle of SCLK0.)
- The CLKIN source bits of the `PCG_SYNC1` register select the input source for PCG. In this case, if the given trigger signal is synchronous to CLKIN, the delay is 3 CLKIN periods. But if they are asynchronous to CLKIN, the delay can vary between 2.5 CLKIN periods to 3.5 CLKIN periods.
- SRU DAI0 is the input source for PCG A, B. If the input clock and trigger signal are synchronous, the delay is exactly 3 input clock periods. If asynchronous, it varies between 2.5 to 3.5 input clock periods depending on the phase difference between the input clock and trigger signal.

Programming Model

This section describes the sequence of software steps required for successful PCG operation.

If the PCG is disabled to reprogram a parameter, use a delay after writing to the disable bit. This delay in core clock (*CCLK*) cycles = (PCG source clock period/*CCLK* period). In summary, use the following general procedure:

1. Clear the PCG enable bits without modifying any other settings.
2. Wait for N *CCLK* cycles (N = PCG source clock period/processor clock period).
3. Program all new parameters without setting the PCG enable bit.
4. Enable the PCG.

Frame Sync Phase Setting

The phase unit requires that the clock and FS are enabled simultaneously in an atomic instruction.

1. Write the clock divider/low 10-bit phase divider to the PCG_CTLA1/PCG_CTLB1 registers.
2. Program the FS divider/high 10-bit phase divider, enabling both the PCG_CTLA0.CLKEN/PCG_CTLA0.FSEN and the PCG_CTLB0.CLKEN/PCG_CTLB0.FSEN bits.

Note that both units must be disabled in the same way.

External Event Trigger

The trigger with the external clock is is enabled by setting bits 0 and 16 of the PCG_SYNC register. Program the phase to 3, so that the rising edge of the external clock is in-sync with the frame sync (***FS Output Synchronization With External Trigger Input***).

Use the following steps.

1. Program the PCG_SYNC1 - PCG_SYNC4 registers and the PCG_CTLA0 through PCG_CTLB1 registers appropriately.
2. Enable the clock or frame sync, or both.

Since the rising edge of the external clock is used to synchronize with the frame sync, the frame sync output is not generated until a rising edge of the external clock is sensed.

Debug Features

Take care in cases where any input to the phase unit is modified. Any individual change of the PCG_CTLA1.CLKDIV or PCG_CTLA0.FSDIV dividers can cause a failure in the PCG sync operation between the serial clock and the frame sync. Only the programming model ensures a correct setup for phase settings.

ADSP-2184x PCG Register Descriptions

Precision Clock Generator (PCG) contains the following registers.

Table 31-5: ADSP-2184x PCG Register List

Name	Description
PCG_CTLA0	Precision Clock A Control 0 Register
PCG_CTLA1	Precision Clock A Control 1 Register
PCG_CTLB0	Precision Clock B Control 0 Register
PCG_CTLB1	Precision Clock B Control 1 Register
PCG_CTLC0	Precision Clock C Control 0 Register
PCG_CTLC1	Precision Clock C Control 1 Register

Table 31-5: ADSP-2184x PCG Register List (Continued)

Name	Description
PCG_CTLD0	Precision Clock D Control 0 Register
PCG_CTLD1	Precision Clock D Control 1 Register
PCG_CTLE0	Precision Clock E Control 0 Register
PCG_CTLE1	Precision Clock E Control 1 Register
PCG_CTLF0	Precision Clock F Control 0 Register
PCG_CTLF1	Precision Clock F Control 1 Register
PCG_CTLG0	Precision Clock G Control 0 Register
PCG_CTLG1	Precision Clock G Control 1 Register
PCG_CTLH0	Precision Clock H Control 0 Register
PCG_CTLH1	Precision Clock H Control 1 Register
PCG_PW1	Precision Clock Pulse Width Control 1 Register
PCG_PW2	Precision Clock Pulse Width Control 2 Register
PCG_PW3	Precision Clock Pulse Width Control 3 Register
PCG_PW4	Precision Clock Pulse Width Control 4 Register
PCG_SYNC1	Precision Clock Frame Sync Synchronization 1 Register
PCG_SYNC2	Precision Clock Frame Sync Synchronization 2 Register
PCG_SYNC3	Precision Clock Frame Sync Synchronization 3 Register
PCG_SYNC4	Precision Clock Frame Sync Synchronization 4 Register

Precision Clock A Control 0 Register

The PCG_CTLA0 register enables the clock, frame sync, and select divisor for the PCG0 clock A signal.

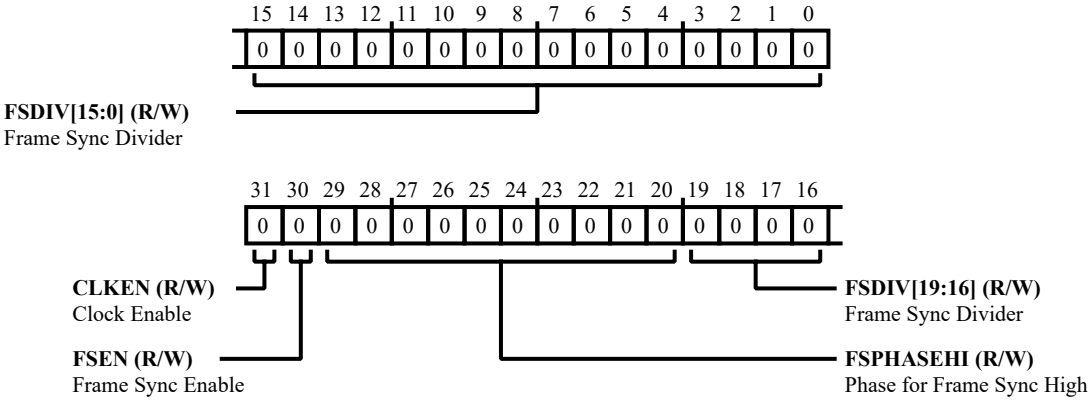


Figure 31-7: PCG_CTLA0 Register Diagram

Table 31-6: PCG_CTLA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLA0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLA0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLA0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLA0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLA0.FSDIV >1 PCGx is in normal mode, PCG_CTLA0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	PCG is in normal mode

Precision Clock A Control 1 Register

The PCG_CTLA1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock A signal.

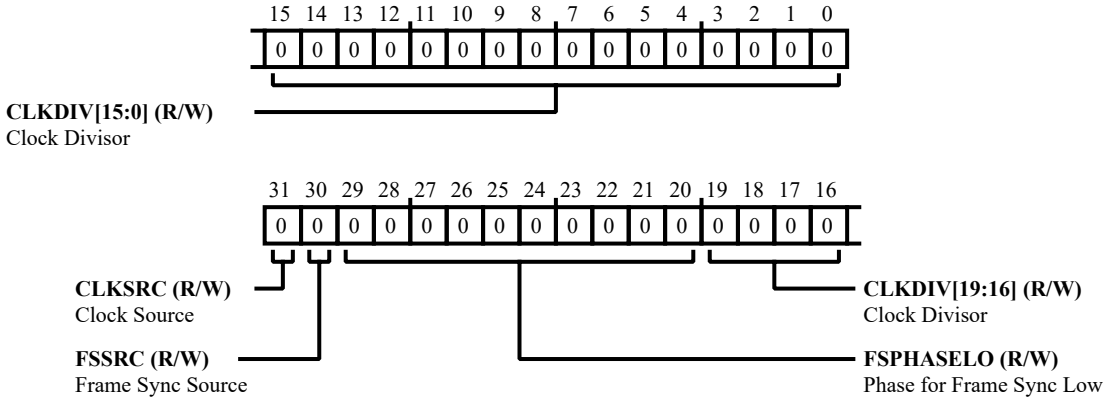


Figure 31-8: PCG_CTLA1 Register Diagram

Table 31-7: PCG_CTLA1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLA1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI0 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLA1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXTX_DAI0 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLA1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLA1.CLKDIV bit field contains the clock divisor value.	

Precision Clock B Control 0 Register

The PCG_CTLB0 register enables the clock, frame sync, and select divisor for the PCG0 clock B signal.

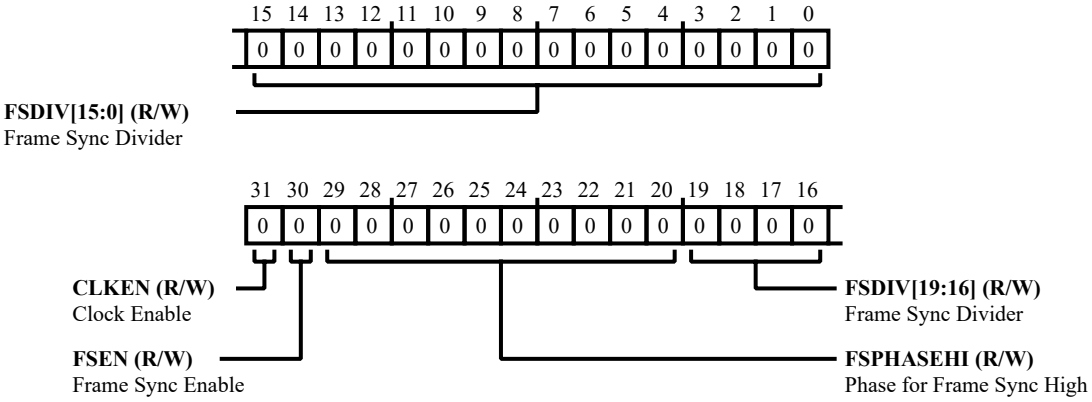


Figure 31-9: PCG_CTLB0 Register Diagram

Table 31-8: PCG_CTLB0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLB0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLB0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLB0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLB0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLB0.FSDIV >1 PCGx is in normal mode, PCG_CTLB0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock B Control 1 Register

The PCG_CTLB1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock B signal.

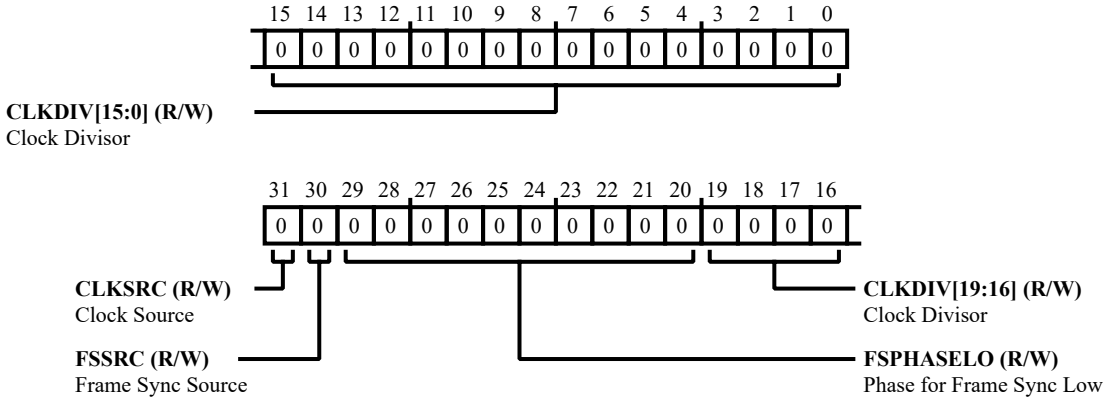


Figure 31-10: PCG_CTLB1 Register Diagram

Table 31-9: PCG_CTLB1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLB1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI0 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLB1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI0 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLB1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLB1.CLKDIV bit field contains the clock divisor value.	

Precision Clock C Control 0 Register

The PCG_CTLC0 register enables the clock, frame sync, and select divisor for the PCG0 clock C signal.

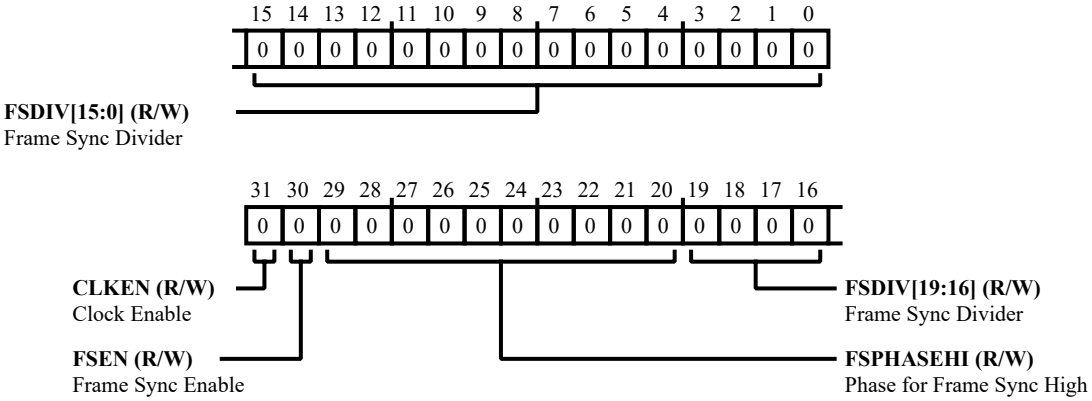


Figure 31-11: PCG_CTLC0 Register Diagram

Table 31-10: PCG_CTLC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLC0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLC0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLC0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLC0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLC0.FSDIV >1 PCGx is in normal mode, PCG_CTLC0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock C Control 1 Register

The PCG_CTLC1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock C signal.

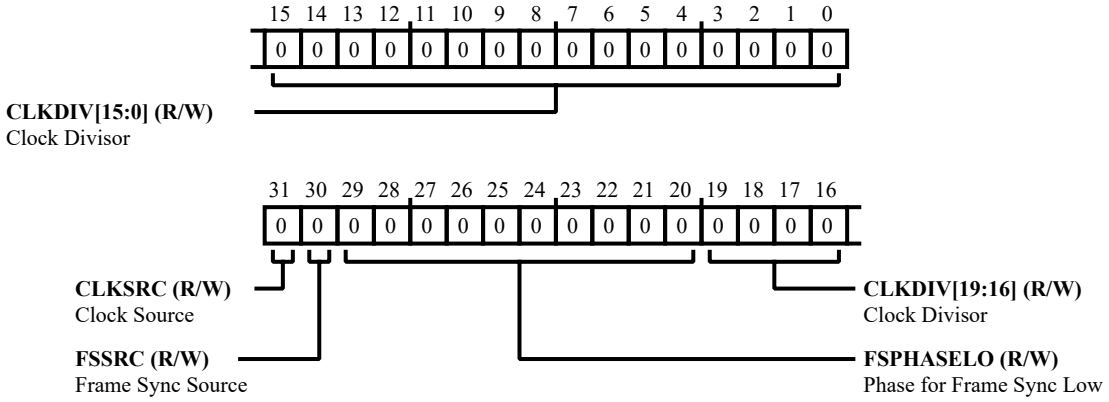


Figure 31-12: PCG_CTLC1 Register Diagram

Table 31-11: PCG_CTLC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLC1.CLKSRC bit specifies the clock source.	
		0	CLKIN1 pin selected for clock
		1	PCG_EXT_DAI1 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLC1.FSSRC bit specifies the frame sync source.	
		0	CLKIN1 pin selected for frame sync
		1	PCG_EXT_DAI1 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLC1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLC1.CLKDIV bit field contains the clock divisor value.	

Precision Clock D Control 0 Register

The PCG_CTLD0 register enables the clock, frame sync, and select divisor for the PCG0 clock D signal.

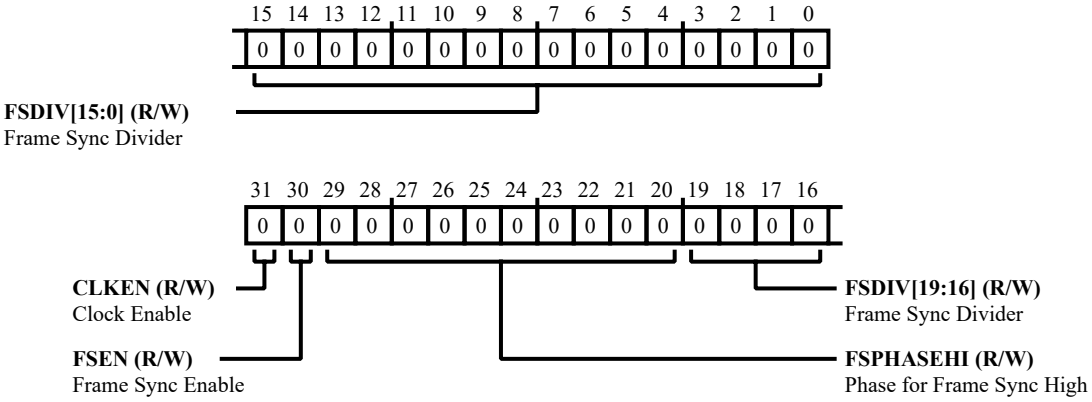


Figure 31-13: PCG_CTLD0 Register Diagram

Table 31-12: PCG_CTLD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLD0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLD0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLD0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLD0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLD0.FSDIV >1 PCGx is in normal mode, PCG_CTLD0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock D Control 1 Register

The PCG_CTLD1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock D signal.

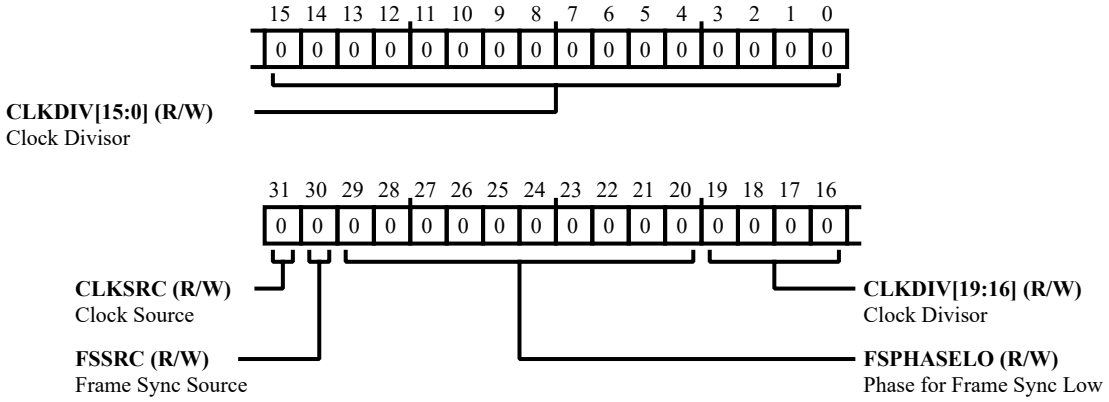


Figure 31-14: PCG_CTLD1 Register Diagram

Table 31-13: PCG_CTLD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLD1.CLKSRC bit specifies the clock source.	
		0	CLKIN1 pin selected for clock
		1	PCG_EXT_DAI1 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLD1.FSSRC bit specifies the frame sync source.	
		0	CLKIN1 pin selected for frame sync
		1	PCG_EXT_DAI1 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLD1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLD1.CLKDIV bit field contains the clock divisor value.	

Precision Clock E Control 0 Register

The PCG_CTLE0 register enables the clock, frame sync, and select divisor for the PCG0 clock E signal.

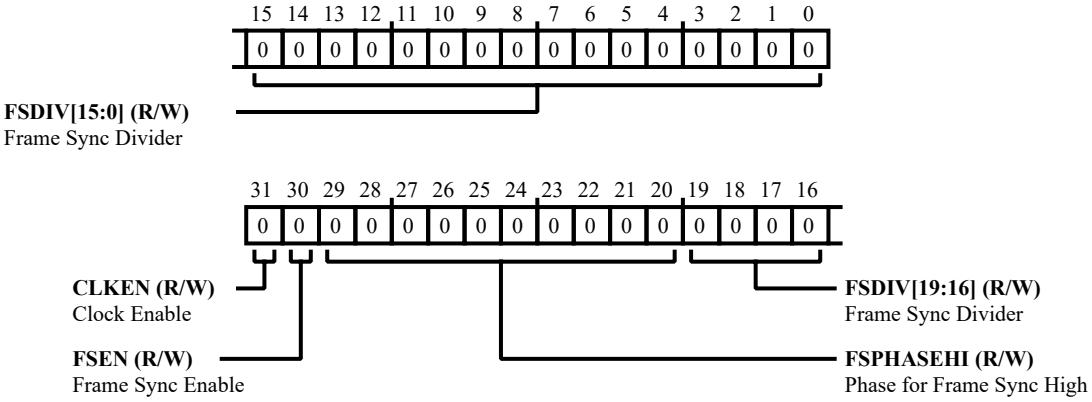


Figure 31-15: PCG_CTLE0 Register Diagram

Table 31-14: PCG_CTLE0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLE0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLE0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLE0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLE0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLE0.FSDIV >1 PCGx is in normal mode, PCG_CTLE0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock E Control 1 Register

The PCG_CTLE1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock E signal.

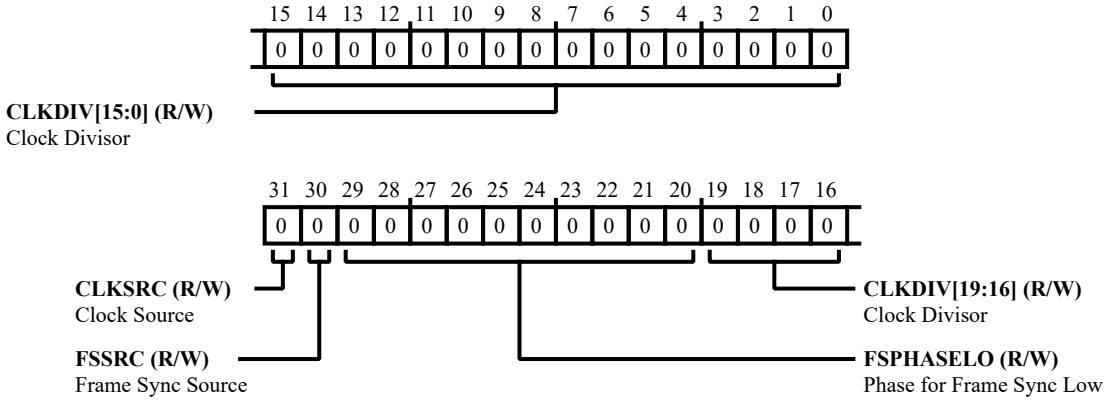


Figure 31-16: PCG_CTLE1 Register Diagram

Table 31-15: PCG_CTLE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLE1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLE1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLE1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLE1.CLKDIV bit field contains the clock divisor value.	

Precision Clock F Control 0 Register

The PCG_CTLF0 register enables the clock, frame sync, and select divisor for the PCG0 clock F signal.

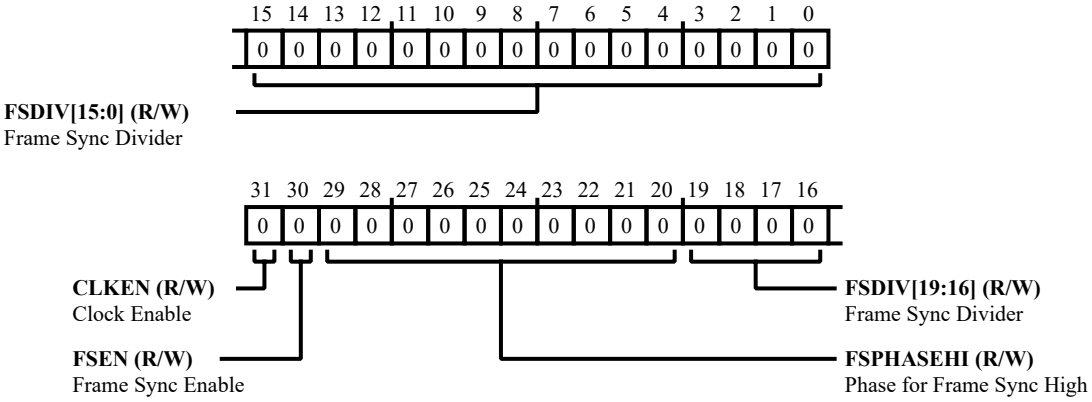


Figure 31-17: PCG_CTLF0 Register Diagram

Table 31-16: PCG_CTLF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLF0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLF0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLF0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLF0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLF0.FSDIV >1 PCGx is in normal mode, PCG_CTLF0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock F Control 1 Register

The PCG_CTLF1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock F signal.

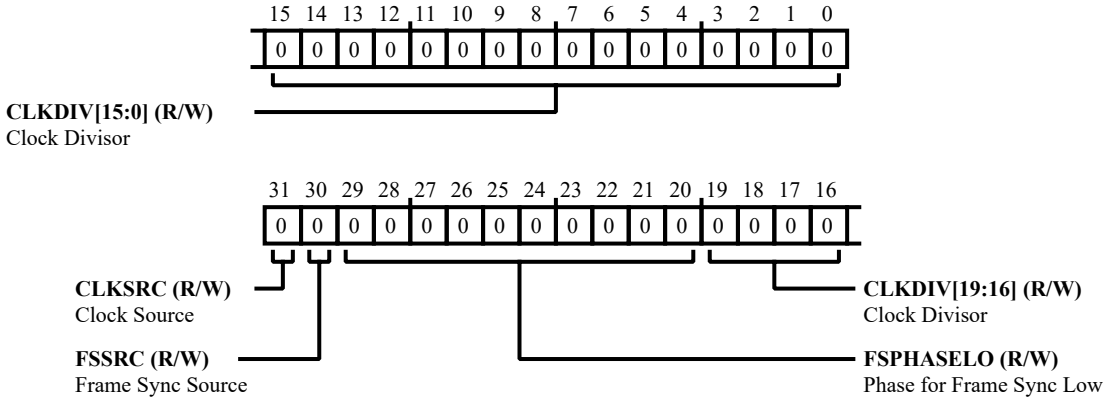


Figure 31-18: PCG_CTLF1 Register Diagram

Table 31-17: PCG_CTLF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLF1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI0 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLF1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI0 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLF1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLF1.CLKDIV bit field contains the clock divisor value.	

Precision Clock G Control 0 Register

The PCG_CTLG0 register enables the clock, frame sync, and select divisor for the PCG0 clock G signal.

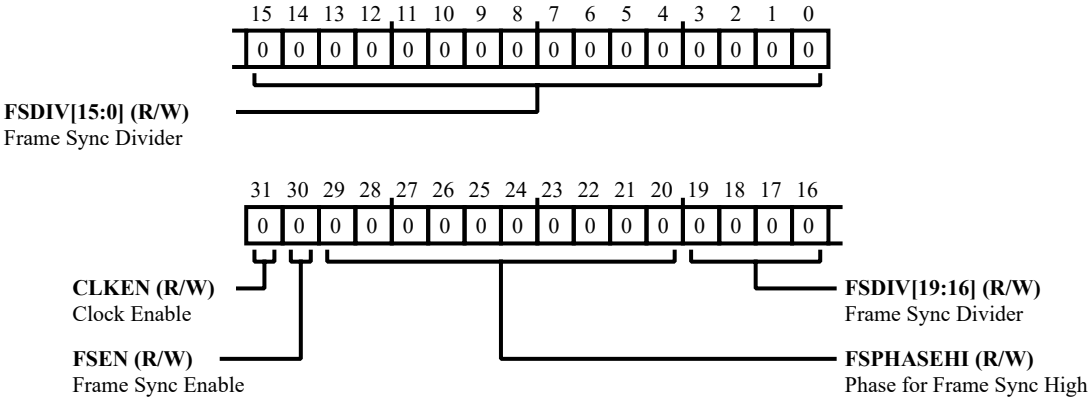


Figure 31-19: PCG_CTLG0 Register Diagram

Table 31-18: PCG_CTLG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLG0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLG0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLG0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLG0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLG0.FSDIV >1 PCGx is in normal mode, PCG_CTLG0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock G Control 1 Register

The PCG_CTLG1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock G signal.

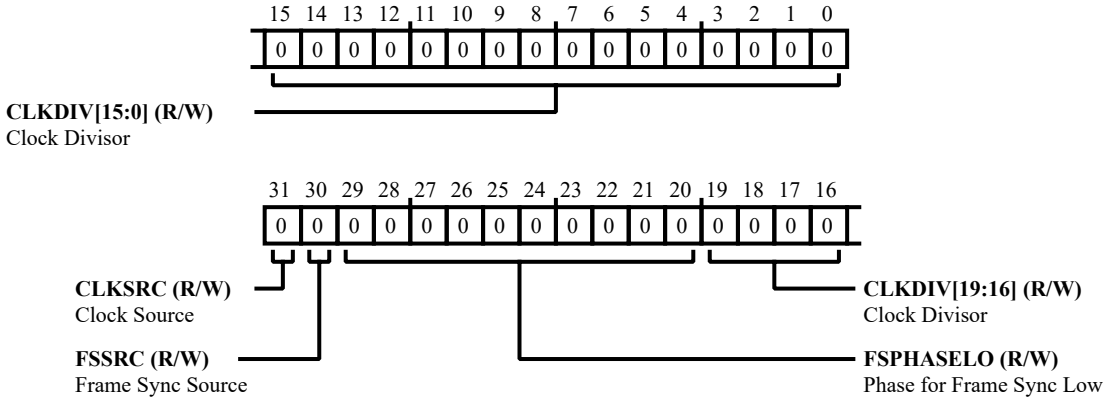


Figure 31-20: PCG_CTLG1 Register Diagram

Table 31-19: PCG_CTLG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLG1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLG1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLG1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLG1.CLKDIV bit field contains the clock divisor value.	

Precision Clock H Control 0 Register

The PCG_CTLH0 register enables the clock, frame sync, and select divisor for the PCG0 clock H signal.

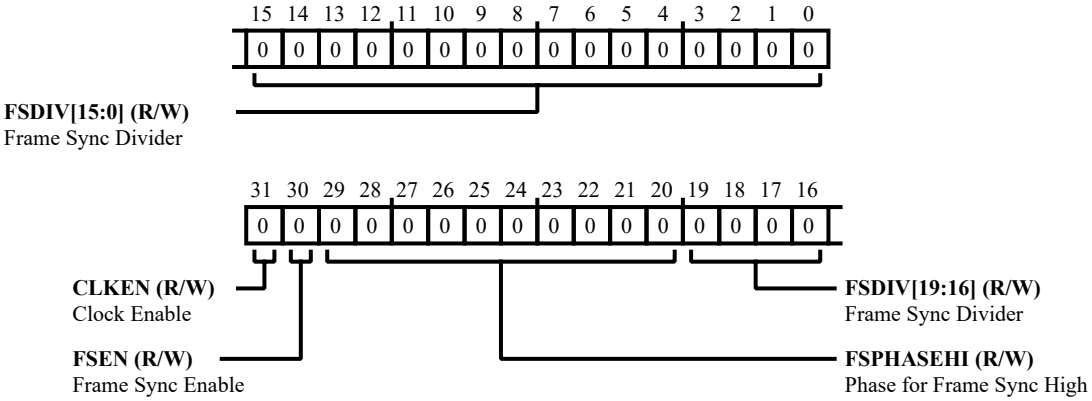


Figure 31-21: PCG_CTLH0 Register Diagram

Table 31-20: PCG_CTLH0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLH0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLH0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLH0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLH0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLH0.FSDIV >1 PCGx is in normal mode, PCG_CTLH0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock H Control 1 Register

The PCG_CTLH1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock H signal.

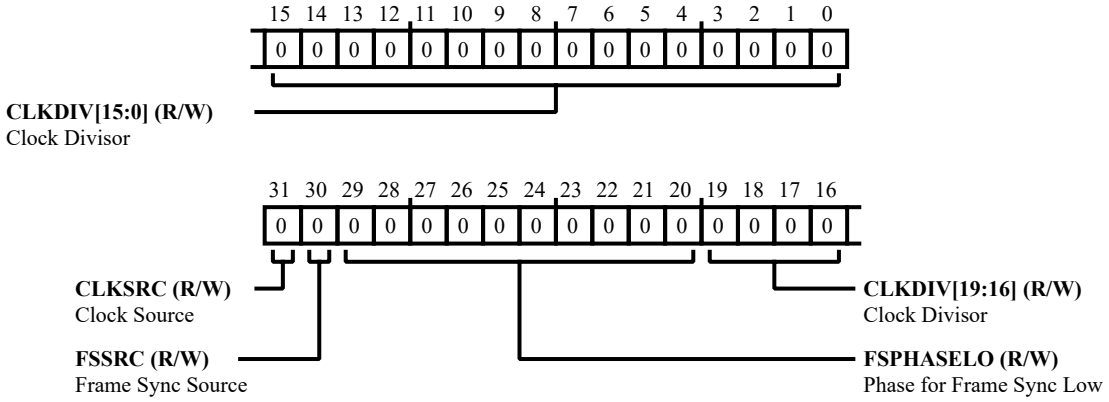


Figure 31-22: PCG_CTLH1 Register Diagram

Table 31-21: PCG_CTLH1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLH1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLH1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLH1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLH1.CLKDIV bit field contains the clock divisor value.	

Precision Clock Pulse Width Control 1 Register

The PCG_PW1 register sets the one shot frame sync and the active low frame sync select for PCG A and PCG B.

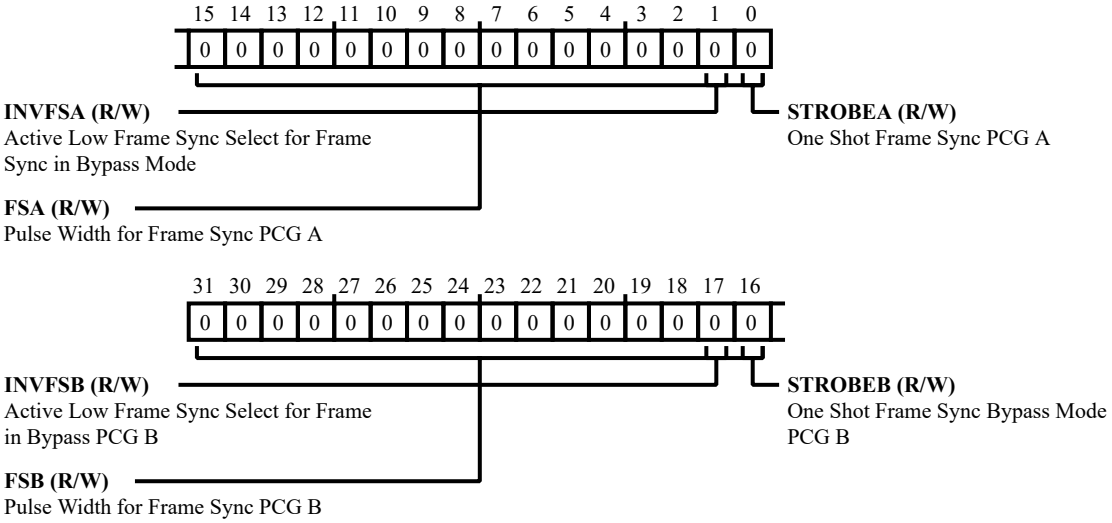


Figure 31-23: PCG_PW1 Register Diagram

Table 31-22: PCG_PW1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSB	Active Low Frame Sync Select for Frame in Bypass PCG B. The PCG_PW1.INVFSB bit selects active low or active high frame sync in bypass mode for PCG B.
16 (R/W)	STROBEB	One Shot Frame Sync Bypass Mode PCG B. The PCG_PW1.STROBEB bit sets the frame sync pulse in bypass mode for PCG B. This is the duration equal to one period of the DAI_MISCA3_I signal (PCG B) repeating at the beginning of every frame.
31:16 (R/W)	FSB	Pulse Width for Frame Sync PCG B. The PCG_PW1.FSB bit field sets the number of input clock periods for which the frame sync output is high. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSA	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW1.INVFSA bit selects active low or active high frame sync for PCG A in bypass mode.
15:0 (R/W)	FSA	Pulse Width for Frame Sync PCG A. The PCG_PW1.FSA bit field sets the number of input clock periods for which the frame sync output is high. Pulse width should be less than the divisor of the frame sync.

Table 31-22: PCG_PW1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	STROBEA	One Shot Frame Sync PCG A. The `PCG_PW1.STROBEA` bit sets the frame sync pulse for PCG A in bypass mode. This is the duration equal to one period of the DAI_MISCA2_I signal (PCG A) repeating at the beginning of every frame.

Precision Clock Pulse Width Control 2 Register

The PCG_PW2 register sets the one shot frame sync and the active low frame sync select for PCG C and PCG D.

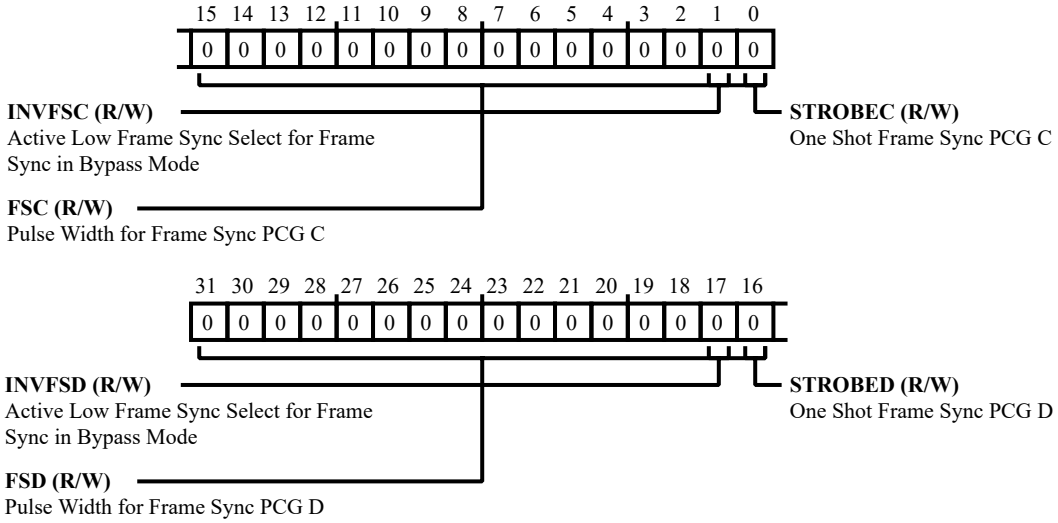


Figure 31-24: PCG_PW2 Register Diagram

Table 31-23: PCG_PW2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSD	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW2.INVFSD bit selects active low or active high frame sync for PCG D in bypass mode.
16 (R/W)	STROBED	One Shot Frame Sync PCG D. The PCG_PW2.STROBED bit sets the frame sync pulse for PCG D in bypass mode. This is the duration equal to one period of the DAI_MISCA3_I signal (PCG D) repeating at the beginning of every frame.
31:16 (R/W)	FSD	Pulse Width for Frame Sync PCG D. The PCG_PW2.FSD bit field sets the number of input clock periods for which the frame sync output is high for PCG D. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSC	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW2.INVFSC bit selects active low or active high frame sync for PCG C in bypass mode.
15:0 (R/W)	FSC	Pulse Width for Frame Sync PCG C. The PCG_PW2.FSC bit field sets the number of input clock periods for which the frame sync output is high for PCG C. Pulse width should be less than the divisor of the frame sync.

Table 31-23: PCG_PW2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	STROBEC	One Shot Frame Sync PCG C. The `PCG_PW2.STROBEC` bit sets the frame sync pulse for PCG C in bypass mode. This is the duration equal to one period of the DAI_MISCA2_I signal (PCG C) repeating at the beginning of every frame.

Precision Clock Pulse Width Control 3 Register

The PCG_PW3 register sets the one shot frame sync and the active low frame sync select for PCG E and PCG F.

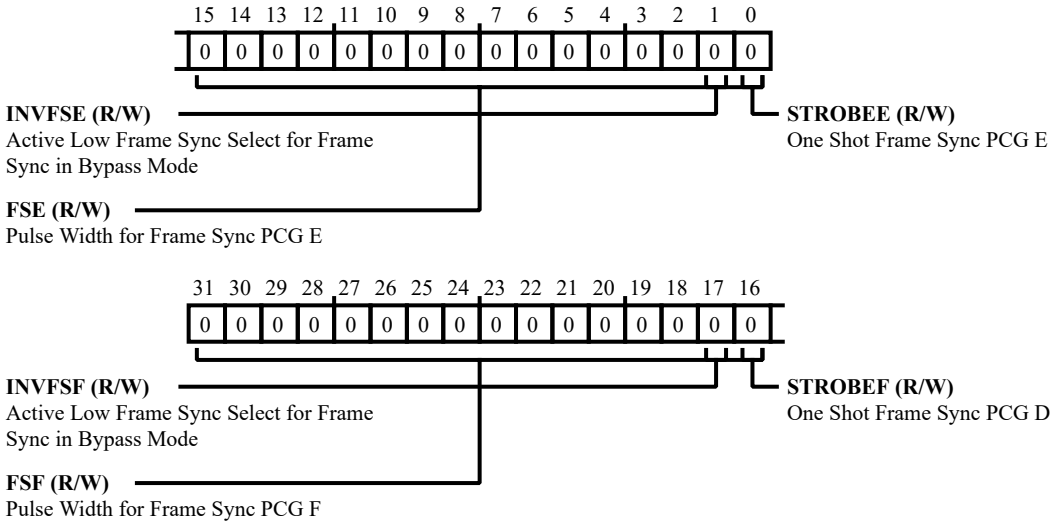


Figure 31-25: PCG_PW3 Register Diagram

Table 31-24: PCG_PW3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSF	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW3.INVFSF bit selects active low or active high frame sync for PCG F in bypass mode.
16 (R/W)	STROBEF	One Shot Frame Sync PCG D. The PCG_PW3.STROBEF bit sets the frame sync pulse for PCG F in bypass mode. This is the duration equal to one period of the DAI_MISCA1_I signal (PCG F) repeating at the beginning of every frame.
31:16 (R/W)	FSF	Pulse Width for Frame Sync PCG F. The PCG_PW3.FSF bit field sets the number of input clock periods for which the frame sync output is high for PCG F. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSE	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW3.INVFSE bit selects active low or active high frame sync for PCG E in bypass mode.
0 (R/W)	STROBEE	One Shot Frame Sync PCG E. The PCG_PW3.STROBEE bit sets the frame sync pulse for PCG E in bypass mode. This is the duration equal to one period of the DAI_MISCA0_I signal (PCG E) repeating at the beginning of every frame.

Table 31-24: PCG_PW3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	FSE	Pulse Width for Frame Sync PCG E. The `PCG_PW3.FSE` bit field sets the number of input clock periods for which the frame sync output is high for PCG E. Pulse width should be less than the divisor of the frame sync.

Precision Clock Pulse Width Control 4 Register

The PCG_PW4 register sets the one shot frame sync and the active low frame sync select for PCG G and PCG H.

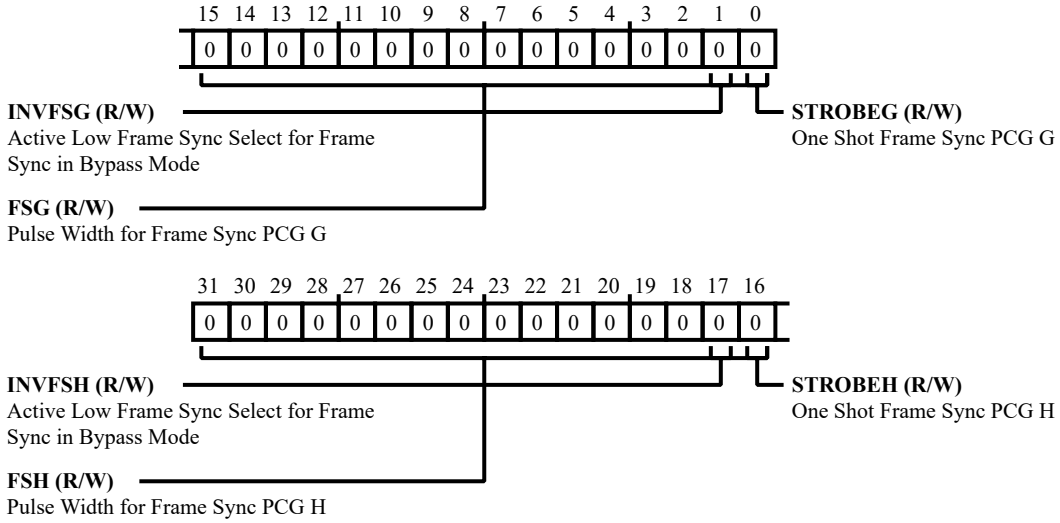


Figure 31-26: PCG_PW4 Register Diagram

Table 31-25: PCG_PW4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSH	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW4.INVFSH bit selects active low or active high frame sync for PCG H in bypass mode.
16 (R/W)	STROBEH	One Shot Frame Sync PCG H. The PCG_PW4.STROBEH bit sets the frame sync pulse for PCG H in bypass mode. This is the duration equal to one period of the DAI_MISCA1_I signal (PCG H) repeating at the beginning of every frame.
31:16 (R/W)	FSH	Pulse Width for Frame Sync PCG H. The PCG_PW4.FSH bit field sets the number of input clock periods for which the frame sync output is high for PCG H. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSG	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW4.INVFSG bit selects active low or active high frame sync for PCG G in bypass mode.
15:0 (R/W)	FSG	Pulse Width for Frame Sync PCG G. The PCG_PW4.FSG bit field sets the number of input clock periods for which the frame sync output is high for PCG G. Pulse width should be less than the divisor of the frame sync.

Table 31-25: PCG_PW4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	STROBEG	One Shot Frame Sync PCG G. The `PCG_PW4.STROBEG` bit sets the frame sync pulse for PCG G in bypass mode. This is the duration equal to one period of the DAI_MISCA0_I signal (PCG G) repeating at the beginning of every frame.

Precision Clock Frame Sync Synchronization 1 Register

The PCG_SYNC1 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLA1.CLKSRC bit is overridden if PCG_SYNC1.CLKASRC bit in the PCG_SYNC1 register is set.

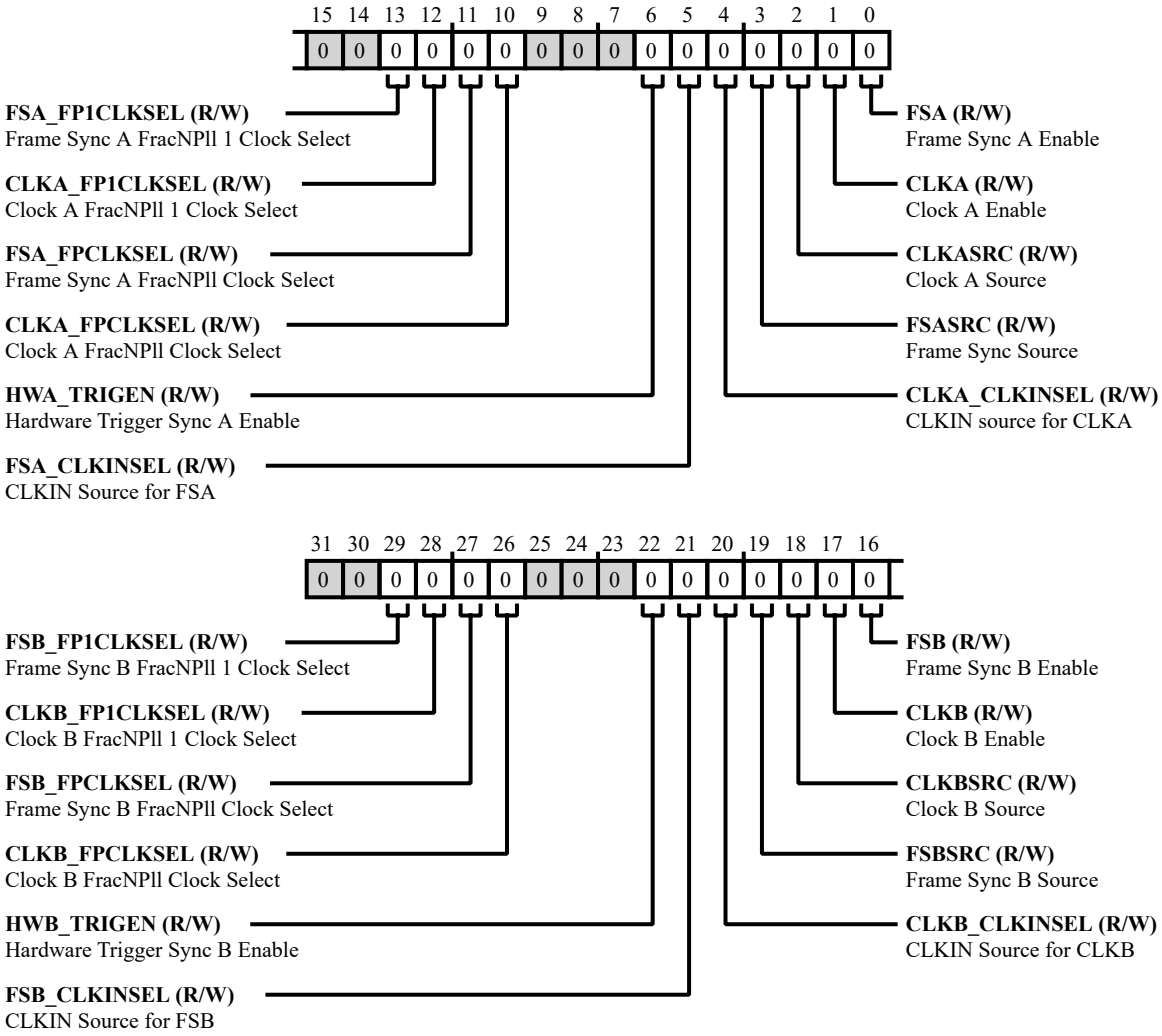


Figure 31-27: PCG_SYNC1 Register Diagram

Table 31-26: PCG_SYNC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	FSB_FP1CLKSEL	Frame Sync B FracNPll 1 Clock Select. The PCG_SYNC1.FSB_FP1CLKSEL bit when enabled, selects FracNPll 1 output clock input clock source to FSB. This feature is enabled only when PCG_BKP_CTLB1.FSSRC bit is set and FSB_FPCLKSEL bit of this register is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll 1 output clock selected for frame sync
28 (R/W)	CLKB_FP1CLKSEL	Clock B FracNPll 1 Clock Select. The PCG_SYNC1.CLKB_FP1CLKSEL bit when enabled, selects FracNPll 1 output clock as input clock source to CLKB. This feature is enabled only when PCG_BKP_CTLB1.CLKSRC bit is set and CLKB_FPCLKSEL bit of this register is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll 1 output clock selected for clock
27 (R/W)	FSB_FPCLKSEL	Frame Sync B FracNPll Clock Select. The PCG_SYNC1.FSB_FPCLKSEL bit when enabled, selects FracNPll output clock input clock source to FSB. This feature is enabled only when PCG_BKP_CTLB1.FSSRC bit is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll output clock selected for frame sync
26 (R/W)	CLKB_FPCLKSEL	Clock B FracNPll Clock Select. The PCG_SYNC1.CLKB_FPCLKSEL bit when enabled, selects FracNPll output clock as input clock source to CLKB. This feature is enabled only when PCG_BKP_CTLB1.CLKSRC bit is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll output clock selected for clock
22 (R/W)	HWB_TRIGEN	Hardware Trigger Sync B Enable. The PCG_SYNC1.HWB_TRIGEN bit enables synchronization of clock B and frame sync B with the external frame sync only after an internal receiver trigger for PCG B is received.	
21 (R/W)	FSB_CLKINSEL	CLKIN Source for FSB. The PCG_SYNC1.FSB_CLKINSEL bit enables the CLKIN input source for FSB.	
		0	CLKIN0
		1	CLKIN1

Table 31-26: PCG_SYNC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKB_CLKINSEL	CLKIN Source for CLKB. The `PCG_SYNC1.CLKB_CLKINSEL` bit enables the CLKIN input source for CLKB.	
		0	CLKIN0
		1	CLKIN1
19 (R/W)	FSBSRC	Frame Sync B Source. The `PCG_SYNC1.FSBSRC` bit enables the frame sync B input source.	
		0	Output selected by FSBSOURCE bit
		1	Clock derived from core PLL selected for frame sync B
18 (R/W)	CLKBSRC	Clock B Source. The `PCG_SYNC1.CLKBSRC` bit enables the clock B input source.	
		0	Output selected by CLKBSOURCE bit
		1	Clock derived from core PLL selected for clock B
17 (R/W)	CLKB	Clock B Enable. The `PCG_SYNC1.CLKB` bit enables synchronization of clock B with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSB	Frame Sync B Enable. The `PCG_SYNC1.FSB` bit enables synchronization of frame sync B with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
13 (R/W)	FSA_FP1CLKSEL	Frame Sync A FracNPll 1 Clock Select. The `PCG_SYNC1.FSA_FP1CLKSEL` bit when enabled, selects FracNPll 1 output clock input clock source to FSA. This feature is enabled only when PCG_BKP_CTLA1.FSSRC bit is set and FSA_FPCLKSEL bit of this register is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll 1 output clock selected for frame sync

Table 31-26: PCG_SYNC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	CLKA_FP1CLKSEL	Clock A FracNPll 1 Clock Select. The `PCG_SYNC1.CLKA_FP1CLKSEL` bit when enabled, selects FracNPll 1 output clock as input clock source to CLKA. This feature is enabled only when PCG_BKP_CTLA1.CLKSRC bit is set and CLKA_FPCLKSEL bit of this register is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll 1 output clock selected for clock
11 (R/W)	FSA_FPCLKSEL	Frame Sync A FracNPll Clock Select. The `PCG_SYNC1.FSA_FPCLKSEL` bit when enabled, selects FracNPll output clock input clock source to FSA. This feature is enabled only when PCG_BKP_CTLA1.FSSRC bit is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll output clock selected for frame sync
10 (R/W)	CLKA_FPCLKSEL	Clock A FracNPll Clock Select. The `PCG_SYNC1.CLKA_FPCLKSEL` bit when enabled, selects FracNPll output clock as input clock source to CLKA. This feature is enabled only when PCG_BKP_CTLA1.CLKSRC bit is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll output clock selected for clock
6 (R/W)	HWA_TRIGEN	Hardware Trigger Sync A Enable. The `PCG_SYNC1.HWA_TRIGEN` bit enables synchronization of clock A and frame sync A with the external frame sync only after an internal receiver trigger for PCG A is received.	
5 (R/W)	FSA_CLKINSEL	CLKIN Source for FSA. The `PCG_SYNC1.FSA_CLKINSEL` bit enables the CLKIN input source for FSA.	
		0	CLKIN0
		1	CLKIN1
4 (R/W)	CLKA_CLKINSEL	CLKIN source for CLKA. The `PCG_SYNC1.CLKA_CLKINSEL` bit enables the CLKIN input source for CLKA	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN

Table 31-26: PCG_SYNC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FSASRC	Frame Sync Source. The `PCG_SYNC1.FSASRC` bit enables the frame sync A input source.	
		0	Output selected by FSASOURCE bit
		1	Clock derived from core PLL selected for frame sync A
2 (R/W)	CLKASRC	Clock A Source. The `PCG_SYNC1.CLKASRC` bit enables the clock A input source.	
		0	Output selected by CLKASOURCE bit
		1	Clock derived from core PLL selected for clock A
1 (R/W)	CLKA	Clock A Enable. The `PCG_SYNC1.CLKA` bit enables synchronization of clock A with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSA	Frame Sync A Enable. The `PCG_SYNC1.FSA` bit enables synchronization of frame sync A with the external frame sync.	
		0	Frame Sync Disabled
		1	Frame Sync Enabled

Precision Clock Frame Sync Synchronization 2 Register

The PCG_SYNC2 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLD1.CLKSRC bit is overridden if PCG_SYNC2.CLKDSRC bit in the PCG_SYNC2 register is set.

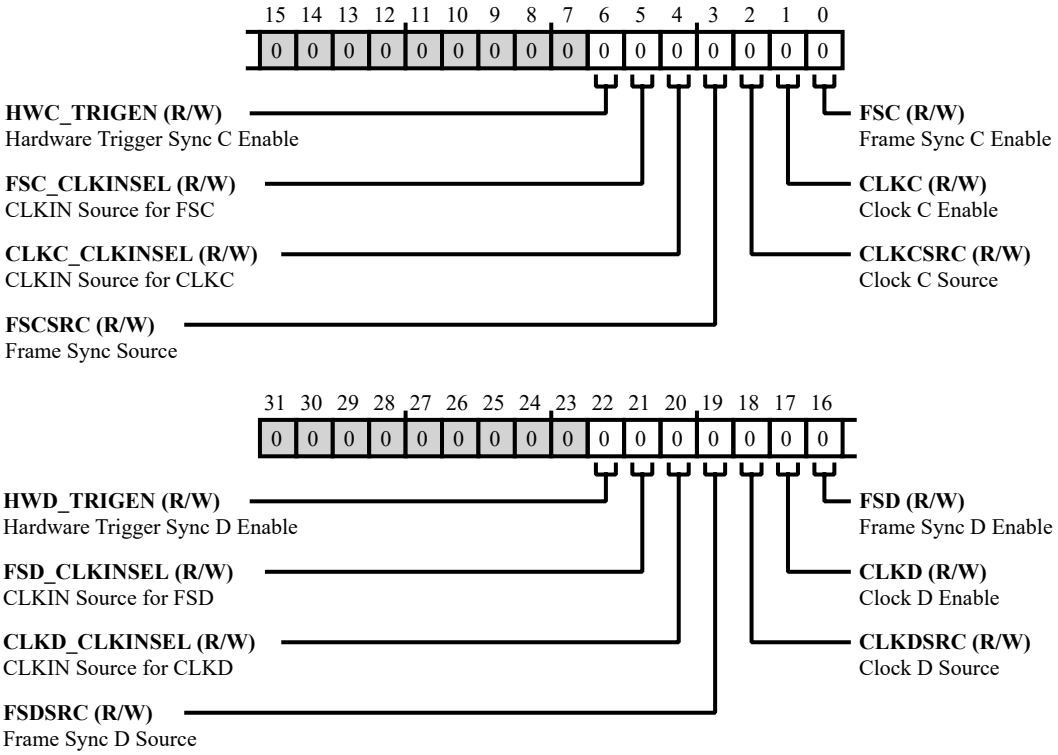


Figure 31-28: PCG_SYNC2 Register Diagram

Table 31-27: PCG_SYNC2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	HWD_TRIGEN	Hardware Trigger Sync D Enable. The PCG_SYNC2.HWD_TRIGEN bit enables synchronization of clock D and frame sync D with the external frame sync only after an internal receiver trigger for PCG D is received.	
21 (R/W)	FSD_CLKINSEL	CLKIN Source for FSD. The PCG_SYNC2.FSD_CLKINSEL bit enables the CLKIN input source for FSD.	
		0	Selects CLKIN0 source for FSB CLKIN
		1	Selects CLKIN1 source for FSB CLKIN

Table 31-27: PCG_SYNC2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKD_CLKINSEL	CLKIN Source for CLKD. The `PCG_SYNC2.CLKD_CLKINSEL` bit enables the CLKIN input source for CLKD.	
		0	Select CLKIN0 source for CLKB CLKIN
		1	Select CLKIN1 source for CLKB CLKIN
19 (R/W)	FSDSRC	Frame Sync D Source. The `PCG_SYNC2.FSDSRC` bit enables the frame sync D input source.	
		0	Output selected by FSDSOURCE bit
		1	Clock derived from core PLL selected for frame sync D
18 (R/W)	CLKDSRC	Clock D Source. The `PCG_SYNC2.CLKDSRC` bit enables the clock D input source.	
		0	Output selected by CLKDSOURCE bit
		1	Clock derived from core PLL selected for clock D
17 (R/W)	CLKD	Clock D Enable. The `PCG_SYNC2.CLKD` bit enables synchronization of clock D with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSD	Frame Sync D Enable. The `PCG_SYNC2.FSD` bit enables synchronization of frame sync D with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
6 (R/W)	HWC_TRIGEN	Hardware Trigger Sync C Enable. The `PCG_SYNC2.HWC_TRIGEN` bit enables synchronization of clock C and frame sync C with the external frame sync only after an internal receiver trigger for PCG C is received.	
5 (R/W)	FSC_CLKINSEL	CLKIN Source for FSC. The `PCG_SYNC2.FSC_CLKINSEL` bit enables the CLKIN input source for FSC.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 31-27: PCG_SYNC2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CLKC_CLKINSEL	CLKIN Source for CLKC. The `PCG_SYNC2.CLKC_CLKINSEL` bit enables the CLKIN input source for CLKC.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
3 (R/W)	FSCSRC	Frame Sync Source. The `PCG_SYNC2.FSCSRC` bit enables the frame sync C input source.	
		0	Output selected by FSCSOURCE bit
		1	Clock derived from core PLL selected for frame sync C
2 (R/W)	CLKCSRC	Clock C Source. The `PCG_SYNC2.CLKCSRC` bit enables the clock C input source.	
		0	Output selected by CLKCSOURCE bit
		1	Clock derived from core PLL selected for clock C
1 (R/W)	CLKC	Clock C Enable. The `PCG_SYNC2.CLKC` bit enables synchronization of clock C with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSC	Frame Sync C Enable. The `PCG_SYNC2.FSC` bit enables synchronization of frame sync C with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled

Precision Clock Frame Sync Synchronization 3 Register

The PCG_SYNC3 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLD1.CLKSRC bit is overridden if CLKDSRC bit in the PCG_SYNC3 register is set.

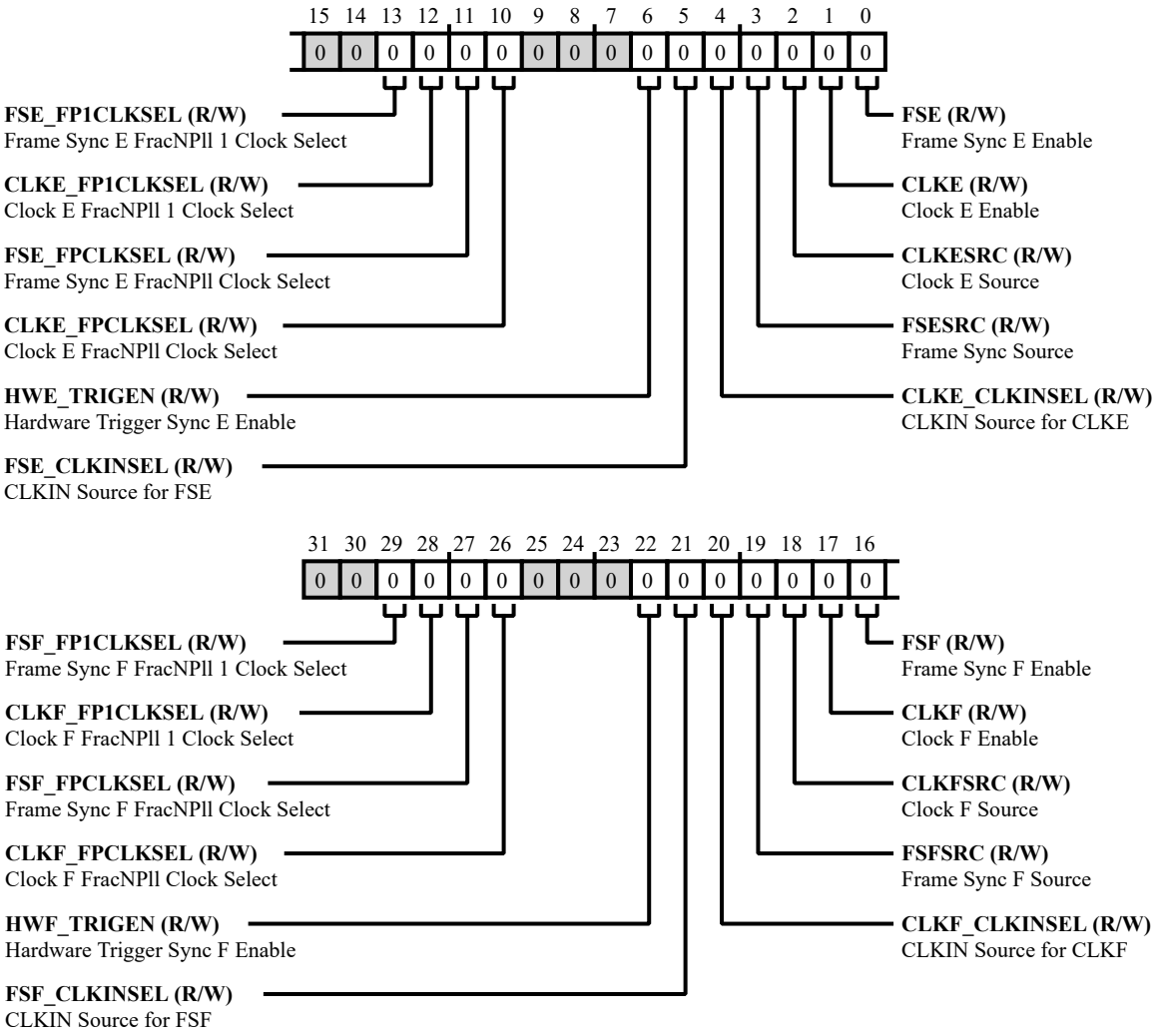


Figure 31-29: PCG_SYNC3 Register Diagram

Table 31-28: PCG_SYNC3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	FSF_FP1CLKSEL	Frame Sync F FracNPll 1 Clock Select. The PCG_SYNC3.FSF_FP1CLKSEL bit when enabled, selects FracNPll 1 output clock input clock source to FSF. This feature is enabled only when PCG_BKP_CTLF1.FSSRC bit is set and FSF_FPCLKSEL bit of this register is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll 1 output clock selected for frame sync
28 (R/W)	CLKF_FP1CLKSEL	Clock F FracNPll 1 Clock Select. The PCG_SYNC3.CLKF_FP1CLKSEL bit when enabled, selects FracNPll 1 output clock as input clock source to CLKF. This feature is enabled only when PCG_BKP_CTLF1.CLKSRC bit is set and CLKF_FPCLKSEL bit of this register is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll 1 output clock selected for clock
27 (R/W)	FSF_FPCLKSEL	Frame Sync F FracNPll Clock Select. The PCG_SYNC3.FSF_FPCLKSEL bit when enabled, selects FracNPll output clock input clock source to FSF. This feature is enabled only when PCG_BKP_CTLF1.FSSRC bit is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll output clock selected for frame sync
26 (R/W)	CLKF_FPCLKSEL	Clock F FracNPll Clock Select. The PCG_SYNC3.CLKF_FPCLKSEL bit when enabled, selects FracNPll output clock as input clock source to CLKF. This feature is enabled only when PCG_BKP_CTLF1.CLKSRC bit is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll output clock selected for clock
22 (R/W)	HWF_TRIGEN	Hardware Trigger Sync F Enable. The PCG_SYNC3.HWF_TRIGEN bit enables synchronization of clock F and frame sync F with the external frame sync only after an internal receiver trigger for PCG-F is received.	
21 (R/W)	FSF_CLKINSEL	CLKIN Source for FSF. The PCG_SYNC3.FSF_CLKINSEL bit enables the CLKIN input source for FSF.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 31-28: PCG_SYNC3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKF_CLKINSEL	CLKIN Source for CLKF. The `PCG_SYNC3.CLKF_CLKINSEL` bit enables the CLKIN input source for CLKF.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
19 (R/W)	FSFSRC	Frame Sync F Source. The `PCG_SYNC3.FSFSRC` bit enables the frame sync F input source.	
		0	Output selected by FSDSOURCE bit
		1	Clock derived from core PLL selected for frame sync D
18 (R/W)	CLKFSRC	Clock F Source. The `PCG_SYNC3.CLKFSRC` bit enables the clock F input source.	
		0	Output selected by CLKDSOURCE bit
		1	Clock derived from core PLL selected for clock D
17 (R/W)	CLKF	Clock F Enable. The `PCG_SYNC3.CLKF` bit enables synchronization of clock F with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSF	Frame Sync F Enable. The `PCG_SYNC3.FSF` bit enables synchronization of frame sync F with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
13 (R/W)	FSE_FP1CLKSEL	Frame Sync E FracNPll 1 Clock Select. The `PCG_SYNC3.FSE_FP1CLKSEL` bit when enabled, selects FracNPll 1 output clock input clock source to FSE. This feature is enabled only when PCG_BKP_CTLE1.FSSRC bit is set and FSE_FPCLKSEL bit of this register is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll 1 output clock selected for frame sync

Table 31-28: PCG_SYNC3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	CLKE_FP1CLKSEL	Clock E FracNPll 1 Clock Select. The `PCG_SYNC3.CLKE_FP1CLKSEL` bit when enabled, selects FracNPll 1 output clock as input clock source to CLKE. This feature is enabled only when PCG_BKP_CTLE1.CLKSRC bit is set and CLKE_FPCLKSEL bit of this register is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll 1 output clock selected for clock
11 (R/W)	FSE_FPCLKSEL	Frame Sync E FracNPll Clock Select. The `PCG_SYNC3.FSE_FPCLKSEL` bit when enabled, selects FracNPll output clock input clock source to FSE. This feature is enabled only when PCG_BKP_CTLE1.FSSRC bit is set.	
		0	PCG_EXTX_DAI0 selected for frame sync
		1	FracNPll output clock selected for frame sync
10 (R/W)	CLKE_FPCLKSEL	Clock E FracNPll Clock Select. The `PCG_SYNC3.CLKE_FPCLKSEL` bit when enabled, selects FracNPll output clock as input clock source to CLKE. This feature is enabled only when PCG_BKP_CTLE1.CLKSRC bit is set.	
		0	PCG_EXT_DAI0 selected for clock
		1	FracNPll output clock selected for clock
6 (R/W)	HWE_TRIGEN	Hardware Trigger Sync E Enable. The `PCG_SYNC3.HWE_TRIGEN` bit enables synchronization of clock E and frame sync E with the external frame sync only after an internal receiver trigger for PCG-E is received.	
5 (R/W)	FSE_CLKINSEL	CLKIN Source for FSE. The `PCG_SYNC3.FSE_CLKINSEL` bit enables the CLKIN input source for FSE.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN
4 (R/W)	CLKE_CLKINSEL	CLKIN Source for CLKE. The `PCG_SYNC3.CLKE_CLKINSEL` bit enables the CLKIN input source for CLKE.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN

Table 31-28: PCG_SYNC3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FSESRC	Frame Sync Source. The `PCG_SYNC3.FSESRC` bit enables the frame sync E input source.	
		0	Output selected by FSCSOURCE bit
		1	Clock derived from core PLL selected for frame sync C
2 (R/W)	CLKESRC	Clock E Source. The `PCG_SYNC3.CLKESRC` bit enables the clock E input source.	
		0	Output selected by CLKCSOURCE bit
		1	Clock derived from core PLL selected for clock C
1 (R/W)	CLKE	Clock E Enable. The `PCG_SYNC3.CLKE` bit enables synchronization of clock E with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSE	Frame Sync E Enable. The `PCG_SYNC3.FSE` bit enables synchronization of frame sync E with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled

Precision Clock Frame Sync Synchronization 4 Register

The PCG_SYNC4 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLD1.CLKSRC bit is overridden if CLKDSRC bit in the PCG_SYNC4 register is set.

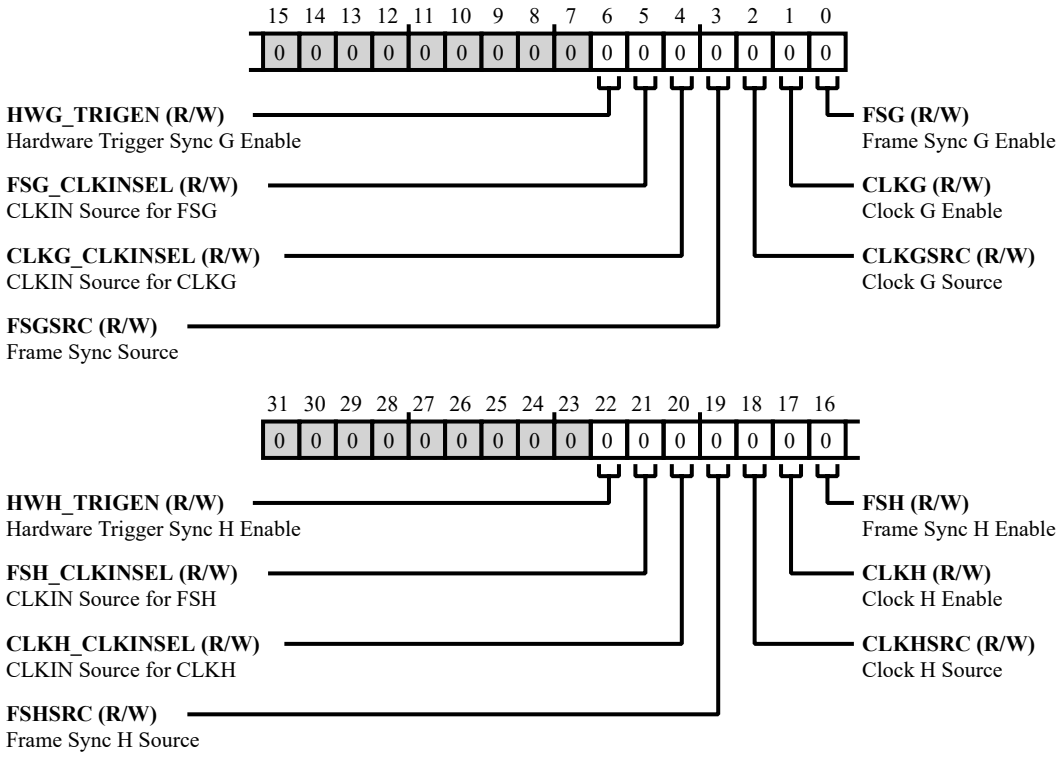


Figure 31-30: PCG_SYNC4 Register Diagram

Table 31-29: PCG_SYNC4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	HWH_TRIGEN	Hardware Trigger Sync H Enable. The PCG_SYNC4.HWH_TRIGEN bit enables synchronization of clock H and frame sync H with the external frame sync only after an internal receiver trigger for PCG-H is received.	
21 (R/W)	FSH_CLKINSEL	CLKIN Source for FSH. The PCG_SYNC4.FSH_CLKINSEL bit enables the CLKIN input source for FSH.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 31-29: PCG_SYNC4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKH_CLKINSEL	CLKIN Source for CLKH. The `PCG_SYNC4.CLKH_CLKINSEL` bit enables the CLKIN input source for CLKH.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
19 (R/W)	FSHSRC	Frame Sync H Source. The `PCG_SYNC4.FSHSRC` bit enables the frame sync H input source.	
		0	Output selected by FSDSOURCE bit
		1	Clock derived from core PLL selected for frame sync D
18 (R/W)	CLKHSRC	Clock H Source. The `PCG_SYNC4.CLKHSRC` bit enables the clock H input source.	
		0	Output selected by CLKDSOURCE bit
		1	Clock derived from core PLL selected for clock D
17 (R/W)	CLKH	Clock H Enable. The `PCG_SYNC4.CLKH` bit enables synchronization of clock H with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSH	Frame Sync H Enable. The `PCG_SYNC4.FSH` bit enables synchronization of frame sync H with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
6 (R/W)	HWG_TRIGEN	Hardware Trigger Sync G Enable. The `PCG_SYNC4.HWG_TRIGEN` bit enables synchronization of clock G and frame sync G with the external frame sync only after an internal receiver trigger for PCG G is received.	
5 (R/W)	FSG_CLKINSEL	CLKIN Source for FSG. The `PCG_SYNC4.FSG_CLKINSEL` bit enables the CLKIN input source for FSG.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 31-29: PCG_SYNC4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CLKG_CLKINSEL	CLKIN Source for CLKG. The `PCG_SYNC4.CLKG_CLKINSEL` bit enables the CLKIN input source for CLKG.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
3 (R/W)	FSGSRC	Frame Sync Source. The `PCG_SYNC4.FSGSRC` bit enables the frame sync G input source.	
		0	Output selected by FSCSOURCE bit
		1	Clock derived from core PLL selected for frame sync C
2 (R/W)	CLKGSRC	Clock G Source. The `PCG_SYNC4.CLKGSRC` bit enables the clock G input source.	
		0	Output selected by CLKCSOURCE bit
		1	Clock derived from core PLL selected for clock C
1 (R/W)	CLKG	Clock G Enable. The `PCG_SYNC4.CLKG` bit enables synchronization of clock G with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSG	Frame Sync G Enable. The `PCG_SYNC4.FSG` bit enables synchronization of frame sync G with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled

32 Asynchronous Sample Rate Converter (ASRC)

Sample rate converters (SRC) are frequently used in digital signal processing audio applications. The most frequently used sample rate conversions are off-loaded into hardware modules that are dedicated for filter processing and reduce the instruction processing load on the core, freeing it up for other tasks.

Features

The ASRC has the following features and capabilities:

- 8 asynchronous stereo SRCs operating in completer mode are available in each DAI.
- Simple programming model
- Controllable muting options (hardware, software and automatic)
- Automatically senses input and output sample frequencies
- Supports left-justified, I^2S, right-justified (16-,18-, 20-, 24-bits), and TDM serial port modes
- Daisy-chain configuration in TDM modes (including between DAI0 and DAI1) for input and output ports to create a serial frame
- Different protocols on input/output port allow format conversions
- De-emphasis filter for 32, 44.1 and 48 kHz sampling frequencies
- Up to 192 kHz sample rate input/output continuous sample ratios from 7.5:1 to 1:8
- Group delay (latency of interpolation filter) is 16 samples
- SNR from 128 to 140 dB (depending on processor model)
- Matched phase mode available to compensate for group delays
- Can be used to de-jitter clocks in systems

Functional Description

Conceptually, the sample rate converter interpolates the serial input data at a rate of 2^{20} and samples the interpolated data stream by the output sample rate. In practice, a 64-tap FIR filter with 2^{20} polyphases, a FIFO, a digital servo loop that measures the time difference between the input and output samples within 5 ps, and a digital circuit to track the sample rate ratio are used to perform the interpolation and output sampling.

ADSP-2184x ASRC Register List

Sample Rate Converter Module

Table 32-1: ADSP-2184x ASRC Register List

Name	Description
ASRC_CTL01	Control Register for ASRC 0 and 1
ASRC_CTL23	Control Register for ASRC 2 and 3
ASRC_CTL45	Control Register For ASRC 4 And 5
ASRC_CTL67	Control Register for ASRC 6 and 7
ASRC_MUTE	Mute Register
ASRC_RAT01	Ratio Register for ASRC 0 and 1
ASRC_RAT23	Ratio Register for ASRC 2 and 3
ASRC_RAT45	Ratio Register For ASRC 4 and 5
ASRC_RAT67	Ratio Register for ASRC 6 and 7

ASRC Interrupt List

The ASRC interrupts are controlled through the DAI.

Table 32-2: ASRC Interrupt List

Interrupt ID	Interrupt Name	Interrupt Condition
64	DAI0_IRQH	ASRC initialization ASRC sample rate change
65	DAI1_IRQH	
66	DAI0_IRQL	
67	DAI1_IRQL	

ASRC Block Diagram

The ***ASRC Block Diagram*** figure shows a top level block diagram of the ASRC module and the ***Core Architecture*** figure shows architecture details.

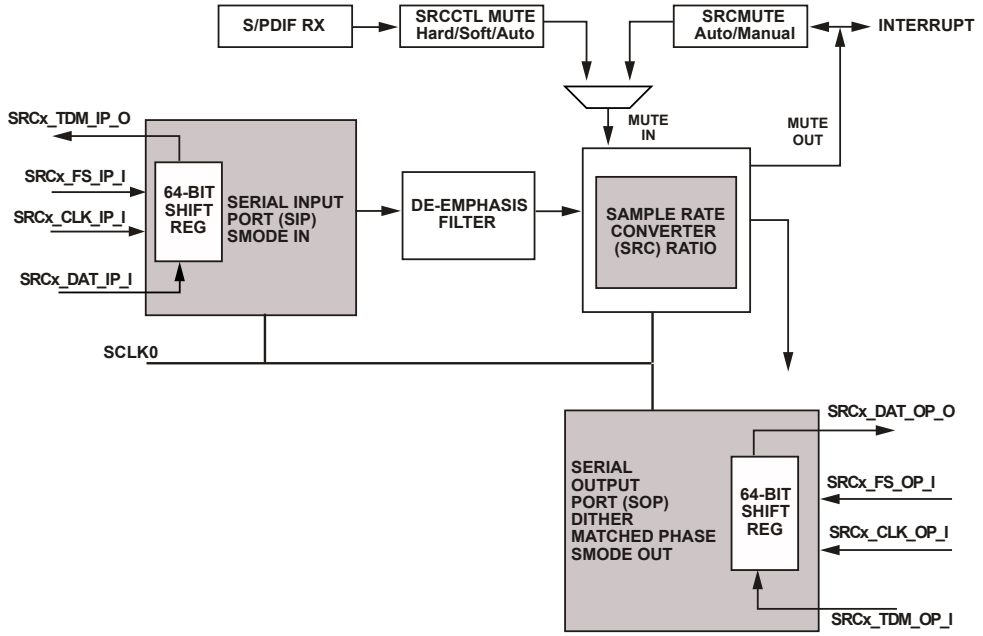


Figure 32-1: ASRC Block Diagram

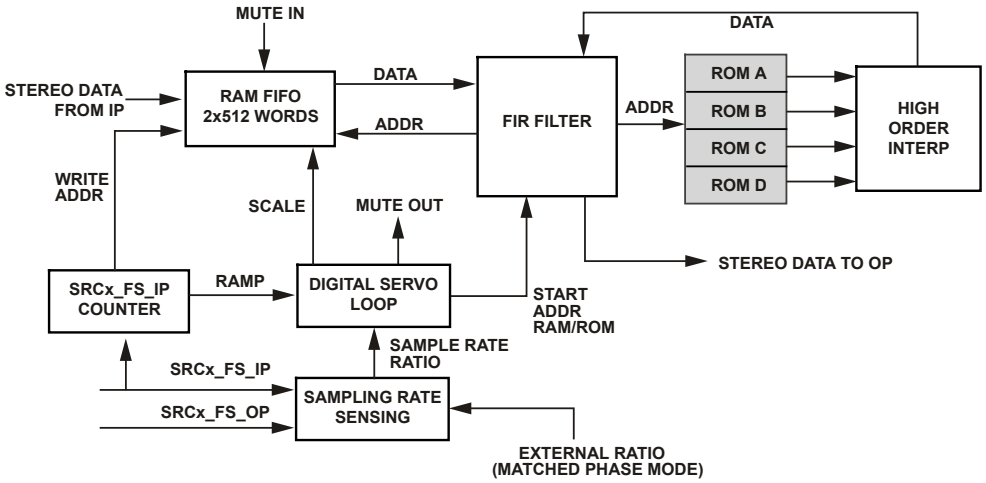


Figure 32-2: ASRC Core Architecture

SRU Programming

The SRU (signal routing unit) needs to be programmed to connect the ASRCs to the output pins or any other peripherals. For more information, see the Digital Audio Interface (DAI) chapter.

Clocking

The ASRC module is in the SCLK0 clock domain. An internal divided version of the SCLK0 clock is generated and used as the fundamental clock for the ASRC module.

I/O Ports

The I/O ports provide the interface through which data is transferred asynchronously into and out of the SRC modules. The SRC has a 3-wire interface for the serial input and output ports that supports left-justified, I^2S, and right-justified (16-, 18-, 20-, 24-bit) modes. Additionally, the serial interfaces support TDM mode for daisy-chaining multiple SRCs to form a frame. The serial output data is dithered down to 20, 18, or 16 bits when 20-, 18-, or 16-bit output data is selected.

NOTE: The SRC converts the data from the serial input port to the sample rate of the serial output port. The sample rate at the serial input port can be asynchronous with respect to the output sample rate of the output serial port.

De-Emphasis Filter

The de-emphasis filter is used to de-emphasize audio data that has been emphasized.

Mute Control

When either the SRC starts up (or there is a change in sample ratio), the mute out signal (`SRCx_MUTEOUT`) is asserted (=1). The mute out signal stays high until the SRC settles on the new sample rates. While mute out is asserted high, the mute in signal should be asserted high as well. The mute in signal performs a soft mute of the audio input data when asserted and un-mutes the input audio data softly when deasserted.

Note that it takes 4096 input port FS samples until the audio input data is completely muted and 4096 FS samples until the audio input data is completely unmuted.

SRC Core

As shown in the ***ASRC Core Architecture*** figure, the sample rate converter's RAM FIFO block adjusts the left and right input samples and stores them for the FIR filter's convolution cycle. The `ASRCx_FS_IP` counter provides the write address (for scaling) to the FIFO block and the ramp input to the digital-servo loop. The ROM stores the coefficients for the FIR filter convolution and performs a high-order interpolation between the stored coefficients. The sample rate ratio block measures the sample rate by dynamically altering the ROM coefficients and scaling the FIR filter length and input data. The digital-servo loop automatically tracks the `SRCx_FS_IP` and `SRCx_FS_OP` sample rates and provides the RAM and ROM start addresses for the start of the FIR filter convolution.

NOTE: Unlike other peripherals, the sample rate converters own local memories (RAM and ROM) which are dedicated for the purpose of sample rate conversion only.

The sample rate converter only operates asynchronously and is always a completer to the input and output ports.

RAM FIFO

The RAM FIFO receives the left and right input data and adjusts the amplitude of the data for both the soft muting of the SRC and the scaling of the input data by the sample rate ratio before storing the samples in RAM. The input data is scaled by the sample rate ratio because as the FIR filter length of the convolution increases, so does the

amplitude of the convolution output. To keep the output of the FIR filter from saturating, the input data is scaled down by multiplying it by (SRCx_FS_OP)/(SRCx_FS_IP) when SRCx_FS_OP<SRCx_FS_IP. The FIFO also scales the input data to mute and stop muting the SRC.

Digital Servo Loop

The digital-servo loop is a ramp filter that provides the initial pointer to the address in RAM and ROM for the start of the FIR convolution. The RAM pointer is the integer output of the ramp filter while the ROM pointer is the fractional part. The digital-servo loop must be able to provide excellent rejection of jitter on the SRCx_FS_IP and SRCx_FS_OP clocks as well as measure the arrival of the SRCx_FS_OP clock within 5 ps. The digital-servo loop also divides the fractional part of the ramp output by the ratio of (SRCx_FS_IP)/(SRCx_FS_OP) for the case when SRCx_FS_IP> SRCx_FS_OP, to dynamically alter the ROM coefficients.

The digital-servo loop is implemented with a multi-rate filter. To settle the digital-servo loop filter quickly at startup or at a change in the sample rate, a fast mode has been added to the filter. When the digital-servo loop starts up or the sample rate is changed, the digital-servo loop kicks into fast mode to adjust and settle on the new sample rate. Upon sensing the digital-servo loop settling down to some reasonable value, the digital-servo loop kicks into normal or slow mode. During fast mode, the SRCx_MUTE_OUT bit of the ASRC is asserted to mute the ASRC input which avoids clicks and pops.

FIR Filter

The FIR filter is a 64-tap filter in the case of SRCx_FS_OP<SRCx_FS_IP and is (SRCx_FS_IP)/(SRCx_FS_OP) × 64 taps for the case when SRCx_FS_IP>SRCx_FS_OP. The FIR filter performs its convolution by loading in the starting address of the RAM address pointer and the ROM address pointer from the digital-servo loop at the start of the SRCx_FS_OP period. The FIR filter then steps through the RAM by decrementing its address by 1 for each tap, and the ROM pointer increments its address by the (SRCx_FS_OP/SRCx_FS_IP) × 2^{20} ratio for SRCx_FS_IP>SRCx_FS_OP or 2^{20} for SRCx_FS_OP <SRCx_FS_IP. Once the ROM address rolls over, the convolution is complete. The convolution is performed for both the left and right channels, and the multiply/accumulate circuit used for the convolution is shared between the channels.

Sample Rate Sensing

The (SRCx_FS_IP)/(SRCx_FS_OP) sample rate ratio circuit is used to dynamically alter the coefficients in the ROM for the case when SRCx_FS_IP>SRCx_FS_OP. The ratio is calculated by comparing the output of an SRCx_FS_OP counter to the output of an SRCx_FS_IP counter. If ASRCx_FS_OP>SRCx_FS_IP, the ratio is held at one. If SRCx_FS_IP>SRCx_FS_OP, the sample rate ratio is updated if it is different by more than two SRCx_FS_OP periods from the previous SRCx_FS_OP to SRCx_FS_IP comparison. This is done to provide some hysteresis to prevent the filter length from oscillating and causing distortion.

Digital Filter Group Delay

The RAM in the FIFO is 512 words deep for both left and right channels. An offset of 16 samples to the write address, provided by the SRCx_FS_IP counter, is added to prevent the RAM read pointer from overlapping the write address. The maximum decimation rate can be calculated from the RAM word: depth = (512 – 16) ÷ 64 taps = 7.75.

The 64 samples effect latency in the interpolation filter. This latency (group delay) depends on interpolation or decimation ratio and is determined as follows:

Interpolation or Decimation Ratio (1): GDL = $16/f_{S_IN} + 32/f_{S_IN}$ seconds for `SRC_FS_OP` > `SRC_FS_IP`

Interpolation or Decimation Ratio (2): GDL = $16/f_{S_IN} + 32/f_{S_IN} \times f_{S_IN}/f_{S_OUT}$ seconds for `SRC_FS_OP` < `SRC_FS_IP`

Data Format

The ***ASRC Data Frame Format by Protocol*** figure shows the data input format for a frame (stereo data). The frame format is valid for all protocols. For models which do not support matched phase mode the 8-bit data field is ignored.

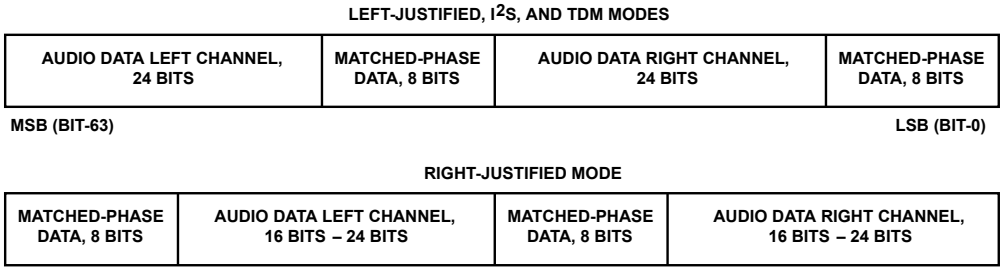


Figure 32-3: ASRC Data Frame Format by Protocol

The *Input/Output Serial Data Formats* figure shows general details for the data formats supported by the ASRC.

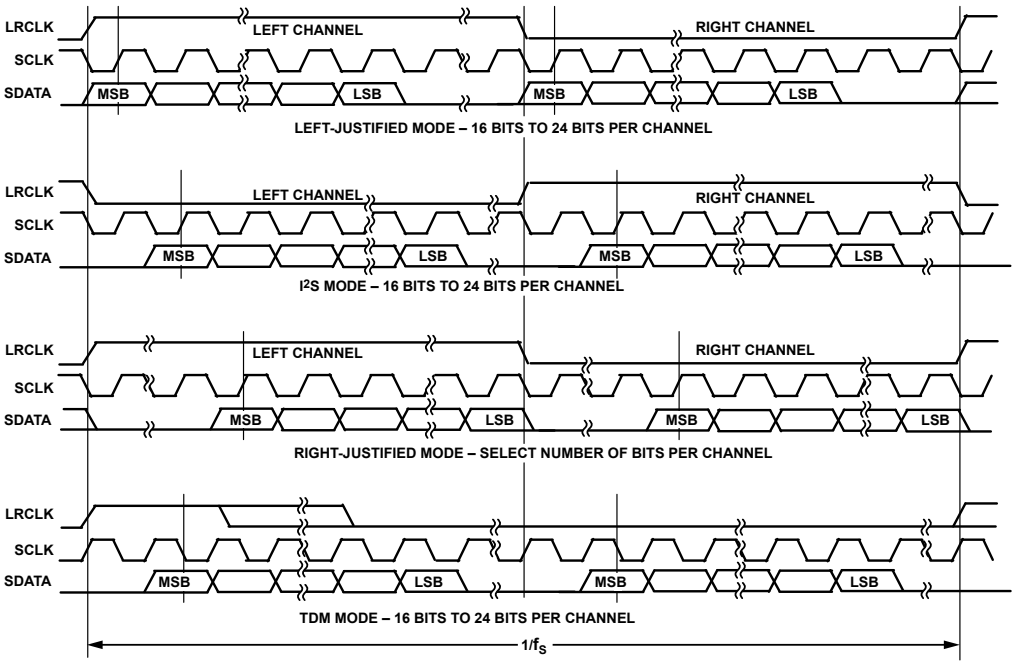


Figure 32-4: Input/Output Serial Data Formats

Operating Modes

The ASRC can operate in TDM, I^2S, left-justified, right-justified, and bypass modes. The serial ports of the processor can be used for moving the ASRC data to/from the internal memory.

In I^2S, left-justified and right-justified modes, the ASRCs operate individually. The serial data provided in the input port is converted to the sample rate of the output port.

TDM Input Mode

In TDM input port, several ASRCs can be daisy-chained together and connected to the serial input port of a SHARC processor or other processor (see the ***TDM Input/Output Modes*** figure). The ASRC IP contains a 64-bit parallel load shift register. When the `SRCx_FS_IP_I` pulse arrives, each ASRC parallel loads its left and right data into the 64-bit shift register. The input to the shift register is connected to `SRCx_DATA_IP_I`, while the output is connected to `SRCx_TDM_IP_O`. By connecting the `SRCx_TDM_IP_O` to the `SRCx_DATA_IP_I` of the next ASRC, a large shift register is created, which is clocked by `SRCx_CLK_IP_I`.

NOTE: In TDM mode, the ASRC drives at the rising edge and samples at the falling edge of the serial clock. In all other modes, the serial clock rising edge is the sampling edge, and the falling edge is the driving edge.

NOTE: The number of ASRCs that can be daisy-chained together is limited by the maximum frequency of `SRCx_CLK_xx_I`, refer to the data sheet for exact values. For example, if the maximum frequency of `SRCx_CLK_xx_I` is *x* MHz, and the output sample rate is *fS*, then number of ASRCs (n) that can be connected in daisy chained fashion is: n 64 FS ≤ *x* MHz.

TDM Output Mode

As shown in the ***TDM Input/Output Modes*** figure, using the TDM output port several ASRCs can be daisy-chained together and connected to the SPORT of this or another processor. The ASRC OP contains a 64-bit parallel load shift register. When the `SRCx_FS_OP_I` pulse arrives, each ASRC loads its left and right data into the 64-bit shift register. The input to the shift register is connected to `SRCx_TDM_OP_I`, and the output is connected to `SRCx_DAT_OP_O`. By connecting the `SRCx_DAT_OP_O` to the `SRCx_TDM_OP_I` of the next ASRC, a large shift register is created, which is clocked by `SRCx_CLK_OP_I`.

As shown in ***TDM Input/Output Modes***, with three ASRCs in a daisy-chain connection, the serial clock for input/or output port is defined as: SCLK = 3 × 64 FS = 192 FS.

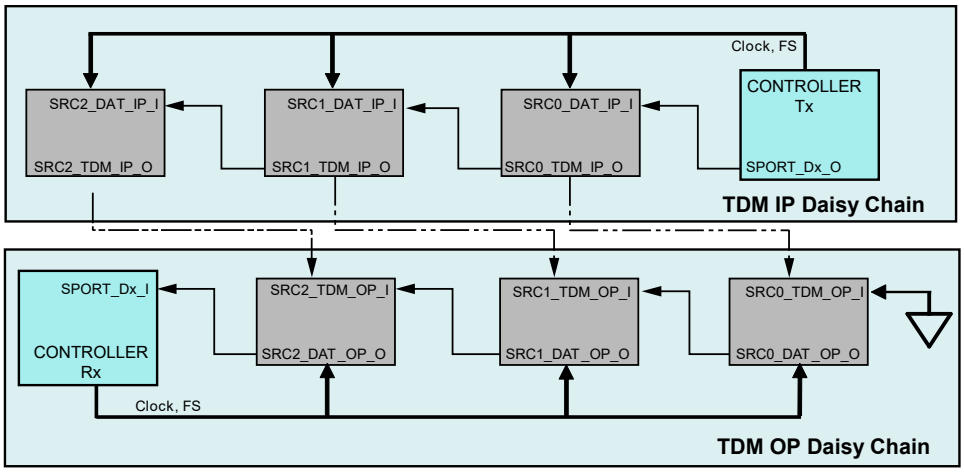


Figure 32-5: TDM Input/Output Modes

Matched-Phase Mode

The matched-phase mode of the sample rate converter, shown in ***Typical Configuration for Matched-Phase Mode Operation***, is enabled by the `ASRC_CTL01.MPHASE0`, `ASRC_CTL01.MPHASE1`, `ASRC_CTL23.MPHASE2` and `ASRC_CTL23.MPHASE3` bits. This mode is used to match the phase (group delay) between two or more adjacent sample rate converters that are operating with the same input and output clocks.

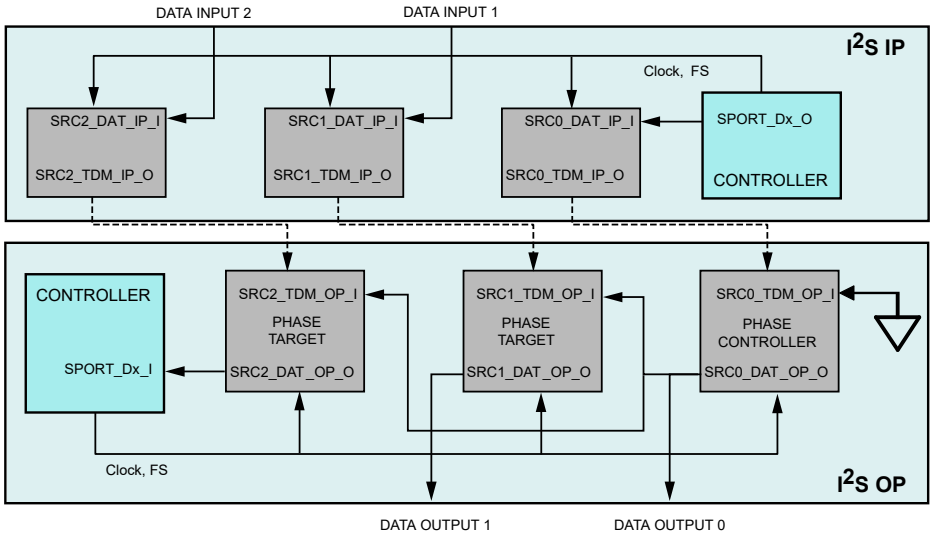


Figure 32-6: Typical Configuration for Matched-Phase Mode Operation

Hysteresis of the (`SRCx_FS_OP`)/(`SRCx_FS_IP`) ratio circuit can cause phase mismatching between two ASRCs operating with the same input and output clocks. Since the hysteresis requires a difference of more than two `SRCx_FS_OP` periods to update the `SRCx_FS_OP` and `SRCx_FS_IP` ratios, two ASRCs may have differences in their ratios from 0 to 4 `SRCx_FS_OP` period counts. The (`SRCx_FS_OP`)/(`SRCx_FS_IP`) ratio adjusts the filter length of the ASRC, which corresponds directly with the group delay. Thus, the magnitude in the phase difference depends upon the resolution of the `SRCx_FS_OP` and `SRCx_FS_IP` counters. The greater the resolution of the counters, the smaller the phase difference error.

When the completer SRC matched-phase mode bit is set (=1), it accepts the sample rate ratio transmitted by another SRC, (the matched-phase requester) which has its matched-phase mode bit cleared (=0), through its serial output.

The phase requester ASRC device transmits its SRCx_FS_OP/SRCx_FS_IP ratio through the data output pin (SRCx_DAT_OP_O) to the completer's ASRC's data input pins (SRCx_TDM_OP_I). The transmitted data (32-bit subframe) contains 24-bit data and 8-bits matched phase (see the ***ASRC Data Frame Format by Protocol*** figure).

The completer SRCs receive the 8-bit matched phase bits (instead of their own internally-derived ratio) if their SRCx_MPHASE bits are set to 1, respectively. The SRCx_FS_IP and SRCx_FS_OP signals may be asynchronous with respect to each other in this mode. Note that there must be 64 SRCx_CLK_OP cycles per frame in matched-phase mode (two 24-bits data and two 8-bits phase match).

NOTE: By default, matched phased data is sent on the SRCx_DAT_OP_O pin, but only if the SRCx_TDM_OP_I pin is tied low. The completers' simply ignore the matched phased data if their ASRC_CTL01.MPHASE1 through ASRC_CTL23.MPHASE3 bits are cleared (= 0).

Bypass Mode

When the ASRC_CTL01.BYP0, ASRC_CTL01.BYP1, ASRC_CTL23.BYP2 and ASRC_CTL23.BYP3 bits are set (=1), the input data bypasses the sample rate converter and is sent directly to the serial output port. Dithering is disabled. This mode is ideal when the input and output sample rates are the same and the SRCx_FS_IP_I and SRCx_FS_OP_I signals are synchronous with respect to each other. In matched phase bypass mode, the SRCx_FS_OP_I signal should come at least one SRCx_CLK_xx_I period before SRCx_FS_IP_I. Cases where this is not met could result in data loss. For example, if internal SPORTS are used then the SRCx_FS_OP_I and SRCx_FS_IP_I signals could be driven by different SPORTs so that the timing of these signals can be controlled by enabling them at different times. This mode can also be used for passing through non-audio data since no processing is performed on the input data.

De-Emphasis Mode

The ASRC_CTL01.DEEMPHASIS0, ASRC_CTL01.DEEMPHASIS1, ASRC_CTL23.DEEMPHASIS2 and ASRC_CTL23.DEEMPHASIS3 bits choose the type of de-emphasis filter based on the input sample rate for 32, 44.1 or 48 kHz sampling rates.

Dithering Mode

The ASRC_CTL01.DITHER0, ASRC_CTL01.DITHER1, ASRC_CTL23.DITHER2, and ASRC_CTL23.DITHER3 control this mode of operation. Serial output data is dithered down to 20, 18, or 16 bits when 20-, 18-, or 16-bit output data is selected. In the case of 20-, 18-, and 16-bit word lengths, the least significant bits of the 24-bit word coming from the SRC into the serial output port are truncated. The DITHER_EN signal (not user configurable) automatically adds dithering to the 24-bit word before truncating to the appropriate output word length. The 21BIT_DITHER signal is used for the consumer version of the SRC to reduce the dynamic range performance to approximately 128 dB.

NOTE: The ASRC can be programmed to add the triangular Probability Distribution Function (PDF) dither to the digital audio samples. It is advisable to add dither when the input word width exceeds the output word width, for example the input word is 20 bits, and the output word is 16 bits. A triangular PDF is considered to create the most favorable noise shaping of the residual quantization noise.

Muting Modes

The mute feature of the ASRC can be controlled automatically in hardware using the MUTE_IN signal by connecting it to the MUTE_OUT signal. Automatic muting can be disabled by setting (=1) the ASRC_MUTE.MUTE0 through ASRC_MUTE.MUTE3 bits.

NOTE: Note that by default, the ASRC_MUTE register connects the MUTE_IN signal to the MUTE_OUT signal, but not conversely.

Soft Mute

When the ASRC_CTL01.SOFTMUTE0, ASRC_CTL01.SOFTMUTE1, ASRC_CTL23.SOFTMUTE2 and ASRC_CTL23.SOFTMUTE3 bits are set, the MUTE_IN signal is asserted, and the ASRC performs a soft mute by linearly decreasing the input data to the ASRC FIFO to zero, (-144 dB) attenuation as described for automatic hardware muting.

A 12-bit counter, clocked by SRCx_FS_IP_I, is used to control the mute attenuation. Therefore, the time it takes from the assertion of the MUTE_IN signal to -144 dB, full mute attenuation is 4096 FS cycles. Likewise, the time it takes to reach 0 dB mute attenuation from the deassertion of the MUTE_IN signal is 4096 FS cycles.

Hard Mute

When the ASRC_CTL01.HARDMUTE0, ASRC_CTL01.HARDMUTE1, ASRC_CTL23.HARDMUTE2 and ASRC_CTL23.HARDMUTE3 bits are set, the ASRC immediately mutes the input data to the ASRC FIFO to zero, (-144 dB) attenuation.

Auto Mute

When the ASRC_CTL01.AUTOMUTE0, ASRC_CTL01.AUTOMUTE1, ASRC_CTL23.AUTOMUTE2 and ASRC_CTL23.AUTOMUTE3 bits are set, the ASRC communicates with the S/PDIF receiver peripheral to determine when the input should mute.

This mode is useful for automatic detection of non-PCM audio data received from the S/PDIF receiver.

Interrupts

The following sections provide information about interrupt sources, masking, and servicing.

Sources

Each ASRC module drives one interrupt signal (mute out asserted). All these signals are connected into the DAI_IRPTL_H or DAI_IRPTL_L latch registers. The ASRC ports generate interrupts as described below.

SRC Mute Out

The SRC mute out signal can be used to generate interrupts on their rising edge, falling edge, or both, depending on how the DAI interrupt mask registers (DAI_IMSK_RE/DAI_IMSK_FE) are programmed. This programming allows the generation of DAI_IRPTL_H/DAI_IRPTL_L interrupts either entering mute, exiting muting or both. The SRCx_MUTE_OUT interrupt is generated only once when the SRC is locked (after 4096 FS input samples) and after changes to the sample ratio. Hard mute, soft mute, and auto mute only control the muting of the input data to the SRC.

Masking

The DAI_IMSK_FE, DAI_IMSK_RE, and DAI_IMSK_PRI registers must be unmasked accordingly. The DAI_IRQH and DAI_IRQL signals are routed to the system event controller (SEC) and general interrupt controller (GIC).

Service

The ISR reads the DAI_IRPTL_H and DAI_IRPTL_L registers to clear the interrupt request.

Programming Model

The following is basic information on programming the ASRC module.

1. Program the ASRC_CTL01 and ASRC_CTL23 registers and keep the ASRC_CTL01.EN0 through ASRC_CTL23.EN3 bits cleared.
2. Set the ASRC_CTL01.EN0 through ASRC_CTL23.EN3 bits. After 4096 input port FS cycles the ASRC has un-muted.

Debug Features

The asynchronous sample rate converter allow the bypass mode. When the ASRC_CTL01.BYP0 through ASRC_CTL23.BYP3 bits are set (=1), the input data bypasses the sample rate converter and is sent directly to the serial output port. This mode can be used for testing both ports when the input and output sample rates are at the same frequency, therefore both input and output ports can be routed to the same serial clock and frame sync.

ADSP-2184x ASRC Register Descriptions

Sample Rate Converter Module (ASRC) contains the following registers.

Table 32-3: ADSP-2184x ASRC Register List

Name	Description
ASRC_CTL01	Control Register for ASRC 0 and 1
ASRC_CTL23	Control Register for ASRC 2 and 3

Table 32-3: ADSP-2184x ASRC Register List (Continued)

Name	Description
ASRC_CTL45	Control Register For ASRC 4 And 5
ASRC_CTL67	Control Register for ASRC 6 and 7
ASRC_MUTE	Mute Register
ASRC_RAT01	Ratio Register for ASRC 0 and 1
ASRC_RAT23	Ratio Register for ASRC 2 and 3
ASRC_RAT45	Ratio Register For ASRC 4 and 5
ASRC_RAT67	Ratio Register for ASRC 6 and 7

Control Register for ASRC 0 and 1

The ASRC_CTL01 register (read/write) controls the operating modes, filters, and data formats used in the ASRC modules 0 and 1.

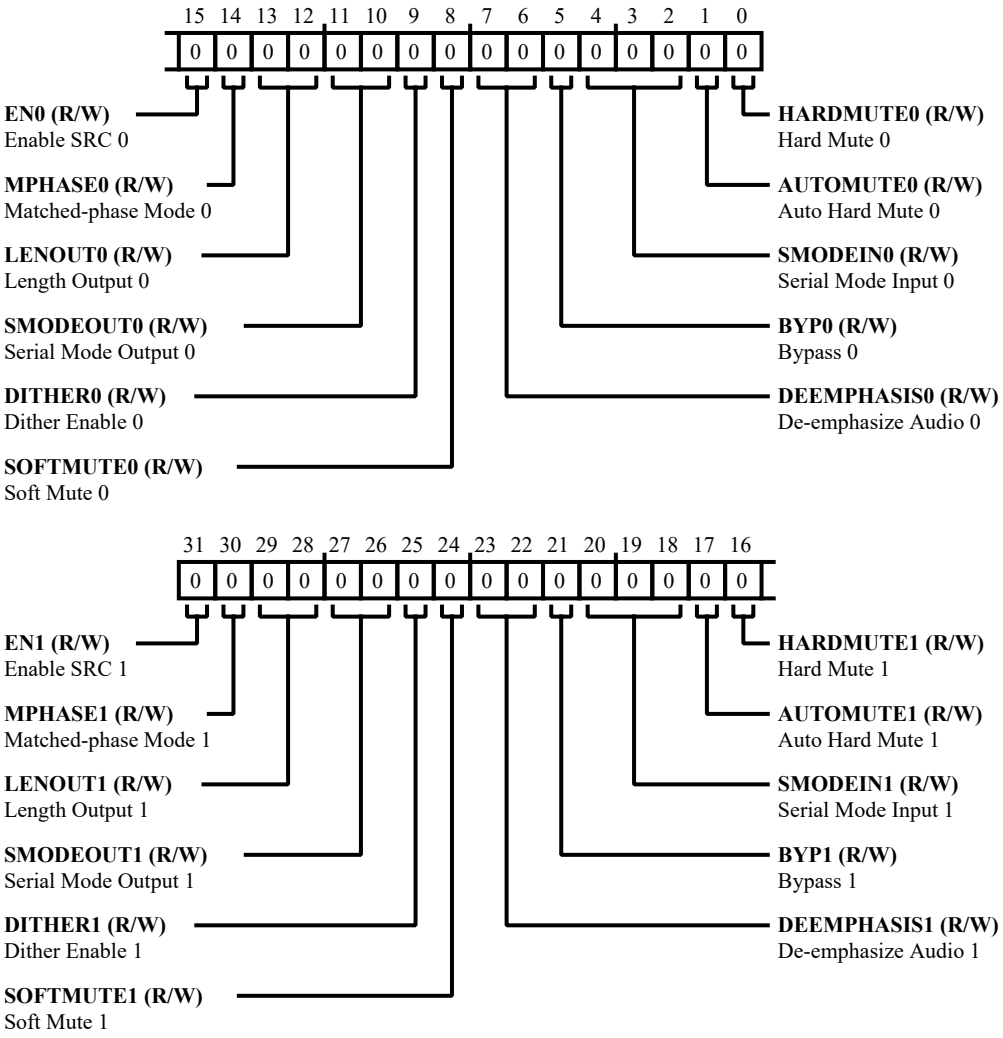


Figure 32-7: ASRC_CTL01 Register Diagram

Table 32-4: ASRC_CTL01 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN1	Enable SRC 1. The ASRC_CTL01.EN1 bit enables SRC 1. When (set = 1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT01.MUTEOUT1 bit. Writes to the ASRC_CTL01 register should be at least one cycle before setting the ASRC_CTL01.EN1 bit. When setting and clearing this bit, it should be held low for a minimum of 5 SCLK cycles.	
30 (R/W)	MPHASE1	Matched-phase Mode 1. The ASRC_CTL01.MPHASE1 bit configures SRC1 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
29:28 (R/W)	LENOUT1	Length Output 1. The ASRC_CTL01.LENOUT1 bit field selects the serial output word length on SRC1.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
27:26 (R/W)	SMODEOUT1	Serial Mode Output 1. The ASRC_CTL01.SMODEOUT1 bit field selects the serial output format on SRC1.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
25 (R/W)	DITHER1	Dither Enable 1. The ASRC_CTL01.DITHER1 bit enables dithering before truncation on SRC1 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-4: ASRC_CTL01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SOFTMUTE1	Soft Mute 1. The `ASRC_CTL01.SOFTMUTE1` bit enables soft mute on SRC1.	
		0	Unmute
		1	Mute
23:22 (R/W)	DEEMPHASIS1	De-emphasize Audio 1. The `ASRC_CTL01.DEEMPHASIS1` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
21 (R/W)	BYP1	Bypass 1. The `ASRC_CTL01.BYP1` bit makes the output of SRC1 the same as the input.	
20:18 (R/W)	SMODEIN1	Serial Mode Input 1. The `ASRC_CTL01.SMODEIN1` bit field selects the serial input format for SRC1.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
17 (R/W)	AUTOMUTE1	Auto Hard Mute 1. The `ASRC_CTL01.AUTOMUTE1` bit auto hard mutes SRC1 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
16 (R/W)	HARDMUTE1	Hard Mute 1. The `ASRC_CTL01.HARDMUTE1` bit hard mutes SRC1.	
		0	Unmute
		1	Mute

Table 32-4: ASRC_CTL01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN0	Enable SRC 0. The ASRC_CTL01.EN0 bit enables SRC 0. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT01.MUTEOUT0 bit. Writes to the ASRC_CTL01 register should be at least one cycle before setting the ASRC_CTL01.EN0 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
14 (R/W)	MPHASE0	Matched-phase Mode 0. The ASRC_CTL01.MPHASE0 bit configures SRC0 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
13:12 (R/W)	LENOUT0	Length Output 0. The ASRC_CTL01.LENOUT0 bit field selects the serial output word length on SRC0.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
11:10 (R/W)	SMODEOUT0	Serial Mode Output 0. The ASRC_CTL01.SMODEOUT0 bit field selects the serial output format on SRC0.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
9 (R/W)	DITHER0	Dither Enable 0. The ASRC_CTL01.DITHER0 bit enables dithering before truncation on SRC0 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-4: ASRC_CTL01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SOFTMUTE0	Soft Mute 0. The `ASRC_CTL01.SOFTMUTE0` bit enables soft mute on SRC0.	
		0	Unmute
		1	Mute
7:6 (R/W)	DEEMPHASIS0	De-emphasize Audio 0. The `ASRC_CTL01.DEEMPHASIS0` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
5 (R/W)	BYP0	Bypass 0. The `ASRC_CTL01.BYP0` bit makes the output of SRC0 the same as the input.	
4:2 (R/W)	SMODEIN0	Serial Mode Input 0. The `ASRC_CTL01.SMODEIN0` bit field selects the serial input format for SRC0.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
1 (R/W)	AUTOMUTE0	Auto Hard Mute 0. The `ASRC_CTL01.AUTOMUTE0` bit auto hard mutes SRC0 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
0 (R/W)	HARDMUTE0	Hard Mute 0. The `ASRC_CTL01.HARDMUTE0` bit hard mutes SRC0.	
		0	Unmute
		1	Mute (default)

Control Register for ASRC 2 and 3

The ASRC_CTL23 register (read/write) controls the operating modes, filters, and data formats used in the sample rate converter modules 2 and 3.

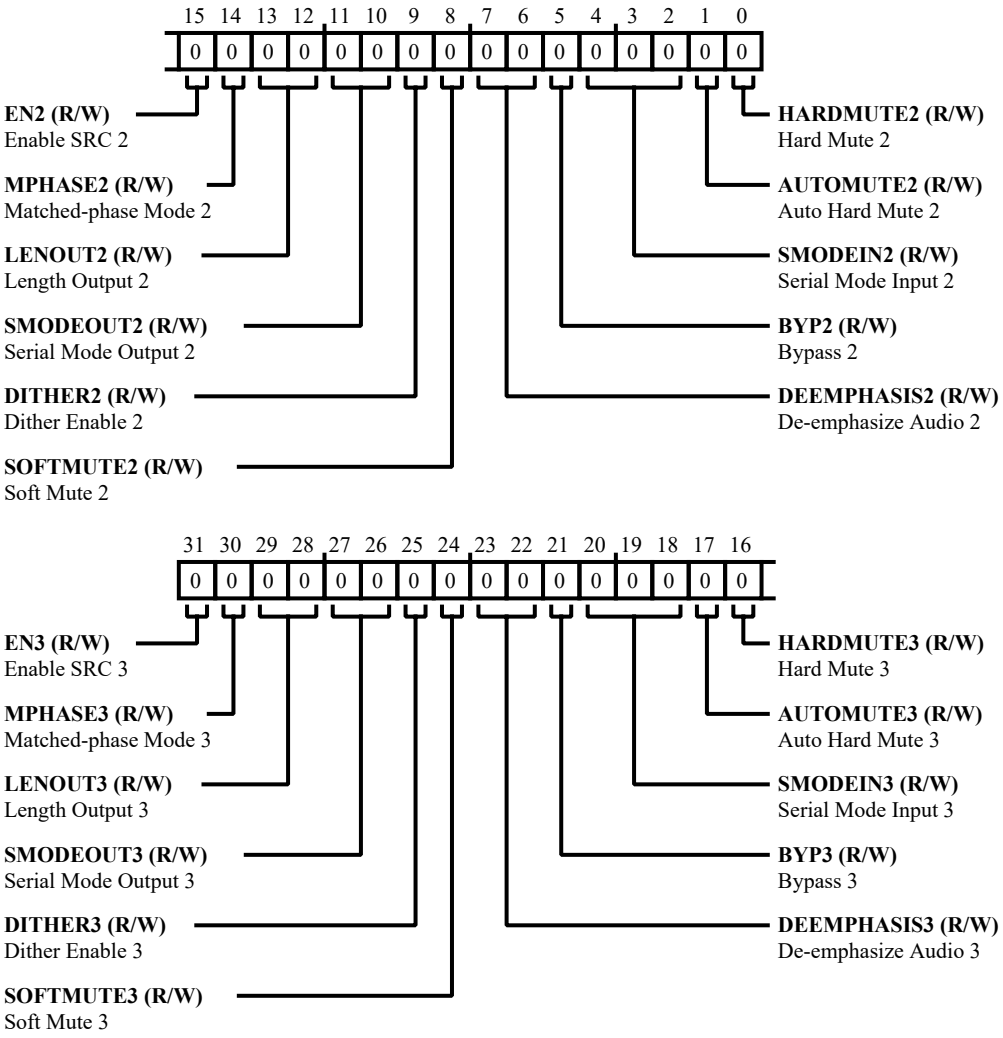


Figure 32-8: ASRC_CTL23 Register Diagram

Table 32-5: ASRC_CTL23 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN3	Enable SRC 3. The ASRC_CTL23.EN3 bit enables SRC 3. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT23.MUTEOUT3 bit. Writes to the ASRC_CTL23 register should be at least one cycle before setting the ASRC_CTL23.EN3 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
30 (R/W)	MPHASE3	Matched-phase Mode 3. The ASRC_CTL23.MPHASE3 bit configures SRC3 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
29:28 (R/W)	LENOUT3	Length Output 3. The ASRC_CTL23.LENOUT3 bit field selects the serial output word length on SRC3.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
27:26 (R/W)	SMODEOUT3	Serial Mode Output 3. The ASRC_CTL23.SMODEOUT3 bit field selects the serial output format on SRC3.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
25 (R/W)	DITHER3	Dither Enable 3. The ASRC_CTL23.DITHER3 bit enables dithering before truncation on SRC3 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-5: ASRC_CTL23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SOFTMUTE3	Soft Mute 3. The ASRC_CTL23.SOFTMUTE3 bit enables soft mute on SRC3.	
		0	Unmute
		1	Mute
23:22 (R/W)	DEEMPHASIS3	De-emphasize Audio 3. The ASRC_CTL23.DEEMPHASIS3 bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
21 (R/W)	BYP3	Bypass 3. The ASRC_CTL23.BYP3 bit makes the output of SRC3 the same as the input.	
20:18 (R/W)	SMODEIN3	Serial Mode Input 3. The ASRC_CTL23.SMODEIN3 bit field selects the serial input format for SRC3.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
17 (R/W)	AUTOMUTE3	Auto Hard Mute 3. The ASRC_CTL23.AUTOMUTE3 bit auto hard mutes SRC3 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
16 (R/W)	HARDMUTE3	Hard Mute 3. The ASRC_CTL23.HARDMUTE3 bit hard mutes SRC3.	
		0	Unmute
		1	Mute

Table 32-5: ASRC_CTL23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN2	Enable SRC 2. The ASRC_CTL23.EN2 bit enables SRC 2. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT23.MUTEOUT2 bit. Writes to the ASRC_CTL23 register should be at least one cycle before setting the ASRC_CTL23.EN2 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
14 (R/W)	MPHASE2	Matched-phase Mode 2. The ASRC_CTL23.MPHASE2 bit configures SRC2 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
13:12 (R/W)	LENOUT2	Length Output 2. The ASRC_CTL23.LENOUT2 bit field selects the serial output word length on SRC2.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
11:10 (R/W)	SMODEOUT2	Serial Mode Output 2. The ASRC_CTL23.SMODEOUT2 bit field selects the serial output format on SRC2.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
9 (R/W)	DITHER2	Dither Enable 2. The ASRC_CTL23.DITHER2 bit enables dithering before truncation on SRC2 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-5: ASRC_CTL23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SOFTMUTE2	Soft Mute 2. The ASRC_CTL23.SOFTMUTE2 bit enables soft mute on SRC2.	
		0	Unmute
		1	Mute
7:6 (R/W)	DEEMPHASIS2	De-emphasize Audio 2. The ASRC_CTL23.DEEMPHASIS2 bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
5 (R/W)	BYP2	Bypass 2. The ASRC_CTL23.BYP2 bit makes the output of SRC2 the same as the input.	
4:2 (R/W)	SMODEIN2	Serial Mode Input 2. The ASRC_CTL23.SMODEIN2 bit field selects the serial input format for SRC2.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
1 (R/W)	AUTOMUTE2	Auto Hard Mute 2. The ASRC_CTL23.AUTOMUTE2 bit auto hard mutes SRC2 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
0 (R/W)	HARDMUTE2	Hard Mute 2. The ASRC_CTL23.HARDMUTE2 bit hard mutes SRC2.	
		0	Unmute
		1	Mute

Control Register For ASRC 4 And 5

The ASRC_CTL45 register (read/write) controls the operating modes, filters, and data formats used in the sample rate converter modules 4 and 5.

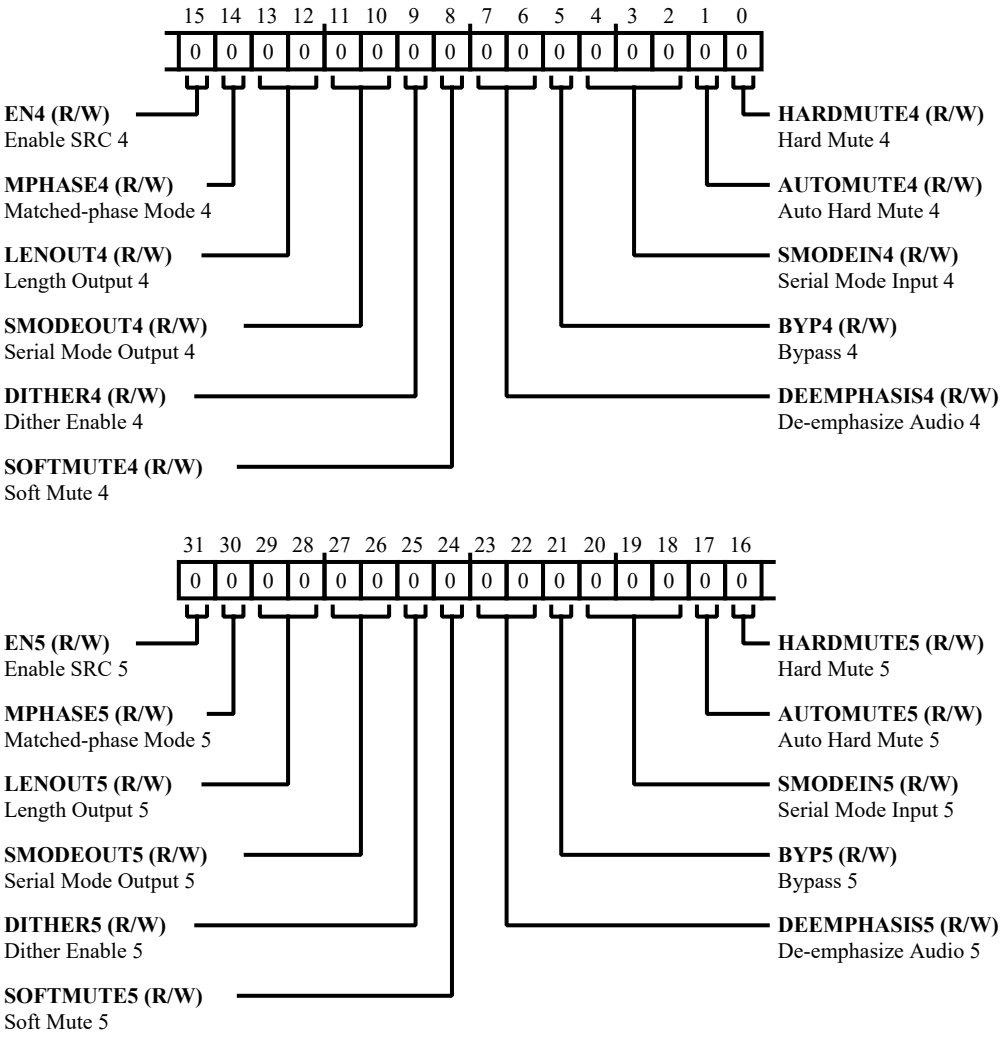


Figure 32-9: ASRC_CTL45 Register Diagram

Table 32-6: ASRC_CTL45 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN5	Enable SRC 5. The ASRC_CTL45.EN5 bit enables SRC 5. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT45.MUTEOUT5 bit. Writes to the ASRC_CTL45 register should be at least one cycle before setting the ASRC_CTL45.EN5 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
30 (R/W)	MPHASE5	Matched-phase Mode 5. The ASRC_CTL45.MPHASE5 bit configures SRC5 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
29:28 (R/W)	LENOUT5	Length Output 5. The ASRC_CTL45.LENOUT5 bit field selects the serial output word length on SRC5.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
27:26 (R/W)	SMODEOUT5	Serial Mode Output 5. The ASRC_CTL45.SMODEOUT5 bit field selects the serial output format on SRC5.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
25 (R/W)	DITHER5	Dither Enable 5. The ASRC_CTL45.DITHER5 bit enables dithering before truncation on SRC3 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-6: ASRC_CTL45 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SOFTMUTE5	Soft Mute 5. The `ASRC_CTL45.SOFTMUTE5` bit enables soft mute on SRC5.	
		0	Unmute
		1	Mute
23:22 (R/W)	DEEMPHASIS5	De-emphasize Audio 5. The `ASRC_CTL45.DEEMPHASIS5` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
21 (R/W)	BYP5	Bypass 5. The `ASRC_CTL45.BYP5` bit makes the output of SRC5 the same as the input.	
20:18 (R/W)	SMODEIN5	Serial Mode Input 5. The `ASRC_CTL45.SMODEIN5` bit field selects the serial input format for SRC5.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
17 (R/W)	AUTOMUTE5	Auto Hard Mute 5. The `ASRC_CTL45.AUTOMUTE5` bit auto hard mutes SRC5 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
16 (R/W)	HARDMUTE5	Hard Mute 5. The `ASRC_CTL45.HARDMUTE5` bit hard mutes SRC5.	
		0	Unmute
		1	Mute

Table 32-6: ASRC_CTL45 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN4	Enable SRC 4. The ASRC_CTL45.EN4 bit enables SRC 4. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT45.MUTEOUT4 bit. Writes to the ASRC_CTL45 register should be at least one cycle before setting the ASRC_CTL45.EN4 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
14 (R/W)	MPHASE4	Matched-phase Mode 4. The ASRC_CTL45.MPHASE4 bit configures SRC4 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
13:12 (R/W)	LENOUT4	Length Output 4. The ASRC_CTL45.LENOUT4 bit field selects the serial output word length on SRC4.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
11:10 (R/W)	SMODEOUT4	Serial Mode Output 4. The ASRC_CTL45.SMODEOUT4 bit field selects the serial output format on SRC4.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
9 (R/W)	DITHER4	Dither Enable 4. The ASRC_CTL45.DITHER4 bit enables dithering before truncation on SRC2 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-6: ASRC_CTL45 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SOFTMUTE4	Soft Mute 4. The `ASRC_CTL45.SOFTMUTE4` bit enables soft mute on SRC4.	
		0	Unmute
		1	Mute
7:6 (R/W)	DEEMPHASIS4	De-emphasize Audio 4. The `ASRC_CTL45.DEEMPHASIS4` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
5 (R/W)	BYP4	Bypass 4. The `ASRC_CTL45.BYP4` bit makes the output of SRC4 the same as the input.	
4:2 (R/W)	SMODEIN4	Serial Mode Input 4. The `ASRC_CTL45.SMODEIN4` bit field selects the serial input format for SRC4.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
1 (R/W)	AUTOMUTE4	Auto Hard Mute 4. The `ASRC_CTL45.AUTOMUTE4` bit auto hard mutes SRC4 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
0 (R/W)	HARDMUTE4	Hard Mute 4. The `ASRC_CTL45.HARDMUTE4` bit hard mutes SRC4.	
		0	Unmute
		1	Mute

Control Register for ASRC 6 and 7

The ASRC_CTL67 register (read/write) controls the operating modes, filters, and data formats used in the sample rate converter modules 6 and 7.

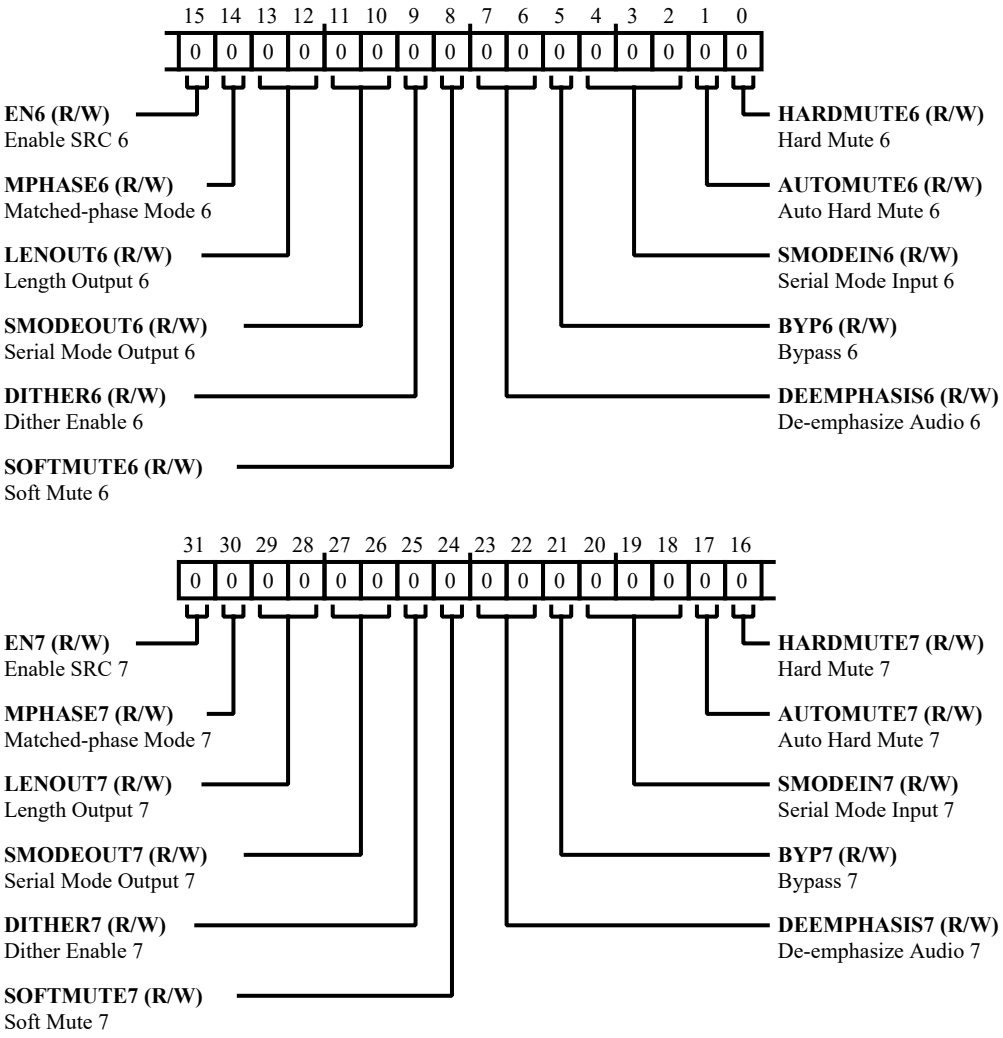


Figure 32-10: ASRC_CTL67 Register Diagram

Table 32-7: ASRC_CTL67 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN7	Enable SRC 7. The ASRC_CTL67.EN7 bit enables SRC 7. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT67.MUTEOUT7 bit. Writes to the ASRC_CTL67 register should be at least one cycle before setting the ASRC_CTL67.EN7 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
30 (R/W)	MPHASE7	Matched-phase Mode 7. The ASRC_CTL67.MPHASE7 bit configures SRC7 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
29:28 (R/W)	LENOUT7	Length Output 7. The ASRC_CTL67.LENOUT7 bit field selects the serial output word length on SRC7.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
27:26 (R/W)	SMODEOUT7	Serial Mode Output 7. The ASRC_CTL67.SMODEOUT7 bit field selects the serial output format on SRC7.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
25 (R/W)	DITHER7	Dither Enable 7. The ASRC_CTL67.DITHER7 bit enables dithering before truncation on SRC7 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-7: ASRC_CTL67 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SOFTMUTE7	Soft Mute 7. The ASRC_CTL67.SOFTMUTE7 bit enables soft mute on SRC7.	
		0	Unmute
		1	Mute
23:22 (R/W)	DEEMPHASIS7	De-emphasize Audio 7. The ASRC_CTL67.DEEMPHASIS7 bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
21 (R/W)	BYP7	Bypass 7. The ASRC_CTL67.BYP7 bit makes the output of SRC7 the same as the input.	
20:18 (R/W)	SMODEIN7	Serial Mode Input 7. The ASRC_CTL67.SMODEIN7 bit field selects the serial input format for SRC7.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
17 (R/W)	AUTOMUTE7	Auto Hard Mute 7. The ASRC_CTL67.AUTOMUTE7 bit auto hard mutes SRC7 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
16 (R/W)	HARDMUTE7	Hard Mute 7. The ASRC_CTL67.HARDMUTE7 bit hard mutes SRC7.	
		0	Unmute
		1	Mute

Table 32-7: ASRC_CTL67 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN6	Enable SRC 6. The ASRC_CTL67.EN6 bit enables SRC 6. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (-144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT67.MUTEOUT6 bit. Writes to the ASRC_CTL67 register should be at least one cycle before setting the ASRC_CTL67.EN6 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
14 (R/W)	MPHASE6	Matched-phase Mode 6. The ASRC_CTL67.MPHASE6 bit configures SRC6 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
13:12 (R/W)	LENOUT6	Length Output 6. The ASRC_CTL67.LENOUT6 bit field selects the serial output word length on SRC6.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
11:10 (R/W)	SMODEOUT6	Serial Mode Output 6. The ASRC_CTL67.SMODEOUT6 bit field selects the serial output format on SRC6.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
9 (R/W)	DITHER6	Dither Enable 6. The ASRC_CTL67.DITHER6 bit enables dithering before truncation on SRC6 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 32-7: ASRC_CTL67 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SOFTMUTE6	Soft Mute 6. The `ASRC_CTL67.SOFTMUTE6` bit enables soft mute on SRC6.	
		0	Unmute
		1	Mute
7:6 (R/W)	DEEMPHASIS6	De-emphasize Audio 6. The `ASRC_CTL67.DEEMPHASIS6` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
5 (R/W)	BYP6	Bypass 6. The `ASRC_CTL67.BYP6` bit makes the output of SRC6 the same as the input.	
4:2 (R/W)	SMODEIN6	Serial Mode Input 6. The `ASRC_CTL67.SMODEIN6` bit field selects the serial input format for SRC6.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
1 (R/W)	AUTOMUTE6	Auto Hard Mute 6. The `ASRC_CTL67.AUTOMUTE6` bit auto hard mutes SRC6 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
0 (R/W)	HARDMUTE6	Hard Mute 6. The `ASRC_CTL67.HARDMUTE6` bit hard mutes SRC6.	
		0	Unmute
		1	Mute

Mute Register

This register connects an ASRCx mute input and output when the mute bit is cleared (=0). This allows ASRCx to automatically mute input while the ASRC is initializing (0 = automatic muting and 1 = manual muting). Bit 0 controls ASRC0, bit 1 controls ASRC1, bit 2 controls ASRC2, and bit 3 controls ASRC3.

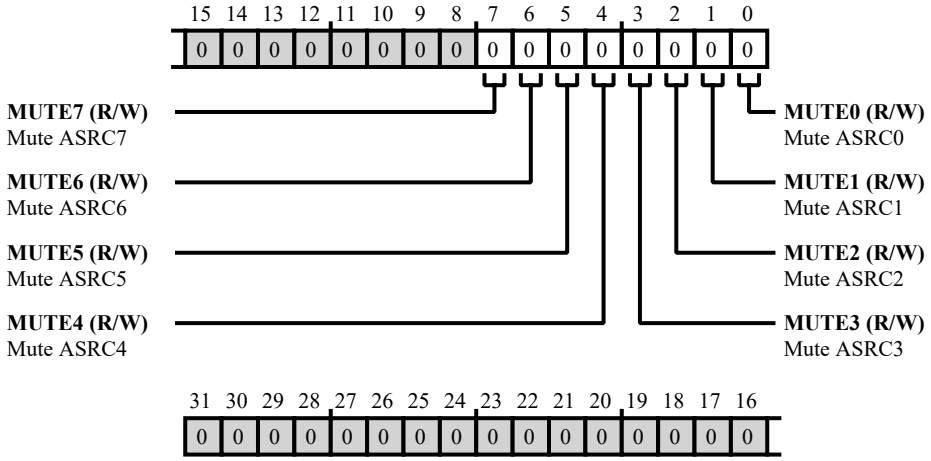


Figure 32-11: ASRC_MUTE Register Diagram

Table 32-8: ASRC_MUTE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	MUTE7	Mute ASRC7. The `ASRC_MUTE.MUTE7` bit automatically mutes ASCR7 output when cleared (=0).
6 (R/W)	MUTE6	Mute ASRC6. The `ASRC_MUTE.MUTE6` bit automatically mutes ASCR6 output when cleared (=0).
5 (R/W)	MUTE5	Mute ASRC5. The `ASRC_MUTE.MUTE5` bit automatically mutes ASCR5 output when cleared (=0).
4 (R/W)	MUTE4	Mute ASRC4. The `ASRC_MUTE.MUTE4` bit automatically mutes ASCR4 output when cleared (=0).
3 (R/W)	MUTE3	Mute ASRC3. The `ASRC_MUTE.MUTE3` bit automatically mutes ASCR3 output when cleared (=0).

Table 32-8: ASRC_MUTE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W)	MUTE2	Mute ASRC2. The `ASRC_MUTE.MUTE2` bit automatically mutes ASCR2 output when cleared (=0).
1 (R/W)	MUTE1	Mute ASRC1. The `ASRC_MUTE.MUTE1` bit automatically mutes ASCR1 output when cleared (=0).
0 (R/W)	MUTE0	Mute ASRC0. The `ASRC_MUTE.MUTE0` bit automatically mutes ASCR0 output when cleared (=0).

Ratio Register for ASRC 0 and 1

The ASRC_RAT01 register report the mute and I/O sample ratio for ASRC0 and ASRC1.

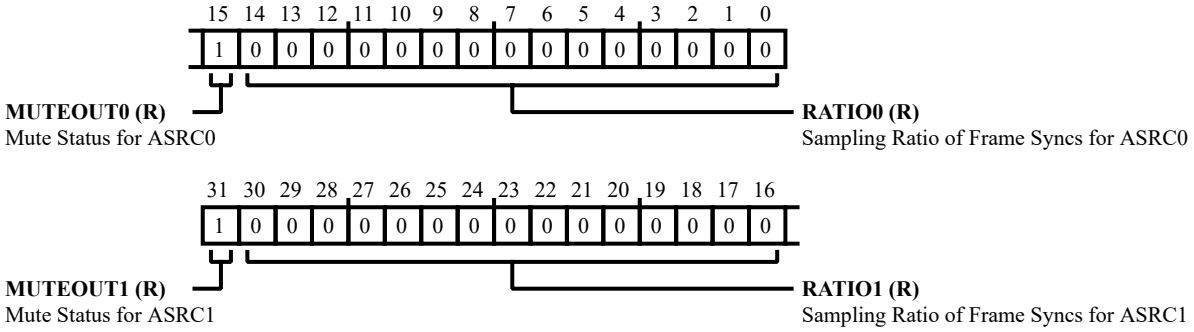


Figure 32-12: ASRC_RAT01 Register Diagram

Table 32-9: ASRC_RAT01 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MUTEOUT1	Mute Status for ASRC1. The ASRC_RAT01.MUTEOUT1 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC1 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.
30:16 (R/NW)	RATIO1	Sampling Ratio of Frame Syncs for ASRC1. The ASRC_RAT01.RATIO1 bit field is read to find the ratio of output to input sampling frequency for ASRC1 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.
15 (R/NW)	MUTEOUT0	Mute Status for ASRC0. The ASRC_RAT01.MUTEOUT0 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC0 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.

Table 32-9: ASRC_RAT01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/NW)	RATIO0	Sampling Ratio of Frame Syncs for ASRC0. The `ASRC_RAT01.RATIO0` bit field is read to find the ratio of output to input sampling frequency for ASRC0 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.

Ratio Register for ASRC 2 and 3

The ASRC_RAT23 register report the mute and I/O sample ratio for ASRC0 and ASRC1.

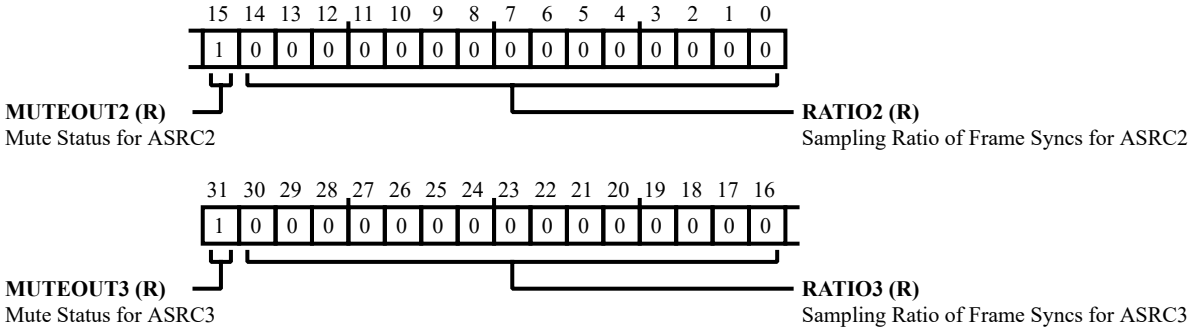


Figure 32-13: ASRC_RAT23 Register Diagram

Table 32-10: ASRC_RAT23 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MUTEOUT3	Mute Status for ASRC3. The ASRC_RAT23.MUTEOUT3 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC3 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.
30:16 (R/NW)	RATIO3	Sampling Ratio of Frame Syncs for ASRC3. The ASRC_RAT23.RATIO3 bit field is read to find the ratio of output to input sampling frequency for ASRC3 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.
15 (R/NW)	MUTEOUT2	Mute Status for ASRC2. The ASRC_RAT23.MUTEOUT2 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC2 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.

Table 32-10: ASRC_RAT23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/NW)	RATIO2	Sampling Ratio of Frame Syncs for ASRC2. The `ASRC_RAT23.RATIO2` bit field is read to find the ratio of output to input sampling frequency for ASRC2 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.

Ratio Register For ASRC 4 and 5

The ASRC_RAT45 register report the mute and I/O sample ratio for ASRC4 and ASRC5.

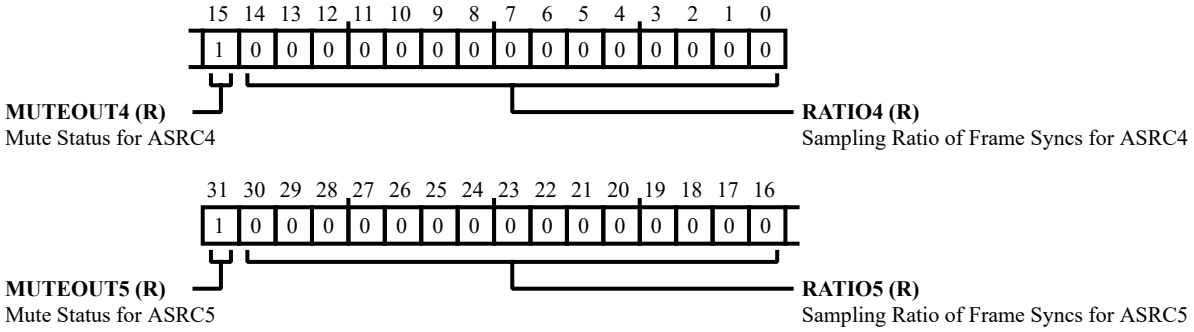


Figure 32-14: ASRC_RAT45 Register Diagram

Table 32-11: ASRC_RAT45 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MUTEOUT5	Mute Status for ASRC5. The ASRC_RAT45.MUTEOUT5 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC5 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.
30:16 (R/NW)	RATIO5	Sampling Ratio of Frame Syncs for ASRC5. The ASRC_RAT45.RATIO5 bit field is read to find the ratio of output to input sampling frequency for ASRC5 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.
15 (R/NW)	MUTEOUT4	Mute Status for ASRC4. The ASRC_RAT45.MUTEOUT4 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC4 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.

Table 32-11: ASRC_RAT45 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/NW)	RATIO4	Sampling Ratio of Frame Syncs for ASRC4. The `ASRC_RAT45.RATIO4` bit field is read to find the ratio of output to input sampling frequency for ASRC4 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.

Ratio Register for ASRC 6 and 7

The ASRC_RAT67 register report the mute and I/O sample ratio for ASRC6 and ASRC7.

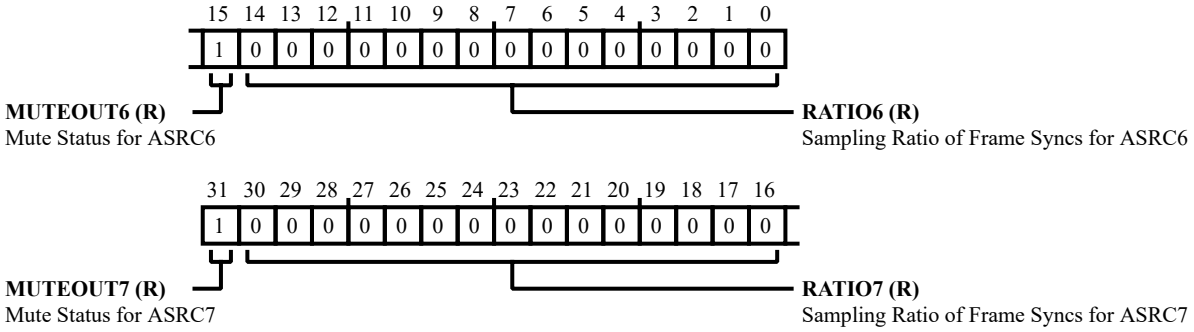


Figure 32-15: ASRC_RAT67 Register Diagram

Table 32-12: ASRC_RAT67 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MUTEOUT7	Mute Status for ASRC7. The ASRC_RAT67.MUTEOUT7 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC7 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.
30:16 (R/NW)	RATIO7	Sampling Ratio of Frame Syncs for ASRC7. The ASRC_RAT67.RATIO7 bit field is read to find the ratio of output to input sampling frequency for ASRC7 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.
15 (R/NW)	MUTEOUT6	Mute Status for ASRC6. The ASRC_RAT67.MUTEOUT6 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC6 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loop's internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.

Table 32-12: ASRC_RAT67 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/NW)	RATIO6	Sampling Ratio of Frame Syncs for ASRC6. The `ASRC_RAT67.RATIO6` bit field is read to find the ratio of output to input sampling frequency for ASRC6 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.

33 Sony/Philips Digital Interface (S/PDIF)

The Sony/Philips Digital Interface (S/PDIF) is a standard audio data transfer format that allows the transfer of digital audio signals from one device to another without having to convert them to an analog signal. The digital audio interface carries three types of information; audio data, non-audio data (compressed data), and timing information.

Features

The S/PDIF interface has the following features.

- Supports one stereo channel or compressed audio streams.
- AES3-compatible S/PDIF transmitter and receiver.
- Transmitting a biphase mark encoded signal that may contain any number of audio channels (compressed or linear pulse code modulation) or non-audio data.
- S/PDIF receiver managing clock recovery with separate S/PDIF on-chip PLL.
- S/PDIF receiver supports the detection of DTS frames of 256, 512, 1024, 2048, and 4096.
- Manage user status information and provide error-handling capabilities in both the transmitter and receiver.
- DAI allows interactions over DAI by serial ports and the external DAI pins to interface to other S/PDIF devices. This includes using the receiver to decode incoming biphase encoded audio streams and passing them via the SPORTs to internal memory for processing-or using the transmitter to encode audio or digital data and transfer it to another S/PDIF receiver in the audio system.

It is important to be familiar with serial digital audio interface standards IEC-60958, EIAJ CP-340, AES3 and AES11.

ADSP-2184x SPDIF Register List

The S/PDIF module is a standard audio data transfer format that allows the transfer of digital audio signals from one device to another without having to convert them to an analog signal. A set of registers govern S/PDIF operations. For more information on S/PDIF functionality, see the S/PDIF register descriptions.

Table 33-1: ADSP-2184x SPDIF Register List

Name	Description
SPDIF_RX_CTL	Receive Control
SPDIF_RX_STAT	Receive Status Register
SPDIF_RX_STAT0_A	Receive Status A0 Register
SPDIF_RX_STAT0_B	Receive Status B0 Register
SPDIF_RX_STAT1_A	Receive Status A1 Register
SPDIF_RX_STAT1_B	Receive Status B1 Register
SPDIF_TX_CTL	Transmit Control Register
SPDIF_TX_STAT_A0	Transmit Status A0 Register
SPDIF_TX_STAT_A1	Transmit Status A1 Register
SPDIF_TX_STAT_A2	Transmit Status A2 Register
SPDIF_TX_STAT_A3	Transmit Status A3 Register
SPDIF_TX_STAT_A4	Transmit Status A4 Register
SPDIF_TX_STAT_A5	Transmit Status A5 Register
SPDIF_TX_STAT_B0	Transmit Status B0 Register
SPDIF_TX_STAT_B1	Transmit Status B1 Register
SPDIF_TX_STAT_B2	Transmit Status B2 Register
SPDIF_TX_STAT_B3	Transmit Status B3 Register
SPDIF_TX_STAT_B4	Transmit Status B4 Register
SPDIF_TX_STAT_B5	Transmit Status B5 Register
SPDIF_TX_UBUFF_A0	Transmit User Buffer A0 Register
SPDIF_TX_UBUFF_A1	Transmit User Buffer A1 Register
SPDIF_TX_UBUFF_A2	Transmit User Buffer A2 Register
SPDIF_TX_UBUFF_A3	Transmit User Buffer A3 Register
SPDIF_TX_UBUFF_A4	Transmit User Buffer A4 Register
SPDIF_TX_UBUFF_A5	Transmit User Buffer A5 Register
SPDIF_TX_UBUFF_B0	Transmit User Buffer B0 Register
SPDIF_TX_UBUFF_B1	Transmit User Buffer B1 Register
SPDIF_TX_UBUFF_B2	Transmit User Buffer B2 Register
SPDIF_TX_UBUFF_B3	Transmit User Buffer B3 Register
SPDIF_TX_UBUFF_B4	Transmit User Buffer B4 Register
SPDIF_TX_UBUFF_B5	Transmit User Buffer B5 Register

Table 33-1: ADSP-2184x SPDIF Register List (Continued)

Name	Description
SPDIF_TX_USRUPDT	User Bit Update Register

SRU Programming

The SRU (signal routing unit) is used to connect the S/PDIF transmitter biphase data out to the output pins or to the S/PDIF receiver. The serial clock, frame sync, data, and external sync (if external synchronization is required) inputs also need to be routed through the SRU. For details of the routing, see DAI Routing Capabilities in the *Digital Audio Interface (DAI)* chapter.

The SRU needs to be programmed in order to connect the S/PDIF receiver to the output pins or any other peripherals and also for the connection to the input biphase stream.

Program the corresponding SRU registers to connect the outputs to the required destinations (see DAI Routing Capabilities inputs to the receiver are routed through the SRU. The extracted clock, frame sync, and data are also routed through the SRU.

S/PDIF Interrupt List

The S/PDIF interrupts are controlled through the DAI.

Table 33-2: S/PDIF Interrupt List

Interrupt Name	Interrupt Condition	Return DAI Register	Interrupt ID
DAI0_IRQH	Block Start	DAIx_IRPTL	64
DAI0_IRQL	Non Audio		66

Clocking

The fundamental timing clock of the S/PDIF receiver is SCLK1_0 from the CDU. When SCLK1_0 is configured, it supports sampling frequencies of 32 kHz to 192 kHz. The fundamental timing clock of the S/PDIF transmitter is *SCLK0*/4.

For information on clock programming, see CDU Programming Model.

S/PDIF Transmitter

The following sections provide information on the S/PDIF transmitter.

Functional Description

The S/PDIF transmitter, shown in the ***S/PDIF Transmitter Block Diagram***, resides within the DAI, and its inputs and outputs can be routed via the SRU. It receives audio data in serial format, encloses the specified user status

information, and converts it into the biphase encoded signal. The serial data input to the transmitter can be formatted as left-justified, I^{2}S, or right-justified with word widths of 16, 18, 20 or 24 bits. ***AES3 Output Block*** shows the detail of the AES block.

The serial data, clock, and frame sync inputs to the S/PDIF transmitter are routed through the signal routing unit (SRU).

The S/PDIF transmitter output may be routed to an output pin via the SRU and then routed to another S/PDIF receiver or to components for off-board connections to other S/PDIF receivers. The output is also available to the S/PDIF receiver for loop-back testing through SRU.

In addition to encoding the audio data in the bi-phase format, the transmitter also provides a way to easily add the channel status information to the outgoing bi-phase stream. There are status/user registers for a frame (192-bits/24 bytes) in the transmitter that correspond to each channel or subframe.

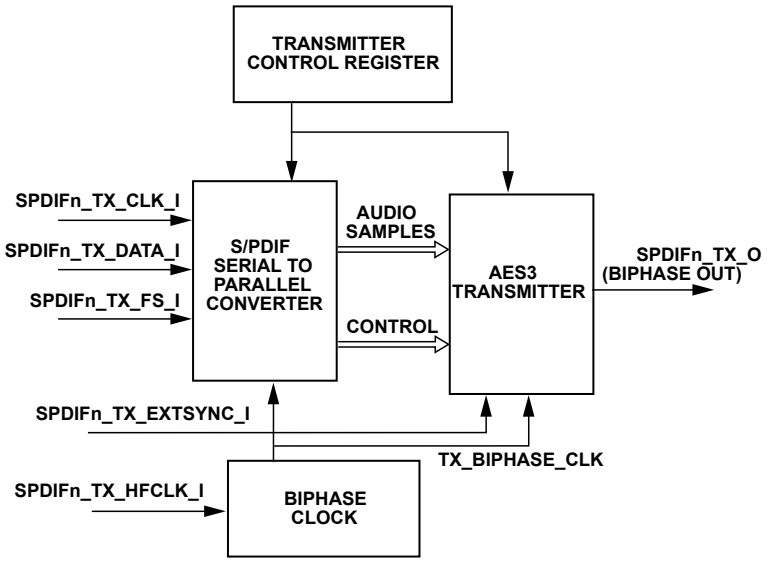


Figure 33-1: S/PDIF Transmitter Block Diagram

Validity bits for both channels may also be controlled by the transmitter control register. Optionally, the user bit, validity bit, and channel status bit are sent to the transmitter with each left/right sample. For each subframe the parity bit is automatically generated and inserted into the biphase encoded data.

A mute control and support for double-frequency single-channel mode are also provided. The serial data input format may be selected as left-justified, I^{2}S, or right-justified with 16-, 18-, 20-, or 24-bit word widths. The over sampling clock is also selected by the transmitter control register.

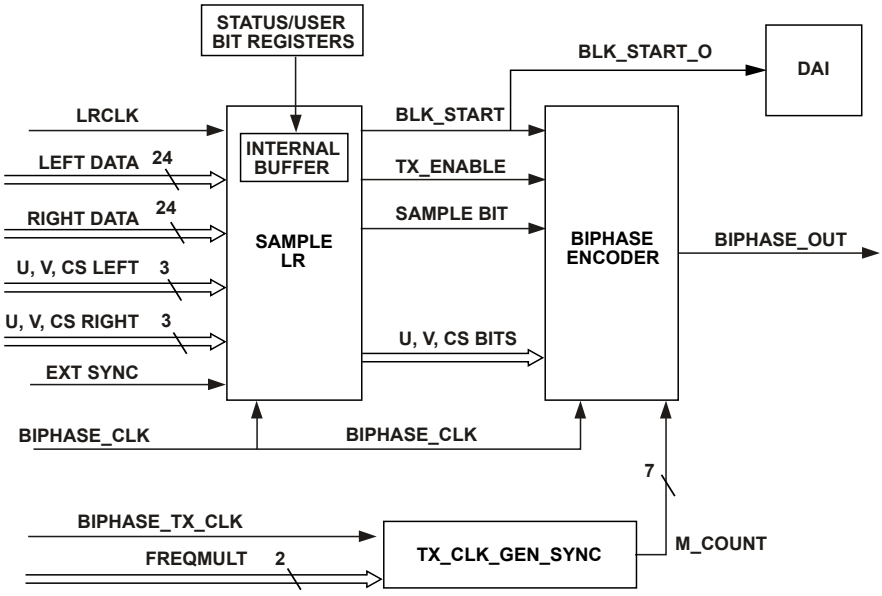


Figure 33-2: AES3 Output Block

Input Data Formats

The *I2S and Left-Justified Formats* and *Right-Justified Formats* figures show the format of data that is sent to the S/PDIF transmitter using a variety of protocol standards.

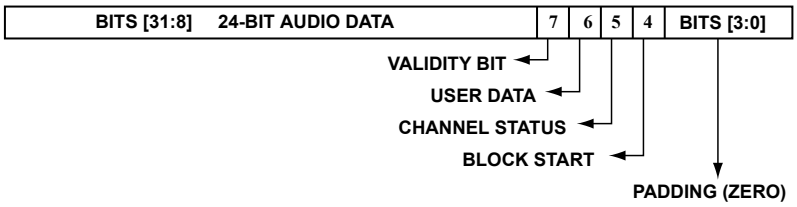


Figure 33-3: I^2S and Left-Justified Formats

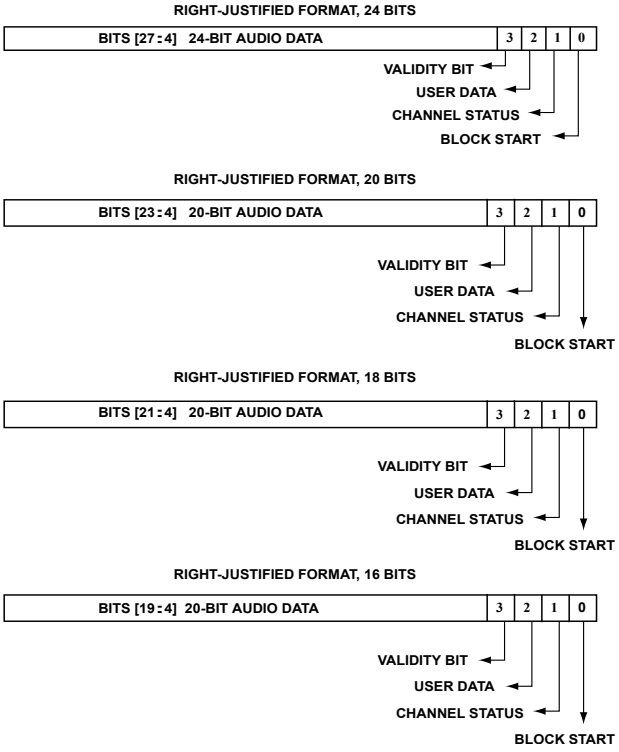


Figure 33-4: Right-Justified Formats

Operating Modes

The S/PDIF transmitter can operate in standalone and full serial modes. The following sections describe these modes in detail.

Full Serial Mode

This mode is selected by clearing the `SPDIF_TX_CTL.AUTO` bit. In this mode all the status bits, audio data and the block start bit (indicating start of a frame), come through the serial data stream (`SPDIF_TX_DATA_I`) pin. The transmitter should be enabled after or at the same time as all of the other control bits.

Standalone Mode

This mode is selected by setting the `SPDIF_TX_CTL.AUTO` bit. In this mode, the block start bit (indicating the start of a frame) is generated internally. The channel status bits come from the channel status buffer registers. The user status bits come from the user bits buffers as shown in the ***AES3 Output Block*** figure.

The validity bits are the `SPDIF_TX_CTL.VALIDR` and `SPDIF_TX_CTL.VALIDL`. In this mode only audio data comes from the `SPDIF_TX_DATA_I` pin. All other data, including the status bit and block start bit is either generated internally or taken from the internal register.

Once the user bits buffer registers (`SPDIF_TX_UBUFF_A0` - `SPDIF_TX_UBUFF_B5`) are programmed, they are used only for the next block of data. This allows programs to change the user bit information in every block of data.

To allow user bit updates, write a 0x1 to the SPDIF_TX_USRUPDT register that is used for further processing. If the SPDIF_TX_CTL.AUTO bit is set:

- and if SPDIF_TX_USRUPDT =1, at every 192nd frame end the user status bits are taken from user bits buffers and transmitted. Simultaneously, the SPDIF_TX_USRUPDT register is cleared automatically by hardware.
- and if SPDIF_TX_USRUPDT =0, at every 192nd frame end the user status bits are updated as zeros and transmitted. The SPDIF_TX_USRUPDT register remains low.

In general, for the next block, programs can update user bits buffers at any time during the transfer of the current block (1 block = 192 frames). There are internal buffers to store the user status bits of the current block of transfer. In other words, at the beginning of every new block, the user status bit (SPDIF_TX_CTL.USRPEND bit) from user bits buffers are copied to internal buffers and transmitted in each frame during the transfer.

Note that since a frame contains 192 bits/8 = 24 bytes, six status/user registers are required to store each four bytes.

Data Output Mode

Two output data formats are supported by the transmitter; *two channel mode* and *single-channel double-frequency* (SCDF) mode. The output format is determined by the transmitter control register (SPDIF_TX_CTL).

In two channel mode, the left channel (channel A) is transmitted when the SPDIF_TX_FS_I is high, and the right channel (channel B) is transmitted when the SPDIF_TX_FS_I is low.

In SCDF mode, the transmitter sends successive audio samples of the same signal across both sub frames, instead of channel A and B. The transmitter transmits at half the sample rate of the input bit stream. The SPDIF_TX_CTL.SCDF bit selects SCDF mode. When in SCDF mode, the SPDIF_TX_CTL.SCDFLR bit determines whether left or right channel data is transmitted.

S/PDIF Receiver

The S/PDIF receiver (***S/PDIF Receiver Block Diagram***) is compliant with all common serial digital audio interface standards including IEC-60958, IEC-61937, AES3, and AES11. For the IEC-60958 standard, all the user-data and channel-status bits (as outlined in this document) are not decoded by the S/PDIF receiver. The interface does make all 192 user-data and channel-status pairs available as an output of the block, for post-decoding.

For the IEC-61937 standard, the S/PDIF only detects compressed AC-3 and DTS formats. No decompression is performed.

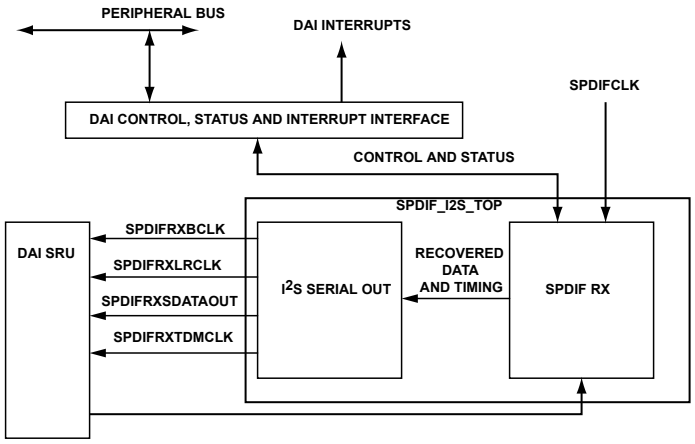


Figure 33-5: S/PDIF Receiver Block Diagram

Functional Description

If the receiver is used, programs need to enable it using the `SPDIF_RX_CTL` register. After the SRU programming is complete, write to the register with control values. At this point, the receiver attempts to lock.

NOTE: The S/PDIF receiver is disabled at default. If the receiver is used in an application, programs should enable the receiver.

The input to the receiver (SPDIFn_RX_I) is a biphase encoded signal that may contain two audio channels (compressed or linear PCM) or non-audio data. The receiver decodes the single biphase encoded stream, producing an I^2S compatible serial data output that consists of a serial clock, a left-right frame sync, and data (channel A/B). It provides the programmer with several methods of managing the incoming status bit information.

The S/PDIF receiver receives any S/PDIF stream with a sampling frequency range of 24 kHz to 192 kHz. Refer to Clocking for more details.

The channel status bits are collected into memory-mapped registers, while other channel status and user bytes must be handled manually. The block start bit, which replaces the parity bit in the serial I^2S stream, indicates the reception of the Z preamble and the start of a new block of channel status and data bits.

Clock Recovery

The S/PDIF receiver recovers the clock that generated the AES3/SPDIF biphase encoded stream from the incoming S/PDIF stream.

This clock is used by the receiver to clock in the biphase encoded data stream and also to provide clocks for either the SPORTs, sample rate converter, or the AES3 and S/PDIF transmitter. The recovered clock may also be used externally to the chip for clocking D/A and A/D converters.

In order to maintain performance, jitter on the clock is sourced to several peripherals.

To comply with the AES11 standard, the recovered left or right clock must be aligned with the preambles within a + or - 5% of the frame period. Since the PLL clock generates a clock 512 times the frame rate clock (512 f_{SCLK}), this clock can be used and divided down to create the phase aligned jitter-free left or right clock.

Output Data Format

The extracted 24-bit audio data, V, U, C, and block start bits are sent on the SPDIF_RX_DAT pin in 32-bit I^2S format as shown in ***I2S and Left-Justified Formats***. The frame sync is transmitted on the SPDIF_RX_FS pin and serial clock is transmitted on the SPDIF_RX_CLK pin. All three pins are routed through the SRU.

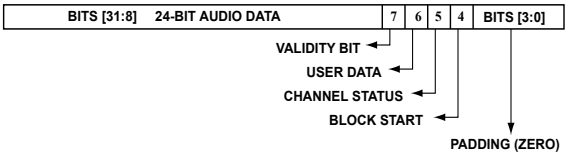


Figure 33-6: I^2S Format

Channel Status

The channel status for the first bytes 4-0 (consumer mode) are collected into memory-mapped registers (SPDIF_RX_STAT0_A, SPDIF_RX_STAT0_B, SPDIF_RX_STAT1_A and SPDIF_RX_STAT1_B). All other channel status bytes 23-5 (professional mode) must be manually extracted from the receiver data stream.

NOTE: Only the first 5 channel status bytes (40-bit) for consumer mode of a frame are stored into the S/PDIF receiver status registers.

Operating Modes

This section describes the receiver channel status for the different modes.

Compressed or Non-linear Audio Data

The S/PDIF receiver processes compressed as well as non-linear audio data according to the supported standards. The following sections describe how this peripheral handles different data.

MPEG-2, AC-3, DTS, and AAC compressed data may be transmitted without setting either the SPDIF_RX_STAT.VALID bit or bit 1 of byte 0. To detect this data, the IEC61937 and SPMTE 337M standards dictate that there be a 96-bit sync code in the 16-, 20- or 24-bit audio data stream. This sync code consists of four words of zeros followed by a word consisting of 0xF872 and another word consisting of 0x4E1F. When this sync code is detected, the SPDIF_RX_STAT.COMPMODE bits hold the information regarding type of compression.

The last two words of the sync code, 0xF872 and 0x4E1F, are called the preamble-A and preamble-B of the burst preamble. Preamble-C of the burst preamble contains burst information and is captured and stored by the receiver. Preamble-D of the burst preamble contains the length code and is captured by the receiver. Even if the validity bit or bit 1 of byte 0 has been set, the receiver still looks for the sync code in order to record the preamble-C and D values. Once the sync code has not been detected in 4096 frames, the preamble-C and D registers are set to zero.

Emphasized Audio Data

Determination as to whether the received audio data is emphasized or not is done in software using the channel status bits as detailed below.

- In professional mode, (bit 0 of byte 0 = 1), channel status bits 2-4 of byte 0 indicate the audio data is emphasized if they are equal to 110 or 111.
- In consumer mode, (bit 0 of byte 0 = 0), channel status bits 3-5 indicate the audio data is emphasized if they are equal to 100, 010 or 110.

Single-Channel Double-Frequency Mode

Unlike previous processors, support for single-channel, double-frequency mode (SCDF) is not supported through specific bits within the `SPDIF_RX_CTL` register, but must be implemented in software using the information provided by the CS (channel status) bits.

- 0111–single channel double frequency mode
- 1000–single channel double frequency mode-stereo left
- 1001–single channel double frequency mode-stereo right

Clock Recovery Modes

The S/PDIF receiver extracts audio data, channel status, and user bits from the biphase encoded AES3 and S/PDIF stream. In addition, a 50% duty cycle reference clock running at the sampling rate of the audio input data is generated for the receiver to recover the oversampling clock.

Number Controlled Oscillator

The receiver can recover the clock from the biphase encoded stream using an on-chip NCO shown in the following figure. The dedicated NCO is separate from the PLL that supplies the clock to the processor core, which is the default operation of the receiver.

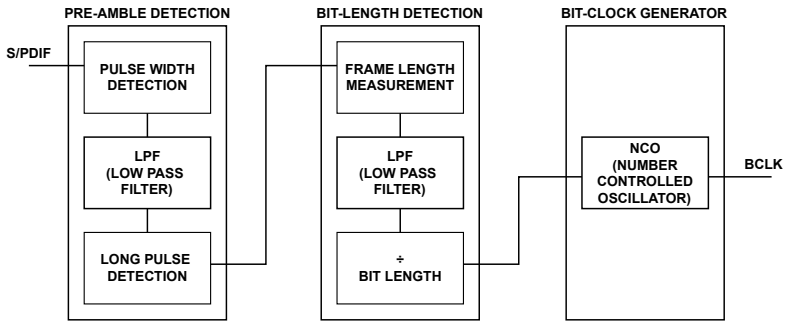


Figure 33-7: S/PDIF Clock Recovery Mechanism

The left/right frame reference clock for the NCO is generated using the preambles. The recovered low jitter left/right frame clock from the NCO attempts to align with the reference clock. However, this recovered left/right clock, like the reference clock, is not phase aligned with the preambles.

Interrupts

The following sections provide information about interrupt sources, masking, and servicing.

Sources

The S/PDIF module of each DAI drives five interrupt signals. Four are status signals driven from SPDIFn_RX and one signal is driven from SPDIFn_TX (block start). These signals are connected into the DAI_IRPTL_L/ DAI_IRPTL_H latch register.

Transmit Block Start

The SPDIFn_TX_BLKSTART output signal, if routed to any miscellaneous interrupt bits (DAIn_INT_31-22 in the DAI_MISC0/DAI_MISC1 registers), triggers a block start interrupt during the last frame of current block.

Receiver Status

The following receiver status bits generate an interrupt.

- Validity (SPDIF_RX_STAT.VALID)
- Receiver locked (SPDIF_RX_STAT.LOCK)
- No audio (SPDIF_RX_STAT.AUDIOTYPE)

Receiver Error

The loss of lock (SPDIF_RX_STAT.LOCKLOSS) bit generates an interrupt.

Masking

For the S/PDIF receiver the DAI_IMSK_RE register must be unmasked accordingly. For the S/PDIF transmit the DAIn_IMASK_x register must be unmasked accordingly.

The INTR_DAI_IRQH and INTR_DAI_IRQL signals are routed to the SEC.

Service

The ISR reads the DAI_IRPTL_H and DAI_IRPTL_L registers to clear the interrupt request.

Programming Model

The following sections provide information on programming the transmitter and receiver.

Programming the Transmitter

Since the S/PDIF transmitter data input is not available to the core, programming the transmitter is as simple as: 1) connecting the SRU to the on-chip (serial ports or input data port) or off-chip (DAI pins) serial devices that provide the clock and data to be encoded, and 2) selecting the desired mode in the transmitter control register. This setup can be accomplished in three steps.

1. Connect the transmitter's four required input signals and one biphase encoded output in the SRU. The four input signals are the serial clock (SPDIF_TX_CLK_I), the serial frame sync (SPDIF_TX_FS_I), the serial data (SPDIF_TX_DAT_I), and the high frequency clock (SPDIF_TX_HFCLK_I) used for the encoding. The only output of the transmitter is SPDIF_TX_O.
2. If user bits are required, write 0x1 to the SPDIF_TX_USRUPDT register for the first block of transfer. Also route the SPDIF_TX_BLK_START_O signal to the DAI_INT_31-22 (DAI_IRPTLx register). This generates interrupts during the last frame of the block (192), allowing changes of user bits for the next block.
3. Initialize the SPDIF_TX_CTL register to enable the data encoding.
4. Manually set the block start bit in the data stream once per block (every 384 words). This is necessary if automatic generation of block start information is not enabled using the SPDIF_TX_CTL.AUTO bit = 0.

NOTE: For more information, see the "DAI Routing Capabilities" section of the *Digital Audio Interface (DAI)* chapter.

Programming the Receiver

Since the S/PDIF receiver data output is not available to the core, programming the peripheral is as simple as connecting the SRU to the on-chip (serial ports) or off-chip (DAI pins) serial devices that provide the clock and data to be decoded, and selecting the desired mode in the receiver control register. This setup can be accomplished in two steps.

1. Connect the input signal and three output signals in the SRU. The only input of the receiver is the biphase encoded stream, SPDIFn_RX_I. The three required output signals are the serial clock (SPDIFn_RX_CLK), the serial frame sync (SPDIFn_RX_FS), and the serial data (SPDIFn_RX_DAT). The high frequency clock (SPDIFn_RX_TDMCLK) derived from the encoded stream is also available if the system requires it.
2. Initialize the SPDIF_RX_CTL register to enable the data decoding. Note that this peripheral is disabled by default.

NOTE: For more information, see the "DAI Routing Capabilities" section of the *Digital Audio Interface (DAI)* chapter.

Interrupted Data Streams on the Receiver

When using the S/PDIF receiver with data streams that are likely to be interrupted, (in other words unplugged and reconnected), it is necessary to take some extra steps to ensure that the S/PDIF receiver's digital PLL will relock to the stream. The steps to accomplish this are described below.

1. Set up interrupts within the DAI so that the S/PDIF receiver can generate an interrupt when the stream is reconnected.
2. Within the interrupt service routine (ISR), stop and restart the NCO. This is accomplished by setting and then clearing the SPDIF_RX_CTL.RST bit.
3. Return from the ISR and continue normal operation.

This method of resetting the NCO has been shown to provide extremely reliable performance when the S/PDIF inputs are interrupted or unplugged momentarily.

Debug Features

The following feature supports S/PDIF debugging.

Loopback Routing

The S/PDIF supports an internal loopback mode by using the SRU. For more information about loopback, see "Loopback Routing" in the *Digital Audio Interface (DAI)* chapter.

ADSP-2184x SPDIF Register Descriptions

The S/PDIF module (SPDIF) contains the following registers.

Table 33-3: ADSP-2184x SPDIF Register List

Name	Description
SPDIF_RX_CTL	Receive Control
SPDIF_RX_STAT	Receive Status Register
SPDIF_RX_STAT0_A	Receive Status A0 Register
SPDIF_RX_STAT0_B	Receive Status B0 Register
SPDIF_RX_STAT1_A	Receive Status A1 Register
SPDIF_RX_STAT1_B	Receive Status B1 Register
SPDIF_TX_CTL	Transmit Control Register
SPDIF_TX_STAT_A0	Transmit Status A0 Register
SPDIF_TX_STAT_A1	Transmit Status A1 Register
SPDIF_TX_STAT_A2	Transmit Status A2 Register
SPDIF_TX_STAT_A3	Transmit Status A3 Register
SPDIF_TX_STAT_A4	Transmit Status A4 Register
SPDIF_TX_STAT_A5	Transmit Status A5 Register
SPDIF_TX_STAT_B0	Transmit Status B0 Register
SPDIF_TX_STAT_B1	Transmit Status B1 Register
SPDIF_TX_STAT_B2	Transmit Status B2 Register
SPDIF_TX_STAT_B3	Transmit Status B3 Register
SPDIF_TX_STAT_B4	Transmit Status B4 Register
SPDIF_TX_STAT_B5	Transmit Status B5 Register

Table 33-3: ADSP-2184x SPDIF Register List (Continued)

Name	Description
SPDIF_TX_UBUFF_A0	Transmit User Buffer A0 Register
SPDIF_TX_UBUFF_A1	Transmit User Buffer A1 Register
SPDIF_TX_UBUFF_A2	Transmit User Buffer A2 Register
SPDIF_TX_UBUFF_A3	Transmit User Buffer A3 Register
SPDIF_TX_UBUFF_A4	Transmit User Buffer A4 Register
SPDIF_TX_UBUFF_A5	Transmit User Buffer A5 Register
SPDIF_TX_UBUFF_B0	Transmit User Buffer B0 Register
SPDIF_TX_UBUFF_B1	Transmit User Buffer B1 Register
SPDIF_TX_UBUFF_B2	Transmit User Buffer B2 Register
SPDIF_TX_UBUFF_B3	Transmit User Buffer B3 Register
SPDIF_TX_UBUFF_B4	Transmit User Buffer B4 Register
SPDIF_TX_UBUFF_B5	Transmit User Buffer B5 Register
SPDIF_TX_USRUPDT	User Bit Update Register

Receive Control

The SPDIF_RX_CTL register is used to enable and control the S/PDIF receiver.

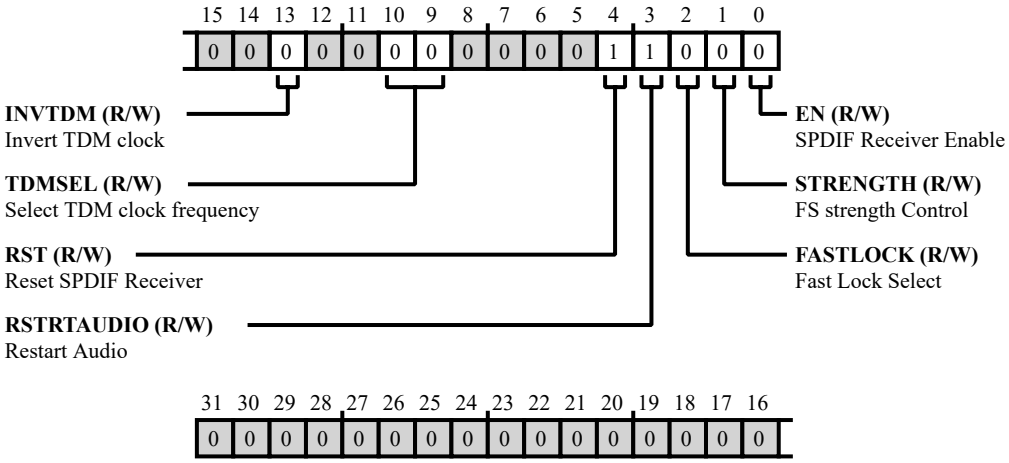


Figure 33-8: SPDIF_RX_CTL Register Diagram

Table 33-4: SPDIF_RX_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	INVTDM	Invert TDM clock. When set to 0, inverted TDM clock or non-inverted Bit clock is selected. When set to 1, non-inverted TDM clock or inverted Blit clock is selected.	
12:9 (R/W)	TDMSEL	Select TDM clock frequency. Select TDM clock frequency	
4 (R/W)	RST	Reset SPDIF Receiver. The SPDIF_RX_CTL.RST bit resets the receiver.	
		0	Takes the receiver out of reset
		1	Resets the SPDIF receiver
3 (R/W)	RSTRTAUDIO	Restart Audio. The SPDIF_RX_CTL.RSTRTAUDIO bit restarts the audio once a re-lock has occurred. When the S/PDIF receiver loses lock the audio output is set to 0. This bit determines the behavior of the audio once lock is re-established. Audio can be manually restarted by toggling this bit high and then low.	
		0	Manually restart audio
		1	Automatically restart audio

Table 33-4: SPDIF_RX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	FASTLOCK	Fast Lock Select. The `SPDIF_RX_CTL.FASTLOCK` bit allows the lock mechanism to lock at normal speed or at faster speed. This has the advantage of recovering very quickly whenever the S/PDIF receiver looses lock due to glitches in the signal. In normal mode the S/PDIF receiver locks after 64 consecutive valid samples, in fast mode the S/PDIF receiver locks after 8 consecutive valid samples	
		0	Enable normal mode
		1	Enable fast mode
1 (R/W)	STRENGTH	FS strength Control. The `SPDIF_RX_CTL.STRENGTH` bit controls the strength of the bit clock and Frame sync outputs from the SPDIF receiver. In strong mode these output signals are continued (as best possible) when the receiver notices a loss-of-lock condition. Note that 'as best possible' refers to the fact that this recovered signal may not be accurate, given the loss-of-lock condition. In weak mode these output signals are interrupted as soon as the receiver notices a loss-of-lock condition.	
		0	Enable strong mode
		1	Enable weak mode
0 (R/W)	EN	SPDIF Receiver Enable. When the `SPDIF_RX_CTL.EN` bit =0 the clock to SPDIF is switched off for power savings.	
		0	Disable receiver
		1	Enable receiver

Receive Status Register

The SPDIF_RX_STAT register consists of bits that indicate the status of various functions supported by S/PDIF receiver.

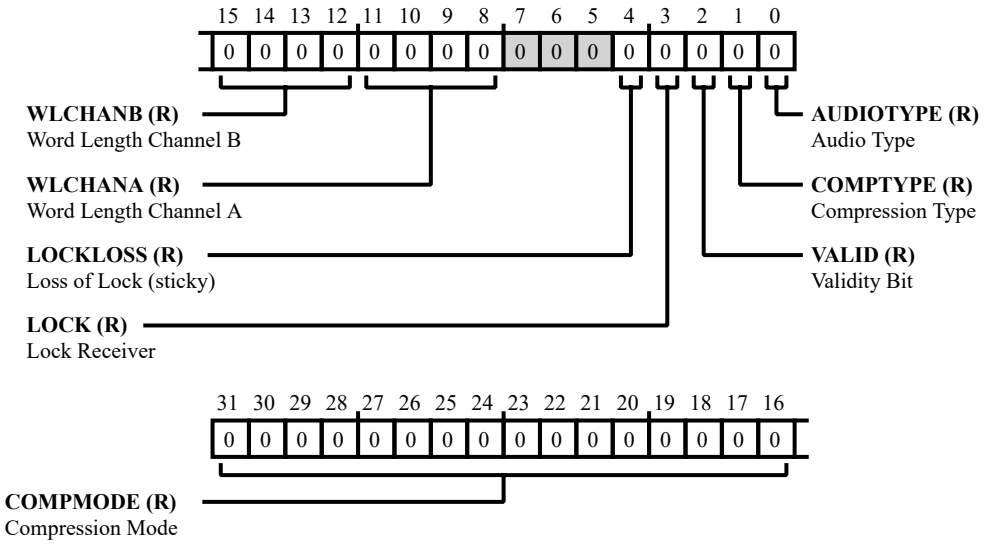


Figure 33-9: SPDIF_RX_STAT Register Diagram

Table 33-5: SPDIF_RX_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/NW)	COMPMODE	Compression Mode. The SPDIF_RX_STAT.COMPMODE bit field indicates the type of compression mode used in the Digital audio stream. The value in this field indicates the 16-bit burst information as specified by the IEC 62937-2 standard. Use this document to decode the value in this bit field.	
15:12 (R/NW)	WLCHANB	Word Length Channel B. The SPDIF_RX_STAT.WLCHANB bit field indicates the S/PDIF word length for channel B. May be decoded as follows (from the S/PDIF standard).	
		0	Reserved
		1	Reserved
		2	SPDIF_LENGTH_16
		3	Reserved
		4	SPDIF_LENGTH_18
		5	SPDIF_LENGTH_22
		6	Reserved

Table 33-5: SPDIF_RX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		7	Reserved
		8	SPDIF_LENGTH_19
		9	SPDIF_LENGTH_23
		10	SPDIF_LENGTH_20
		11	SPDIF_LENGTH_24
		12	SPDIF_LENGTH_17
		13	SPDIF_LENGTH_21
		14	Reserved
		15	Reserved
11:8 (R/NW)	WLCHANA	Word Length Channel A. The `SPDIF_RX_STAT.WLCHANA` bit indicates the S/PDIF word length for channel A. May be decoded as follows (from the S/PDIF standard).	
		0	Reserved
		1	Reserved
		2	SPDIF_LENGTH_16
		3	Reserved
		4	SPDIF_LENGTH_18
		5	SPDIF_LENGTH_22
		6	Reserved
		7	Reserved
		8	SPDIF_LENGTH_19
		9	SPDIF_LENGTH_23
		10	SPDIF_LENGTH_20
		11	SPDIF_LENGTH_24
		12	SPDIF_LENGTH_17
		13	SPDIF_LENGTH_21
		14	Reserved
		15	Reserved

Table 33-5: SPDIF_RX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	LOCKLOSS	Loss of Lock (sticky). The SPDIF_RX_STAT.LOCKLOSS bit indicates that the system has lost lock. This is different to the lock register, as it is sticky it goes high when system looses lock, but returns to low once SPDIF_RX_CTL.RSTRTAUDIO bit is toggled. This is to allow the programs to poll the lock status.	
		0	SPDIF receiver locked
		1	SPDIF Receiver lost lock
3 (R/NW)	LOCK	Lock Receiver. The SPDIF_RX_STAT.LOCK bit indicates the S/PDIF receiver has successfully locked to the S/PDIF stream and is outputting valid data.	
		0	Receiver not locked
		1	Receiver locked
2 (R/NW)	VALID	Validity Bit. The SPDIF_RX_STAT.VALID bit indicates the ORed bits of channel A and B.	
		0	Linear PCM data
		1	Non-linear audio data
1 (R/NW)	COMPTYPE	Compression Type. The SPDIF_RX_STAT.COMPTYPE bit indicates AC3 or DTS compression. Valid only if SPDIF_RX_STAT.AUDIOTYPE indicates compressed data.	
		0	AC3 compressed data
		1	DTS compressed data
0 (R/NW)	AUDIOTYPE	Audio Type. The SPDIF_RX_STAT.AUDIOTYPE bit indicates PCM or compressed data.	
		0	PCM data
		1	Compressed Data

Receive Status A0 Register

The SPDIF_RX_STAT0_A register holds the receive channel 0 status for bytes 0-3 for sub frame A.

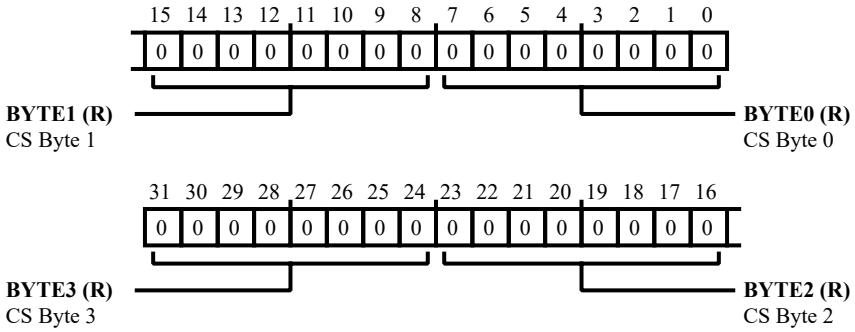


Figure 33-10: SPDIF_RX_STAT0_A Register Diagram

Table 33-6: SPDIF_RX_STAT0_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	BYTE3	CS Byte 3. The SPDIF_RX_STAT0_A.BYTE3 bit field contains byte 3 of received channel A status.
23:16 (R/NW)	BYTE2	CS Byte 2. The SPDIF_RX_STAT0_A.BYTE2 bit field contains byte 2 of received channel A status.
15:8 (R/NW)	BYTE1	CS Byte 1. The SPDIF_RX_STAT0_A.BYTE1 bit field contains byte 1 of received channel A status.
7:0 (R/NW)	BYTE0	CS Byte 0. The SPDIF_RX_STAT0_A.BYTE0 bit field contains the status of byte 0-3 of received channel A.

Receive Status B0 Register

The SPDIF_RX_STAT0_B register holds the receive channel 0 status for bytes 0-3 for sub frame B.

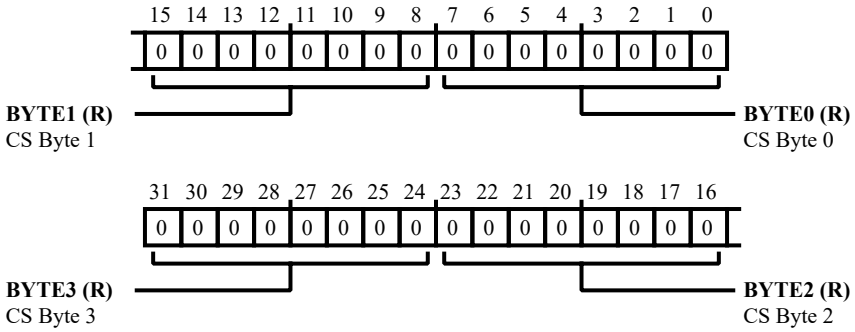


Figure 33-11: SPDIF_RX_STAT0_B Register Diagram

Table 33-7: SPDIF_RX_STAT0_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	BYTE3	CS Byte 3. The SPDIF_RX_STAT0_B.BYTE3 bit field contains byte 3 of received channel B status.
23:16 (R/NW)	BYTE2	CS Byte 2. The SPDIF_RX_STAT0_B.BYTE2 bit field contains byte 2 of received channel B status.
15:8 (R/NW)	BYTE1	CS Byte 1. The SPDIF_RX_STAT0_B.BYTE1 bit field contains byte 1 of received channel B status.
7:0 (R/NW)	BYTE0	CS Byte 0. The SPDIF_RX_STAT0_B.BYTE0 bit field contains byte 0 of received channel B status.

Receive Status A1 Register

The SPDIF_RX_STAT1_A register holds the receive channel 1 status for byte 4 for sub frame A.

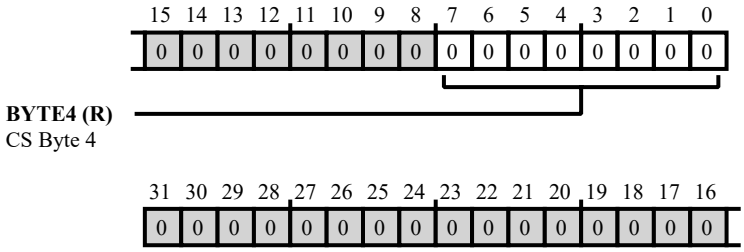


Figure 33-12: SPDIF_RX_STAT1_A Register Diagram

Table 33-8: SPDIF_RX_STAT1_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	BYTE4	CS Byte 4. The SPDIF_RX_STAT1_A.BYTE4 bit field contains byte 4 of received channel A status.

Receive Status B1 Register

The SPDIF_RX_STAT1_B register holds the receive channel 1 status for byte 4 for sub frame B.

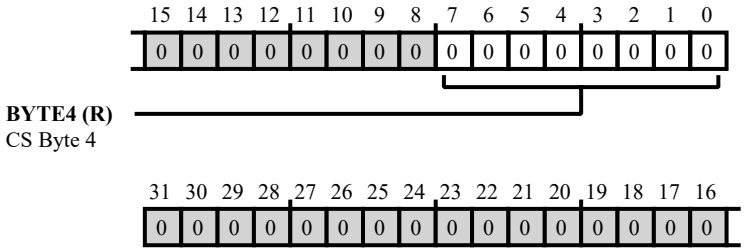


Figure 33-13: SPDIF_RX_STAT1_B Register Diagram

Table 33-9: SPDIF_RX_STAT1_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	BYTE4	CS Byte 4. The SPDIF_RX_STAT1_B.BYTE4 bit field contains byte 4 of received channel B status.

Transmit Control Register

The SPDIF_TX_CTL register provides bits the enable/disable the transmitter and configure several options related to data transmission.

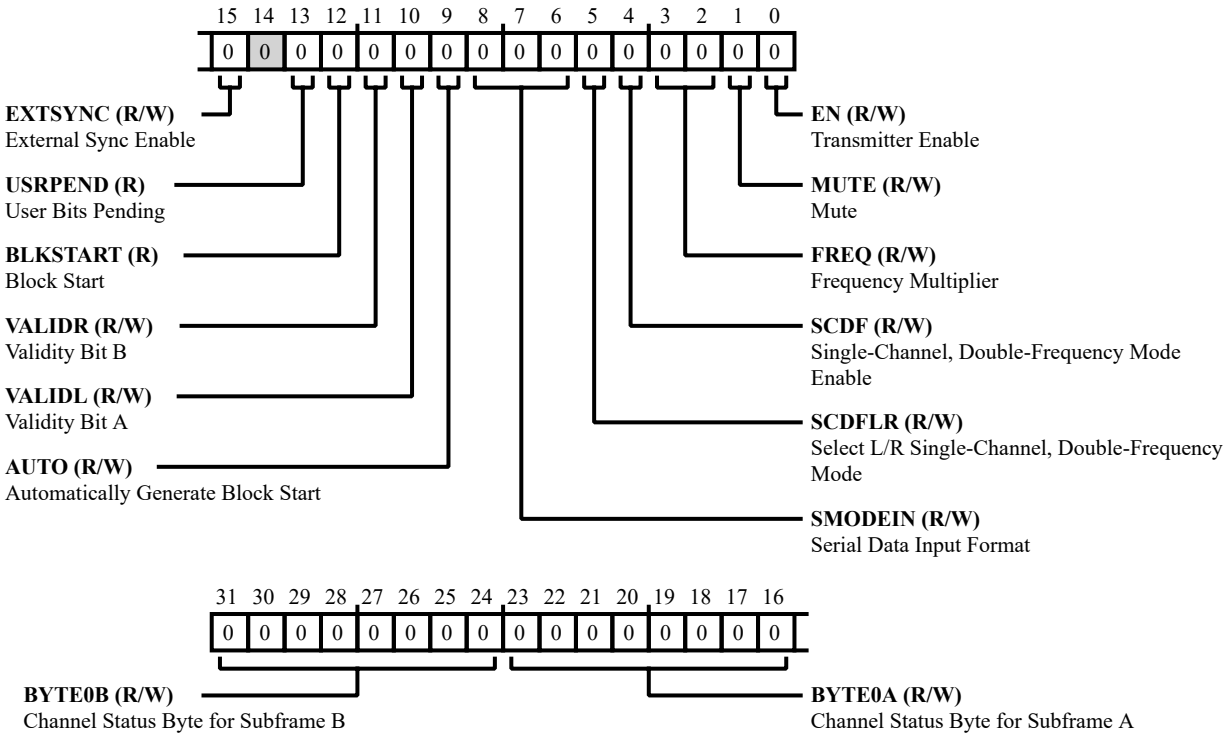


Figure 33-14: SPDIF_TX_CTL Register Diagram

Table 33-10: SPDIF_TX_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE0B	Channel Status Byte for Subframe B. The SPDIF_TX_CTL.BYTE0B bit field contains the channel status for the second bytes 9-5.
23:16 (R/W)	BYTE0A	Channel Status Byte for Subframe A. The SPDIF_TX_CTL.BYTE0A bit field contains the channel status for the first bytes 4-0.
15 (R/W)	EXTSYNC	External Sync Enable. When the SPDIF_TX_CTL.EXTSYNC bit is set (regardless of the SPDIF_TX_CTL.AUTO bit setting) the internal frame counter is set to zero at an internal LRCLK rising edge followed by an EXTSYNC_I rising edge.

Table 33-10: SPDIF_TX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/NW)	USRPEND	User Bits Pending. The SPDIF_TX_CTL.USRPEND bit is set if the update of the internal buffer from the Transmit User Bits Buffer registers has not yet completed.	
12 (R/NW)	BLKSTART	Block Start. The SPDIF_TX_CTL.BLKSTART bit is a status bit that indicates block start (when the SPDIF_TX_CTL.AUTO bit = 1).	
		0	Current word is not block start
		1	Current word is block start
11 (R/W)	VALIDR	Validity Bit B. Use the SPDIF_TX_CTL.VALIDR bit with the channel status buffer.	
10 (R/W)	VALIDL	Validity Bit A. The SPDIF_TX_CTL.VALIDL bit either manually start block transfer according to input stream status bits or automatically start block transfer. Use the SPDIF_TX_CTL.VALIDL bit with the channel status buffer.	
		0	Manually start block transfer
		1	Automatically start block transfer
9 (R/W)	AUTO	Automatically Generate Block Start. When enabled, the transmitter is in standalone mode where it inserts block start, channel status, and validity bits on its own. If the channel status or validity buffer needs to be enabled (after the DAI programming is complete), first write to the buffers with the required data and then enable the buffers by setting the SPDIF_TX_CTL.AUTO bit.	
8:6 (R/W)	SMODEIN	Serial Data Input Format. The SPDIF_TX_CTL.SMODEIN bit selects the data input format.	
		0	Left-justified
		1	I2S
		2	Reserved
		3	Reserved
		4	Right-justified, 24 bits
		5	Right-justified, 20 bits
		6	Right-justified, 18 bits
		7	Right-justified, 16 bits

Table 33-10: SPDIF_TX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SCDFLR	Select L/R Single-Channel, Double-Frequency Mode. The SPDIF_TX_CTL.SCDFLR bit selects the left or right channel in SCDF mode.	
		0	Left channel
		1	Right channel
4 (R/W)	SCDF	Single-Channel, Double-Frequency Mode Enable. The SPDIF_TX_CTL.SCDF bit enables single-channel, double-frequency mode.	
		0	Two-channel mode
		1	SCDF mode
3:2 (R/W)	FREQ	Frequency Multiplier. The SPDIF_TX_CTL.FREQ bit field sets the oversampling ratio.	
		0	256 x frame sync oversampling
		1	384 x frame sync oversampling
		2-3	Reserved
1 (R/W)	MUTE	Mute. The SPDIF_TX_CTL.MUTE bit mutes the serial data output.	
		0	Disable Mute
		1	Enable Mute
0 (R/W)	EN	Transmitter Enable. The SPDIF_TX_CTL.EN bit enables the transmitter and resets the control registers to their defaults.	
		0	Transmitter disabled
		1	Transmitter enabled

Transmit Status A0 Register

The SPDIF_TX_STAT_A0 register holds the transmit channel 0 status for bytes 1-4 for sub frame A.

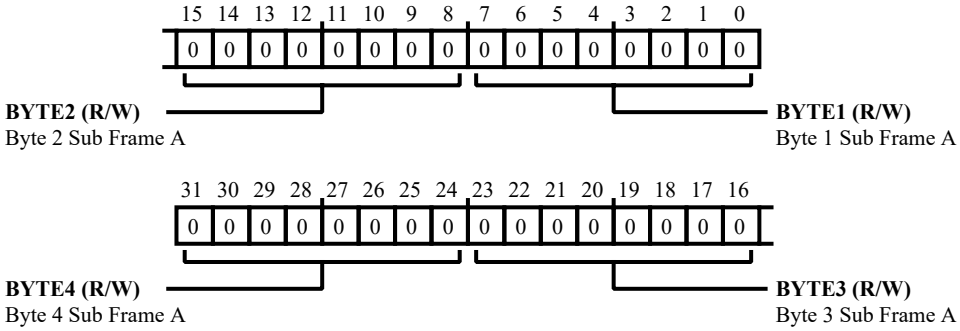


Figure 33-15: SPDIF_TX_STAT_A0 Register Diagram

Table 33-11: SPDIF_TX_STAT_A0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE4	Byte 4 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE4 bit field holds the transmit channel 0 status for byte 4 for sub frame A.
23:16 (R/W)	BYTE3	Byte 3 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE3 bit field holds the transmit channel 0 status for byte 3 for sub frame A.
15:8 (R/W)	BYTE2	Byte 2 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE2 bit field holds the transmit channel 0 status for byte 2 for sub frame A.
7:0 (R/W)	BYTE1	Byte 1 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE1 bit field holds the transmit channel 0 status for byte 1 for sub frame A.

Transmit Status A1 Register

The SPDIF_TX_STAT_A1 register holds the transmit status for bytes 5-8 for sub frame A.

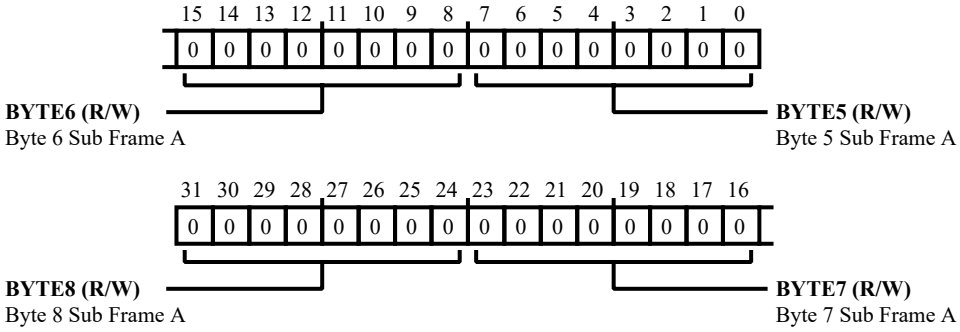


Figure 33-16: SPDIF_TX_STAT_A1 Register Diagram

Table 33-12: SPDIF_TX_STAT_A1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE8	Byte 8 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE8 bit field contains transmit status for byte 8 of sub frame A.
23:16 (R/W)	BYTE7	Byte 7 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE7 bit field contains transmit status for byte 7 of sub frame A.
15:8 (R/W)	BYTE6	Byte 6 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE6 bit field contains transmit status for byte 6 of sub frame A.
7:0 (R/W)	BYTE5	Byte 5 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE5 bit field contains transmit status for byte 5 of sub frame A.

Transmit Status A2 Register

The SPDIF_TX_STAT_A2 register holds the transmit status for bytes 9-12 for sub frame A.

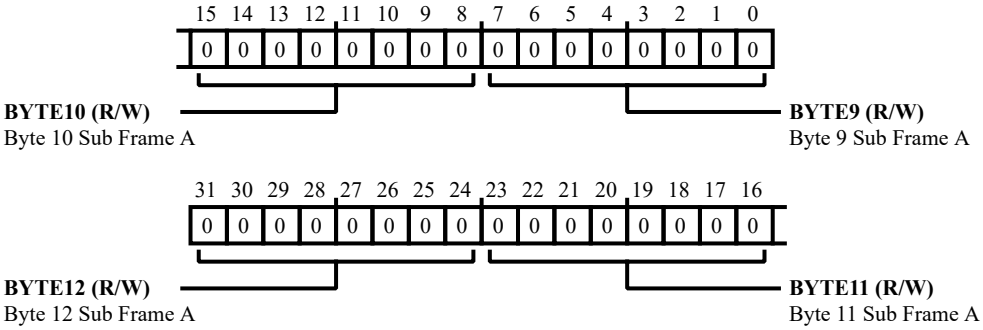


Figure 33-17: SPDIF_TX_STAT_A2 Register Diagram

Table 33-13: SPDIF_TX_STAT_A2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE12	Byte 12 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE12 bit field contains transmit status for byte 12 of sub frame A.
23:16 (R/W)	BYTE11	Byte 11 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE11 bit field contains transmit status for byte 11 of sub frame A.
15:8 (R/W)	BYTE10	Byte 10 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE10 bit field contains transmit status for byte 10 of sub frame A.
7:0 (R/W)	BYTE9	Byte 9 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE9 bit field contains transmit status for byte 9 of sub frame A.

Transmit Status A3 Register

The SPDIF_TX_STAT_A3 register holds the transmit status for bytes 13-16 for sub frame A.

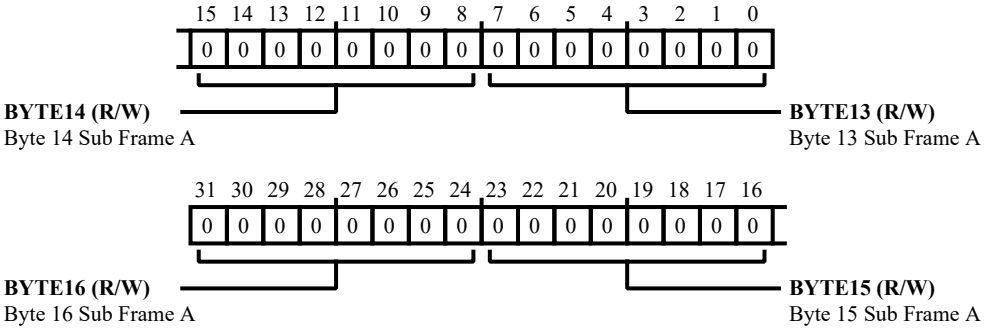


Figure 33-18: SPDIF_TX_STAT_A3 Register Diagram

Table 33-14: SPDIF_TX_STAT_A3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE16	Byte 16 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE16 bit field contains transmit status for byte 16 of sub frame A.
23:16 (R/W)	BYTE15	Byte 15 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE15 bit field contains transmit status for byte 15 of sub frame A.
15:8 (R/W)	BYTE14	Byte 14 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE14 bit field contains transmit status for byte 14 of sub frame A.
7:0 (R/W)	BYTE13	Byte 13 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE13 bit field contains transmit status for byte 13 of sub frame A.

Transmit Status A4 Register

The SPDIF_TX_STAT_A4 register holds the transmit status for bytes 17-20 for sub frame A.

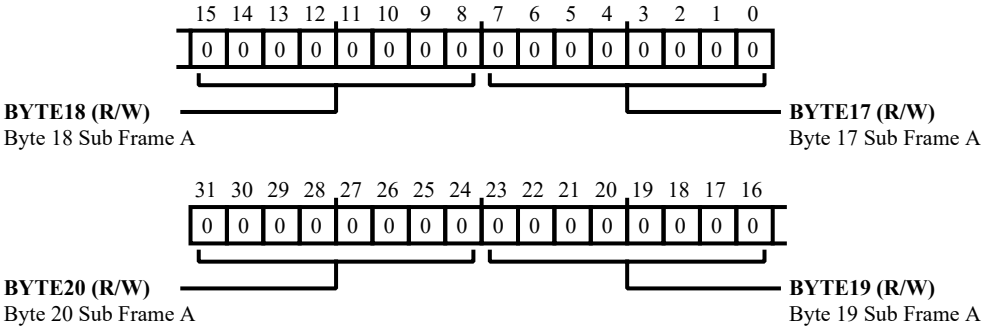


Figure 33-19: SPDIF_TX_STAT_A4 Register Diagram

Table 33-15: SPDIF_TX_STAT_A4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE20	Byte 20 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE20 bit field contains transmit status for byte 20 of sub frame A.
23:16 (R/W)	BYTE19	Byte 19 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE19 bit field contains transmit status for byte 19 of sub frame A.
15:8 (R/W)	BYTE18	Byte 18 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE18 bit field contains transmit status for byte 18 of sub frame A.
7:0 (R/W)	BYTE17	Byte 17 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE17 bit field contains transmit status for byte 17 of sub frame A.

Transmit Status A5 Register

The SPDIF_TX_STAT_A5 register holds the transmit status for bytes 21-23 for sub frame A.

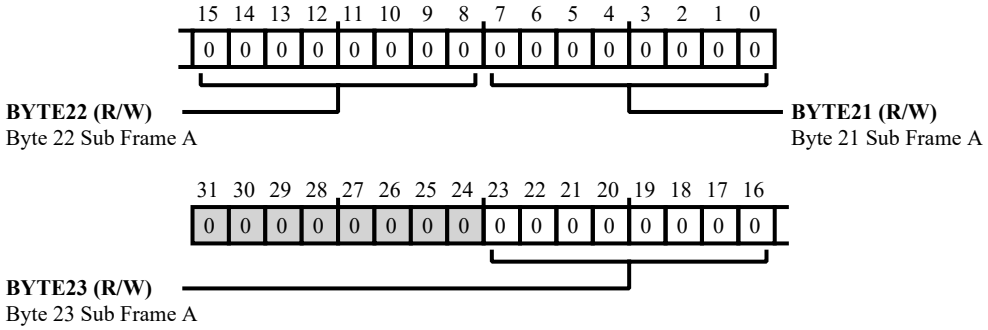


Figure 33-20: SPDIF_TX_STAT_A5 Register Diagram

Table 33-16: SPDIF_TX_STAT_A5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	BYTE23	Byte 23 Sub Frame A. The SPDIF_TX_STAT_A5.BYTE23 bit field contains transmit status for byte 23 of sub frame A.
15:8 (R/W)	BYTE22	Byte 22 Sub Frame A. The SPDIF_TX_STAT_A5.BYTE22 bit field contains transmit status for byte 22 of sub frame A.
7:0 (R/W)	BYTE21	Byte 21 Sub Frame A. The SPDIF_TX_STAT_A5.BYTE21 bit field contains transmit status for byte 21 of sub frame A.

Transmit Status B0 Register

The SPDIF_TX_STAT_B0 register holds the transmit channel 0 status for bytes 1-4 for sub frame B.

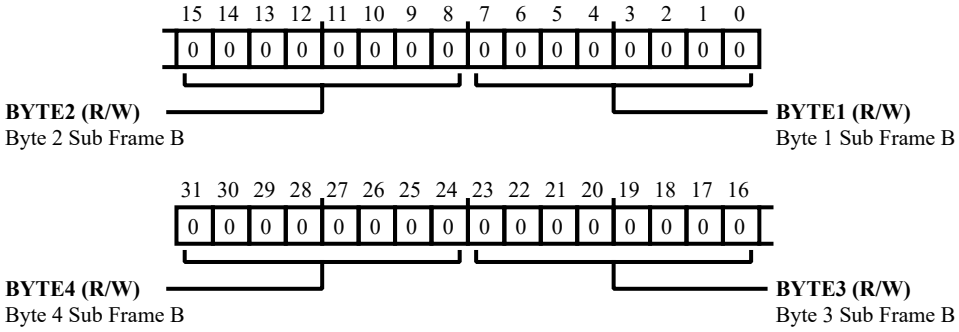


Figure 33-21: SPDIF_TX_STAT_B0 Register Diagram

Table 33-17: SPDIF_TX_STAT_B0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE4	Byte 4 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE4 bit field holds the transmit channel 0 status for byte 4 for sub frame B.
23:16 (R/W)	BYTE3	Byte 3 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE3 bit field holds the transmit channel 0 status for byte 3 for sub frame B.
15:8 (R/W)	BYTE2	Byte 2 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE2 bit field holds the transmit channel 0 status for byte 2 for sub frame B.
7:0 (R/W)	BYTE1	Byte 1 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE1 bit field holds the transmit channel 0 status for byte 1 for sub frame B.

Transmit Status B1 Register

The SPDIF_TX_STAT_B1 register holds the transmit status for bytes 5-8 for sub frame B.

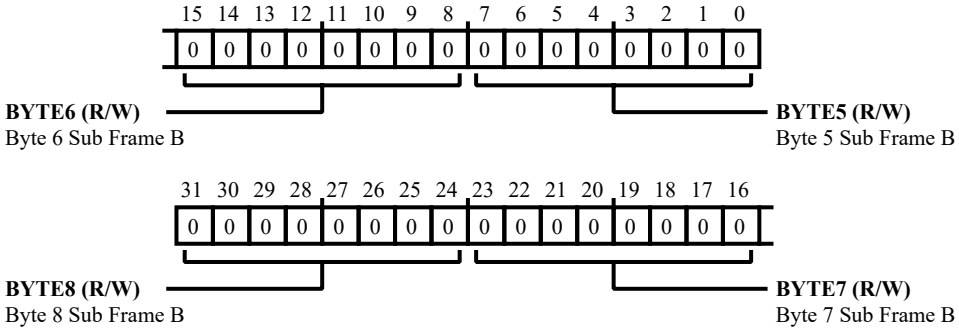


Figure 33-22: SPDIF_TX_STAT_B1 Register Diagram

Table 33-18: SPDIF_TX_STAT_B1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE8	Byte 8 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE8 bit field contains transmit status for byte 8 of sub frame B.
23:16 (R/W)	BYTE7	Byte 7 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE7 bit field contains transmit status for byte 7 of sub frame B.
15:8 (R/W)	BYTE6	Byte 6 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE6 bit field contains transmit status for byte 6 of sub frame B.
7:0 (R/W)	BYTE5	Byte 5 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE5 bit field contains transmit status for byte 5 of sub frame B.

Transmit Status B2 Register

The SPDIF_TX_STAT_B2 register holds the transmit status for bytes 9-12 for sub frame B.

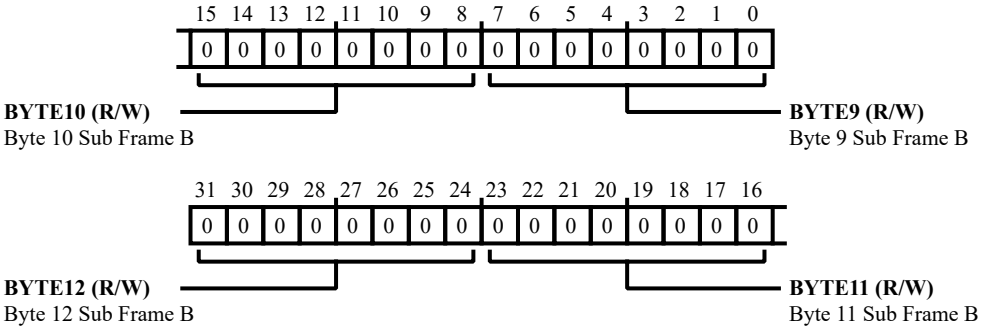


Figure 33-23: SPDIF_TX_STAT_B2 Register Diagram

Table 33-19: SPDIF_TX_STAT_B2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE12	Byte 12 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE12 bit field contains transmit status for byte 12 of sub frame B.
23:16 (R/W)	BYTE11	Byte 11 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE11 bit field contains transmit status for byte 11 of sub frame B.
15:8 (R/W)	BYTE10	Byte 10 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE10 bit field contains transmit status for byte 10 of sub frame B.
7:0 (R/W)	BYTE9	Byte 9 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE9 bit field contains transmit status for byte 9 of sub frame B.

Transmit Status B3 Register

The SPDIF_TX_STAT_B3 register holds the transmit status for bytes 13-16 for sub frame B.

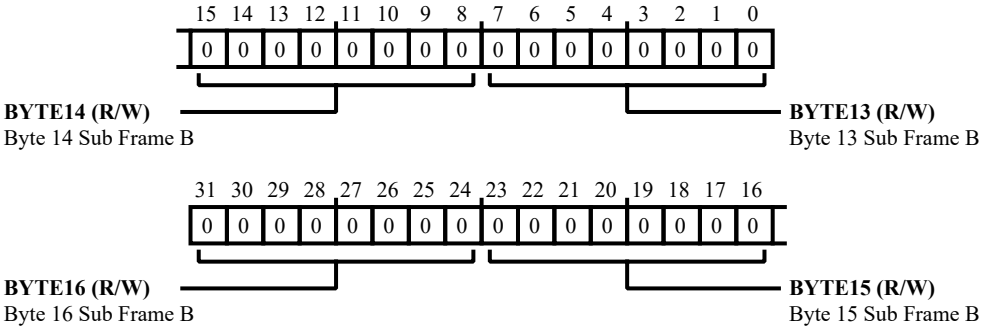


Figure 33-24: SPDIF_TX_STAT_B3 Register Diagram

Table 33-20: SPDIF_TX_STAT_B3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE16	Byte 16 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE16 bit field contains transmit status for byte 16 of sub frame B.
23:16 (R/W)	BYTE15	Byte 15 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE15 bit field contains transmit status for byte 15 of sub frame B.
15:8 (R/W)	BYTE14	Byte 14 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE14 bit field contains transmit status for byte 14 of sub frame B.
7:0 (R/W)	BYTE13	Byte 13 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE13 bit field contains transmit status for byte 13 of sub frame B.

Transmit Status B4 Register

The SPDIF_TX_STAT_B4 register holds the transmit status for bytes 17-20 for sub frame B.

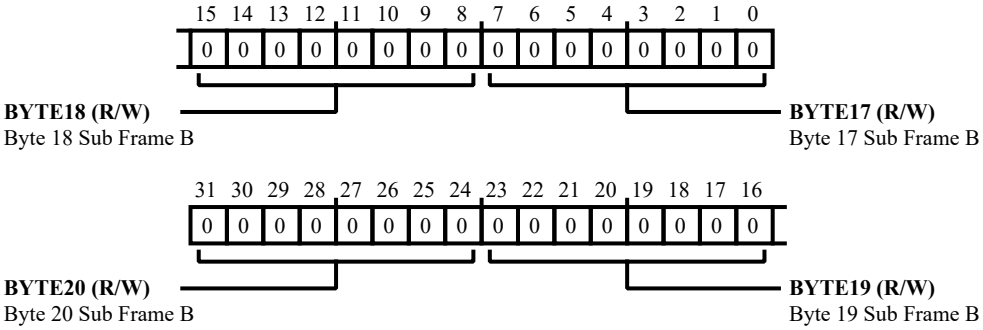


Figure 33-25: SPDIF_TX_STAT_B4 Register Diagram

Table 33-21: SPDIF_TX_STAT_B4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE20	Byte 20 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE20 bit field contains transmit status for byte 20 of sub frame B.
23:16 (R/W)	BYTE19	Byte 19 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE19 bit field contains transmit status for byte 19 of sub frame B.
15:8 (R/W)	BYTE18	Byte 18 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE18 bit field contains transmit status for byte 18 of sub frame B.
7:0 (R/W)	BYTE17	Byte 17 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE17 bit field contains transmit status for byte 17 of sub frame B.

Transmit Status B5 Register

The SPDIF_TX_STAT_B5 register holds the transmit status for bytes 21-23 for sub frame B.

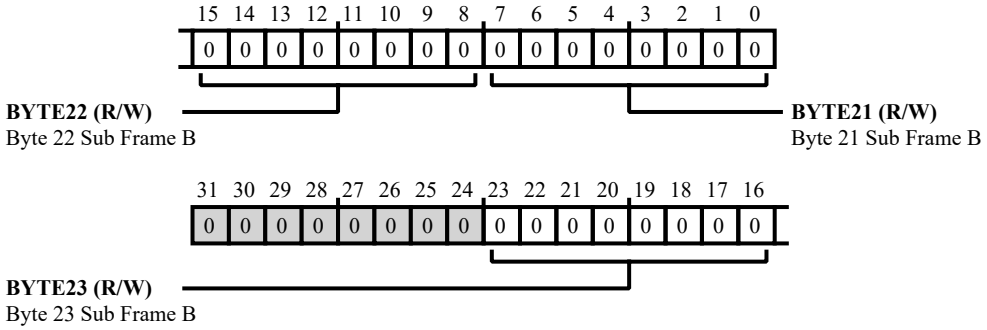


Figure 33-26: SPDIF_TX_STAT_B5 Register Diagram

Table 33-22: SPDIF_TX_STAT_B5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	BYTE23	Byte 23 Sub Frame B. The SPDIF_TX_STAT_B5.BYTE23 bit field contains transmit status for byte 23 of sub frame B.
15:8 (R/W)	BYTE22	Byte 22 Sub Frame B. The SPDIF_TX_STAT_B5.BYTE22 bit field contains transmit status for byte 22 of sub frame B.
7:0 (R/W)	BYTE21	Byte 21 Sub Frame B. The SPDIF_TX_STAT_B5.BYTE21 bit field contains transmit status for byte 21 of sub frame B.

Transmit User Buffer A0 Register

The SPDIF_TX_UBUFF_A0 register holds the transmit user buffer data for bytes 0-3 for sub frame A.

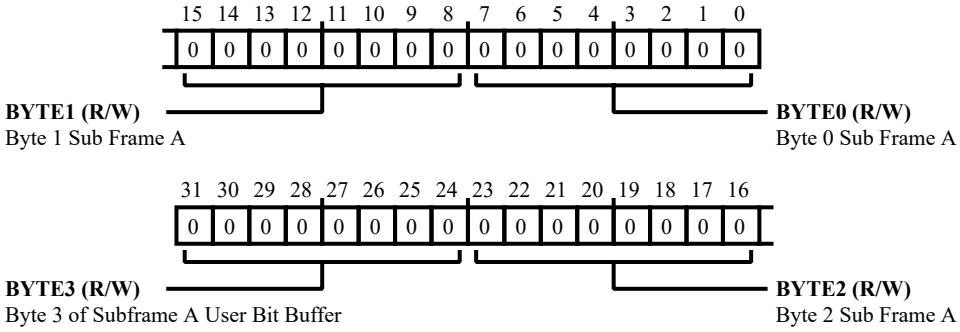


Figure 33-27: SPDIF_TX_UBUFF_A0 Register Diagram

Table 33-23: SPDIF_TX_UBUFF_A0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE3	Byte 3 of Subframe A User Bit Buffer. The SPDIF_TX_UBUFF_A0.BYTE3 bit field contains user bit data for byte 0 of sub frame A.
23:16 (R/W)	BYTE2	Byte 2 Sub Frame A. The SPDIF_TX_UBUFF_A0.BYTE2 bit field contains user bit data for byte 2 of sub frame A.
15:8 (R/W)	BYTE1	Byte 1 Sub Frame A. The SPDIF_TX_UBUFF_A0.BYTE1 bit field contains user bit data for byte 1 of sub frame A.
7:0 (R/W)	BYTE0	Byte 0 Sub Frame A. The SPDIF_TX_UBUFF_A0.BYTE0 bit field contains user bit data for byte 0 of sub frame A.

Transmit User Buffer A1 Register

The SPDIF_TX_UBUFF_A1 register holds the transmit user buffer data for bytes 4-7 for sub frame A.

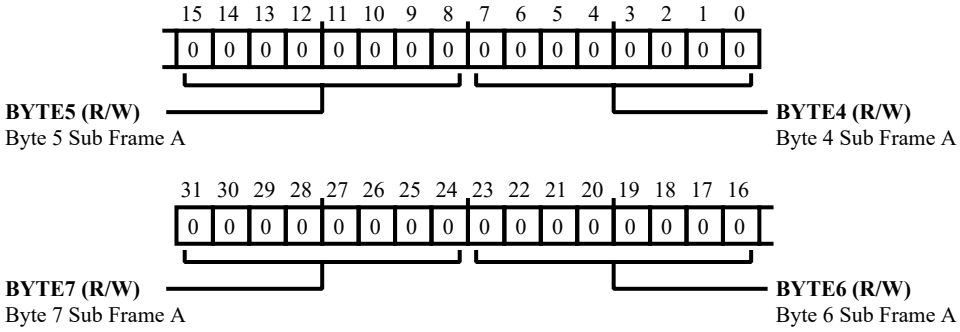


Figure 33-28: SPDIF_TX_UBUFF_A1 Register Diagram

Table 33-24: SPDIF_TX_UBUFF_A1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE7	Byte 7 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE7 bit field contains user bit data for byte 7 of sub frame A.
23:16 (R/W)	BYTE6	Byte 6 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE6 bit field contains user bit data for byte 6 of sub frame A.
15:8 (R/W)	BYTE5	Byte 5 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE5 bit field contains user bit data for byte 5 of sub frame A.
7:0 (R/W)	BYTE4	Byte 4 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE4 bit field contains user bit data for byte 0 of sub frame A.

Transmit User Buffer A2 Register

The SPDIF_TX_UBUFF_A2 register holds the transmit user buffer data for bytes 8-11 for sub frame A.

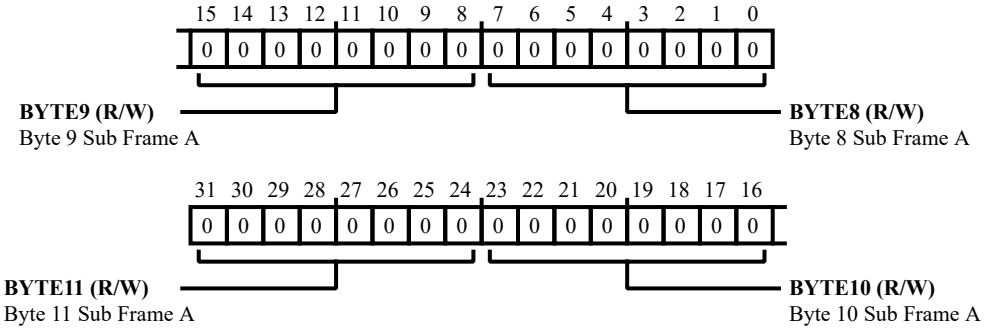


Figure 33-29: SPDIF_TX_UBUFF_A2 Register Diagram

Table 33-25: SPDIF_TX_UBUFF_A2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE11	Byte 11 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE11 bit field contains user bit data for byte 11 of sub frame A.
23:16 (R/W)	BYTE10	Byte 10 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE10 bit field contains user bit data for byte 10 of sub frame A.
15:8 (R/W)	BYTE9	Byte 9 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE9 bit field contains user bit data for byte 9 of sub frame A.
7:0 (R/W)	BYTE8	Byte 8 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE8 bit field contains user bit data for byte 8 of sub frame A.

Transmit User Buffer A3 Register

The SPDIF_TX_UBUFF_A3 register holds the transmit user buffer data for bytes 12-15 for sub frame A.

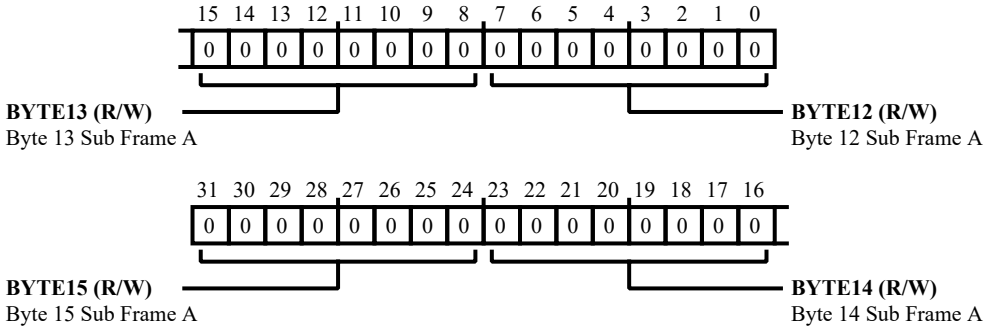


Figure 33-30: SPDIF_TX_UBUFF_A3 Register Diagram

Table 33-26: SPDIF_TX_UBUFF_A3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE15	Byte 15 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE15 bit field contains user bit data for byte 15 of sub frame A.
23:16 (R/W)	BYTE14	Byte 14 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE14 bit field contains user bit data for byte 14 of sub frame A.
15:8 (R/W)	BYTE13	Byte 13 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE13 bit field contains user bit data for byte 13 of sub frame A.
7:0 (R/W)	BYTE12	Byte 12 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE12 bit field contains user bit data for byte 12 of sub frame A.

Transmit User Buffer A4 Register

The SPDIF_TX_UBUFF_A4 register holds the transmit user buffer data for bytes 16-19 for sub frame A.

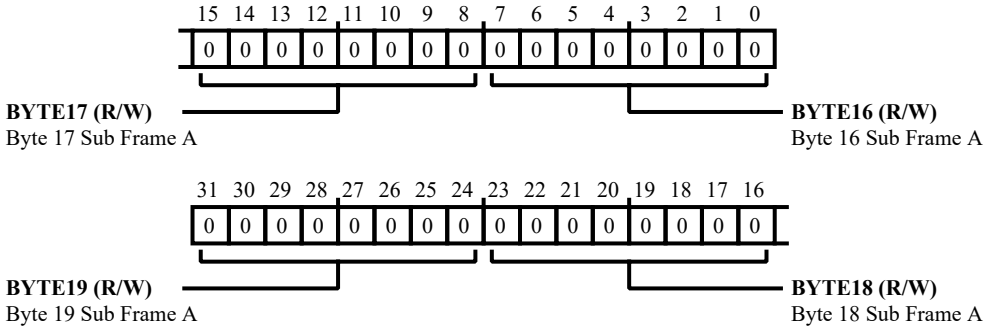


Figure 33-31: SPDIF_TX_UBUFF_A4 Register Diagram

Table 33-27: SPDIF_TX_UBUFF_A4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE19	Byte 19 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE19 bit field contains user bit data for byte 19 of sub frame A.
23:16 (R/W)	BYTE18	Byte 18 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE18 bit field contains user bit data for byte 18 of sub frame A.
15:8 (R/W)	BYTE17	Byte 17 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE17 bit field contains user bit data for byte 17 of sub frame A.
7:0 (R/W)	BYTE16	Byte 16 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE16 bit field contains user bit data for byte 16 of sub frame A.

Transmit User Buffer A5 Register

The SPDIF_TX_UBUFF_A5 register holds the transmit user buffer data for bytes 20-23 for sub frame A.

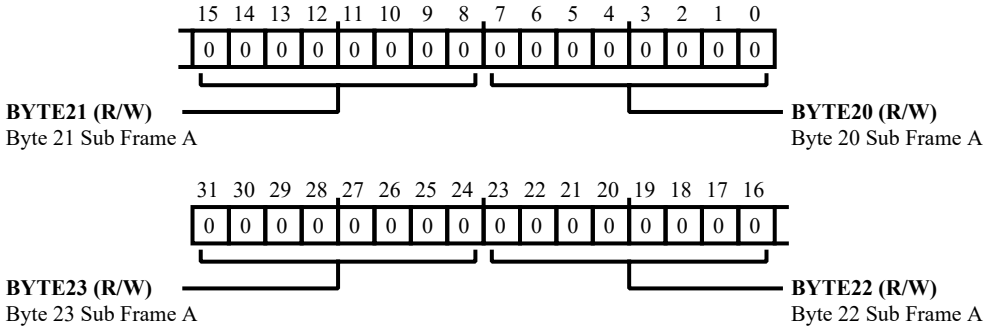


Figure 33-32: SPDIF_TX_UBUFF_A5 Register Diagram

Table 33-28: SPDIF_TX_UBUFF_A5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE23	Byte 23 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE23 bit field contains user bit data for byte 23 of sub frame A.
23:16 (R/W)	BYTE22	Byte 22 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE22 bit field contains user bit data for byte 22 of sub frame A.
15:8 (R/W)	BYTE21	Byte 21 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE21 bit field contains user bit data for byte 21 of sub frame A.
7:0 (R/W)	BYTE20	Byte 20 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE20 bit field contains user bit data for byte 20 of sub frame A.

Transmit User Buffer B0 Register

The SPDIF_TX_UBUFF_B0 register holds the transmit user buffer data for bytes 0-3 for sub frame B.

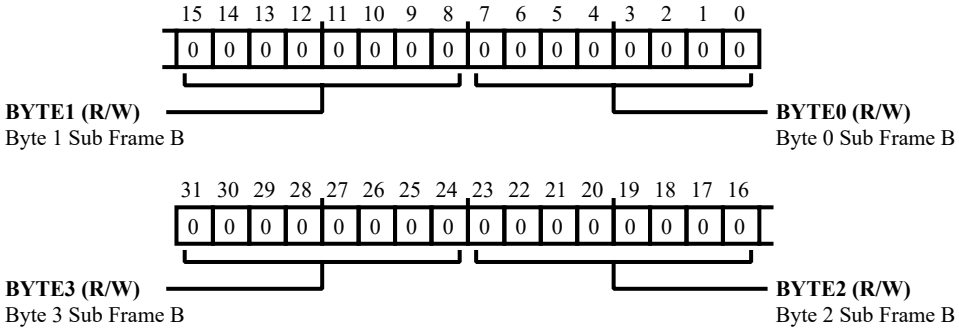


Figure 33-33: SPDIF_TX_UBUFF_B0 Register Diagram

Table 33-29: SPDIF_TX_UBUFF_B0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE3	Byte 3 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE3 bit field contains user bit data for byte 3 of sub frame B.
23:16 (R/W)	BYTE2	Byte 2 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE2 bit field contains user bit data for byte 2 of sub frame B.
15:8 (R/W)	BYTE1	Byte 1 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE1 bit field contains user bit data for byte 1 of sub frame B.
7:0 (R/W)	BYTE0	Byte 0 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE0 bit field contains user bit data for byte 0 of sub frame B.

Transmit User Buffer B1 Register

The SPDIF_TX_UBUFF_B1 register holds the transmit user buffer data for bytes 4-7 for sub frame B.

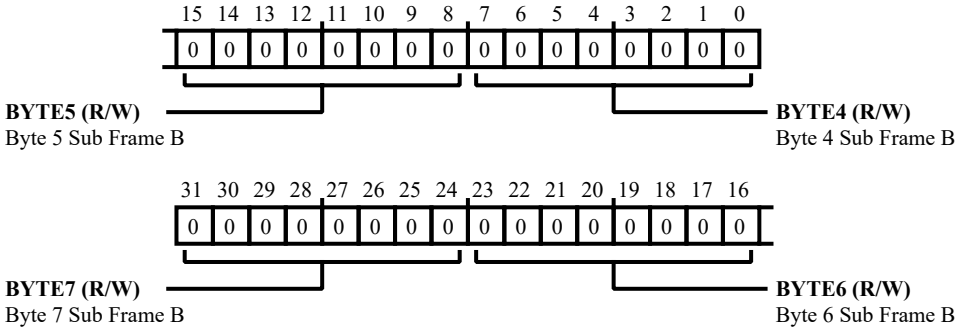


Figure 33-34: SPDIF_TX_UBUFF_B1 Register Diagram

Table 33-30: SPDIF_TX_UBUFF_B1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE7	Byte 7 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE7 bit field contains user bit data for byte 7 of sub frame B.
23:16 (R/W)	BYTE6	Byte 6 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE6 bit field contains user bit data for byte 6 of sub frame B.
15:8 (R/W)	BYTE5	Byte 5 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE5 bit field contains user bit data for byte 5 of sub frame B.
7:0 (R/W)	BYTE4	Byte 4 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE4 bit field contains user bit data for byte 4 of sub frame B.

Transmit User Buffer B2 Register

The SPDIF_TX_UBUFF_B2 register holds the transmit user buffer data for bytes 8-11 for sub frame B.

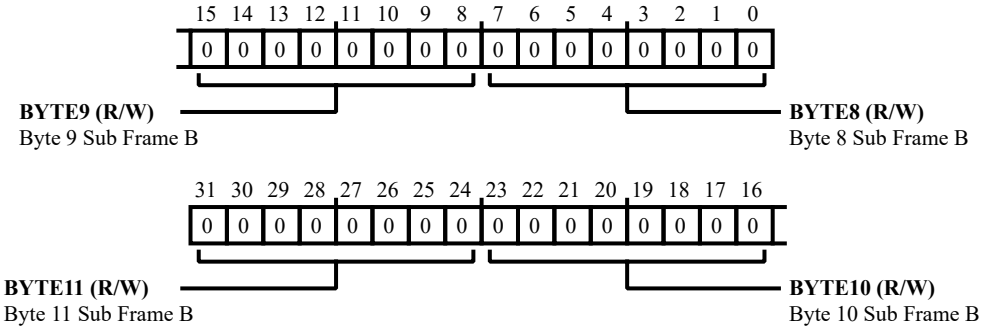


Figure 33-35: SPDIF_TX_UBUFF_B2 Register Diagram

Table 33-31: SPDIF_TX_UBUFF_B2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE11	Byte 11 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE11 bit field contains user bit data for byte 11 of sub frame B.
23:16 (R/W)	BYTE10	Byte 10 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE10 bit field contains user bit data for byte 10 of sub frame B.
15:8 (R/W)	BYTE9	Byte 9 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE9 bit field contains user bit data for byte 9 of sub frame B.
7:0 (R/W)	BYTE8	Byte 8 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE8 bit field contains user bit data for byte 8 of sub frame B.

Transmit User Buffer B3 Register

The SPDIF_TX_UBUFF_B3 register holds the transmit user buffer data for bytes 12-15 for sub frame B.

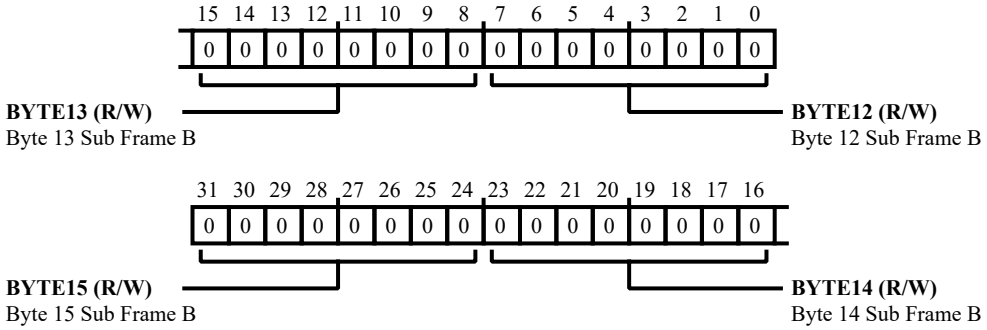


Figure 33-36: SPDIF_TX_UBUFF_B3 Register Diagram

Table 33-32: SPDIF_TX_UBUFF_B3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE15	Byte 15 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE15 bit field contains user bit data for byte 15 of sub frame B.
23:16 (R/W)	BYTE14	Byte 14 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE14 bit field contains user bit data for byte 14 of sub frame B.
15:8 (R/W)	BYTE13	Byte 13 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE13 bit field contains user bit data for byte 13 of sub frame B.
7:0 (R/W)	BYTE12	Byte 12 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE12 bit field contains user bit data for byte 12 of sub frame B.

Transmit User Buffer B4 Register

The SPDIF_TX_UBUFF_B4 register holds the transmit user buffer data for bytes 16-19 for sub frame B.

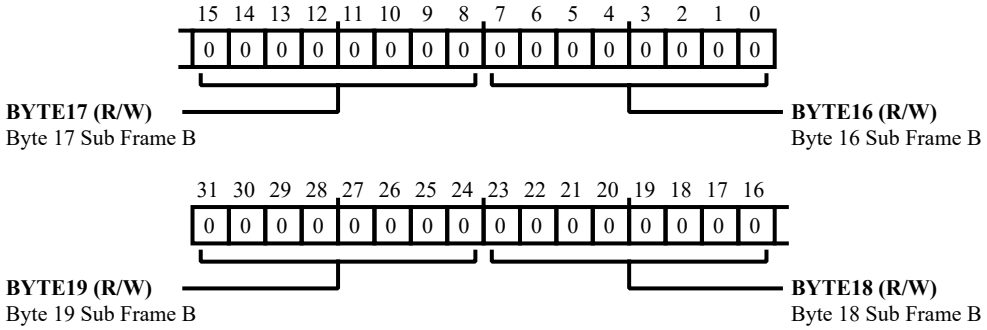


Figure 33-37: SPDIF_TX_UBUFF_B4 Register Diagram

Table 33-33: SPDIF_TX_UBUFF_B4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE19	Byte 19 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE19 bit field contains user bit data for byte 19 of sub frame B.
23:16 (R/W)	BYTE18	Byte 18 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE18 bit field contains user bit data for byte 18 of sub frame B.
15:8 (R/W)	BYTE17	Byte 17 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE17 bit field contains user bit data for byte 17 of sub frame B.
7:0 (R/W)	BYTE16	Byte 16 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE16 bit field contains user bit data for byte 16 of sub frame B.

Transmit User Buffer B5 Register

The SPDIF_TX_UBUFF_B5 register holds the transmit user buffer data for bytes 20-23 for sub frame B.

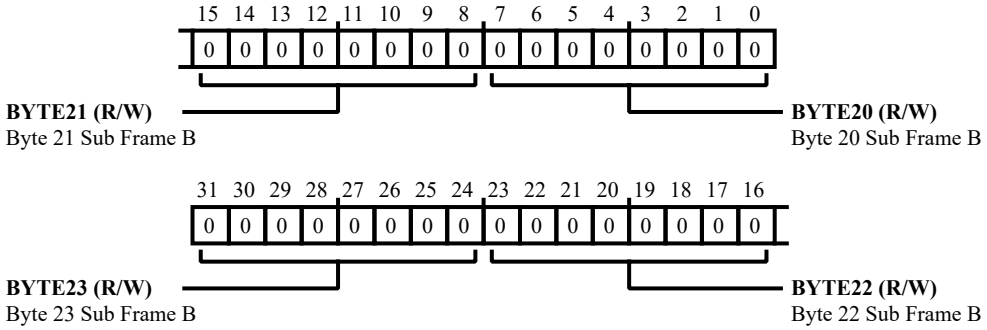


Figure 33-38: SPDIF_TX_UBUFF_B5 Register Diagram

Table 33-34: SPDIF_TX_UBUFF_B5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE23	Byte 23 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE23 bit field contains user bit data for byte 23 of sub frame B.
23:16 (R/W)	BYTE22	Byte 22 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE22 bit field contains user bit data for byte 22 of sub frame B.
15:8 (R/W)	BYTE21	Byte 21 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE21 bit field contains user bit data for byte 21 of sub frame B.
7:0 (R/W)	BYTE20	Byte 20 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE20 bit field contains user bit data for byte 20 of sub frame B.

User Bit Update Register

After writing to the transmit user buffer registers, a value of 0x1 must be written into the SPDIF_TX_USRUPDT register to enable the use of these user buffer bits in the next transfer block.

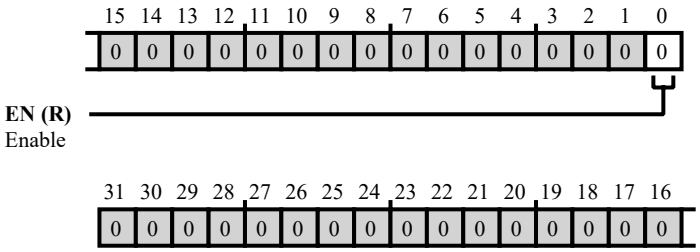


Figure 33-39: SPDIF_TX_USRUPDT Register Diagram

Table 33-35: SPDIF_TX_USRUPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	EN	Enable. After writing to the transmit user buffer registers, a value of 0x1 must be written into the SPDIF_TX_USRUPDT register to enable the use of these SPDIF_TX_USRUPDT.EN bits in the next transfer block.

34 Direct Memory Access (DMA)

The processor architecture distributes the DMA channels throughout the infrastructure. Often, the channels cluster together through system crossbars (SCB), sharing a single interface with the main system crossbar.

The DMA channels can perform transfers between memory and a peripheral or between one memory and another memory. Memory-to-memory DMA transfers (MDMA) require two DMA channels. One channel is the source channel, and the second, the destination channel.

All DMA channels can transport data to and from virtually all on-chip and off-chip memories.

DMA transfers on the processor use either a descriptor-based method or register-based method. Register-based DMA allows the processor directly to program DMA controller registers to initiate a DMA transfer. On completion, the controller registers can automatically update with their original setup values for continuous transfer, if needed. Descriptor-based DMA transfers require a set of parameters stored within memory to initiate a DMA sequence. Descriptor-based transfers allow the chaining together of multiple DMA sequences. In descriptor-based DMA operations, DMA channel programming can automatically set up and start another DMA transfer after the current sequence completes.

The DMA channel does not connect external memories and devices directly. Rather, data passes through an external-memory interface port. DMA operations can access any device the external memory interface supports. These interfaces typically include:

- Flash memory
- SRAM
- FIFOs
- Memory-mapped peripheral devices
- Dynamic Memory (if present)

DMA Channel Features

The processor uses Direct Memory Access (DMA) to transfer data within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA channel carries out the data transfers independent of processor activity. The DMA channels are dispersed throughout the infrastructure and interface with the system crossbar unit (SCB).

The following is a list of DMA interface features.

- Supports integer byte strides including byte strides of 0 and negative byte strides
- Register-based configuration
  - Core writes DMA configuration
  - Supports automatic reloading for continuous operation
- Flexible descriptor-based configuration
  - DMA descriptors are fetched from memory
  - Support for variable descriptor sizes
- Flexible flow control – Transitions between the various descriptor-based modes and for DMA termination
- Orthogonal transfers
  - Support for three transfer dimensions
  - One and two dimensional (1D and 2D) transfers supported per descriptor set
  - Three dimensional (3D) support provided by chained descriptor sets
- Configurable memory and peripheral-transfer word sizes
  - Memory interface supports 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit transfers
  - Peripheral interface supports for 8-bit, 16-bit, and 32-bit transfers
- Interrupt notification
  - Row or work unit completion
  - Error conditions
- Incoming and outgoing trigger support
  - Trigger generation for row or work unit completion
  - Work unit can wait for incoming trigger
- MMR access bus – Provides access to memory-mapped registers for configuration, monitoring, and debug
- SCB crossbar interface connects the DMA channel to the system crossbar
- Peripheral DMA bus – Interfaces the DMA channel to a peripheral or another DMA channel
- Peripheral data-request interrupt support
- Bandwidth monitoring and limiting for MDMA channels

DMA Channel Functional Description

This section provides a functional description of the DMA channel interface.

NOTE: There are two types of peripherals that use DMA. The first have dedicated DMA channels controlled by the Dedicated DMA Engine (DDE) and are described in this chapter. The second type is not controlled by the DDE. These peripherals have their own operating modes and programming models (see the peripheral chapter for this information). The complete list of DMA supported peripherals are shown in .

ADSP-2184x DMA Register List

The DMA channel controller (DMA) supports data transfers within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA channel carries out the data transfers independent of processor activity. The DMA channels are dispersed throughout the infrastructure, as DMAn's. A set of registers governs DMA operations. For more information on DMA functionality, see the DMA register descriptions.

Table 34-1: ADSP-2184x DMA Register List

Name	Description
DMA_ADDRSTART	Start Address of Current Buffer Register
DMA_ADDR_CUR	Current Address Register
DMA_BWLCNT	Bandwidth Limit Count Register
DMA_BWLCNT_CUR	Bandwidth Limit Count Current Register
DMA_BWMCNT	Bandwidth Monitor Count Register
DMA_BWMCNT_CUR	Bandwidth Monitor Count Current Register
DMA_CFG	Configuration Register
DMA_DSCPTR_CUR	Current Descriptor Pointer Register
DMA_DSCPTR_NXT	Pointer to Next Initial Descriptor Register
DMA_DSCPTR_PRV	Previous Initial Descriptor Pointer Register
DMA_STAT	Status Register
DMA_XCNT	Inner Loop Count Start Value Register
DMA_XCNT_CUR	Current Count (1D) or Intra-row XCNT (2D) Register
DMA_XMOD	Inner Loop Address Increment Register
DMA_YCNT	Outer Loop Count Start Value (2D only) Register
DMA_YCNT_CUR	Current Row Count (2D only) Register
DMA_YMOD	Outer Loop Address Increment (2D only) Register

ADSP-2184x DMA Channel List

Table 34-2: ADSP-2184x DMA Channel List

DMA ID	DMA Channel Name	Description
DMA0	SPORT0_A_DMA	SPORT0 Channel A DMA
DMA1	SPORT0_B_DMA	SPORT0 Channel B DMA
DMA2	SPORT1_A_DMA	SPORT1 Channel A DMA
DMA3	SPORT1_B_DMA	SPORT1 Channel B DMA
DMA4	SPORT2_A_DMA	SPORT2 Channel A DMA
DMA5	SPORT2_B_DMA	SPORT2 Channel B DMA
DMA6	SPORT3_A_DMA	SPORT3 Channel A DMA
DMA7	SPORT3_B_DMA	SPORT3 Channel B DMA
DMA8	MDMA0_SRC	Memory DMA Stream 0 Source Channel
DMA9	MDMA0_DST	Memory DMA Stream 0 Destination Channel
DMA10	SPORT4_A_DMA	SPORT4 Channel A DMA
DMA11	SPORT4_B_DMA	SPORT4 Channel B DMA
DMA12	SPORT5_A_DMA	SPORT5 Channel A DMA
DMA13	SPORT5_B_DMA	SPORT5 Channel B DMA
DMA14	SPORT6_A_DMA	SPORT6 Channel A DMA
DMA15	SPORT6_B_DMA	SPORT6 Channel B DMA
DMA16	SPORT7_A_DMA	SPORT7 Channel A DMA
DMA17	SPORT7_B_DMA	SPORT7 Channel B DMA
DMA18	MDMA1_SRC	Memory DMA Stream 1 Source Channel
DMA19	MDMA1_DST	Memory DMA Stream 1 Destination Channel
DMA20	UART0_TXDMA	UART0 Transmit DMA
DMA21	UART0_RXDMA	UART0 Receive DMA
DMA22	SPI0_TXDMA	SPI0 TX DMA Channel
DMA23	SPI0_RXDMA	SPI0 RX DMA Channel
DMA24	SPI1_TXDMA	SPI1 TX DMA Channel
DMA25	SPI1_RXDMA	SPI1 RX DMA Channel
DMA26	SPI2_TXDMA	SPI2 TX DMA Channel

Table 34-2: ADSP-2184x DMA Channel List (Continued)

DMA ID	DMA Channel Name	Description
DMA27	SPI2_RXDMA	SPI2 RX DMA Channel
DMA30	LP0_DMA	LP0 DMA Channel
DMA34	UART1_TXDMA	UART1 Transmit DMA
DMA35	UART1_RXDMA	UART1 Receive DMA
DMA36	LP1_DMA	LP1 DMA Channel
DMA37	UART2_TXDMA	UART2 Transmit DMA
DMA38	UART2_RXDMA	UART2 Receive DMA
DMA39	MDMA2_SRC	Memory DMA Stream 2 Source Channel
DMA40	MDMA2_DST	Memory DMA Stream 2 Destination Channel
DMA43	MDMA3_SRC	Memory DMA Stream 3 Source Channel
DMA44	MDMA3_DST	Memory DMA Stream 3 Destination Channel
DMA45	MDMA4_SRC	Memory DMA Stream 4 Source Channel
DMA46	MDMA4_DST	Memory DMA Stream 4 Destination Channel
DMA47	MDMA5_SRC	Memory DMA Stream 5 Source Channel
DMA48	MDMA5_DST	Memory DMA Stream 5 Destination Channel
DMA49	MDMA6_SRC	Memory DMA Stream 6 Source Channel
DMA50	MDMA6_DST	Memory DMA Stream 6 Destination Channel
DMA51	MDMA7_SRC	Memory DMA Stream 7 Source Channel
DMA52	MDMA7_DST	Memory DMA Stream 7 Destination Channel
DMA59	SPI5_TXDMA	SPI5 TX DMA Channel
DMA60	SPI5_RXDMA	SPI5 RX DMA Channel

DMA Definitions

To make the best use of the DMA controller, it is useful to understand the following terms.

Descriptor

A single element of a descriptor set that maps to a specific register of a DMA channel.

Descriptor Fetch

The action of retrieving descriptors from memory through memory read operations and loading then into the DMA channel registers upon their read return.

Descriptor Set

An array of descriptors associated with a single work unit. The user can configure the size of the descriptor set. See Descriptor-Based Flow Modes .

Disabled State

The channel is disabled because the enable bit = 0 or because of an error.

DMAC

An acronym used for a DMA cluster.

DMA Channel

A single DMA engine that has all the capabilities and registers as defined for a given processor. A DMA channel or engine is connected to a single peripheral.

DMA Cluster

A grouping of multiple DMA channels with a shared SCB crossbar interface, controller, and arbiter. Also known as a DMAC.

Initial Descriptor

The first descriptor in the descriptor set.

MDMA

Memory-to-Memory DMA data transfer. Two DMA channels are paired to perform a memory read from one address location and a memory write of that data to another address location.

Stop State

A time where the channel is enabled but not currently programmed to perform a data transfer. Programming the flow to STOP causes the channel to enter the stop state at the end of the work unit.

User

Any person, debug, emulator, software routine, or action taken by the core that accesses the MMR registers of the DMA channel or peripherals, or sets up data and descriptors in memory.

Wait State

If instructed to wait for a trigger, the channel enters this state once it has completed a work unit. The channel remains in this state until a trigger occurs. If a trigger came in before reaching the wait state, the channel skips over the wait state upon completion of the work unit.

Work Unit

A single data transaction or series of data transactions performed based on the configuration of the DMA channel. For autobuffer mode, a new work unit is defined at the time all current count registers are initialized to start values. Once all the current count registers count down to zero, the work unit has completed.

Work Unit Chain

A single work unit or a series of work units separated by a `STOP` or disabled state. The work units in the chain (except the last one) are programmed to the required descriptor flow. The last work unit in the chain is programmed to a flow of `STOP` or `AUTO`. `STOP` terminates the state at the end of that work unit. `AUTO` must be terminated by disabling the DMA channel. A work unit chain is also known as a descriptor chain.

Block Diagram

The ***DMA Channel Block Diagram*** shows the functional blocks within the DMA interface.

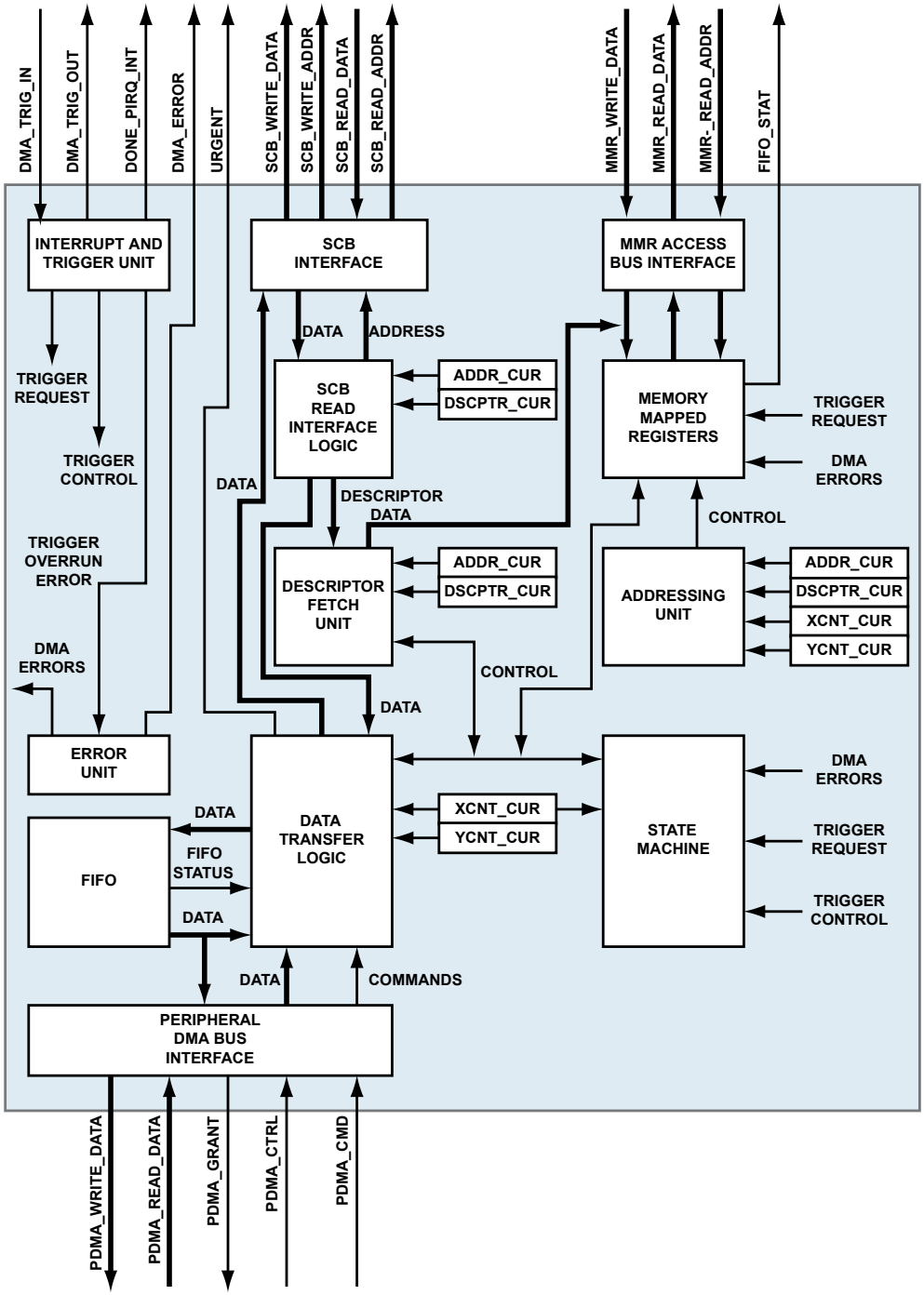


Figure 34-1: DMA Channel Block Diagram

For more information on the interfaces, see:

- DMA Channel Peripheral DMA Bus
- Medium Band Width DMA Channel MMR Access Bus
- DMA Channel Event Control

- DMA Channel SCB Interface

Architectural Concepts

The DMA channel provides a method to transfer data between memory spaces or between memory and a peripheral using a number of system interfaces. The DMA channel provides an efficient method of distributing data throughout the system, freeing up the processor core for other operations. Each peripheral that supports DMA transfers has its own dedicated DMA channel or channels with its own register set. The register set configures and controls the operating modes of the DMA transfers.

DMA Channel SCB Interface

The SCB interface connects the DMA channel to the SCB crossbar allowing for transfers to and from the processors internal memory and other suitable system resources.

The DMA channel connects to the system interconnect through the SCB interface. This connection lets the DMA channel perform work-unit data transfers with memories such as L1, L2 (internal), and L3 (external). In addition to work unit data transfers, the SCB interface also is used for fetching descriptor sets for all the descriptor-based transfer modes.

The DMA channel can support data bus widths of 16, 32, 64, or 128 bits. The data bus widths for a given DMA channel on a specific processor can vary and are not configurable. Read the `DMA_STAT.MBWID` field to determine the assigned bus widths.

SCB Interface Signals

The DMA channel operates at *SYSCLK* frequencies. The SCB crossbar handles the internal arbitration of the transfer requests of all the controllers interfaced to the SCB crossbar instance as shown in the ***SCB Interface Signals*** table.

Table 34-3: SCB Interface Signals

Signal	Width (bits)	Description
SCB_WRITE_DATA	16, 32, 64, or 128	Data bus used for write operations. The width of the bus can be determined from `DMA_STAT.MBWID`.
SCB_WRITE_ADDRESS	32	Write address bus. Provides the address of the first transfer in a burst transaction.
SCB_READ_DATA	16, 32, 64, or 128	Data bus used for read operations. The width of the bus can be determined from `DMA_STAT.MBWID`.
SCB_READ_ADDRESS	32	Read address bus. Provides the address of the first transfer in a burst transaction.

SCB Burst Transfers

The SCB interface supports burst transfers for memory read and write operations. The burst length is a function of the configurable memory size of the DMA channel for the work unit and the fixed bus width of the SCB data bus of the DMA channel.

- If the DMA channel configuration selects a memory transfer size less than or equal to the DMA channels bus width, the burst length is always 1.
- If the configured memory size is greater than the SCB interface bus width, the burst length is sufficient to transfer a transaction as specified by the configured memory size.

Table 34-4: DMA Channel SCB Burst Lengths

Configured Memory Size	Burst Length			
	16-bit Bus	*32-bit Bus*	*64-Bit Bus*	*128-bit Bus*
1 Byte	1	1	1	1
2 Bytes	1	1	1	1
4 Bytes	2	1	1	1
8 Bytes	4	2	1	1
16 Bytes	8	4	2	1
32 Bytes	16	8	4	2

Data Address Alignment

To prevent addressing errors and to maximize bandwidth of the SCB interface to the DMA channel, data addresses align with a multiple of the programmable memory size of the DMA channels configuration. These configuration options appear in the Descriptor Set Address Alignment table.

There are situations in which entire work units may not transfer at the maximum configurable memory size. In this case, the entire work unit can transfer by reducing the configured memory size at the expense of bus bandwidth using descriptor sets as follows:

- The first descriptor set can be configured to transfer data until the larger memory size alignments are met.
- A second descriptor set with a larger memory size configuration then can be used to transfer the bulk of the data in the work unit.
- Finally, a third descriptor set can be used with a smaller memory size to complete any final data transfers that cannot meet the alignment requirements of the previous descriptor set configuration.

Table 34-5: DMA Channel Address Alignment Requirements

Configured Memory Size	Address Restriction
1 Byte	No restriction
2 Bytes	ADDR[0] == 0
4 Bytes	ADDR[1:0] == 0
8 Bytes	ADDR[2:0] == 0
16 Bytes	ADDR[3:0] == 0
32 Bytes	ADDR[4:0] == 0

Descriptor Set Address Alignment

All descriptor set addresses and descriptors within a descriptor set must align to a 32-bit address. For descriptor set fetches, the DMA engine ignores the memory-size configuration of the DMA channel. This feature avoids the need to align descriptor sets based on the memory width configuration of the previous descriptor set.

For descriptor sets containing only a single descriptor, the transfer takes place as a single 32-bit transfer. For descriptor sets containing multiple descriptors, the DMA engine fetches each 32-bit descriptor individually and treats it as multiple 32-bit transfers.

Peripheral Control Commands

The peripheral DMA bus of the DMA channel provides a means for peripherals on the processor to issue commands to the DMA channel. These commands provide greater control over the DMA channel operation. This control improves real-time performance and relieves control and interrupt demands on the core. Peripherals can send commands to the DMA controller over the 3-bit PERI_CMD bus. The DMA control commands extend the set of operations available to the peripheral beyond the simple “request data” command used by peripherals in general. Refer to the appropriate peripheral chapter for a description on how that peripheral uses DMA control commands.

These DMA control commands (see the ***PDMA_CMD Peripheral DMA Control Commands*** table) are not visible to or controlled by the program. But, their use by a peripheral has implications for the structure of the DMA transfers that the peripheral can support. It is important to write application software such that it complies with certain restrictions, regarding work units and descriptor chains. Complying with this guideline makes the peripheral operate properly whenever it issues DMA control commands.

The ***PDMA_CMD Peripheral DMA Control Commands*** table describes the commands the DMA controller issues. The following sections describe these commands in more detail.

Table 34-6: PDMA_CMD Peripheral DMA Control Commands

Command	Name	Description
b#000	NOP	No operation
b#001	Restart	Restarts the current work unit from the beginning
b#010	Finish	Finishes the current work unit and starts the next
b#011	Interrupt	Immediately sets the DMA completion interrupt in the DMA channel
b#100	Request Data	Typical DMA data request
b#101	Request Data Urgent	Urgent DMA data request
b#110	Reserved	Reserved
b#111	Reserved	Reserved

Idle Command

The DMA channel drives this command when the enabled peripheral has no data requests required.

Request-Data Command

The request data command is a request for data transfers between the DMA channel and the peripheral. The request is held by the peripheral until granted or acknowledged by the DMA channel.

Request-Data Urgent Command

The request-data urgent command behaves identically to the request data command, except that---during the commands assertion---the DMA channel performs its memory accesses with urgent priority. This priority includes both data and descriptor fetch memory accesses. For example, a DMA management capable peripheral can use this control command if an internal FIFO approaches a critical condition.

The request is held by the peripheral until granted or acknowledged by the DMA channel.

Peripheral-Control Command Restrictions

The proper operation of the DMA channel FIFO leads to certain restrictions in the sequence of DMA peripheral control commands issued by a peripheral. The following sections describe these restrictions.

DMA Channel Peripheral DMA Bus

The DMA channel connects to peripherals or other DMA channels through the peripheral DMA bus. This bus is a dedicated point-to-point interface supporting data bus widths of 8, 16, 32, or 64 bits. The data bus widths for a given DMA channel on a particular processor can vary and are not configurable. Reading the `DMA_STAT.PBWID` field permits determining the assigned bus width.

The DMA channel operates at the SYSCLK frequencies, as does the peripheral DMA bus. The ***Peripheral DMA Bus Signals*** table provides descriptions of the peripheral DMA bus signals.

Table 34-7: Peripheral DMA Bus Signals

Signal	Width (bits)	Description
PDMA_WRITE_DATA	8, 16, 32, or 64	Data bus used for write operations. The width of the bus can be determined from `DMA_STAT.PBWID`.
PDMA_READ_DATA	8, 16, 32, or 64	Data bus used for read operations. The width of the bus can be determined from `DMA_STAT.PBWID`.
PDMA_DMA_GRANT		Control signals to indicate that data is valid for DMA channel read operations (peripheral transmit). These signals indicate that the DMA channel is ready to receive data for write operations (peripheral receive).
PDMA_CMD	3	The peripheral uses the signal for issuing DMA channel control commands.
PDMA_CTRL		The peripheral uses the control signals to send various commands to the DMA channel and control the direction of flow.

Memory DMA and Triggering

A memory DMA (MDMA) channel provides a means of doing memory-to-memory DMA transfers among the various memory spaces that have DMA support.

The DMA controller implements memory DMA (MDMA) channels by interfacing two DMA channels through the peripheral DMA bus interface. One DMA channel serves for memory read operations, and the second channel servers for memory writes. Depending on the processor, a memory DMA channel can have an additional peripheral, such as a CRC peripheral. The additional peripheral is inserted into the peripheral DMA bus that optionally can be enabled.

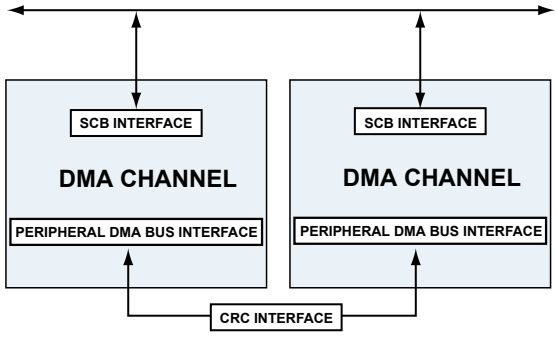


Figure 34-2: MDMA Channel Dedicated Pair

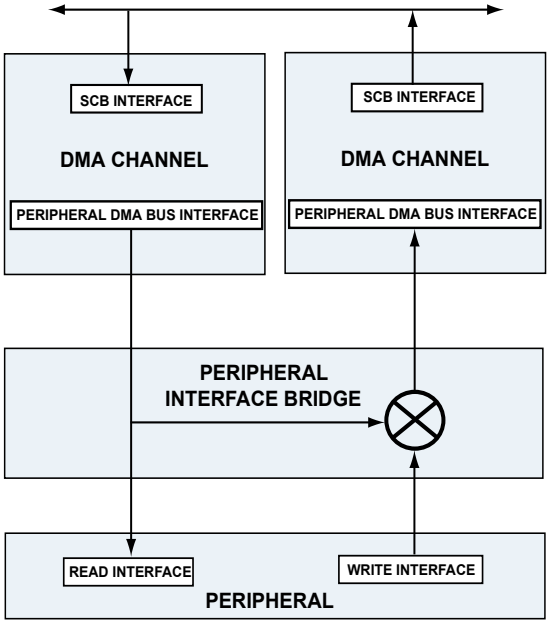


Figure 34-3: MDMA Channel Pair with Peripheral

A memory-to-memory transfer always requires enabled source and destination channels. Because the channels interface through the peripheral DMA bus and can have an additional peripheral inserted into the peripheral DMA bus, programs must make sure to set the same values in the `DMA_CFG.PSIZE` of both the source and destination channels.

The memory DMA channels support the full range of the `DMA_CFG.MSIZE` options for the DMA transfers to and from the memories.

Because the MDMA channel consists of two DMA channels, the entire MDMA channel has two sets of FIFOs, one in the read channel and one in the write channel. This FIFO usage allows for more efficient bursting of both read and write transactions using the available bandwidth. While the DMA_CFG.PSIZE configuration must be identical for both source and destination DMA channels, this restriction does not apply for the DMA_CFG.MSIZE configuration.

Configure the DMA_CFG.PSIZE bits to a value no larger than the supported bus width of the peripheral DMA bus. From a performance perspective, use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the PSIZE value used.

NOTE: This is applicable for all of the DMA channels, except the MDMA channels 2 and 3. For the MDMA 2 and 3, the minimum *PSIZE* and *MSIZE* is 4 bytes (32 bits).

The independent source and destination DMA channels also have their own dedicated interrupt and trigger events. While it is normal practice to have only event generation performed at destination DMA completion, programs also can use other means of interrupt generation.

Configuration of an MDMA transfer is done in a similar manner to peripheral DMA transfers, except for writing two DMA channel registers instead of one.

To control the pace of data transfers, use triggers on either the memory read or the memory write channel pair used in an MDMA operation. Setting the DMA_CFG.TWAIT bit in the memory read channel prevents both channels from transferring data before the system is ready. However, only configuring the memory write channel to wait for a trigger allows for data fetch from the memory in anticipation of the memory write operation.

The processor supports these categories of MDMA channels:

Enhanced Bandwidth MDMA. The channels run in SYSCLK domain and support 32-bit memory and peripheral bus width. The maximum theoretical bandwidth of this MDMA stream is up to 4 × 500 = 2000 MB/s.

Maximum Bandwidth MDMA. The channels run in SYSCLK domain and support 64-bit memory and peripheral bus width. The maximum theoretical bandwidth of these MDMA streams is up to 8 × 500 = 4000 MB/s.

Table 34-8: MDMA Streams

MDMA Stream	Source Channel	Destination Channel	CRC Support	Other Names	Operating Clock	Bus Width (bits)	Maximum Theoretical BW (Mbytes/ second)
0	8	9	Yes	CRC0	SYSCLK (500 MHz max)	32	2000
1	18	19	Yes	CRC1			
2	39	40	No	MSMDMA			
3	43	44	No	HSMDMA		64	4000

Table 34-8: MDMA Streams (Continued)

MDMA Stream	Source Channel	Destination Channel	CRC Support	Other Names	Operating Clock	Bus Width (bits)	Maximum Theoretical BW (Mbytes/ second)
4	45	46	Yes	CRC2		32	2000
5	47	48	Yes	CRC3			
6	49	50	No	MSMDMA1			
7	51	52	No	HSMDMA1		64	4000

Medium Band Width DMA Channel MMR Access Bus

The MMR access bus provides access to all the DMA channels memory-mapped registers for DMA channel configuration, monitoring, and debug. The interface has a fixed 32-bit data bus for read and write accesses.

The ***MMR Access Bus Signals*** table provides descriptions of the MMR access bus signals.

Table 34-9: MMR Access Bus Signals

Signal	Width (bits)	Description
MMR_WRITE_DATA	32	Data bus used for write operations to the MMRs from the core
MMR_READ_DATA	32	Data bus used to return read data from the MMRs
MMR_READ_ADDR	7	Address used to select the MMR to access

DMA Channel Operation Flow

A detailed description of the flow of operation of the DMA channel appears in the following topics:

- Startup Flow
- Refresh Flow
- DMA Operating Modes
- Stop Mode
- DMA Channel Errors

Startup Flow

Enabling a DMA operation on a given channel first requires directly writing some or all of the DMA parameter registers. The minimum set of register required to be initialized depends on the desired mode of operation as described in the following sections.

Startup Minimum-Enable Requirements

To start a DMA operation on a given channel, some or all of the DMA parameter registers must first be initialized and configured to the desired DMA channels operating mode.

- For descriptor-array-based flow modes, at minimum, write the DMA_DSCPTR_CUR register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.
- For descriptor-list-based flow modes, at minimum, write the DMA_DSCPTR_NXT register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.
- For non-descriptor-based flow modes, write the DMA_ADDRSTART, DMA_XCNT, and DMA_XMOD registers prior to writing the DMA_CFG register.

Programs can write other registers that can remain static throughout the course of the DMA activity. The write to the DMA_CFG register begins the DMA operation.

ATTENTION: When software directly writes the DMA_CFG register, the DMA controller recognizes this action as the special startup condition. This condition occurs when starting the DMA controller for the first time on this channel or occurs after the DMA channel stops. It is possible for the channel to flag a DMA error condition regardless of the DMA_CFG.EN bit setting.

Startup Operation

The startup operation is initiated by software directly writing the DMA_CFG register when starting DMA for the first time on a channel or after the channel has entered to the stop state.

When the descriptor fetch is complete and the DMA channel is enabled, the DMA_CFG descriptor element replaces the DMA_CFG register content and assumes control. Before this point, the direct write to the DMA_CFG register had control.

At startup, the selected flow mode and the descriptor size determine the course of the DMA initialization process. The DMA_CFG.FLOW field determines whether to load more current registers from descriptor sets in memory. The DMA_CFG.NDSIZE field details how many descriptor elements to fetch before starting the DMA operation. This process does not affect DMA registers that are not in the descriptor; no modifications are made to their prior values.

For descriptor-list flow modes, the channel copies the DMA_DSCPTR_NXT register value into the DMA_DSCPTR_CUR register. Then, the channel fetches new descriptor elements from memory. The DMA_DSCPTR_CUR register indexes each fetch, and the channel increments the index after each fetch. After completion of the descriptor fetch, the DMA_DSCPTR_CUR register points to the next 32-bit word in memory past the end of the descriptor.

If the descriptor fetch is for a descriptor-array mode transfer, the channel does *not* copy the DMA_DSCPTR_NXT register into the DMA_DSCPTR_CUR register. *Instead*, the descriptor fetch indexing begins with the value in the DMA_DSCPTR_CUR register.

If DMA_CFG is not part of the fetched descriptor set, the previous value (originally as written on startup) controls the work unit operation. If the DMA_CFG register is part of the fetched descriptor set, the value programmed by the MMR access controls only the loading of the first descriptor set fetched from memory. The configuration of the DMA_CFG register controls the subsequent DMA work units of the fetched descriptor set.

After the descriptor fetch is complete or if the flow configuration was originally for one of the register-based flow modes, the DMA operation begins. The DMA channel immediately fills its FIFO. For a memory-write operation,

the DMA channel begins accepting data from the peripheral. For a memory-read operation, the DMA channel begins memory reads when the SCB bus grants access to the DMA channel.

When the DMA channel performs its first data-memory access, its address and count computations take their input operands from the start registers. These registers can include DMA_ADDRSTART, DMA_XCNT, and DMA_YCNT, if necessary. The channel writes results back to the current registers. These registers include DMA_ADDR_CUR, DMA_XCNT_CUR, and DMA_YCNT_CUR. Note that the current registers are not valid until the channel performs the first memory access, which can be some time after the write to the DMA_CFG register starts the channel. Once started, the channel automatically loads the current registers from the appropriate descriptor elements, overwriting their previous contents. These automatic-load operations include:

- The channel copies the DMA_ADDRSTART value to DMA_ADDR_CUR.
- The channel copies the DMA_XCNT value to DMA_XCNT_CUR.
- The channel copies the DMA_YCNT to DMA_YCNT_CUR.

Refresh Flow

When the channel completes processing of a work unit, the DMA channel performs the following operations:

- Completes the transfer of all data between memory and the DMA channel.
- Performs a synchronized transition (if the DMA channel configuration is a memory read operation with the DMA_CFG.SYNC bit enabled) *and* transfers all data to the peripheral before continuing.
- Forwards the signals from the DMA channel (if interrupts or triggers are enabled) *and* updates the DMA_STAT register to indicate the interrupt request or trigger events.
- Clears the DMA_STAT.RUN bit field to stop DMA operation (if the flow was set to stop mode) *and* transfers any remaining data in the FIFO of the DMA channel to the peripheral.
- Loads a new descriptor from memory into the DMA registers by way of the contents of the DMA_DSCPTR_CUR register (for descriptor-array mode) *and* increments the DMA_DSCPTR_CUR register

  The channel takes the descriptor size from the DMA_CFG.NDSIZE value before the fetch.
- Copies the DMA_DSCPTR_NXT register into the DMA_DSCPTR_CUR register (for descriptor-list mode), fetches the descriptor from the new contents of the DMA_DSCPTR_CUR register, *and* places these contents into the DMA registers while incrementing the DMA_DSCPTR_CUR register.
- Checks for detection of an incoming trigger event (for descriptor-on-demand array mode):
  - If the channel detects a trigger event, the DMA channel loads a new descriptor from memory into the DMA registers from the contents of the DMA_DSCPTR_CUR register, while incrementing the DMA_DSCPTR_CUR register. The channel takes the descriptor size from the DMA_CFG.NDSIZE value before the fetch.
  - If the channel detects no trigger event, the DMA channel begins the next work unit by reloading the current registers.

- Checks for detection of an incoming trigger event (for descriptor-on-demand list mode):
  - If the channel detects a trigger event, the DMA channel copies the DMA_DSCPTR_NXT register value to the DMA_DSCPTR_CUR register, fetches the descriptor memory from the DMA_DSCPTR_CUR register, *and* places the contents into the DMA registers while incrementing the DMA_DSCPTR_CUR register.
  - If the channel detects no trigger event, the DMA channel begins the next work unit by reloading the current registers as described in the next step.
- Begins the next work unit (if flow configuration is anything other than stop mode) by reloading the current registers (DMA_ADDR_CUR, DMA_XCNT_CUR, and DMA_YCNT_CUR) from their descriptor registers (DMA_ADDRSTART, DMA_XCNT, and DMA_YCNT)

Work Unit Transition Flow

The DMA_CFG.SYNC bit controls transitions from one work unit to the next work unit. In general, continuous transitions have lower latency at the cost of restrictions on changes of data format or addressed memory space in the two work units. These latency gains and data restrictions arise from the way the channel handles the DMA FIFO while fetching the next descriptor.

In continuous transitions, with disabled synchronization, the DMA FIFO pipeline continues to transfer data to and from the peripheral or destination memory. These transfers continue during the descriptor fetch and during the DMA channel pause between descriptor chains. By comparison, synchronized transitions provide better real-time synchronization of interrupts and triggers with a given peripheral state. Synchronized transitions also provide greater flexibility in the data formats and memory spaces of the two work units. This flexibility comes at the cost of higher latency in the transition. In synchronized transitions, the DMA FIFO pipeline drains to the destination or flushes (received data discarded) between work units.

NOTE: The DMA_CFG.SYNC bit of the MDMA source channel controls work unit transitions for MDMA streams. Clear this reserved bit of the MDMA destination channel, placing it in the disabled state. In transmit (memory read) channels, the DMA_CFG.SYNC bit of the last descriptor before the transition controls the transition behavior. In contrast, in receive channels, the DMA_CFG.SYNC bit of the first descriptor of the next descriptor chain controls the transition.

Work Unit Transmit and MDMA Source Transitions

In DMA transmit (memory read) and MDMA source channels, the DMA_CFG.SYNC bit controls the interrupt timing at the end of the work unit. This bit also controls the handling of the DMA FIFO between the current and the next work unit.

If the DMA_CFG.SYNC bit configuration disables synchronization, the DMA channel operates in continuous transition. In a continuous transition, just after reading the last data item from memory, the DMA channel starts all of the following operations parallel:

- Signals the interrupt request or trigger
- Updates the DMA_STAT register to indicate DMA completion status
- Begins fetching the next descriptor

- Delivers the final data items from the DMA FIFO to the destination memory or peripheral

This process lets the DMA channel provide data from the FIFO to the peripheral continuously during the descriptor fetch latency period.

If the configuration disables synchronization, the final interrupt request or trigger (if enabled) occurs when the channel reads the last data from memory. This event occurs at the earliest time that the channel safely can modify the output memory buffer without affecting the previous data transmission. There can be a number of data items remaining in the FIFO and not yet at the peripheral. This number depends on the FIFO depth of the DMA channel. In this configuration, do not use the DMA interrupt request as the sole means of synchronizing the shutdown or reconfiguration of the peripheral following a transmission.

NOTE: If the configuration selects continuous transition on a transmit (memory read) descriptor, the next descriptor must have the same:

- Peripheral transfer size (`DMA_CFG.PSIZE`)
- Read or write direction
- Source memory (internal versus external) as the current descriptor

It is possible to disable synchronization by selecting continuous transition on a work unit with configuration for stop-flow mode and with enabled interrupts or triggers. This approach can result in the execution of the event service routine while draining of the final data is ongoing from the FIFO to the peripheral. If data transfers are in-progress, the FIFO is not yet empty. The `DMA_STAT.RUN` bits of the DMA channels indicate this status. Do not start a new work unit with a different peripheral transfer size or direction while data transfers are in-progress.

CAUTION: Disabling the channel with the `DMA_CFG.EN` bit while data transfers are in-progress causes the loss of the data in the FIFO.

A synchronized transition configuration directs the channel to drain the DMA FIFO to the destination memory or peripheral. This FIFO operation occurs before the channel signals any interrupt and before the channel fetches any subsequent descriptor or data. This operation incurs greater latency, but provides direct synchronization between the DMA interrupt and the state of the data at the peripheral.

If the configuration enables synchronization and enables interrupts, on the last descriptor in a work unit, the interrupt occurs when the channel transfers the final data to the peripheral. This event allows the service routine to switch properly to non-DMA transmit operation. When the event vectors to the interrupt service routine, the DMA channel FIFO is empty, and the DMA channel is no longer running (indicated by the `DMA_STAT.RUN` bits).

A synchronized transition also allows greater flexibility in the format of the DMA descriptor chain. When enabled, the next descriptor can have any `DMA_CFG.PSIZE` configuration or read/write direction supported by the peripheral and can come from either memory space (internal or external). This feature can be useful in managing MDMA work unit queues, since it is no longer necessary to interrupt the queue between dissimilar work units.

Work Unit Receive and MDMA Destination Transitions

In DMA receive channels (memory write operations), the `DMA_CFG.SYNC` bit controls the handling of the DMA FIFO between descriptor chains (not individual descriptor sets), during the DMA channel pause. The

DMA channel pauses after the descriptor sets configured with stop flow mode are complete. Restart the channel (for example, after an interrupt) by writing the DMA_CFG register of the channel with a value that enables the DMA channel. If the configuration disables synchronization in the DMA_CFG value of the new work unit, the configuration selects a continuous transition. In this mode, the DMA FIFO retains any data items received during the channel pause, and they are the first items written to memory in the new work unit. This mode of operation provides lower latency at work unit transitions and ensures no dropping of data items during a DMA pause. The channel provides this operation at the cost of certain restrictions on the DMA descriptors.

NOTE: If the DMA_CFG.SYNC bit disables synchronization on the first descriptor of a chain after a DMA pause, do not change the configuration of the DMA_CFG.PSIZE field of the new chain from the previous descriptor chain (active before the pause). This restriction applies unless the DMA channel is reset between chains by disabling and then re-enabling the DMA channel.

If the DMA_CFG.SYNC bit configuration enables synchronization, the channel uses a synchronized transition. In this mode, only the data that the DMA channel receives from the peripheral after the write to the DMA_CFG register gets to memory. The channel discards any prior data items transferred from the peripheral to the DMA FIFO before this register write occurs. This operation provides direct synchronization between the data stream received from the peripheral and the timing of the channel restart, which occurs on the write to the DMA_CFG register.

For receive DMA operations, the synchronization has no effect in transitions between work units in the same descriptor chain. When the flow mode of previous descriptor was not stopped, the DMA channel did not pause.

If a descriptor chain begins with synchronization enabled, there is no restriction on the DMA_CFG.PSIZE of the new chain in comparison with the previous chain.

NOTE: The peripheral transfer size (DMA_CFG.PSIZE) must not change between one descriptor and the next in any DMA receive (memory write) channel within a single descriptor chain, regardless of the DMA_CFG.SYNC bit setting. In other words, all memory write descriptor sets in a descriptor chain must have the same DMA_CFG.PSIZE value. For any DMA receive channel (memory write operation), there is no restriction on changes of peripheral transfer size (internal versus external) between descriptors or descriptor chains.

Transfer Termination and Shutdown Flow

This section describes channel transfer termination and shutdown in stop flow mode and in autobuffer flow mode.

Stop Flow Mode

In stop flow mode, the DMA channel stops automatically after the work unit is complete. If using a list or array of descriptors to control DMA transfers and if every descriptor contains a DMA_CFG descriptor element, configure the flow of the final DMA_CFG descriptor element to stop mode, stopping the channel gracefully. After completion, the DMA channel remains in the stop state. Do not confuse this state with the disabled state, which either occurs due to a DMA error or occurs through disabling the DMA channel by configuring the DMA_CFG.EN bit.

The intention of disabling the DMA channel through a write to the DMA_CFG.EN bit is to shut down the DMA channel and to enter the disabled state. All memory and peripheral data transfers cease, and only peripheral interrupts pass through the DMA channels interrupt signals. However, the DMA channel maintains the

DMA_STAT.RUN bits. For a write to memory, the outstanding memory-transaction counter tracks returning memory write acknowledgments and updates as required.

For memory reads, the outstanding memory-transaction count also tracks returning memory reads. The channel does not write the memory reads into the FIFO. The channel updates the counter to reflect the completion of the transaction, but the channel ignores the data. The DMA_STAT.RUN bits remain in the *waiting for write ACK or FIFO drain to peripheral* state and do not change to *stop or idle state* until the return of all outstanding transactions.

When the DMA_CFG.EN bit again enables the DMA channel, the channel performs a full reset and clears all counters. If an outstanding memory transaction returns an acknowledgment or read data after this event, a memory transaction error occurred, which generates an error event. Programs must ensure that all outstanding memory transactions complete before reconfiguring the DMA channel. For example, programs can poll the DMA_STAT.RUN bits to return to the stop or idle state before proceeding.

Autobuffer Flow Mode

In this mode, the flow does not use any descriptors in stored memory. Instead, the channel performs DMA in a continuous circular buffer fashion, based on user-programmed DMA register settings. On completion of the work unit, the channel reloads the parameter registers into the current registers, and the DMA controller resumes immediately with zero overhead. Consider this mode as a succession of automatically restarted work units.

For autobuffer-flow modes, the only way to cease operations is to disable the DMA channel through the DMA_CFG.EN bit. One method of changing to a new work unit is:

- Disable the DMA channel
- Set up all the registers (and descriptors in memory, if used) except for DMA_CFG
- Poll DMA_STAT.RUN to wait for the status to reflect stop or idle state, and
- Write DMA_CFG to the new configuration to begin the next work unit

In autobuffer-flow mode or for a list or array of descriptor sets without DMA_CFG descriptors, use an MMR write to the DMA_CFG register to terminate the DMA transfer process. Configure the value of the DMA_CFG.EN bit in this register to disable the DMA channel.

CAUTION: When the configuration disables a DMA channel, the DMA controller disables interrupt logic that is based on work unit transitions. Be aware of the system environment and current actions, so that additional interrupts are not required from the DMA channel.

CAUTION: If disabled through DMA_CFG.EN in the middle of a transaction, the DMA channel completes any transactions that have begun and avoids generating bus errors. However, the channel considers the action of re-enabling the DMA as a hard reset for all internal DMA channel components. Therefore, pay attention to that particular action to avoid unexpected results.

DMA Channel Errors

When an error occurs, the DMA channel maintains all the state and register values that allow programs to diagnose error causes more thoroughly. The greatest benefit to the programmer is to know exactly what operational state the DMA channel was in at the exact moment the error occurred.

Take care to address the root cause of the error, whether or not the problem originated in the DMA channel. If not properly resolved, the error can result in an additional error shortly after operations resume. The problem can cause other errors elsewhere in the DMA channel or associated modules and circuitry. So, take care also to address those potential problems. Ensure that all outstanding memory reads and writes are complete or cleared before resuming DMA channel operation.

After addressing all issues and neutralizing all side effects of any errors, clear the DMA_STAT.ERRC status field and restart the DMA channel by disabling then re-enabling the DMA channel through the DMA_CFG.EN bit.

The following sections describe the error types.

Status and Debug Errors

DMA channel error conditions can cause the DMA process to end abnormally. The DMA channel provides error detection as a tool for system development and debug, helping to identify DMA-related programming errors. When the DMA channel detects an error, the channel immediately stops and discards any returned memory-read transactions. The DMA_STAT.RUN field of the DMA channel indicates the idle state after acknowledging all outstanding memory transactions. In addition, the channel asserts an error interrupt request and updates the DMA_STAT.IRQERR field. Also, the channel updates the DMA_STAT.ERRC field, indicating the error cause of the first detected error. Unless the error occurs at the exact moment that modification of register values occurs, the registers contain the error values.

All the DMA error interrupt requests are combined into a single shared interrupt request output. Combined error signals require reading the DMA_STAT register of each DMA channel associated with a combined error interrupt request to determine the DMA channel responsible for the generation of the interrupt.

The DMA channel error interrupt handler performs the following actions:

- Read the DMA_STAT register of each DMA channel, seeking a channel with the DMA_STAT.IRQERR set to indicate an error.
- Read the DMA_STAT.ERRC field of each DMA channel, determining the cause of the error.
- Clear the problem with the DMA channel. For example, fix the register values.
- Clear the error in the DMA channel through a write-1-to-clear operation to the DMA_STAT.IRQERR bit.

If the channel flags any uncleared error other than a bandwidth monitor error, the channel reports no other error. If the channel reports an uncleared bandwidth monitor error, the channel reports any newly detected error through updating the DMA_STAT.ERRC field.

DMA Configuration Register Errors

The channel only flags these configuration errors when the DMA_CFG.EN bit enables the DMA channel. Error flagging occurs when the configuration:

- Uses a reserved setting
- Enables DMA_CFG.TWAIT in descriptor on-demand flow mode
- Uses an illegal DMA_CFG.NDSIZE
- Uses an illegal DMA_CFG.MSIZE
- Configures DMA_XCNT = 0 or, when DMA_YCNT = 0 in 2D DMA mode
- Uses non-zero value in DMA_CFG.NDSIZE when DMA is configured in stop mode or auto mode
- Enables interrupt or outgoing triggers on DMA_YCNT when DMA is configured in 1D mode
- Use a DMA_CFG.MSIZE that exceeds the FIFO size of the DMA channel
- Uses an illegal DMA_CFG.PSIZE
- Uses a DMA_CFG.PSIZE that exceeds the FIFO size
- Uses a DMA_CFG.PSIZE that exceeds the bus width
- Attempts to change from a transmit operation (memory read) to a receive operation without properly synching in the previous work unit or when it is the first work unit in a new chain
- Attempts to change DMA_CFG.PSIZE of a transmit operation (memory read) without properly synching in previous work unit or when it is the first work unit in a new chain
- Attempts to change from receive operation (memory write) to a transmit operation during a descriptor chain.

  The channel only can change from receive to transmit if the new transmit is synchronized and is the first work unit.
- Attempts to change DMA_CFG.PSIZE of a receive operation (memory write) when the operation was not the first work unit (with DMA_CFG.SYNC enabled)

Illegal Register Write During Run

The channel generates an error when a write occurs to writable registers of an enabled, running DMA channel. The channel blocks the write. The DMA_STAT, DMA_BWLCNT, and DMA_BWMCNT registers are exempt from this behavior.

Address Alignment Error

The channel generates an address alignment error when any of the following apply:

- Alignment of a descriptor address is not on a 32-bit boundary.
- The current DMA_CFG.MSIZE configuration contains an unaligned transfer address. The DMA_ADDRSTART register is not aligned according to the DMA_CFG.MSIZE field.

Memory Access Error

The channel generates a memory access error when the DMA process:

- attempts to access an unpopulated address,

- attempts to access a location that provokes a security violation

The error returned from the memory triggers the memory access error.

Trigger Overrun Error

A trigger overrun error is generated when a new trigger input occurred while an outstanding trigger is waiting. This error is only generated if `DMA_CFG.TOVEN` is enabled.

Bandwidth-Monitor Error

The channel generates this error when the bandwidth-monitor count expires. This error is not fatal, and the DMA channel continues operation.

Control Interface Error

The channel reports control-interface errors as bus errors to the bus controller. This error can result from:

- An address error
- A register write error (write to a read-only register)

DMA Operating Modes

The DMA channel supports a number of different flow modes that control how the DMA channel progresses from one work unit to the next.

The flow mode of a DMA channel is not a global setting. A DMA descriptor set can include the descriptor responsible for configuring the flow of the work unit. There is no restriction, limiting the flow configuration to be the same for the entire descriptor chain. If the descriptor chain is not endless, the last descriptor set configures the flow to stop mode, which results in termination of the descriptor chain after the work unit completes. Another example for mixing flow modes is to create an endless descriptor-array. The configuration of the last descriptor set in the array selects the descriptor-list mode. The next descriptor pointer in this set of descriptors points to the first descriptor in the array.

Register-Based Flow Modes

Register-based DMA operations require configuration by directly writing to the memory-mapped registers of the DMA channel.

Register-based DMA is the traditional method of DMA operation. Software writes all of the configuration of the DMA channel into the memory-mapped registers. This configuration includes information such as the source or destination address and length of the data in the transfer. The DMA controller then starts channel operation. The DMA channel supports the following register-based flow modes.

- Stop Mode
- Autobuffer Mode

The DMA channel supports variable descriptor set sizes within the configuration. The size of a descriptor set can contain as little as a single descriptor. The supported descriptor set sizes can differ between the various

descriptor-based flow modes. In addition to the descriptor set size being configurable, descriptor-based DMA also allows altering the flow mode of the next descriptor set. This feature allows for the transition from descriptor-array mode to descriptor-list mode and permits configuring the flow to stop or autobuffer mode.

Stop Mode

In stop mode, the DMA operation executes only once. If started, the DMA channel transfers the desired number of data words and stops itself again when finished. If the DMA channel is no longer used, software configures the enable bit to disable a paused channel. The channel also can generate interrupts and triggers for each row or work unit completion, depending on the desired operation.

Autobuffer Mode

In autobuffer mode, the DMA operates repeatedly in a circular manner. If the transfer of all data words completes, the channel reloads the address pointer (`DMA_ADDR_CUR`) automatically with the `DMA_ADDRSTART` value. The channel also can generate an interrupt.

The `DMA_CFG.FLOW` field enables autobuffer mode. The configuration must load the `DMA_CFG.NDSIZE` field value, such that the next descriptor size is zero.

Descriptor-Based Flow Modes

Descriptor-based DMA operations fetch descriptor sets from memory allowing for autonomous loading of work units on other work units. Software does not need to set up the DMA sequences directly by writing into the DMA controller registers. Rather, software keeps DMA descriptor sets in memory.

Descriptor-based DMA operations have the following additional attributes.

- The DMA controller autonomously loads the descriptor set from memory to the affected DMA controller registers on demand.
- The channel can fetch descriptor sets from any memory space that supports DMA read operations.
- The descriptor set describes the next operation that the DMA controller performs.
- The descriptor set can include information such as the DMA configuration word as well as data source or destination address, transfer count, and address modify values.

A descriptor set describes a single work unit. The next work unit can reuse some values from the previous one descriptor set. But, this reusage is possible only if they are not overwritten in the subsequent descriptor set fetches and only if the work unit requires the use of this descriptor.

The DMA channel supports the following flow modes with descriptor-based operations.

- Descriptor-Array Mode
- Descriptor-List Mode
- Descriptor-On-Demand Modes

The DMA channel supports variable descriptor set sizes within the configuration. The size of a descriptor set can contain as little as a single descriptor and the supported descriptor set sizes can differ between the various descriptor-based flow modes. In addition to configurable descriptor set size, descriptor-based DMA also allows for altering of the flow mode of the next descriptor set. Programs can transition from one descriptor-based mode to another descriptor-based mode and can also transition to any of the register-based flow modes.

Descriptor-Array Mode

When configured in this mode, the descriptor sets do not contain further descriptor pointers. Software writes the initial descriptor-pointer value, which points to an array of descriptors. This operation assumes that the individual descriptors reside next to each other and assumes that their addresses are known.

The ***Offsets for Descriptor-Array Mode Parameters and Descriptors*** table illustrates how to structure a descriptor set in memory. The descriptor sets must reside in a contiguous block or memory in the format shown in the table. Locate the first descriptor of the next descriptor set in the memory location immediately following the last descriptor of the current descriptor set. The values have the same order as the corresponding offset addresses of the memory-mapped register.

Table 34-10: Offsets for Descriptor -Array Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_ADDRSTART
0x04	DMA_CFG
0x08	DMA_XCNT
0x0C	DMA_XMOD
0x10	DMA_YCNT
0x14	DMA_YMOD

All other DMA channel registers that were not loaded as a result of the descriptor set fetch, retain their previous values. The channel reloads all of the current registers between the descriptor set fetch and the start of the DMA operation for the work unit.

NOTE: At a minimum, write the DMA_DSCPTR_CUR register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.

Descriptor-List Mode

In this flow mode, multiple descriptors form a chained list in which each descriptor set contains a pointer to the next descriptor set, allowing greater flexibility in memory layout options. When the channel fetches the descriptor set, the operation loads this pointer value into the next descriptor pointer register of the DMA channel.

Descriptor Sets

The ***Offsets for Descriptor-List Mode Parameters and Descriptors*** table shows how to structure a descriptor set in memory. Disperse the placement of the descriptor sets throughout memory, having sets reside in different memory

blocks. But, each descriptor of the descriptor set must reside in a contiguous section of memory in the format shown in the table. The values have the same order as the corresponding offset addresses of the memory-mapped registers.

Table 34-11: Offsets for Descriptor-List Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_DSCPTR_NXT
0x04	DMA_ADDRSTART
0x08	DMA_CFG
0x0C	DMA_XCNT
0x10	DMA_XMOD
0x14	DMA_YCNT
0x18	DMA_YMOD

All other DMA channel registers that were not loaded as a result of the descriptor set fetch, retain their previous values. The channel reloads all of the current values of the registers between the descriptor set fetch and the start of the DMA operation for the work unit.

Minimum Startup Requirements

At a minimum, write the `DMA_DSCPTR_NXT` register prior to write to the `DMA_CFG` register, which is the special action required to start the DMA channel.

Descriptor-On-Demand Modes

The Descriptor-Array Mode and Descriptor-List Mode each have an on-demand mode of operation.

In on-demand mode, at the end of the work unit, if the DMA channel has not detected an incoming trigger event, the channel repeats the current work unit. If the DMA channel receives an incoming trigger before completion of the work unit, the channel fetches a new descriptor set.

The ***Offsets for Descriptor-Array Mode Parameters and Descriptors*** and ***Offsets for Descriptor-List Mode Parameters and Descriptors*** tables illustrate how to structure each descriptor set in memory.

Table 34-12: Offsets for Descriptor-Array Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_ADDRSTART
0x04	DMA_CFG
0x08	DMA_XCNT
0x0C	DMA_XMOD
0x10	DMA_YCNT

Table 34-12: Offsets for Descriptor-Array Mode Parameters and Descriptors (Continued)

Descriptor Offset	Parameter Register
0x14	DMA_YMOD

NOTE: For descriptor-array mode, at a minimum, write the DMA_DSCPTR_CUR register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.

Table 34-13: Offsets for Descriptor-List Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_DSCPTR_NXT
0x04	DMA_ADDRSTART
0x08	DMA_CFG
0x0C	DMA_XCNT
0x10	DMA_XMOD
0x14	DMA_YCNT
0x18	DMA_YMOD

NOTE: For descriptor-list mode, at a minimum, write the DMA_DSCPTR_NXT register prior to write to the DMA_CFG register, which is the special action required to start the DMA channel.

Data Transfer Modes

In addition to supporting basic one-dimensional DMA transfers, the DMA channel also supports two-dimensional functionality.

Two-Dimensional DMA

Register-based flow modes and descriptor-based flow modes support two-dimensional data transfers.

In two-dimensional (2D) mode, the X-direction count (DMA_XCNT), the X-direction modifier (DMA_XMOD), the Y-direction count (DMA_YCNT), and the Y-direction modifier (DMA_YMOD) support arbitrary row and column sizes. Also, the modify values can be negative, allowing implementation of interleaved data streams. The DMA_XCNT value specifies the row size, and the DMA_YCNT value specifies the column size; where the DMA_XCNT value must be 2 or greater.

The DMA start address (DMA_ADDRSTART), the X-direction modifier (DMA_XMOD), and the Y-direction modifier (DMA_YMOD) specifications all are in bytes. The alignment must be a multiple of the DMA transfer word size; configured using the DMA_CFG.MSIZE bit. Misalignment results in a DMA channel error.

The DMA_XMOD register value is the byte-address increment that the channel applies after each transfer, decrementing the DMA_XCNT register. The channel does not apply the DMA_XCNT when the inner loop count ends with the DMA_XCNT_CUR register decrementing to 0 from 1. Except, the channel does apply the DMA_XCNT on the final transfer, when the DMA_YCNT register is 1 and the DMA_XCNT register decrements from 1 to 0.

The DMA_YMOD register value is the byte-address increment that the channel applies after each decrement of the value in DMA_YCNT_CUR. However, the channel does not apply the DMA_YMOD value to the last item in the array on which the outer loop count (DMA_YCNT_CUR) also expires by decrementing from 1 to 0.

After the last transfer completes, DMA_YCNT_CUR is 1 and the DMA_XCNT_CUR register is 0. The DMA channels current address points to the last items address plus the DMA_XMOD register value. If the DMA channel programming selects automatic refresh (such as in autobuffer mode), the channel reloads the DMA_XCNT_CUR, DMA_YCNT_CUR, and DMA_ADDR_CUR for the first data transfer of the next work unit.

Interrupt notification is configurable for end-of-row or end-of-work unit completion.

For example, two-dimensional DMA can be used to extract interleaved data (such as RGB values for a video frame) by modifying both DMA_XMOD and DMA_YMOD values. The ***Capturing a Video Data Stream 2D DMA Example*** depicts the process of receiving a stream of the R, G, B values from an N*M frame. The inner loop of the 2D DMA configuration has three values (DMA_XCNT = 3) and a stride (DMA_XMOD) of N*M, chosen such that successive elements in each row are 1-2-3, 4-5-6 and so forth. The outer loop of the 2D DMA configuration has N*M values (DMA_YCNT = N*M) and a negative stride (DMA_YMOD) of 1-2*N*M chosen to instruct the DMA controller to jump from element 3 to 4, 6 to 7 and so forth at the end of each inner loop.

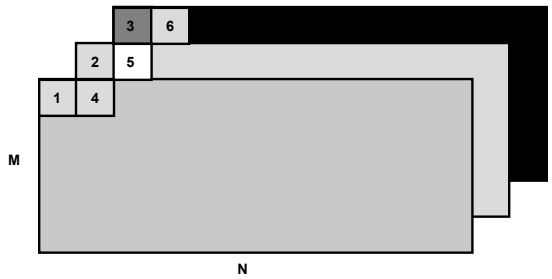


Figure 34-4: Capturing a Video Data Stream 2D DMA Example

DMA Channel Event Control

The DMA channel supports a number of events that provide notification of work unit state, peripheral data request, peripheral interrupt request and completion events, and DMA channel error conditions. In addition to flexible interrupt configuration, the DMA channel also supports incoming and outgoing triggers which are useful in synchronizing the DMA channel with other system resources.

The DMA channel has two interrupt signals for support of a number of events such as work-unit state events, peripheral interrupt request (PIRQ) events, peripheral data request (PDR) events, and DMA channel errors. The channel reports DMA channel errors on a dedicated interrupt signal. All other interrupt sources share an interrupt signal. In addition to flexible interrupt configuration, the DMA channel also supports incoming and outgoing triggers which are useful in synchronizing the DMA channel with other system resources.

The channel can signal the processor on DMA channel events using status information and optional interrupt requests. Programs can use these events to update the progress of data transfers and to request intervention from the processor core. Configure most DMA channel interrupts using bits in the DMA_CFG register. Dedicated bits in

the DMA_STAT register report the occurrence of various events. Use write-one-to-clear (W1C) operations to clear interrupt requests from the status register.

NOTE: Hardware does not clear the interrupt status bits automatically, even when programs disable then reenable the DMA channel. In this situation, the channel deasserts the interrupt signal, after the program disables the DMA channel. But, the status bit remains set until software either re-enables the DMA channel or clears the status bit.

The DMA channel supports the following categories of events on the interrupt signals:

- Work-unit state events generate interrupts on row or on work unit DMA completion.
- A peripheral uses peripheral interrupt request (PIRQ) events to signal when it has completed the transfer of all data.
- A peripheral uses peripheral data request (PDR) events to request data from a disabled or idle DMA channel.
- Error events signal a failure in the work unit.

ATTENTION: While in an error state, the DMA channel does not generate an interrupt to the processor for a work-unit state event or a PIRQ event, nor does the channel forward a PDR event.

Event Signals

The ***Event Signals*** table provides descriptions of DMA channel events.

Table 34-14: Event Signals

Signal	Width (bits)	Description
DMA_ERROR	1	Used to signal an error condition in the DMA channel. The source of the error can be determined by reading the DMA_STAT.ERRC bit.
DONE_PIRQ_INT	1	Signal used to indicate DMA completions events, PIRQ events and also for forwarding PDR events based on configuration. Read the corresponding fields in DMA_STAT to determine the source of the event.
DMA_TRIG_OUT	1	Trigger output that gets routed to the TRU and can be configured to provide notification on row or work unit completion.
DMA_TRIG_IN	1	Trigger input from the TRU that can be used to control the start of a work unit.

Work Unit State Events

Completing a row or a work unit generates a work-unit state event. For either of these events to generate an interrupt request, the configuration of the interrupt of the DMA channel must select one of the available work-unit completion modes.

- Current X count reaching 0 for row completion or 1D DMA work unit completion
- Current Y count reaching 0 for work unit completion of 2D DMA

NOTE: For 1D DMA, a DMA channel configuration error results if the configuration generates the interrupt request when the current Y counter reaches 0.

The DMA channel issues the last memory read or write transaction for the row or work unit, then pauses until the return of the read or write acknowledge. After successful acknowledge of the transfer, the DMA channel issues the interrupt request and continues to process the next row or work unit.

Waiting for acknowledgement of the memory access results in a delay. However, programs can read or modify data in the memory without adversely affecting or being affected by the DMA transfer.

NOTE: While the DMA channel pauses waiting for acknowledgement of the memory transfer, the DMA channel is still capable of fetching the next descriptor set. This fetch gets the channel ready to process the next work unit as soon as the memory access completes.

The channel configuration of the synchronization feature also affects interrupt timing. For memory-read operations with synchronization enabled, the channel delays the interrupt request until the completion of the last transfer from the DMA channel FIFO to the peripheral. The synchronization feature does not affect interrupt timing for memory write operations.

Peripheral Interrupt Request Events

For peripheral-transmit operations, a peripheral connected to the DMA channel can use peripheral interrupt request (PIRQ) events to indicate that data has left the channel FIFO and to indicate transfer completion.

In order to support PIRQ interrupts, correctly configure the interrupt of the DMA channel. This configuration disables the generation of interrupt requests based on the work unit state and, instead, results in generating an interrupt request when the DMA channel receives the command from the peripheral.

The channel only generates the interrupt request under the following conditions:

- The configuration enables the DMA channel
- The DMA channel is in the stop state
- The configuration of the DMA channel interrupt selects PIRQ operation

Peripheral Data Request Events

Peripheral data request (PDR) events occur when an interfaced peripheral requests data from the DMA channel and the DMA channel (either disabled or enabled) is in the stop state.

When a peripheral sends a data request command to a disabled DMA channel, the DMA channel generates an interrupt to the System Event Controller (SEC). There is no status information reported about this event in the status register of the DMA channel. Instead, the channel identifies the PDR event from the fact that the DMA channel generated an interrupt while disabled. It is possible to further confirm event status by verifying the status of the peripheral interfaced to the DMA channel.

This operation forwards data requests as interrupts when the DMA channel is in the disabled state. The DMA channel can forward PDR events as an interrupt request when the DMA channel is in the stop state after the

completion of a work unit. The forwarding of this interrupt when the DMA channel is in the stop state is optional and configured by the program during DMA channel configuration.

DMA Channel Triggers

DMA channel triggers are useful for synchronizing the DMA channel with other events in the system. One usage is to combine channel triggers with each other to create ping-pong buffers. Another usage is to combine the triggers with interrupt requests to notify the processor on reaching a particular milestone that requires service. The channel also can use triggers to enforce a handshake DMA operation in which the trigger acts as a signal for a DMA request.

NOTE: Using the trigger to control the pace of data transfers, such as for handshake DMA, requires that all the data for the entire work unit is ready for transfer.

The DMA channel has a single incoming trigger that can control the pace of the data transfers performed by the DMA channel. The configuration can direct the DMA channel to wait for the incoming trigger before starting the work unit transfer or fetching a descriptor set from memory.

The DMA channel also has a single outgoing trigger signal. This configuration can direct this trigger to signal the end of row or an entire work unit. The DMA channel issues the last memory read or memory write transaction for the row or work unit, then pauses until return of the transfer acknowledge. After acknowledgement of the transfer, the DMA channel issues the trigger before processing the next row or work unit.

NOTE: The peripheral interrupt (PIRQ) and trigger out are independent functions. PIRQ is asserted when the respective peripheral completes the transaction. Trigger out is asserted when the DMA module completes the transaction.

Issuing Triggers

The DMA channel configuration can direct the channel to generate an outgoing trigger signal at the end of row or the end of a work unit. The DMA channel issues the last memory read or memory write transaction for the row or work unit, then pauses until the return of the transfer acknowledge. After acknowledgement of the transfer, the DMA channel issues the trigger before processing the next row or work unit.

NOTE: While the DMA channel pauses waiting for acknowledgement of the memory transfer, the DMA channel is still capable of fetching the next descriptor set. This fetch gets the channel ready to process the next work unit as soon as the memory access completes.

Waiting For Triggers

Programs can use triggering to control the pace of data transfers performed by the DMA channel. The DMA channel enters a wait state before beginning the next work unit if the configuration enables `DMA_CFG.TWAIT` and either of the following apply:

- The channel receives a trigger since the last time the DMA channel left the wait state.
- The channel receives a trigger since its transition from disable to enable.

In the wait state, the DMA channel also does not perform a descriptor fetch. After receiving a trigger, the DMA channel leaves the wait state and begins the next work unit or fetches the next descriptor if configured for a descriptor-based mode of operation.

If a memory-mapped register write operation programs the channel with stop flow mode enabled (DMA_CFG.TWAIT bit) and the channel has not already received a trigger, the DMA channel enters a wait state before performing the data transfer. On receiving the trigger, the DMA channel begins the data transfer portion of the work unit. Once the data transfer is complete, the DMA channel enters the stop state.

If a memory-mapped register write operation programs the DMA channel with the flow mode configured to one of the descriptor-based modes, the DMA channel enters the wait state before performing the descriptor fetch. After completing the descriptor fetch, the DMA channel immediately proceeds to the data transfer, regardless of the value of the DMA_CFG.TWAIT bit. If another (next) descriptor fetch follows the descriptor fetch, the DMA channel enters a wait state before fetching the next descriptor.

If the descriptor fetch returns a descriptor with stop flow mode, the DMA_CFG.TWAIT value for that descriptor does not affect the DMA as the channel enters the stop state after completing the data transfer. The DMA channel only enters the wait state based on DMA_CFG.TWAIT before the next work unit or descriptor fetch.

If the descriptor fetch returns a descriptor configured for autobuffer flow mode, the DMA_CFG.TWAIT for that descriptor does not affect the DMA for the first work unit of the autobuffer transfer. After completing the first work unit and not receiving another trigger, the DMA channel enters the wait state before reinitializing its counters and address registers (if not configured for current addressing). The channel performs the next work unit after receiving the trigger.

The incoming trigger can occur when the DMA channel has not entered the wait state. The trigger can occur while the DMA channel is executing a work unit, is performing descriptor fetch, or is in the stop state. The trigger is held internally. After the work unit is complete, the DMA channel skips the wait state and proceeds directly to executing the following work unit. If the DMA_CFG.TWAIT bit is not enabled, the DMA channel also skips the wait state. However, the trigger is held internally and is used the next time the configuration enables DMA_CFG.TWAIT. This trigger retention allows programs to enable the DMA_CFG.TWAIT functionality several work units apart without concern for losing a trigger. The DMA channels trigger-overrun enable functionality can be enabled in all work units to ensure that multiple triggers do not occur between the work units with the DMA_CFG.TWAIT bit enabled.

DMA Channel Programming Model

Several synchronization and control methods are available for use in development of software tasks which manage peripheral DMA and memory DMA. Software must accept requests for new DMA transfers from other software tasks, integrate these transfers into existing transfer queues, and reliably notify other tasks when the transfers are complete.

In the processor, it is possible to manage each peripheral DMA and memory DMA stream with a separate task or to manage them together with any other stream. Each DMA channel has independent, orthogonal control registers, resources, and interrupts. So, the selection of the control scheme for one channel does not affect the choice of control scheme on other channels. For example, one peripheral can use a linked-descriptor-list, interrupt-driven

scheme while another peripheral can simultaneously use a demand-driven, buffer-at-a-time scheme synchronized by polling DMA events.

The topics that follow describe the steps required to configure the DMA channel for the various modes in addition to the programming concepts required for software synchronization.

Mode Configuration

Use the step-by-step directions that follow to set up the DMA channel for operating modes.

Register-Based Linear-Buffer Stop Flow Mode

This procedure configures the DMA channel of a peripheral to read data from internal memory and to send it to the peripheral for transmission. Upon DMA completion, the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

Assume that the peripheral is in a state where it is ready to transmit data received from the DMA channel.

The task involves writing to a number of DMA channel MMR registers to configure a DMA channel to:

- Read data from internal memory, and
- Send it to a peripheral connected to the peripheral DMA bus.

1. Write the DMA_ADDRSTART register.

   ADDITIONAL INFORMATION: Software can use the address to calculate the most optimum possible DMA_CFG.MSIZE.

2. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE, and the calculation also must consider the start address alignment.

3. Write the DMA_XCNT register based on the calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: The DMA_XCNT value is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

4. Write the DMA_XMOD register.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. Always specify this register as a number of bytes.

5. Write the DMA_CFG register with DMA_CFG.EN configured to enable the DMA channel.

   ADDITIONAL INFORMATION: Set the DMA_CFG.FLOW bit for STOP mode. Configure the DMA_CFG.WNR bit for memory read operation. Configure the DMA_CFG.PSIZE bits to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended

to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

- The DMA_CFG.SYNC bit can be configured to control DMA completion notification timing.
- Interrupts and triggers also can be configured at this step depending on requirements.

Now, the DMA channel is enabled, and the buffer is transferred. The DMA channel enters the IDLE state upon completion of the work unit.

Register-Based Autobuffer Flow Mode

This procedure configures the DMA channel of a peripheral to read data from internal memory and send it to the peripheral for transmission. The transmission of the buffer repeats endlessly. Upon DMA completion, the DMA channel restarts the DMA operation, creating an endless circular buffer transfer.

Assume the peripheral is in a state where it is ready to transmit data received from the DMA channel.

The task involves writing to a number of DMA channel MMR registers to configure a DMA channel to:

- Read data from internal memory, and
- Send it to a peripheral connected to the peripheral DMA bus.

1. Write the DMA_ADDRSTART register.

   ADDITIONAL INFORMATION: Use the address to calculate the optimum possible DMA_CFG.MSIZE.

2. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE, and the calculation must consider the start address alignment.

3. Write the DMA_XCNT register based on calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: The DMA_XCNT register value is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

4. Write the DMA_XMOD register.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. Always specify this register as a number of bytes.

5. Write the DMA_CFG register with the DMA_CFG.EN bit configured to enable the DMA channel.

   ADDITIONAL INFORMATION: Set the DMA_CFG.FLOW bit for autobuffer mode. Configure the DMA_CFG.WNR bit for memory read operation. Configure the DMA_CFG.PSIZE bit to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended

to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

- The DMA_CFG.SYNC bit can be configured to control DMA completion notification timing.
- Interrupts and triggers also can be configured at this step depending on requirements.

Now, the DMA channel is enabled, and the buffer transfers until the DMA channel is disabled.

Descriptor-Array Flow Mode

This procedure configures the DMA channel of a peripheral to:

- Read data from memory as described by the descriptor sets in the array, and
- Send the data to the peripheral for transmission.

Descriptor sets are read from an array in memory to configure the individual work units.

Assume the peripheral is in a state where it is ready to transmit data received from the DMA channel. Assume that the array of descriptors is to be initialized with the last descriptor set configured for STOP flow mode.

The task involves writing to a number of DMA channel MMR registers to:

- Configure a DMA channel to read the array in memory, containing the first descriptor set that configured the DMA channel to retrieve, and
- Send the data to a peripheral connected to the peripheral DMA bus.

On DMA completion, the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the DMA_DSCPTR_CUR register with the address of the array in which the descriptor sets are stored.

   ADDITIONAL INFORMATION: The array address must meet any processor alignments restrictions imposed by descriptor fetches.

2. Write the DMA_CFG register with the DMA_CFG.EN bit configured to enable the DMA channel.

   ADDITIONAL INFORMATION: Configure the DMA_CFG.PSIZE bits to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used. Set the DMA_CFG.FLOW bit for descriptor-array mode. Configure the DMA_CFG.NDSIZE bits to describe the number of descriptor elements contained within the first descriptor set. Configure the DMA_CFG.WNR bit for memory read operation.

   - The descriptor set that is fetched controls the DMA_CFG.SYNC configuration and the interrupt or trigger configurations.

The first descriptor set is fetched from memory location provided by the DMA_DSCPTR_CUR register and loaded to the MMR registers of the DMA channel.

Now, the DMA channel is processing all the work units provided in the descriptor array. The DMA channel enters the IDLE state on completion of the final work unit that was configured for STOP flow mode.

Descriptor-List Flow Mode

This procedure configures the DMA channel of a peripheral to:

- Read data from memory as described by the descriptor sets in the list, and
- Send it to the peripheral for transmission.

The DMA controller reads the descriptor sets from a list of descriptors. With the list, each descriptor set has a descriptor that points to the next descriptor set location in memory.

Assume the peripheral must be in a state where it is ready to transmit data received from the DMA channel. Assume that the list of descriptors must be initialized with the last descriptor set in the list configured for Stop flow mode.

The task involves writing to a number of DMA channel MMR registers to:

- Configure a DMA channel to read the list in memory, containing the first descriptor set that configured the DMA channel to retrieve, and
- Send the data to a peripheral connected to the peripheral DMA bus.

On DMA completion, the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the DMA_DSCPTR_NXT register with the address of the first descriptor in the list to be processed.

   ADDITIONAL INFORMATION: The array address must meet any processor alignments restrictions imposed by descriptor fetches.

2. Write the DMA_CFG register with the DMA_CFG.EN configured to enable the DMA channel.

   ADDITIONAL INFORMATION: Set the DMA_CFG.FLOW for descriptor-list mode. Configure the DMA_CFG.NDSIZE bit to describe the number of descriptor elements contained within the first descriptor set. Configure the DMA_CFG.WNR bit for memory read operation. Configure the DMA_CFG.PSIZE bit to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

   - The descriptor set that is fetched controls the DMA_CFG.SYNC configuration and controls the interrupt or trigger configurations.

The first descriptor set is fetched from the memory location provided by DMA_DSCPTR_NXT and is loaded to the MMR registers of the DMA channel.

Now, the DMA channel is processing all the work units provided in the descriptor list. The DMA channel enters the idle state when the final work unit that was configured for stop-flow mode is complete.

Register-Based Memory-to-Memory Transfer in Stop Flow Mode

This procedure configures a memory DMA channel pair in stop flow mode. One DMA channel is configured for memory read operations, while the other DMA channel is configured for memory write.

The task involves writing to a number of DMA channels on two DMA channels that create a memory DMA pair. On DMA completion, the DMA channel enters the idle state, until either the DMA channel is disabled or is reconfigured for a new transfer.

1. Write the DMA_ADDRSTART register of the source DMA channel.

   ADDITIONAL INFORMATION: The address can be used to calculate the optimum DMA_CFG.MSIZE possible.

2. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE and the start address alignment must also be considered.

3. Write the DMA_XCNT register of the source DMA channel based on calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: DMA_XCNT is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

4. Write the DMA_XMOD register of the source DMA channel.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. This register is always specified in the number of bytes.

5. Write the DMA_ADDRSTART register of the destination DMA channel.

   ADDITIONAL INFORMATION: The address can be used to calculate the most optimum DMA_CFG.MSIZE possible.

6. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE and the start address alignment must also be considered.

7. Write the DMA_XCNT register of the destination DMA channel based on the calculated DMA_CFG.MSIZE.

ADDITIONAL INFORMATION: DMA_XCNT is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

8. Write the DMA_XMOD register of the destination DMA channel.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. This register is always specified in the number of bytes.

9. Write the DMA_CFG register of the source DMA channel with DMA_CFG.EN configured to enable the DMA channel.

   ADDITIONAL INFORMATION: The DMA_CFG.FLOW bit field must be configured for stop mode. The DMA_CFG.WNR bit must be cleared for memory read operation. The DMA_CFG.PSIZE bits must be configured to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

   - The DMA_CFG.SYNC bit can be configured to control DMA completion notification timing.
   - Interrupts and triggers also can be configured at this step, depending on requirements. The interrupts and triggers are enabled within the destination DMA channel configuration.

   The memory read DMA transfer begins.

10. Write the DMA_CFG register of the destination DMA channel with DMA_CFG.EN configured to enable the DMA channel.

    ADDITIONAL INFORMATION: The DMA_CFG.FLOW bit field must be configured for stop mode. The DMA_CFG.WNR bit must be set for memory write operation. The DMA_CFG.PSIZE bits must be configured to a value no larger than the supported bus width of the peripheral DMA bus. This value must also match the value written for the source DMA channel configuration.

    - Interrupts and triggers also can be configured at this step depending on requirements.

    The memory write DMA transfer begins.

Both memory DMA channels are now running and the data is transferred from the source address to the destination address. The DMA channel enters the IDLE state upon completion of the work unit.

Programming Concepts

Using the features, operating modes, and event control for the DMA channel to their greatest potential requires an understanding of some DMA channel-related concepts.

Synchronization of Software and DMA

A critical element of software DMA management is the synchronization of DMA work unit completion with software. This synchronization can be achieved using DMA channel interrupt request and trigger events and using a poll of the status bits of these events within the DMA channel registers, or combining these techniques.

Processor polling of DMA address/count/status for completion is not a recommended programming practice. The requirements and limitations of processor polling place significant responsibility onto the code developer to be deeply aware of the underlying hardware. The interrupt requests and triggers are designed for efficient code development and reuse.

Interrupt and Trigger Event-Based Synchronization

Interrupt and trigger based synchronization methods must avoid overrun. An overrun occurs when some events fail to invoke the event handler of a DMA channel for every event due to excessive latency in processing of events. The system design must ensure to either:

- Schedule only one event per channel (for example, at the end of a descriptor list), or
- Space the generated events sufficiently far apart in time that system processing budgets can guarantee service of every event.

The DMA channel issues status information through an interrupt request or trigger event or changes event status bits in the DMA_STAT register. This status guarantees that the last memory operation of the work unit is complete. For memory read DMA transactions, this status means that the FIFO of the DMA channel safely receives the final memory read data. For DMA transactions writing to memory, this status indicates that the DMA channel received an acknowledge of completion of the last write transfer of the work unit.

Register Polling Based Synchronization

Do not poll the DMA channel registers (DMA_ADDR_CUR, DMA_DSCPTR_CUR, DMA_XCNT_CUR, or DMA_YCNT_CUR) as a method of precisely synchronizing DMA with data processing. This approach is inaccurate due to the operation of the DMA channel FIFOs and DMA or memory pipelining. The current address, pointer, and count registers change several cycles in advance of the completion of the corresponding memory operation. This timing is measurable from the time at which the results of the operation are first visible to the core by memory read or write instructions.

For example, in a DMA channel memory write operation to external memory, assume DMA channel *A* initiates a DMA channel write operation. For memories with access latency, this operation requires many system-clock cycles. Meanwhile, DMA channel *B* (which does not in itself incur latency) initiates a transfer, which stalls behind the slow operation of channel *A*. Software monitoring channel *B* could not safely conclude whether the memory location pointed to by the DMA_ADDR_CUR of channel *B*. Also, the software cannot conclude whether the register has been written based solely on the contents of this register.

Polling of the current address, pointer, and count registers can permit loose synchronization of DMA with software. But, the software must allow for the lengths of the DMA or memory pipeline. Also, software must consider the length of the DMA FIFO for a particular peripheral. If the FIFOs are filled with incomplete work, the DMA channel does not advance current address, pointer, or count registers. The incomplete work includes reads that have been started but have not yet finished.

Additionally, software must consider the length of the pipelines to the destination memory. If the DMA FIFO length and channel memory-pipeline length are added, software can estimate the maximum number of incomplete memory operations in progress.

NOTE: The estimate would be a maximum, as the DMA or memory pipeline can include traffic from other DMA channels.

Descriptor Queues

A system designer may want to write a DMA manager facility which accepts DMA requests from other software. The DMA manager software does not know in advance when new work requests are received or what these requests contain. The software could manage these transfers using a circular linked list of DMA descriptors. In such a list, each descriptor sets the DMA_DSCPTR_NXT descriptor, which points to the next descriptor set. And the last descriptor set in the list points to the first descriptor set.

The code that writes into this descriptor list could use the circular addressing modes of the processor. This approach does not need to use comparison and conditional instructions to manage the circular structure. In this case, the DMA_DSCPTR_NXT descriptor of each descriptor set can be written once at startup, and skipped over as new contents are written for each descriptor.

The recommended method for synchronization of a descriptor queue is to use an interrupt or trigger. The descriptor queue is structured, such that (at least) the final valid descriptor set is always programmed to generate an interrupt or trigger event upon completion. More detail is provided in the following sections.

- Queues Using Event Generation for Every Descriptor Set
- Queues Using Minimal Events

Queues Using Event Generation for Every Descriptor Set

In this system, the DMA manager software synchronizes with the DMA channel by enabling an interrupt request or trigger on every descriptor set. Only use this method if the system design can guarantee that each work unit completion event is serviced separately (no interrupt or trigger overrun).

To maintain synchronization of the descriptor set queue, the non-interrupt software maintains a count of descriptor sets added to the queue. The event handler (either interrupt or trigger) maintains a count of completed descriptor sets removed from the queue. The counts are equal only when the DMA channel is paused after having processed all the descriptor sets.

When each new work unit event is received, the DMA manager software initializes a new descriptor set, taking care to set the flow to stop mode. Next, the software compares the descriptor set counts to determine whether the DMA channel is running. If the DMA channel is paused (counts equal), the software increments its count. Then, the software starts the DMA channel by writing the DMA_CFG of the new descriptor set.

If the counts are unequal, the software instead modifies the DMA_CFG of the next-to-last descriptor set, such that it now describes the newly queued descriptor set. This operation does not disrupt the DMA channel provided the rest of the descriptors of the set are initialized in advance. It is necessary to synchronize the software to the DMA to determine whether the DMA channel read the new or the old DMA_CFG value.

The event handler performs the synchronization operation. When an event is detected, the handler reads the DMA_STAT register of the DMA channel. If the DMA_STAT.RUN bit indicates that the DMA channel is running, the channel has moved on to processing another descriptor. The event handler can increment its count and exit. If the DMA_STAT.RUN bit indicates that the channel is not running, the channel is paused because either:

- There are no more descriptor sets to process, or
- The last descriptor set was queued too late

Where *too late* means that the modification of the DMA_CFG of the next-to-last descriptor set occurred *after* that descriptor was read into the DMA channel. In this case, the event handler does the following:

- Writes the DMA_CFG value appropriate for the last descriptor set to DMA_CFG register of the DMA channel,
- Increments the completed descriptor count, and
- Exits

If the event latencies of the system are large enough to cause any of the events to be dropped, this system can fail. An event handler capable of safely synchronizing multiple descriptor set interrupt requests is complex, performing several MMR accesses to ensure robust operation. In such a system environment, a minimal event synchronization method is preferred.

Queues Using Minimal Events

In this system, only one DMA interrupt request or trigger event is generated in the queue at any time. The DMA event handler for this system can also be extremely short. Here, the descriptor queue is organized into an *active* and a *waiting* portion, where events are enabled only on the last descriptor set in each portion.

When each new DMA request is processed, the software fills in the content of a new descriptor set and adds it to the waiting portion of the queue. The DMA_CFG descriptor of the descriptor set must have the flow set to stop mode. If more than one request is received before the DMA queue completion event occurs, the non-interrupt code queues later descriptor sets. It forms a waiting portion of the queue separate from the active portion of the queue that the DMA channel is processing. In other words, all but the last active descriptor sets contain flow values for a descriptor-based mode and have no event enable set.

The last active descriptor set has the stop flow mode and an event generation enabled. Also, all but the last waiting descriptor sets are configured for descriptor-based flow modes with no event generation. Only the last waiting descriptor set is configured for stop flow mode and event generation enabled. This configuration ensures that the DMA channel can automatically process the whole active queue before then issuing one event. Also, this arrangement makes it easy to start the waiting queue within the event handler by a single DMA_CFG register write.

After queuing a new waiting descriptor, the non-interrupt software leaves a message for its interrupt handler in a memory mailbox location. The location contains the desired DMA_CFG value for starting the first waiting descriptor set in the waiting queue (or 0, indicating no waiting descriptors).

The software must not modify the contents of the active descriptor set queue directly once processing by the DMA channel has started, unless careful synchronization measures are taken. In the most straightforward implementation of a descriptor set queue, the DMA manager software never modifies descriptors on the active queue. Instead, the DMA manager waits until the DMA queue completion event indicates that the processing of the entire active queue is complete.

When a DMA queue completion event is received, the event handler reads the mailbox from the non-interrupt software and writes the value to the DMA_CFG register of the DMA channel. This write to a register restarts the

queue, effectively transforming the waiting queue to an active queue. The event handler then passes a message back to the non-interrupt software indicating the location of the last descriptor set accepted into the active queue.

However, the event handler can read its mailbox and find a DMA_CFG value of zero, indicating there is no more work to perform. It then passes an appropriate message back to the non-interrupt software indicating that the queue has stopped.

The non-interrupt software which accepts new DMA work unit requests must synchronize the activation of a new work unit with the interrupt handler. If the queue has stopped (the mailbox from the event handler is zero), the non-interrupt software must start the queue. (The queue starts by writing the first descriptor sets DMA_CFG value to the DMA_CFG register of the channel). If the queue is not stopped, the non-interrupt software must not write the DMA_CFG register. (This write causes a DMA error). Instead, it must queue the descriptor onto the waiting queue and update its mailbox directed to the event handler.

ADSP-2184x DMA Register Descriptions

The Direct Memory Access module (DMA) contains the following registers.

Table 34-15: ADSP-2184x DMA Register List

Name	Description
DMA_ADDRSTART	Start Address of Current Buffer Register
DMA_ADDR_CUR	Current Address Register
DMA_BWLCNT	Bandwidth Limit Count Register
DMA_BWLCNT_CUR	Bandwidth Limit Count Current Register
DMA_BWMCNT	Bandwidth Monitor Count Register
DMA_BWMCNT_CUR	Bandwidth Monitor Count Current Register
DMA_CFG	Configuration Register
DMA_DSCPTR_CUR	Current Descriptor Pointer Register
DMA_DSCPTR_NXT	Pointer to Next Initial Descriptor Register
DMA_DSCPTR_PRV	Previous Initial Descriptor Pointer Register
DMA_STAT	Status Register
DMA_XCNT	Inner Loop Count Start Value Register
DMA_XCNT_CUR	Current Count (1D) or Intra-row XCNT (2D) Register
DMA_XMOD	Inner Loop Address Increment Register
DMA_YCNT	Outer Loop Count Start Value (2D only) Register
DMA_YCNT_CUR	Current Row Count (2D only) Register
DMA_YMOD	Outer Loop Address Increment (2D only) Register

Start Address of Current Buffer Register

The DMA_ADDRSTART register contains the start address of the work unit currently targeted for DMA. This register is read/write prior to enabling the channel, but is read-only after enabling channel.

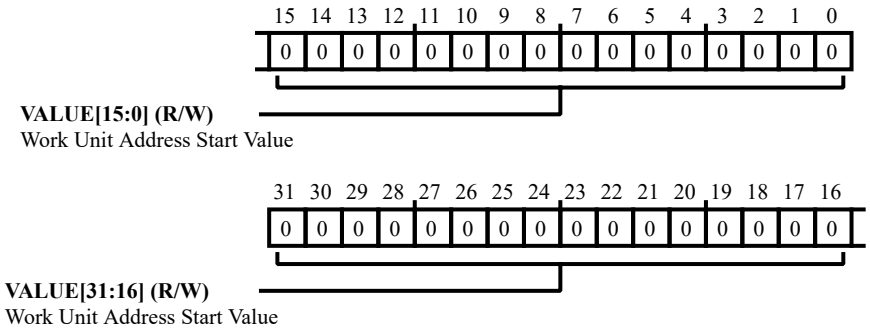


Figure 34-5: DMA_ADDRSTART Register Diagram

Table 34-16: DMA_ADDRSTART Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Address Start Value. The DMA_ADDRSTART.VALUE bit field contains the start address of the work unit currently targeted for DMA.

Current Address Register

The DMA_ADDR_CUR register contains the present memory transfer address for a given work unit. At the start of a work unit, the DMA_ADDR_CUR register is loaded from the DMA_ADDRSTART register, and the DMA_ADDR_CUR register is incremented as each transfer occurs. The DMA_ADDR_CUR register is read/write prior to enabling the channel, but is read-only after enabling the channel.

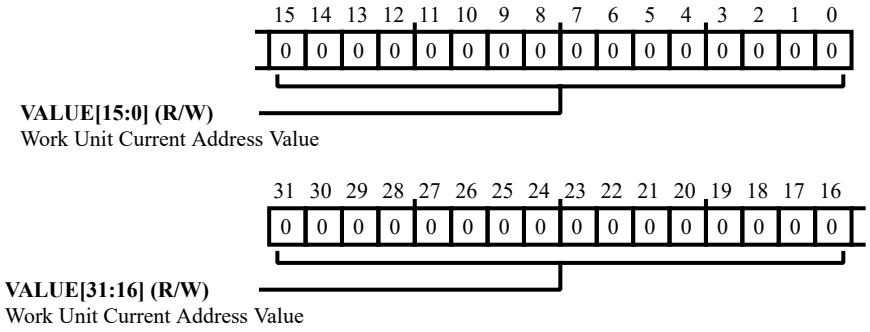


Figure 34-6: DMA_ADDR_CUR Register Diagram

Table 34-17: DMA_ADDR_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Current Address Value. The DMA_ADDR_CUR.VALUE bit field contains the present memory transfer address for a given work unit.

Bandwidth Limit Count Register

The DMA_BWLCNT register contains a count that determines how often the DMA issues memory transactions. The DMA loads the value from the DMA_BWLCNT register into the DMA_BWLCNT_CUR register and decrements the current value each SCLK cycle. When DMA_BWLCNT_CUR reaches 0x0000, the next request is issued, and the DMA reloads DMA_BWLCNT_CUR. This bandwidth limit functionality is not applied to descriptor fetch requests. Programming 0x0000 allows the DMA to request as often as possible. 0xFFFF is a special case and causes requests to stop.

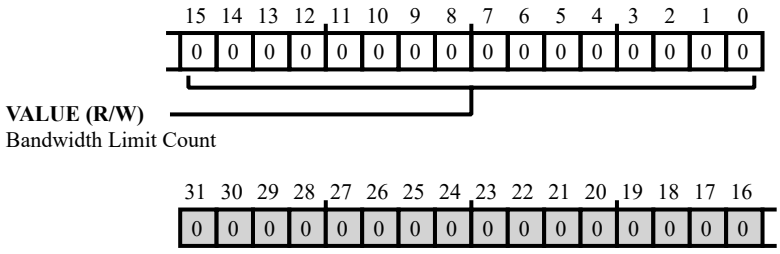


Figure 34-7: DMA_BWLCNT Register Diagram

Table 34-18: DMA_BWLCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Bandwidth Limit Count. The DMA_BWLCNT.VALUE bit field contains a count that determines how often the DMA issues memory transactions.

Bandwidth Limit Count Current Register

The DMA_BWLCNT_CUR register contains the number of SCLK count cycles remaining before the next request is issued.

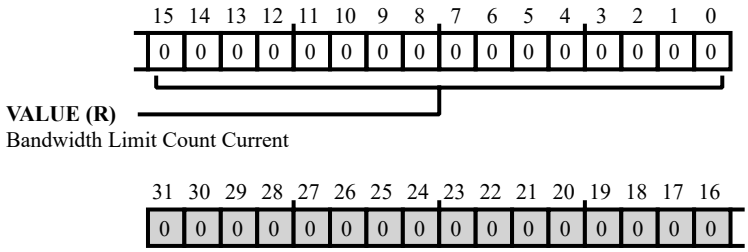


Figure 34-8: DMA_BWLCNT_CUR Register Diagram

Table 34-19: DMA_BWLCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	Bandwidth Limit Count Current. The DMA_BWLCNT_CUR.VALUE bit field contains the number of SCLK count cycles remaining before the next request is issued.

Bandwidth Monitor Count Register

The DMA_BWMCNT register contains the maximum number of SCLK cycles allowed for a work unit to complete. Each time the DMA_CFG register is written (MMR access only), a work unit ends or an autobuffer wraps. The DMA loads the value in this register into the DMA_BWMCNT_CUR register.

The DMA decrements DMA_BWMCNT_CUR every SCLK a work unit is active. If the DMA_BWMCNT_CUR register reaches 0x0000_0000, the DMA_STAT.IRQERR bit is set, and the DMA_STAT.ERRC bit field is set to 0x6. The DMA_BWMCNT_CUR remains at 0x0000_0000 until it is reloaded when the work unit completes.

Unlike other errors, a bandwidth monitor error does not stop work unit processing. Programming 0x0000_0000 disables bandwidth monitor functionality.

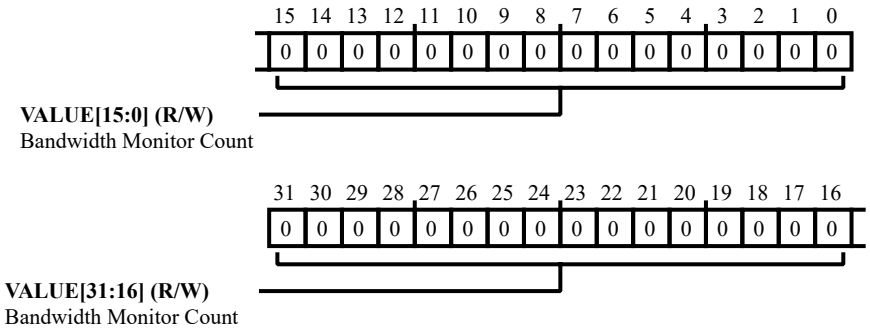


Figure 34-9: DMA_BWMCNT Register Diagram

Table 34-20: DMA_BWMCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Bandwidth Monitor Count. The DMA_BWMCNT.VALUE bit field contains the maximum number of SCLK cycles allowed for a work unit to complete.

Bandwidth Monitor Count Current Register

The DMA_BWMCNT_CUR register contains the number of cycles remaining for the current descriptor to complete.

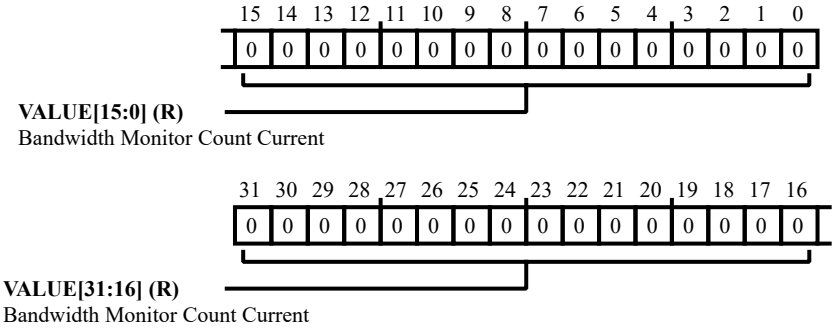


Figure 34-10: DMA_BWMCNT_CUR Register Diagram

Table 34-21: DMA_BWMCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Bandwidth Monitor Count Current. The DMA_BWMCNT_CUR.VALUE bit field contains the number of cycles remaining for the current descriptor to complete.

Configuration Register

The DMA_CFG register sets up DMA parameters and operation modes. Writing to the DMA_CFG register while a DMA process is already running causes a DMA error (except when clearing the DMA_CFG.EN bit).

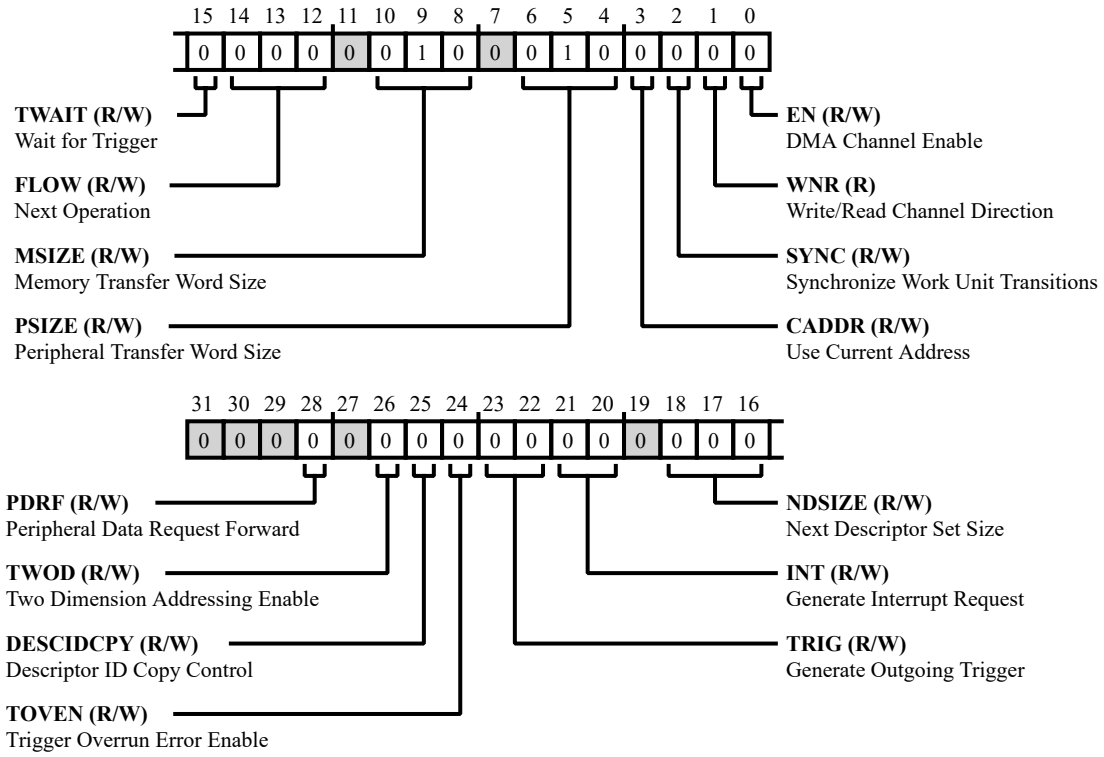


Figure 34-11: DMA_CFG Register Diagram

Table 34-22: DMA_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	PDRF	Peripheral Data Request Forward. The DMA_CFG.PDRF bit defines how the DMA handles data requests from the peripheral while in idle state after a stop mode or memory read work unit. If set, the DMA forwards the peripheral data request as an interrupt. This bit applies only to the DMA_CFG.FLOW bits configured for stop and DMA_CFG.WNR bits configured for memory read.	
		0	Peripheral Data Request Not Forwarded
		1	Peripheral Data Request Forwarded

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	TWOD	Two Dimension Addressing Enable. The DMA_CFG.TWOD bit selects whether the DMA addressing involves only DMA_XCNT and DMA_XMOD (one-dimensional DMA) or also involves DMA_YCNT and DMA_YMOD (two-dimensional DMA).	
		0	One-Dimensional Addressing
		1	Two-Dimensional Addressing
25 (R/W)	DESCIDCPY	Descriptor ID Copy Control. The DMA_CFG.DESCIDCPY bit specifies when to copy the initial descriptor pointer to the DMA_DSCPTR_PRV register. A bus write to the DMA_CFG register to clear the DMA_CFG.EN bit causes the DMA to immediately use the new value of the DMA_CFG.DESCIDCPY bit. To preserve consistency (if required by application), match the new value of this bit to the previous value.	
		0	Never Copy
		1	Copy on Work Unit Complete
24 (R/W)	TOVEN	Trigger Overrun Error Enable. A trigger overrun occurs if more than one trigger was received before the DMA reached the trigger wait state. If DMA_CFG.TOVEN is set, a trigger overrun causes the DMA to flag an error. In cases where a trigger overrun is not a problem, clearing DMA_CFG.TOVEN prevents the overrun from causing an error and halting the DMA. The DMA_CFG.TOVEN operates independently of the DMA_CFG.TWAIT bit selection.	
		0	Ignore Trigger Overrun
		1	Error on Trigger Overrun

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/W)	TRIG	Generate Outgoing Trigger. The DMA_CFG.TRIG selects whether the DMA issues an outgoing trigger, based on the work unit counter values. In one-dimensional mode, the only options are to trigger at the end of the whole work unit (trigger when DMA_XCNT_CUR reaches 0) or not to trigger at all. If in one-dimensional addressing mode, programming the DMA_CFG.TRIG bit field to trigger when DMA_YCNT_CUR reaches 0 (or to reserved) causes the DMA to flag a configuration error. In two-dimensional addressing mode, the trigger options are: at the end of each row of the inner loop (when DMA_XCNT_CUR reaches 0), only after completing the whole work unit (when DMA_YCNT_CUR reaches 0), or no trigger. If in two-dimensional mode and set to trigger when DMA_XCNT_CUR reaches 0, the DMA also issues a trigger at the end of the work unit. If in two-dimensional addressing mode, programming DMA_CFG.TRIG to reserved causes the DMA to flag a configuration error. If DMA_CFG.TRIG is non-zero and the peripheral issues a finish command, the DMA issues a trigger after the finish procedure is complete. For more information about trigger generation timing, see the trigger section of the DMA functional description.	
		0	Never Assert Trigger
		1	Trigger When XCNTCUR Reaches 0
		2	Trigger When YCNTCUR Reaches 0
		3	Reserved

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:20 (R/W)	INT	Generate Interrupt Request. The DMA_CFG.INT bit field selects whether an interrupt request goes to the core based on work unit status or a peripheral interrupt request. For one-dimensional mode, setting the DMA_CFG.INT bits to generate an interrupt request when the DMA_YCNT_CUR register reaches 0 causes the DMA to flag a configuration error. The peripheral interrupt setting lets the DMA generate the last grant indication and to accept and or forward the peripheral interrupt command. The peripheral interrupt selection applies only to the DMA_CFG.FLOW bits set for stop and the DMA_CFG.WNR bits set for memory read. If the DMA_CFG.INT is set for interrupt request on count completion (DMA_XCNT_CUR or DMA_YCNT_CUR reach 0) and the peripheral issues a finish command, the DMA issues an interrupt request after the finish procedure is complete. For more information, see the sections on interrupt generation and peripheral control in the DMA functional description.	
		0	Never Assert Interrupt
		1	Interrupt When X Count Expires
		2	Interrupt When Y Count Expires
		3	Peripheral Interrupt request
18:16 (R/W)	NDSIZE	Next Descriptor Set Size. The DMA_CFG.NDSIZE bit field specifies the number of descriptor elements in memory to load during the next descriptor fetch. The DMA loads the descriptors in a specific order. The descriptor set contains the next descriptor pointer when it is a descriptor list. The descriptor set does not contain the next descriptor pointer when it is a descriptor array.	
		0	Fetch One Descriptor Element
		1	Fetch Two Descriptor Elements
		2	Fetch Three Descriptor Elements
		3	Fetch Four Descriptor Elements
		4	Fetch Five Descriptor Elements
		5	Fetch Six Descriptor Elements
		6	Fetch Seven Descriptor Elements
		7	Reserved

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	TWAIT	Wait for Trigger. The DMA_CFG.TWAIT bit controls whether the DMA waits for an incoming trigger from another channel or user. If the DMA_CFG.TWAIT bit is set, the DMA enters the wait state before starting the next work unit, including descriptor fetch when in descriptor mode. Do not use the wait for trigger control for descriptor-on-demand mode which causes an error. For more information, see the trigger section of the DMA functional description.	
		0	Begin Work Unit Automatically (No Wait)
		1	Wait for Trigger (Halt before Work Unit)
14:12 (R/W)	FLOW	Next Operation. The DMA_CFG.FLOW bit field selects the descriptor handling options.	
		0	STOP. See the Stop Flow Mode section.
		1	AUTO. See the Autobuffer Flow Mode section.
		2	Reserved
		3	Reserved
		4	DSCL. See the Descriptor List Mode section.
		5	DSCA. See the Descriptor Array Mode section.
		6	Descriptor On-Demand List. See the Descriptor List Mode section.
		7	Descriptor On Demand Array. See the Descriptor On Demand section.
10:8 (R/W)	MSIZE	Memory Transfer Word Size. The DMA_CFG.MSIZE bits select memory transfer sizes of 8-, 16-, 32-, 64-, 128-, or 256-bit words. The transfer start address (DMA_ADDRSTART) and transfer increment values (DMA_XMOD, and, if needed, DMA_YMOD) must be a multiple of the memory transfer unit size.	
		2	4 Bytes
		3	8 Bytes
		4	16 Bytes
		5	32 Bytes

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	PSIZE	Peripheral Transfer Word Size. The DMA_CFG.PSIZE bits select peripheral transfer sizes as 8, 16, 32, or 64 bits wide. Each request and grant results in a single peripheral access. There are no bursts on the peripheral bus, so the DMA_CFG.PSIZE selection must be less than or equal to the width of the bus. If the selection is greater than the bus width, a configuration error occurs. The peripheral bus of the processor is dedicated to DMA and peripheral accesses.	
		2	4 Bytes
		3	8 Bytes
3 (R/W)	CADDR	Use Current Address. When the DMA_CFG.CADDR bit is cleared, the DMA loads the DMA_ADDRSTART register on the first access of the work unit. When the DMA_CFG.CADDR bit is set, the DMA uses the DMA_ADDR_CUR register value for the starting address for the work unit and writes the same value to the DMA_ADDRSTART register. This operation permits continuation of a previous work unit. When DMA uses this mode, the fetched start address value (as part of the descriptor set at the end of a descriptor list or array) is ignored.	
		0	Load Starting Address
		1	Use Current Address

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	SYNC	Synchronize Work Unit Transitions. Setting the DMA_CFG.SYNC bit clears the DMA FIFO and pointers before starting the first work unit of a work unit chain. When the transfer direction is memory read/transmit (DMA_CFG.WNR =0), the DMA waits until all data transmits to a peripheral before moving on to the next work unit, clearing the FIFO and pointers. When the transfer direction is memory write/receive (DMA_CFG.WNR =1), the DMA ignores the DMA_CFG.SYNC bit value after processing the first work unit of a work unit chain. As a channel can receive data when turned on but idle, data from the peripheral can still be in the FIFO even though the channel was not programmed. When the DMA_CFG.SYNC bit field is set at the beginning of a work unit chain (during the first work unit), the DMA clears the FIFO, erasing the data put into the FIFO while the channel was idle. Syncing lets programs change the DMA_CFG.PSIZE between individual work units and (in some cases) work unit chains. The sync resets the pointers in the FIFO, preventing misaligned FIFO access. Programs can change the DMA_CFG.MSIZE field between consecutive work units, independent of the DMA_CFG.SYNC bit setting. Syncing also permits changes to transfer direction. Because the data in the FIFO is eliminated, the data that went into the FIFO from one direction (transmit or receive) is not sent back in the other direction after the direction change.	
		0	No Synchronization
		1	Synchronize Channel
1 (R/NW)	WNR	Write/Read Channel Direction. The DMA_CFG.WNR selects receive (write to memory) or transmit (read from memory) channel direction.	
		0	Transmit (Read from memory)
		1	Receive (Write to memory)

Table 34-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	DMA Channel Enable. The DMA_CFG.EN bit enables the selected DMA channel. When a peripheral DMA channel is enabled, data requests from the peripheral denote DMA requests. When a channel is disabled, the DMA unit ignores the peripheral data request and passes it directly to the system event controller. To avoid unexpected results, enable the DMA channel before enabling the peripheral, and disable the peripheral before disabling the DMA channel. A transition of the DMA_CFG.EN bit from 0 to 1 creates a hard reset of all internal counters and states, including the DMA_STAT register. (All other register values remain unaffected.) A transition from 1 to 0 maintains all counters and registers for reading and analysis. Note that if a descriptor loads when this bit is cleared (see the DMA_CFG.FLOW field), the DMA transitions to the off or idle state after the descriptor load is complete.	
		0	Disable
		1	Enable

Current Descriptor Pointer Register

The DMA_DSCPTR_CUR register contains the memory address for the next descriptor to be loaded. The DMA_DSCPTR_CUR register is read/write prior to enabling the channel, but is read-only after enabling the channel. For DMA_CFG.FLOW mode settings that involve descriptor fetches, this register is used to read descriptors into appropriate MMRs before a work unit begins. For descriptor list mode, the DMA_DSCPTR_CUR register is initialized from the DMA_DSCPTR_NXT register before fetching each descriptor set. Then, the address in the DMA_DSCPTR_CUR register increments as each descriptor is read.

When the entire descriptor set has been read, the DMA_DSCPTR_CUR register contains this value:

DMA_DSCPTR_CUR = Descriptor Start Address + Descriptor Size (# of elements)

For descriptor array mode, the DMA_DSCPTR_CUR register, and not the DMA_DSCPTR_NXT register, must be programmed by a MMR access before starting DMA operation.

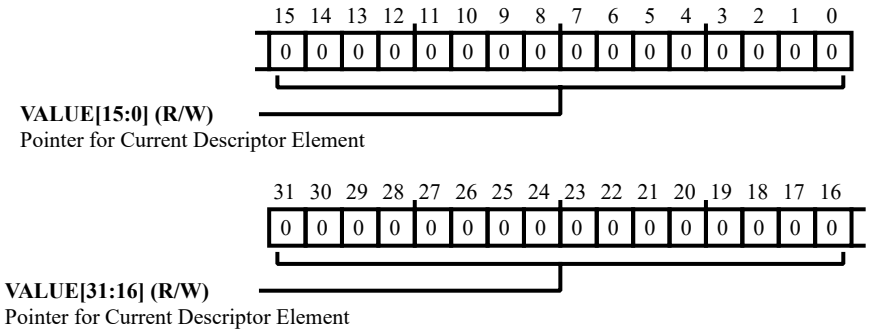


Figure 34-12: DMA_DSCPTR_CUR Register Diagram

Table 34-23: DMA_DSCPTR_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Pointer for Current Descriptor Element. The DMA_DSCPTR_CUR.VALUE bit field contains the memory address for the next descriptor to be loaded.

Pointer to Next Initial Descriptor Register

The DMA_DSCPTR_NXT register specifies the start location of the next descriptor set, which begins when the DMA activity specified by the current descriptor set completes. This register is read/write prior to enabling the channel, but is read-only after enabling channel.

The DMA_DSCPTR_NXT register is only used in descriptor list mode. At the start of a descriptor fetch in this mode, the DMA_DSCPTR_NXT register is copied into the DMA_DSCPTR_CUR register. During descriptor fetch, the DMA increments the DMA_DSCPTR_CUR register value after reading each element of the descriptor set.

In descriptor list mode, the DMA_DSCPTR_NXT register (not the DMA_DSCPTR_CUR register) must be programmed directly through MMR access, before the DMA operation is started. In descriptor array mode, the DMA disregards the DMA_DSCPTR_NXT register and uses the DMA_DSCPTR_CUR register to control descriptor fetch.

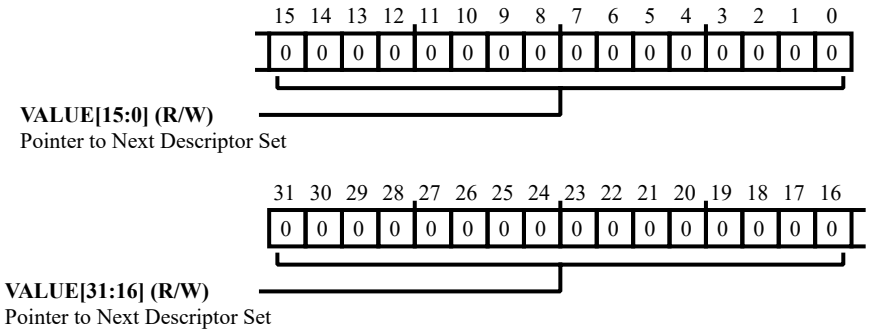


Figure 34-13: DMA_DSCPTR_NXT Register Diagram

Table 34-24: DMA_DSCPTR_NXT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Pointer to Next Descriptor Set. The DMA_DSCPTR_NXT.VALUE bit field specifies the start location of the next descriptor set.

Previous Initial Descriptor Pointer Register

The DMA_DSCPTR_PRV register contains the initial descriptor pointer for the previous work unit. If DMA_CFG.DESCIDCPYis set, the DMA copies the initial descriptor pointer to DMA_DSCPTR_PRV after the work unit completes. Otherwise, the value is not updated.

To indicate an overrun, bit 0 of the DMA_DSCPTR_PRV register is used as a previous descriptor pointer overrun (PDPO) status bit. Due to aligned addressing combined with all descriptors being 32 bits in width, bits 0 and 1 of all descriptor pointers must be zero. Otherwise, an alignment error occurs when used for descriptor fetches. As a result, bit 1 and 0 of the DMA_DSCPTR_PRV register can be used for status. For more information, see the section on descriptor pointer capture in the DMA functional description.

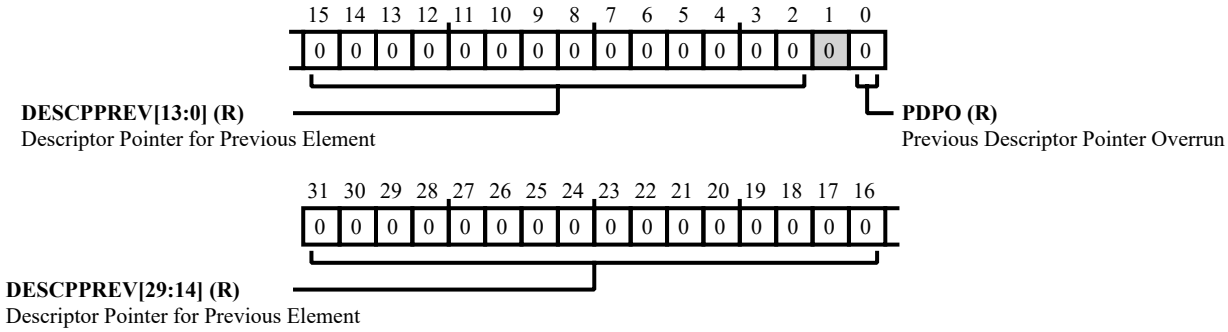


Figure 34-14: DMA_DSCPTR_PRV Register Diagram

Table 34-25: DMA_DSCPTR_PRV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	DESCPPREV	Descriptor Pointer for Previous Element. The DMA_DSCPTR_PRV.DESCPPREV bit field contains the initial descriptor pointer for the previous work unit.	
0 (R/NW)	PDPO	Previous Descriptor Pointer Overrun. The DMA_DSCPTR_PRV.PDPO bit signifies an alignment error. Due to aligned addressing combined with all descriptors being 32 bits in width, bits 0 and 1 of all descriptor pointers must be zero. Otherwise, an alignment error would occur when used for descriptor fetches.	
		0	No Error Occurred
		1	Error Occurred

Status Register

The DMA_STAT register indicates the status of DMA work units, the FIFO, errors, interrupts, and triggers.

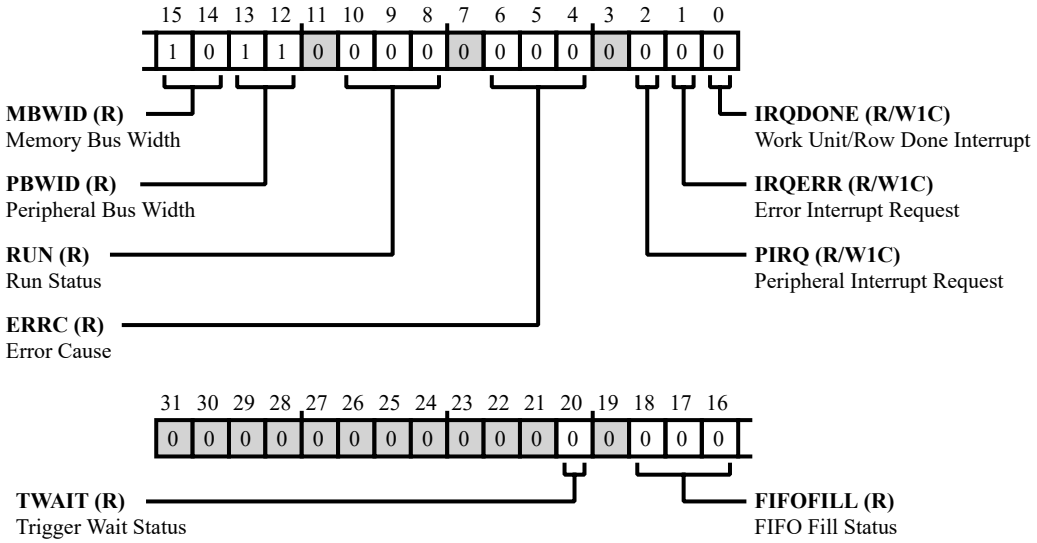


Figure 34-15: DMA_STAT Register Diagram

Table 34-26: DMA_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	TWAIT	Trigger Wait Status. The DMA_STAT.TWAIT bit indicates whether the DMA has or has not received a trigger. This bit is set until it reaches the next wait state. At that point, the bit is cleared, the DMA stops processing that work unit, and the following work unit processes. The DMA does not distinguish between one or more triggers received.	
		0	No Trigger Received
		1	Trigger Received
18:16 (R/NW)	FIFOFILL	FIFO Fill Status. The DMA_STAT.FIFOFILL bit field reports the quantity of data in the FIFO relative to available space.	
		0	Empty
		1	Empty < FIFO = 1/4 Full
		2	1/4 Full < FIFO = 1/2 Full
		3	1/2 Full < FIFO = 3/4 Full
		4	3/4 Full < FIFO = Full
		5	Reserved

Table 34-26: DMA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Reserved
		7	Full
15:14 (R/NW)	MBWID	Memory Bus Width. The DMA_STAT.MBWID bit field indicates the width of the memory bus connected to this DMA.	
		0	2 Bytes
		1	4 Bytes
		2	8 Bytes
		3	16 Bytes
13:12 (R/NW)	PBWID	Peripheral Bus Width. The DMA_STAT.PBWID bit field indicates the width of the peripheral bus connected to this DMA.	
		0	1 Byte
		1	2 Bytes
		2	4 Bytes
		3	8 Bytes
10:8 (R/NW)	RUN	Run Status. The DMA_STAT.RUN bit field reports the DMA's current operational state. If the DMA is in idle or stop state, the DMA_CFG.EN bit is either 0 or 1. This bit field does not clear when a transition of the DMA_CFG.EN bit from 0 to 1 occurs. The DMA_STAT.RUN clears automatically when the DMA completes.	
		0	Idle/Stop State
		1	Descriptor Fetch
		2	Data Transfer
		3	Waiting for Trigger
		4	Waiting for Write ACK/FIFO Drain to Peripheral
		5	Reserved
		6	Reserved
		7	Reserved
6:4 (R/NW)	ERRC	Error Cause. When an interrupt request error occurs (DMA_STAT.IRQERR), the DMA updates the DMA_STAT.ERRC bit field to identify the type of error. For more information, see the errors section of the DMA functional description.	

Table 34-26: DMA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Configuration Error
		1	Illegal Write Occurred While Channel Running
		2	Address Alignment Error
		3	Memory Access or Fabric Error
		4	Reserved
		5	Trigger Overrun
		6	Bandwidth Monitor Error
		7	Reserved
2 (R/W1C)	PIRQ	Peripheral Interrupt Request. The DMA_STAT.PIRQ bit indicates a peripheral interrupt request. Programs can use the DMA_STAT.PIRQ bit status to help determine which DMA asserted the interrupt request. This bit also helps to distinguish between an interrupt request caused by the state of the work unit and an interrupt request caused by the peripheral.	
		0	No Interrupt request
		1	Interrupt Request signaled by peripheral
1 (R/W1C)	IRQERR	Error Interrupt Request. The DMA_STAT.IRQERR bit indicates that the DMA has detected a documented rule violation during DMA programming or operation. The DMA cannot, however, flag all possible programming or operation issues to indicate errors. Programmers can use the DMA_STAT.IRQERR bit to help determine which DMA issued the error interrupt request. The DMA_STAT.IRQERR does not clear a transition of the DMA_CFG.EN bit from 0 to 1. Clear the DMA_STAT.IRQERR bit with a write-1-to-clear operation prior to the DMA_CFG.EN transition for the fields to be reset.	
		0	No Error
		1	Error Occurred
0 (R/W1C)	IRQDONE	Work Unit/Row Done Interrupt. The DMA_STAT.IRQDONE bit indicates that the DMA has detected the completion of a work unit or row (inner loop count) and has issued an interrupt request. Programs can use the DMA_STAT.IRQDONE status to help determine which DMA asserted the interrupt request. Programs can also use these bits to help distinguish between an interrupt request based on the state of the work unit and an interrupt request made by the peripheral. For more information, see the interrupts section of the DMA functional description.	
		0	Inactive
		1	Active

Inner Loop Count Start Value Register

For 2D DMA, the DMA_XCNT register contains the inner loop count. This value selects the number of DMA_CFG.MSIZE size data transfers that make up the length of a row. For 1D DMA, the DMA_XCNT register specifies the number of DMA_CFG.MSIZE size data transfers for the entire work unit. The DMA_XCNT register is read/write prior to enabling the channel, but is read-only after enabling channel. Note that the DMA generates a configuration error if this register is 0x0 when a work unit begins.

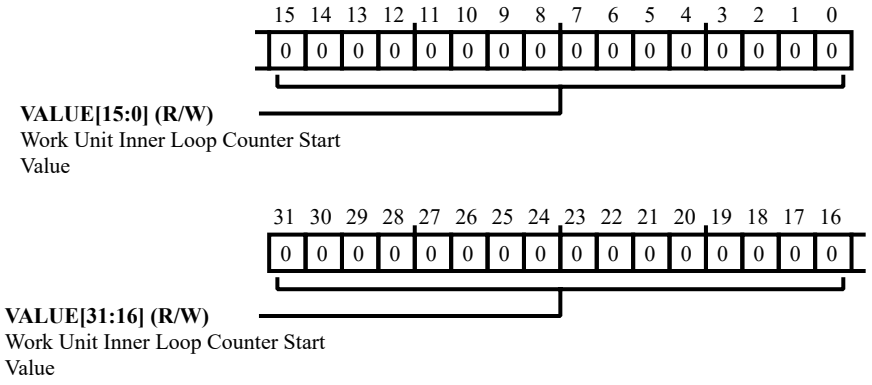


Figure 34-16: DMA_XCNT Register Diagram

Table 34-27: DMA_XCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Inner Loop Counter Start Value. For 2D DMA, the DMA_XCNT.VALUE bit field contains the inner loop count. For 1D DMA, DMA_XCNT.VALUE specifies the number of DMA_CFG.MSIZE size data transfers for the entire work unit.

Current Count (1D) or Intra-row XCNT (2D) Register

For 1D DMA, the DMA loads the DMA_XCNT_CUR register from the DMA_XCNT register at the beginning of each work unit. For 2D DMA, the DMA loads the DMA_XCNT_CUR register from the DMA_XCNT register after the end of each row. The DMA decrements the value in DMA_XCNT_CUR register each time a DMA_CFG.MSIZE size data transfer occurs. When the count in DMA_XCNT_CUR register expires, the work unit is complete. In 2D DMA, the DMA_XCNT_CUR value is 0 only when the entire transfer is complete.

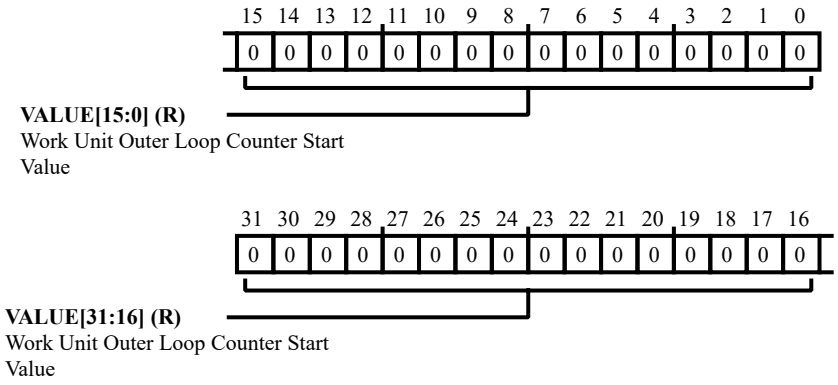


Figure 34-17: DMA_XCNT_CUR Register Diagram

Table 34-28: DMA_XCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Work Unit Outer Loop Counter Start Value. For 1D DMA, the DMA_XCNT_CUR.VALUE bit field holds the DMA_XCNT value at the beginning of each work unit. For 2D DMA, the DMA_XCNT_CUR.VALUE bit field holds the DMA_XCNT value after the end of each row.

Inner Loop Address Increment Register

The DMA_XMOD register contains a signed, two's-complement byte address increment. In 1D DMA, this increment is the stride that is applied after each DMA_CFG.MSIZE size data transfer. The DMA_XMOD register is read/write prior to enabling the channel, but is read-only after enabling the channel.

The DMA_XMOD register value is specified in bytes, regardless of the work unit size. In 2D DMA, this increment is applied after each DMA_CFG.MSIZE size data transfer in the inner loop, up to but not including the last DMA_CFG.MSIZE size data transfer in each inner loop. After the last DMA_CFG.MSIZE size data transfer in each inner loop, the DMA_YMOD register is applied instead, including the last DMA_CFG.MSIZE size data transfer of a work unit.

The DMA_XMOD field can be set to 0. In this case, DMA is performed repeatedly to or from the same address. This approach can be useful for transferring data between a data register and an external memory-mapped peripheral.

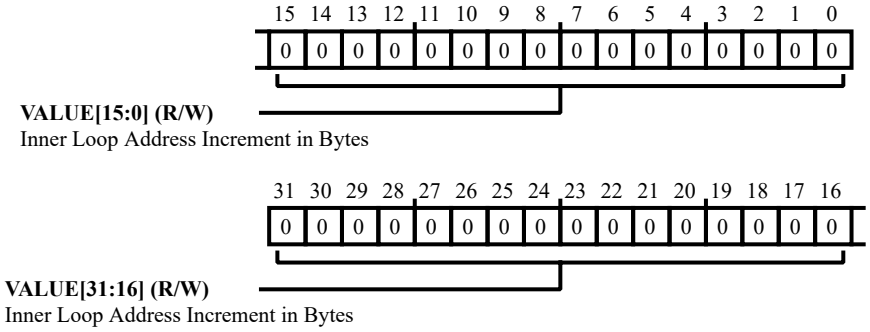


Figure 34-18: DMA_XMOD Register Diagram

Table 34-29: DMA_XMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Inner Loop Address Increment in Bytes. The DMA_XMOD.VALUE bit field contains a signed, two's-complement byte address increment.

Outer Loop Count Start Value (2D only) Register

For 2D DMA, the DMA_YCNT register contains the outer loop count. This register is not used in 1D DMA mode. The DMA_YCNT register specifies the number of rows in the outer loop of a 2D DMA sequence. The DMA_YCNT register is read/write prior to enabling the channel, but is read-only after enabling channel. Note that the DMA generates a configuration error if this register is 0x0 when a work unit begins.

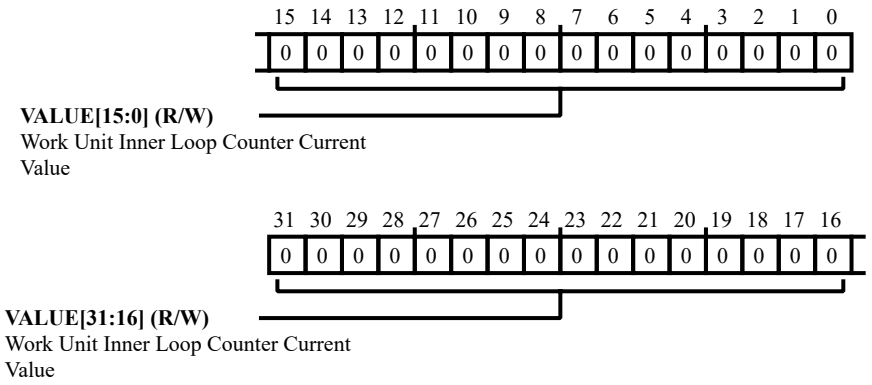


Figure 34-19: DMA_YCNT Register Diagram

Table 34-30: DMA_YCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Inner Loop Counter Current Value. For 2D DMA, the DMA_YCNT.VALUE bit field contains the outer loop count.

Current Row Count (2D only) Register

For 2D DMA, the DMA loads the DMA_YCNT_CUR register from the DMA_YCNT register at the beginning of each 2D DMA session. The DMA_YCNT_CUR register is not used for 1D DMA. The DMA decrements the DMA_YCNT_CUR register each time the DMA_XCNT_CUR register expires during 2D DMA operation, signifying the completion of an entire row transfer.

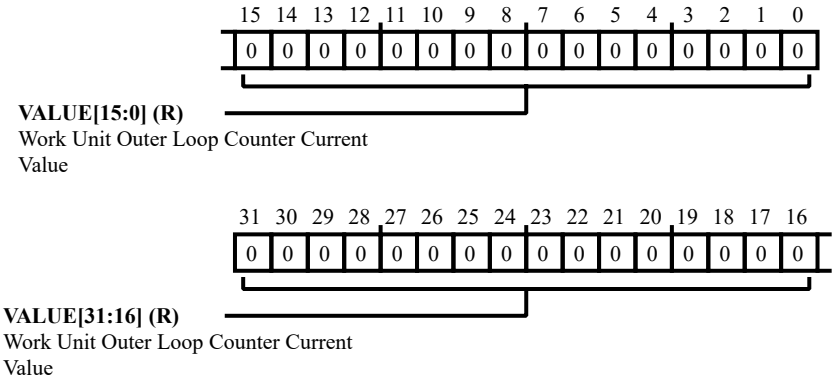


Figure 34-20: DMA_YCNT_CUR Register Diagram

Table 34-31: DMA_YCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Work Unit Outer Loop Counter Current Value. For 2D DMA, the DMA_YCNT_CUR.VALUE bit field holds the value from the DMA_YCNT register at the beginning of each 2D DMA session.

Outer Loop Address Increment (2D only) Register

The DMA_YMOD register contains a signed, two's-complement value. This byte address increment is applied after each decrement of the DMA_YCNT_CUR register. The value is the offset between the last word of one row and the first word of the next row. Note that DMA_YMOD is specified in bytes, regardless of the work unit size. The DMA_YMOD register is read/write prior to enabling the channel, but is read-only after enabling the channel.

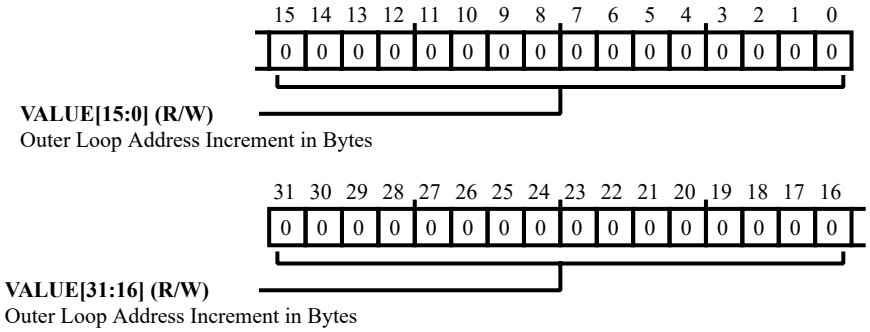


Figure 34-21: DMA_YMOD Register Diagram

Table 34-32: DMA_YMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Outer Loop Address Increment in Bytes. The DMA_YMOD.VALUE bit field contains a signed, two's-complement value.

35 Extended Memory DMA (EMDMA)

The Extended Memory DMA engine can be used in applications that copy data in a non-sequential manner. This includes delay lines, scatter and gather, and circular access types.

NOTE: Previous SHARC processors featured external port DMA. Current SHARC products use EMDMA that can access all memory locations for source and destination DMA operations.

EMDMA Features

EMDMA frees the processor core, allowing it to perform other operations while the data transfers between memories occurs as a background task.

EMDMA has the following features and capabilities.

- Standard mode DMA Transfer
- Chained mode with direction on-the-fly
- Tap list mode (scatter/gather)
- Delay line mode (write to read)
- All the DMA modes can operate in circular fashion
- In circular operation, some modes allow a write back of the index pointer for correct addressing of the next Transfer Control Block (TCB)

EMDMA Functional Description

The following sections provide information on the functional operations and operating modes of EMDMA.

Internal-to-internal DMA

This is accomplished by indexing all external parameter registers with internal addresses.

DMA bursting

DMA supports burst transfers only when the modifier is 1, for any other modifier values, single accesses are performed. The burst size is not user configurable and is chosen by the processor for optimal performance. The maximum burst size is 8 and the minimum is 1.

Transfer control blocks

The structure of a TCB is conceptually the same as that of a traditional linked list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB may point to itself to continuously re-run the same DMA.

Chain pointer DMA

In chained DMA operations, the processor automatically initializes and then begins another DMA transfer when the current DMA transfer is complete.

ADSP-2184x EMDMA Register List

The EMDMA controllers support a variety of direct memory access operations which can access any system memory and transfer the entire block of data. A set of registers govern EMDMA operations. For more information on EMDMA functionality, see the EMDMA register descriptions.

Table 35-1: ADSP-2184x EMDMA Register List

Name	Description
EMDMA_BASE	External Base Address Register
EMDMA_BUFLEN	Circular Buffer Length Register
EMDMA_CHNPTR	Chain Pointer Register
EMDMA_CNT0	Internal Count Register
EMDMA_CNT1	External Count Register
EMDMA_CTL	External Memory DMA Control Register
EMDMA_INDX0	Internal Index Register
EMDMA_INDX1	External Index Register
EMDMA_MOD0	Internal Modifier Register
EMDMA_MOD1	External Modifier Register
EMDMA_TCNT	Delay Line Tap Count Register
EMDMA_TPTR	Tap List Pointer Register

ADSP-2184x EMDMA Interrupt List

Table 35-2: ADSP-2184x EMDMA Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
106	EMDMA0_DONE	EMDMA0 Transfer Done	Edge	
107	EMDMA1_DONE	EMDMA1 Transfer Done	Edge	

ADSP-2184x EMDMA Trigger List

Table 35-3: ADSP-2184x EMDMA Trigger List Generators

Trigger ID	Name	Description	Sensitivity
51	EMDMA0_DONE	EMDMA0 DMA Done	Edge
52	EMDMA1_DONE	EMDMA1 DMA Done	Edge

Table 35-4: ADSP-2184x EMDMA Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

DMA Addressing

Besides the traditional internal-to-external addressing type, the DMA module also supports internal-to-internal transfers. This is accomplished by indexing all external parameter registers with internal addresses. The DMA controller recognizes the transfer by addresses and not by an additional control bit setting.

All the DMA addresses given by EMDMA are word-aligned byte addresses. The programming for the index registers is provided in the *Register Descriptions* section.

DMA Burst Transfers

DMA supports burst transfers only when the modifier is 1. For any other modifier values, single accesses are performed.

The burst size is not user-configurable and is chosen by the processor for optimal performance. The maximum burst size is 8 and the minimum is 1.

The EMDMA uses appropriate burst transfer sizes for optimal throughput. For example, if the word count is 15, then 5 bursts are performed with burst sizes of 8+4+1+1+1 transfers.

Transfer Control Block (TCB) Memory Storage

The location of the DMA parameters for the next sequence comes from the chain pointer register that points to the next set of DMA parameters stored in the processor's internal memory. In chained DMA operations, the processor automatically initializes and then begins another DMA transfer when the current DMA transfer is complete. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Assignment

The structure of a TCB is conceptually the same as that of a traditional linked list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB may point to itself to continuously rerun the same DMA. The EMDMA reads each word of the TCB and loads it into the corresponding register. The end of the chain (no further TCBs are loaded) is indicated by a TCB with a chain pointer register value of zero.

The address field of the chain pointer registers is only 30 bits wide. If a program writes a symbolic address to bit 30 of the chain pointer there may be a conflict with the `EMDMA_CHNPTR.PCI` bit. Programs should clear the upper bits of the address then AND the `EMDMA_CHNPTR.PCI` bit separately, if needed.

Starting Chain Loading

A DMA sequence is defined as the sum of the DMA transfer from when the parameter registers initialize to when the count register decrements to zero. The EMDMA module has a chaining enable bit (`EMDMA_CTL.CHEN`).

To start the chain, write the internal (channel 0) index address of the first TCB to the chain pointer register (`EMDMA_CHNPTR`). When chaining is enabled, DMA transfers are initiated by writing a memory address to the chain pointer register. This is also an easy way to start a single DMA sequence, with no subsequent chained DMAs.

During TCB chain loading, the EMDMA loads the DMA channel parameter registers with values retrieved from system memory.

The address in the chain pointer register points to the highest address of the TCB. This contains the internal (channel 0) index parameter. This means that if a program declares an array to hold the TCB, the chain pointer register should point to the last location of the array and not to the first TCB location.

Buffered Chain Loading Register

The chain pointer register (`EMDMA_CHNPTR`) is buffered. Before the chain loading starts, the buffer is copied into the chain pointer register and is decremented after each register is loaded.

The chain pointer register can be loaded with a new address at any time during the DMA sequence (`EMDMA_CTL.CHEN` =1). This allows a DMA channel to have chaining status deactivated (chain pointer register = 0x0) until some event occurs that loads the chain pointer register with a non-zero value. Writing all zeros to the address field of the chain pointer register also deactivates chaining for the next TCB.

TCB Storage

The EMDMA supports several types of DMA, resulting in different lengths of TCBs. The TCB size varies from six locations (chained DMA) to 13 locations (delay line DMA). The ***EMDMA TCBs*** table shows the required TCBs for chained DMA.

In the following tables, TCB refers to the start address of the TCB array.

Table 35-5: EMDMA TCBs for Standard DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_MOD1
TCB + 0x2	EMDMA_INDX1
TCB + 0x3	EMDMA_CNT0
TCB + 0x4	EMDMA_MOD0
TCB + 0x5	EMDMA_INDX0

The order the descriptors are fetched with circular buffering enabled is shown in the ***EMDMA TCBs for Standard Circular DMA*** table.

Table 35-6: EMDMA TCBs for Standard Circular DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_BUFLEN
TCB + 0x2	EMDMA_BASE
TCB + 0x3	EMDMA_MOD1
TCB + 0x4	EMDMA_INDX1
TCB + 0x5	EMDMA_CNT0
TCB + 0x6	EMDMA_MOD0
TCB + 0x7	EMDMA_INDX0

For delay line DMA, TCB loading is split into two sequences to improve overall priority. The first TCB loads the write parameters (EMDMA_INDX0 – EMDMA_BUFLEN) and the second loads the read parameters (EMDMA_INDX0 – EMDMA_CHNPTR). This two stage loading is transparent to the application. The order the descriptors are fetched for delay line DMA, as shown in the ***EMDMA TCBs for Delay Line DMA*** table.

Table 35-7: EMDMA TCBs for Delay Line DMA

Address	Register
Delay Line Read	
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_TPTR
TCB + 0x2	EMDMA_TCNT
TCB + 0x3	EMDMA_MOD1
TCB + 0x4	EMDMA_CNT0

Table 35-7: EMDMA TCBs for Delay Line DMA (Continued)

Address	Register
TCB + 0x5	EMDMA_INDX0
Delay Line Write	
TCB + 0x6	EMDMA_BUFLEN
TCB + 0x7	EMDMA_BASE
TCB + 0x8	EMDMA_MOD1
TCB + 0x9	EMDMA_INDX1
TCB + 0xA	EMDMA_CNT0
TCB + 0xB	EMDMA_MOD0
TCB + 0xC	EMDMA_INDX0

The order the descriptors are fetched for scatter/gather DMA with circular buffering disabled is shown in the ***EMDMA TCBs for Scatter/Gather DMA*** table.

Table 35-8: EMDMA TCBs for Scatter/Gather DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_TPTR
TCB + 0x2	EMDMA_TCNT
TCB + 0x3	EMDMA_MOD1
TCB + 0x4	EMDMA_INDX1
TCB + 0x5	EMDMA_CNT0
TCB + 0x6	EMDMA_MOD0
TCB + 0x7	EMDMA_INDX0

The order the descriptors are fetched for scatter/gather DMA with circular buffering enabled is shown in the ***EMDMA TCBs for Circular Scatter/Gather DMA*** table.

Table 35-9: EMDMA TCBs for Circular Scatter/Gather DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_BUFLEN
TCB + 0x2	EMDMA_BASE
TCB + 0x3	EMDMA_TPTR
TCB + 0x4	EMDMA_TCNT

Table 35-9: EMDMA TCBs for Circular Scatter/Gather DMA (Continued)

Address	Register
TCB + 0x5	EMDMA_MOD1
TCB + 0x6	EMDMA_INDX1
TCB + 0x7	EMDMA_CNT0
TCB + 0x8	EMDMA_MOD0
TCB + 0x9	EMDMA_INDX0

EMDMA Operating Modes

This section and the ***EMDMA_CTL Register Bit to Operating Modes*** table show the different DMA modes which can be used. The complete register bit descriptions are in the ***External Memory DMA Control Registers (EMDMA_CTL)***.

Table 35-10: EMDMA_CTL Register Bit to Operating Modes

Bit (Name)	Standard	Chained	Scatter/Gather	Delay Line
Control Bits				
0 (EN)	Valid			
1 (TRAN)	Valid			N/A
2 (CHEN)	Valid			
3 (DLEN)	N/A			Valid
4 (CBEN)	Valid			
5 (DFLSH)	Valid			
6	N/A			
7 (WRBEN)	N/A	Valid	(=0)	(=1)
8 (OFCEN)	Valid			N/A
9 (TLEN)	N/A		Valid	N/A
11–10	N/A			
12 (INTDONE0)	Valid			
15–13	N/A			
Status Bits				
17–16 (DFS)	Valid			
19–18	N/A			
20 (DMAS0)	Valid			
21 (CHS)	N/A	Valid	N/A	Valid

Table 35-10: EMDMA_CTL Register Bit to Operating Modes (Continued)

Bit (Name)	Standard	Chained	Scatter/Gather	Delay Line
22 (TLS)	N/A		Valid	N/A
23 (WBS)	N/A			Valid
24 (DMAS1)	Valid			
25 (DIRS)	Valid			
31–26	N/A			

NOTE: Reading additional bit-field information from N/A (Not applicable) bits does not generate a meaningful result.

A program sets up a DMA channel by writing the transfer's parameters to the DMA parameter registers.

NOTE: The `EMDMA_CNT1` parameter register (read only) is a copy of the `EMDMA_CNT0` register. If `EMDMA_CNT0` is written, the `EMDMA_CNT1` register is updated automatically.

Standard DMA

A standard DMA (once it is configured) transfers data from location A to location B. An interrupt can be used to indicate the end of the transfer. To start a new DMA sequence after the current one is finished, a program must first clear the DMA enable bit (`EMDMA_CTL.EN`), write new parameters to the index, modify, and count registers (parameter registers), then set the DMA enable bit to re-enable DMA.

This DMA type resembles the traditional DMA type to initialize the different internal and external parameters (channel0 and channel1) (index, modify and count) registers and configuration of the DMA control registers.

Circular Buffered DMA

Circular buffered DMA resembles the traditional core DAG circular buffered mode by using registers for circular buffering. In this mode, the DMA needs two additional registers (base and length) to support reads and writes to a circular buffer. The ***Circular Buffering Write DMA*** and ***Circular Buffering Read DMA*** figures illustrate circular buffered DMA, in contrast with the ***Standard Write DMA*** figure.

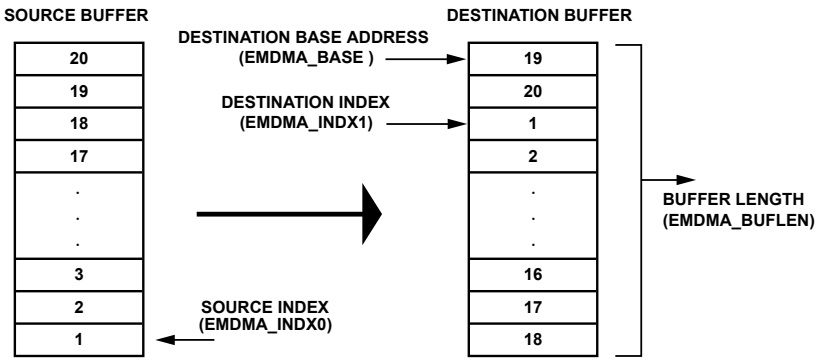


Figure 35-1: Circular Buffering Write DMA

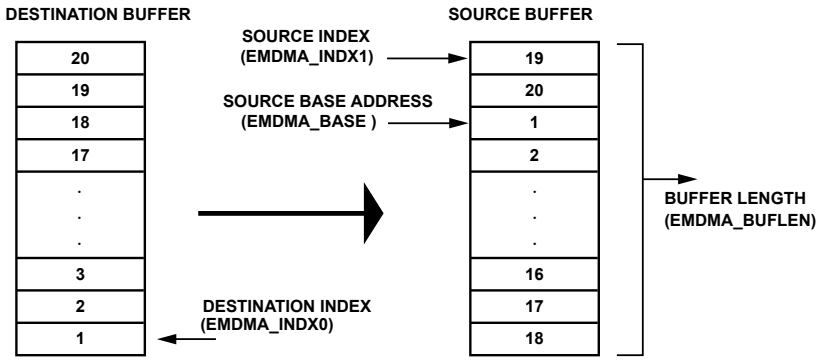


Figure 35-2: Circular Buffering Read DMA

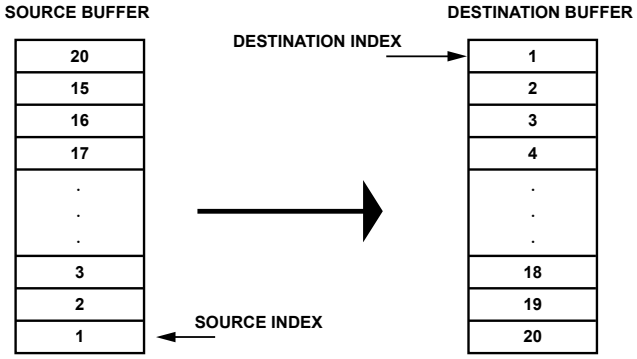


Figure 35-3: Standard Write DMA

NOTE: Circular buffering is available for all operating modes (standard, chained, tap list and delay line DMA).

Chained DMA Mode

Chained DMA sequences are a set of multiple DMA operations, each auto-initializing the next in line. It is used to support automated access by a linked list (repetitive reads and writes to a defined location defined by the individual TCBs). To start a new DMA sequence after the current one is finished, the EMDMA automatically loads new index, modify, and count values pointed to by that channel's `EMDMA_CHNPTR` register. Using chaining, programs can set up consecutive DMA operations and each operation can have different attributes.

DMA data transfers can be set up as continuous or periodic. With chained DMA, the attributes of a specific DMA are stored in internal memory and are referred to as a Transfer Control Block or TCB. Extended Memory DMA loads these attributes in chains for execution. This allows for multiple chains that are finite or infinite.

NOTE: When chaining is enabled, polling should not be used to determine DMA status only because the DMA appears inactive if it is sampled while the next TCB is loading. In such cases where chaining is enabled, along with the polling of DMA status bit, polling of the chaining status bit should also occur to so that the correct status of the DMA is known. For example, the `EMDMA_CTL.CHS` bit should be polled as well as the `EMDMA_CTL.DMAS0` and `EMDMA_CTL.DMAS1` bits when EMDMA is configured in DMA chaining mode.

Data Direction On-the-Fly

A change of external memory data direction for each individual TCB in a chain sequence is allowed.

The EMDMA_CHNPTR.CPDR bit changes the data flow direction. If the EMDMA_CHNPTR.CPDR =0, writes through channel 0 are performed; if EMDMA_CHNPTR.CPDR =1, channel 0 reads are performed. This works similarly to the EMDMA_CHNPTR.PCI bit. The EMDMA_CTL.OFCEN and EMDMA_CTL.CHEN bits must be set (=1) to enable this functionality.

NOTE: If chaining is enabled with the EMDMA_CTL.OFCEN =1, then the EMDMA_CTL.TRAN bit has no effect, and the data direction is determined by the EMDMA_CHNPTR.CPDR bit.

Write Back Circular Index Pointer

Operating the DMA in circular mode requires some special considerations. The index pointer of start address within the buffer may wrap around for the case when IC × IM > EL or it does not finish if IC × IM < EL where:

IC is the value of the EMDMA_INDX0 register

IM is the value of the EMDMA_MOD0 register

EL is the value of the EMDMA_BUFLEN register

In both cases, the TCB start address is no longer valid.

Setting the EMDMA_CTL.WRBEN bit writes (at the end of current TCB block) the current index address + 1 into the TCB memory which is the start address for the next TCB. This bit is only selectable for chained DMA mode. For tap list and delay line modes, this bit is hardwired to 0 or 1.

Scatter/Gather DMA

The purpose of scatter/gather DMA is the transfer of data from/to non-contiguous memory blocks.

The scatter/gather DMA type is a fixed block size scatter/gather DMA that relies on tap list entries in system memory to calculate the (Channel 1 DMA Address) to scatter/gather the DMA. When the DMA direction is Channel 1 write (EMDMA_CTL.TRAN =1) it is a scatter DMA. When (EMDMA_CTL.TRAN =0), it is a gather DMA. This mode also supports chained and circular buffer chained DMAs.

See the ***Scatter DMA (Writes)***, ***Gather DMA (Reads)***, ***Circular Buffering Scatter DMA (Writes)***, and ***Circular Gather DMA (Reads)*** figures.

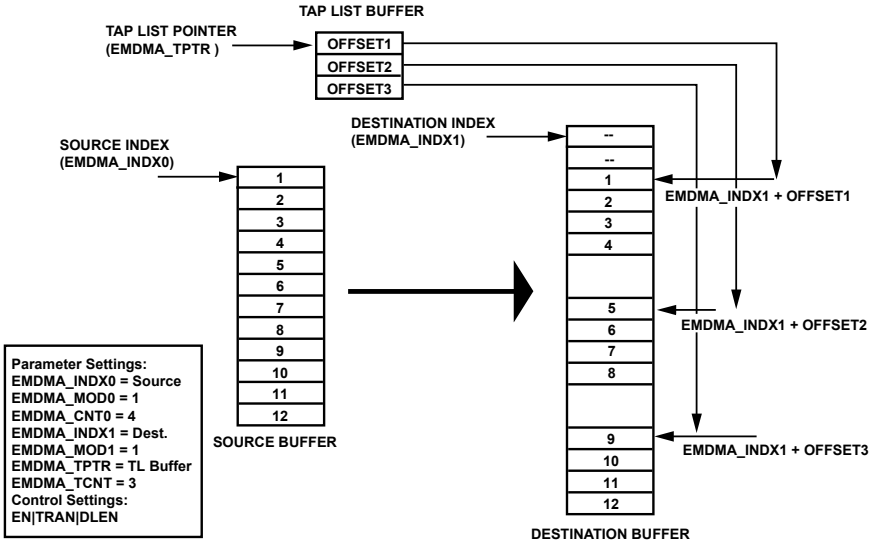


Figure 35-4: Scatter DMA (Writes)

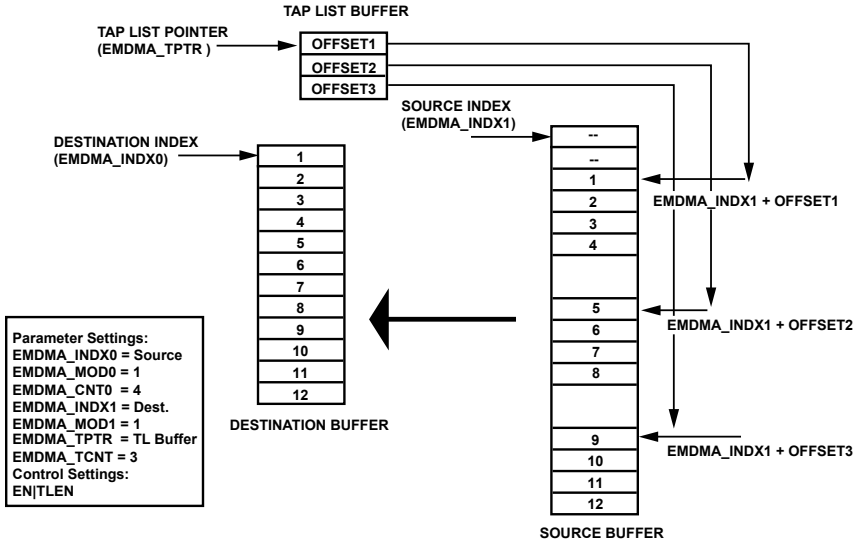


Figure 35-5: Gather DMA (Reads)

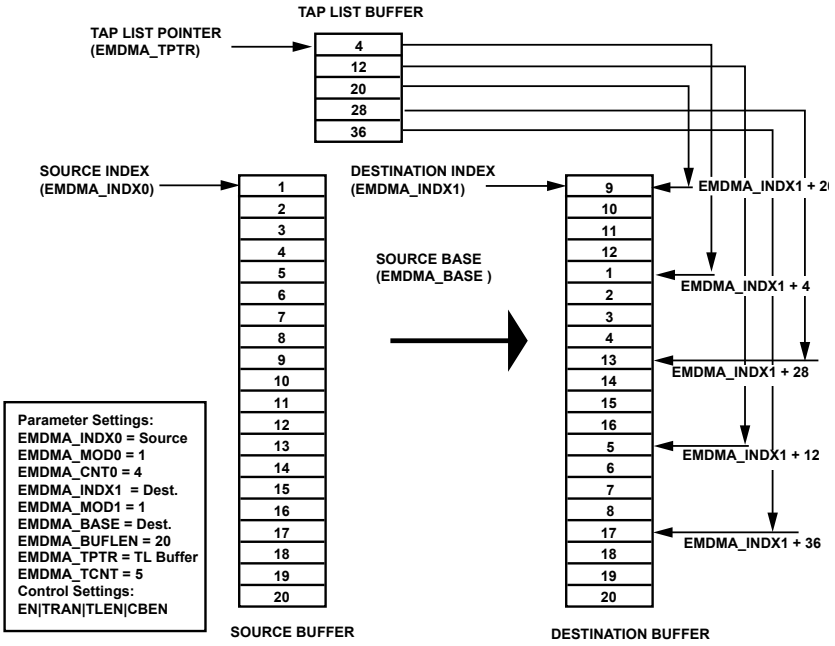


Figure 35-6: Circular Buffering Scatter DMA (Writes)

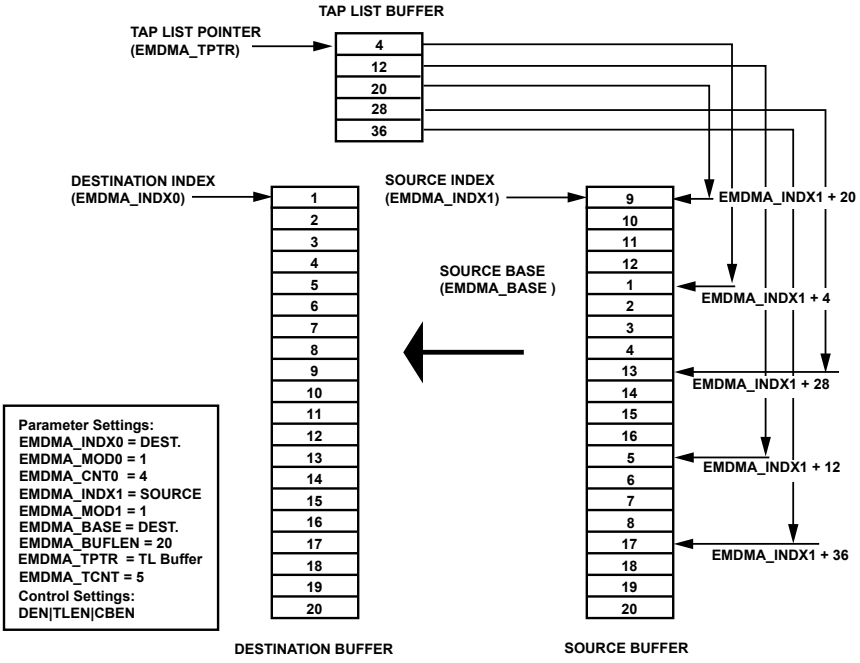


Figure 35-7: Circular Gather DMA (Reads)

For each 32-bit tap read, the Channel 1 read index is shown in the ***Read/Write Index Pre-Modify (Scatter/Gather DMA)*** table. Note that one tap list entry starts multiple reads.

Table 35-11: Read/Write Index Pre-Modify (Scatter/Gather DMA)

Pre-Modify Address Equation **EMDMA_INDX0** + **EMDMA_TPTR** [**EMDMA_TCNT**] + (**EMDMA_MOD1** × **EMDMA_CNT1**)	Result	
	Block Size	Tap
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × 0	N	0
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × 1		
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × 2		
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × N		
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × 0	N	1
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × 1		
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × 2		
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × N		
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × 0	N	M
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × 1		
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × 2		
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × N		

Pre Modified Read/Write Index

For scatter/gather DMA, the tap list modifiers are employed, and the number of taps is determined by the tap list count register (EMDMA_TCNT). The number of sequential reads (block size) from every tap is determined by the internal count register (EMDMA_CNT0), and is the same for every tap. The read/write pointer in external index register (EMDMA_INDX1) serves as the index address for these read/writes.

TL[N] is the first tap list entry in the memory as pointed to by the tap list pointer register (EMDMA_TPTR). The tap list entries are 27-bit signed integers. For each read/write block, the DMA state machine fetches the offset from the tap list. The offset is added to the EMDMA_INDX1 register value to get the start address of the next block. The Channel 1 addresses are circular buffered if circular buffering is enabled (see the ***Circular Buffering Scatter DMA (Writes)*** and ***Circular Buffering Gather DMA (Reads)*** figures in Scatter/Gather DMA).

Once the EMDMA_CNT0 register for the final tap decrements to zero (EMDMA_TCNT and EMDMA_CNT0 are zero), then the tap list DMA access is complete, and the DMA completion interrupt is generated (when chaining is enabled the interrupt depends on the EMDMA_CHNPTR.PCI bit setting).

The write back mode (EMDMA_CTL.WRBEN bit) is hardwired to zero for tap list based DMA (as the addressing is pre-modify, and therefore the EMDMA_INDX1 value coincides with the TCB value even at the end of the DMA).

Delay Line DMA

Delay line DMA is used to support reads and writes to delay line buffers with limited core interaction. In this sense, delay line DMA is a quantity of integrated writes followed by reads from channel 1 (a delay line DMA access). Delay line DMA is described in the following sections.

NOTE: Delay line DMA can only operate by using chained DMA mode (EMDMA_CTL.CHEN bit set). In order to use delay line DMA for a single DMA sequence, initialize the EMDMA_CHNPTR register to zero in the TCB.

NOTE: Delay line DMA can be used in any system memory.

The delay line DMA operation follows these steps:

1. Load the first half of TCB for writing (seven parameters).
2. DMA writes to the delay line buffer until IC = 0.
3. Update the EI index pointer if circular mode is enabled.
4. Load the second half of TCB for reading (six parameters).
5. DMA tap based reads from the delay line buffer until RC = 0.

   Jump to step 1.

Writes to delay line. The number of writes is determined by the EMDMA_CNT0 register. The data is fetched from the EMDMA_INDX0 register and the EMDMA_MOD0 register is used as the internal modifier. The EMDMA_INDX1 register serves as the external index and is incremented by the EMDMA_MOD1 register after each write. These writes are circular buffered if circular buffering is enabled. See the ***Write to Delay Line Buffer*** figure.

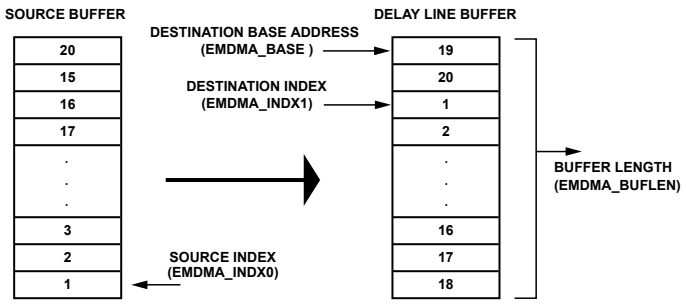


Figure 35-8: Write to Delay Line Buffer

When the writes are complete, (EMDMA_CNT0 =0) the EMDMA_INDX1 register, which serves as the write pointer of the delay line, is written back (EMDMA_CTL.WRBEN is hardwired to 1) to the TCB location from where it was fetched.

Reads from the delay line. For reads, the tap list (TL) modifiers are used, and the number of reads is determined by the EMDMA_CNT0 register. The write pointer in the EMDMA_INDX1 register serves as the index address for these reads (reads start from where writes end). The EMDMA_INDX1 register, along with tap list modifiers, are used in a pre-modify addressing mode to create the external address for the reads. Therefore, for each read, the DMA controller fetches the external modifier (EMDMA_TCNT register) from the tap list and the reads are circular buffered (if enabled). See the ***Read From Delay Line Buffer*** figure.

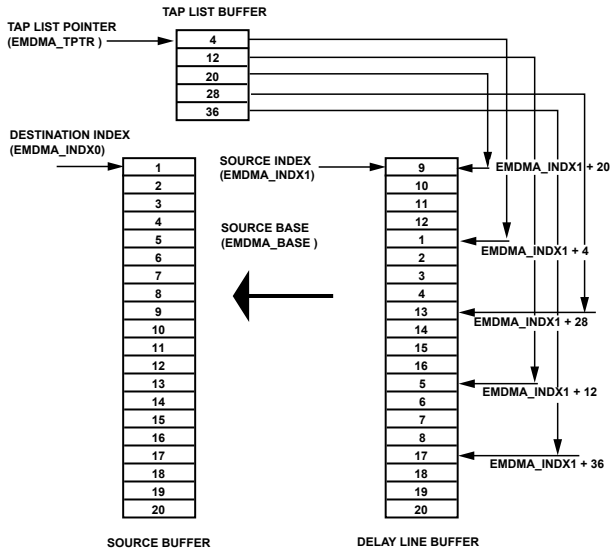


Figure 35-9: Read From Delay Line Buffer

Pre-Modified Read Index

Note that TL[N] is the first tap list entry in memory pointed to by the tap list pointer register (EMDMA_TPTR). Tap list entries are 27-bit signed integers. Therefore, for each read-block, the DMA state machine fetches the offset external modifier from the tap list. The reads are circular buffered if circular buffering is enabled.

NOTE: The channel 1 DMA address generation follows pre-modify addressing for reads in delay line DMA and therefore the EMDMA_INDX1 register values are not updated. Also the EMDMA_MOD1 register does not have any effect during these delay line reads. Once the read count completes, the EMDMA_CNT0 register decrements to zero (both EMDMA_CNT0 and EMDMA_TCNT are zero) for the final tap. Finally, the delay line DMA access completes, and the DMA completion interrupt is generated. When chaining is enabled, the interrupt is dependent on the EMDMA_CHNPTR.PCI bit setting. The delay line DMA can only be initialized using the TCB. To use the delay line DMA for a single DMA sequence, initialize the EMDMA_CHNPTR register to zero in the TCB.

For each 32-bit tap read, the channel 1 read index is shown in the ***Read/Write Index Pre-Modify (Scatter/Gather DMA)*** table. Note that one tap list entry starts multiple reads.

Table 35-12: Read/Write Index Pre-Modify (Scatter/Gather DMA)

Pre-Modify Address Equation	Result	
EMDMA_INDX1 + **EMDMA_TPTR**[**EMDMA_TCNT**] + (**EMDMA_MOD1** × **EMDMA_CNT1**)	Block Size	Tap
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × 0	N	0
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × 1		
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × 2		
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × N		

Table 35-12: Read/Write Index Pre-Modify (Scatter/Gather DMA) (Continued)

Pre-Modify Address Equation **EMDMA_INDX1** + **EMDMA_TPTR**[**EMDMA_TCNT**] + (**EMDMA_MOD1** × **EMDMA_CNT1**)	Result	
	Block Size	Tap
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × 0	N	1
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × 1		
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × 2		
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × N		
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × 0	N	M
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × 1		
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × 2		
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × N		

ADSP-2184x EMDMA Register Descriptions

Extended Memory DMA (EMDMA) contains the following registers.

Table 35-13: ADSP-2184x EMDMA Register List

Name	Description
EMDMA_BASE	External Base Address Register
EMDMA_BUFLEN	Circular Buffer Length Register
EMDMA_CHNPTR	Chain Pointer Register
EMDMA_CNT0	Internal Count Register
EMDMA_CNT1	External Count Register
EMDMA_CTL	External Memory DMA Control Register
EMDMA_INDX0	Internal Index Register
EMDMA_INDX1	External Index Register
EMDMA_MOD0	Internal Modifier Register
EMDMA_MOD1	External Modifier Register
EMDMA_TCNT	Delay Line Tap Count Register
EMDMA_TPTR	Tap List Pointer Register

External Base Address Register

The EMDMA_BASE register contains the external base address of the delay line buffer. This is used for maintaining circular buffered read/writes to the delay line.

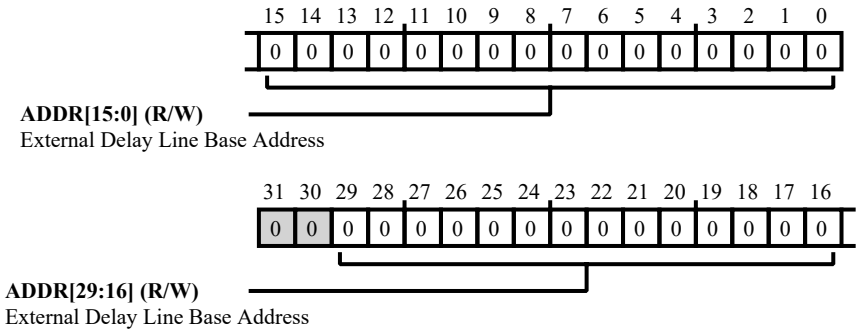


Figure 35-10: EMDMA_BASE Register Diagram

Table 35-14: EMDMA_BASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	ADDR	External Delay Line Base Address. The EMDMA_BASE.ADDR bit field contains the external base address of the delay line buffer.

Circular Buffer Length Register

The EMDMA_BUFLEN register holds the circular buffer length for the delay line DMA.

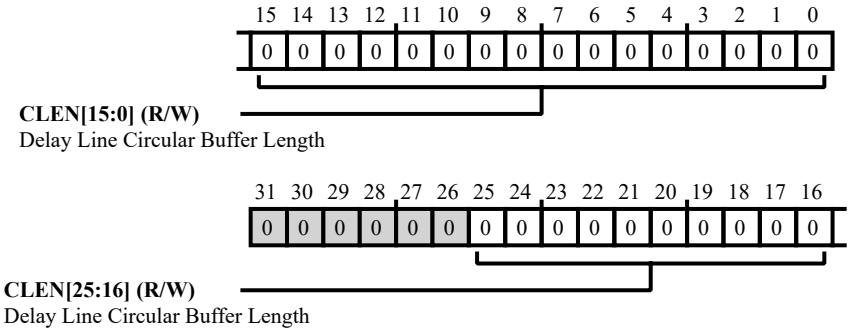


Figure 35-11: EMDMA_BUFLEN Register Diagram

Table 35-15: EMDMA_BUFLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:0 (R/W)	CLEN	Delay Line Circular Buffer Length. The EMDMA_BUFLEN.CLEN bit field holds the circular buffer length for the delay line DMA.

Chain Pointer Register

The EMDMA_CHNPTR register contains the address of the next descriptor in memory when the EMDMA_CTL.CHEN bit =1. This register also has bits to change DMA directions for each descriptor and the PCI bit.

Note that the lower 30-bits of this register are to be written with 30 MSB's of the word-aligned byte addresses of the corresponding next descriptor address.

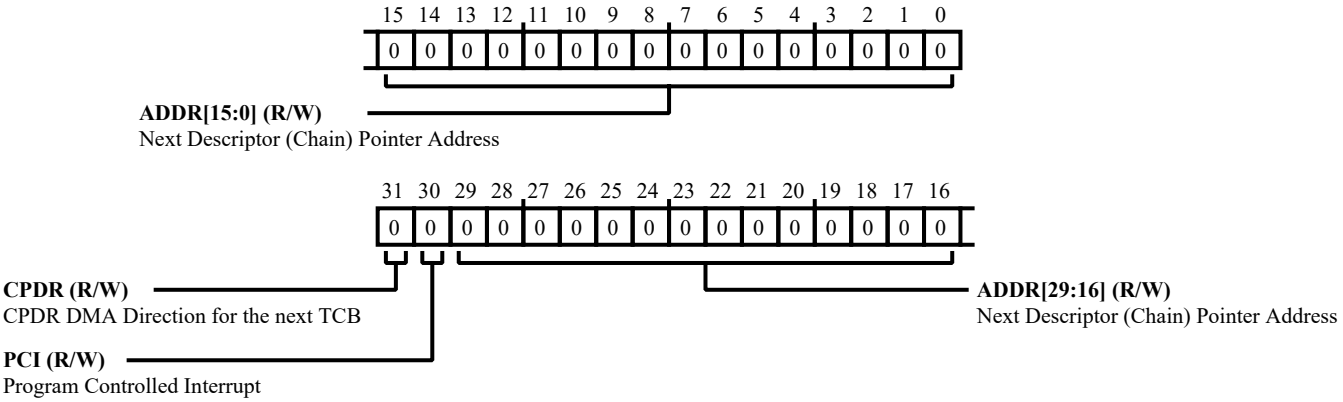


Figure 35-12: EMDMA_CHNPTR Register Diagram

Table 35-16: EMDMA_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CPDR	CPDR DMA Direction for the next TCB. The EMDMA_CHNPTR.CPDR bit configures whether the DMA is a write to internal memory or a read from internal memory. Note: this setting is applicable only if EMDMA_CTL.OFCEN =1 and is not applicable for delay line DMA.	
		0	Write to Channel 0 (Channel 1 reads)
		1	Read from Channel 0 (Channel 1 Writes)
30 (R/W)	PCI	Program Controlled Interrupt. The EMDMA_CHNPTR.PCI bit PCI sets whether an interrupt is generated after the current TCB or if no interrupt is generated. (Only affects DMA if chaining is enabled).	
		0	Enable DMA Channel interrupt to occur at the completion of the entire DMA chained transfer
		1	Enable DMA Channel interrupt to occur at the completion of current DMA sequence

Table 35-16: EMDMA_CHNPTR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	ADDR	Next Descriptor (Chain) Pointer Address. The `EMDMA_CHNPTR.ADDR` bit field provides the address of the next descriptor in memory.

Internal Count Register

The EMDMA_CNT0 register contains the number of words to be transferred for channel 0 DMA.

Note: If delay line DMA is enabled then the EMDMA_CNT0 register serves as the count register for the delay line writes.

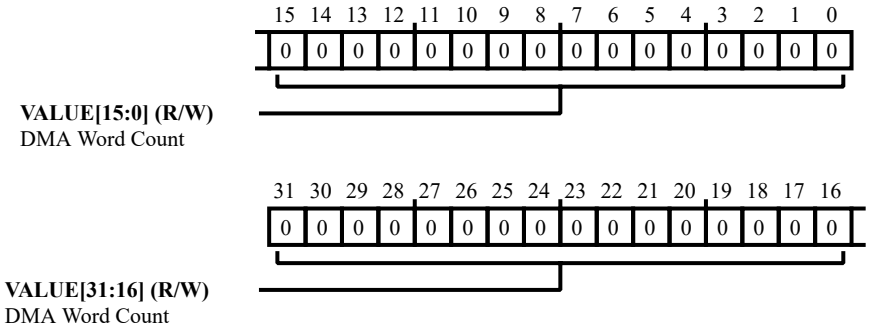


Figure 35-13: EMDMA_CNT0 Register Diagram

Table 35-17: EMDMA_CNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	DMA Word Count. The EMDMA_CNT0.VALUE bit field is the DMA word count.

External Count Register

The EMDMA_CNT1 register contains the number of words to be transferred for channel 1 DMA.

Note: If delay line DMA is enabled then the EMDMA_CNT1 register serves as the count register for the delay line writes.

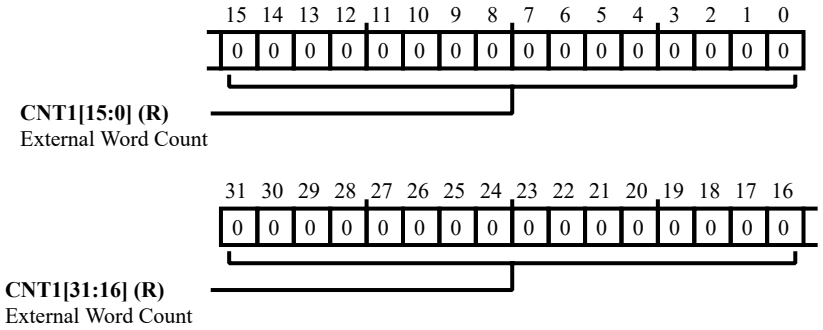


Figure 35-14: EMDMA_CNT1 Register Diagram

Table 35-18: EMDMA_CNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CNT1	External Word Count. The EMDMA_CNT1.CNT1 bit field is the external word count.

External Memory DMA Control Register

The EMDMA_CTL register contains bits that enable and configure EMDMA and indicate DMA transfer status.

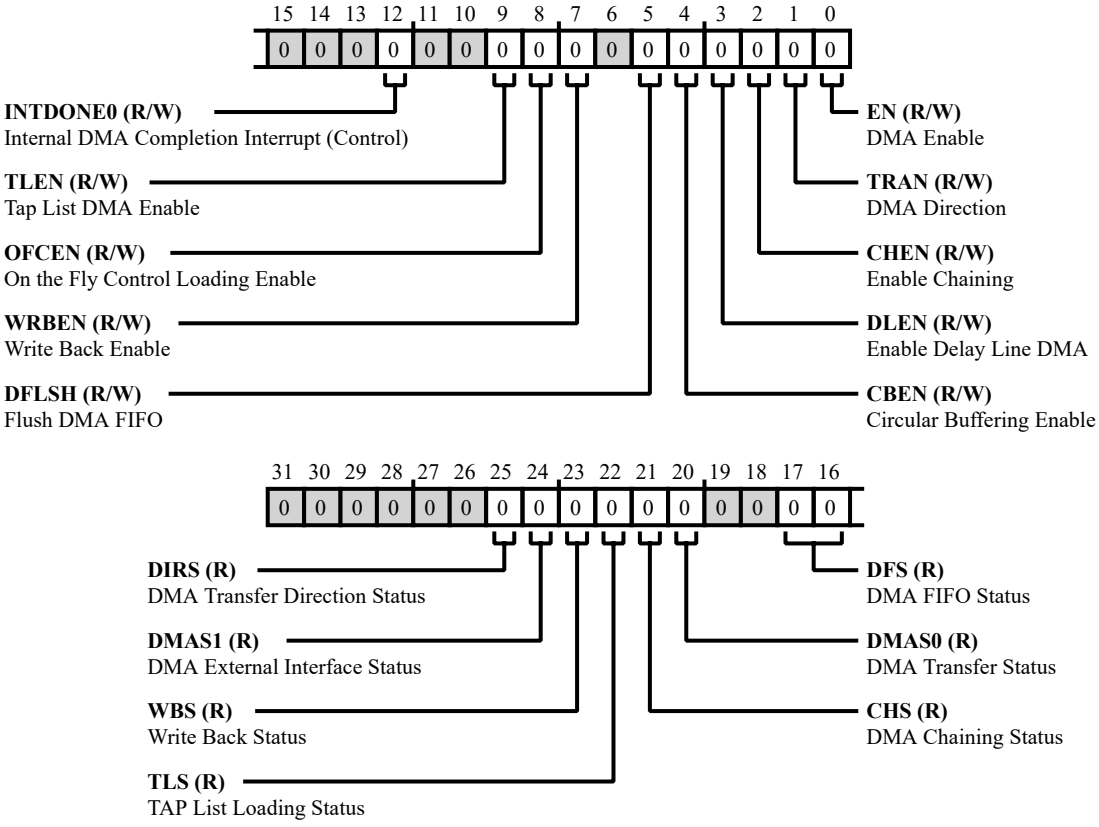


Figure 35-15: EMDMA_CTL Register Diagram

Table 35-19: EMDMA_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/NW)	DIRS	DMA Transfer Direction Status. The EMDMA_CTL.DIRS bit provides the DMA transfer status direction. This is useful for delay line DMA where the transfer direction changes with the state of the DMA state machine. For standard DMA, the EMDMA_CTL.DIRS bit reflects the state of the EMDMA_CTL.TRAN bit.	
		0	DMA direction is Channel 1 Reads
		1	DMA direction is Channel 1 Writes

Table 35-19: EMDMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	DMAS1	DMA External Interface Status. The EMDMA_CTL.DMAS1 bit provides the DMA channel 1 transfer status.	
		0	Channel 1 DMA does not have any access pending
		1	Channel 1 DMA access are pending
23 (R/NW)	WBS	Write Back Status. The EMDMA_CTL.WBS bit provides the delay line write pointer write back status.	
		0	Write pointer write back is not active
		1	Write pointer write back is active
22 (R/NW)	TLS	TAP List Loading Status. The EMDMA_CTL.TLS bit provides the DMA tap list loading status.	
		0	TAP list loading is not active
		1	TAP list loading is active
21 (R/NW)	CHS	DMA Chaining Status. The EMDMA_CTL.CHS bit provides the DMA chaining status.	
		0	DMA chain loading is not active
		1	DMA chain loading is active
20 (R/NW)	DMAS0	DMA Transfer Status. The EMDMA_CTL.DMAS0 bit provides the DMA channel 0 transfer status.	
		0	DMA idle
		1	DMA in progress
17:16 (R/NW)	DFS	DMA FIFO Status. The EMDMA_CTL.DFS bit field provides the DMA FIFO status.	
		0	FIFO EMPTY
		1	FIFO Partially Full
		2	Reserved
		3	FIFO Full

Table 35-19: EMDMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	INTDONE0	Internal DMA Completion Interrupt (Control). The EMDMA_CTL.INTDONE0 bit configures when the DMA complete interrupt is generated. The EMDMA_CTL.INTDONE0 =1 setting is provided for backward compatibility with older SHARC processors.	
		0	Interrupt on access completion (Channel 0 or Channel 1 DMA completion)
		1	Interrupt on Channel 0 DMA completion.
9 (R/W)	TLEN	Tap List DMA Enable. The EMDMA_CTL.TLEN bit enables scatter/gather tap list DMA.	
		0	Disables the tap list based scatter/gather DMA
		1	Enables the tap list based scatter/gather DMA
8 (R/W)	OFCEN	On the Fly Control Loading Enable. The control bits in EMDMA_CHNPTR register are used to describe the next TCB behavior if the EMDMA_CTL.OFCEN bit is set. Therefore, the DMA controls can be changed from TCB to TCB. 0 = disables the control bits in the EMDMA_CHNPTR register. 1 = enables the control bits in the EMDMA_CHNPTR register. If chaining is enabled with EMDMA_CTL.OFCEN bit set, the EMDMA_CTL.TRAN bit has no effect, and direction is determined by the EMDMA_CHNPTR.CPDR bit.	
7 (R/W)	WRBEN	Write Back Enable. The EMDMA_CTL.WRBEN bit enables write back of the EIEP register after reads and or writes. Write back is automatically enabled for delay line DMA. WRBEN is applicable only if chaining is enabled (EMDMA_CTL.CHEN =1).	
5 (R/W)	DFLSH	Flush DMA FIFO. The EMDMA_CTL.DFLSH bit flushes the DMA FIFO. The buffer is only flushed when this bit is set. It can be set with the enable bit. Setting this bit also clears the EMDMA_CTL.DFS bit.	
4 (R/W)	CBEN	Circular Buffering Enable. The EMDMA_CTL.CBEN bit enables circular buffering. Circular buffering can be used with normal DMA as well. If circular buffering is enabled with chaining for normal DMA, then ELEP and EBEP should be part of the TCB.	
		0	Disables circular buffering with delay line DMA
		1	Enables circular buffering with delay line DMA

Table 35-19: EMDMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DLEN	Enable Delay Line DMA. The EMDMA_CTL.DLEN bit enables delay line DMA. This bit is applicable only when the EMDMA_CTL.CHEN bit =1.	
		0	Delay-line DMA disabled
		1	Delay-line DMA enabled
2 (R/W)	CHEN	Enable Chaining. The EMDMA_CTL.CHEN bit enables DMA chaining.	
		0	Chaining disabled
		1	Chaining enabled
1 (R/W)	TRAN	DMA Direction. The EMDMA_CTL.TRAN bit determines the DMA data direction. Note: If the delay line DMA is enabled, this bit does not have any effect. For delay line DMA, transfer direction depends on the state of delay line transfers. For internal-internal or external-external DMA, this bit must be set.	
		0	Write through Channel 0 (Channel 1 reads)
		1	Read from Channel 0(Channel 1 writes)
0 (R/W)	EN	DMA Enable. The EMDMA_CTL.EN bit enables DMA.	
		0	Disable DMA
		1	Enable DMA

Internal Index Register

The EMDMA_INDX0 register contains the start address of the buffer for channel 0 DMA.

Note: For delay line DMA the EMDMA_INDX0 register serves as the delay line write index which is the start address of the channel 0 DMA buffer for the channel 1 write data.

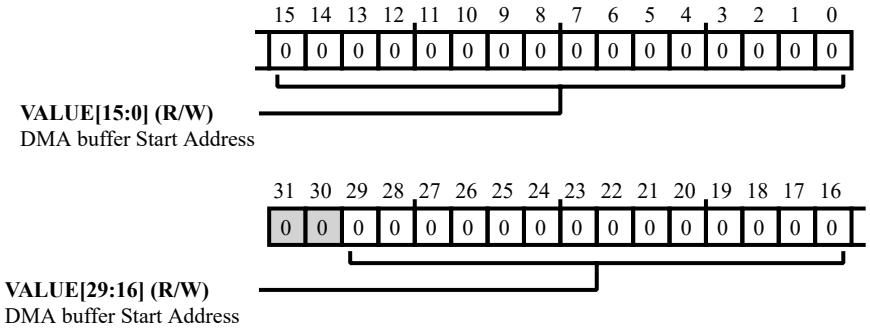


Figure 35-16: EMDMA_INDX0 Register Diagram

Table 35-20: EMDMA_INDX0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	DMA buffer Start Address. The EMDMA_INDX0.VALUE bit field is written with the 30 MSBs of the word -aligned byte addresses.

External Index Register

The EMDMA_INDX1 register contains the start address of the buffer for channel 1 DMA.

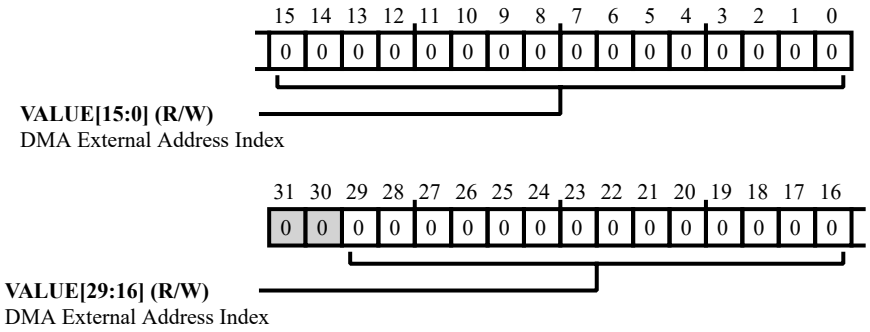


Figure 35-17: EMDMA_INDX1 Register Diagram

Table 35-21: EMDMA_INDX1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	DMA External Address Index. The EMDMA_INDX1.VALUE bit field is the DMA external address index.

Internal Modifier Register

The EMDMA_MOD0 register contains the channel 0 DMA address modifier.

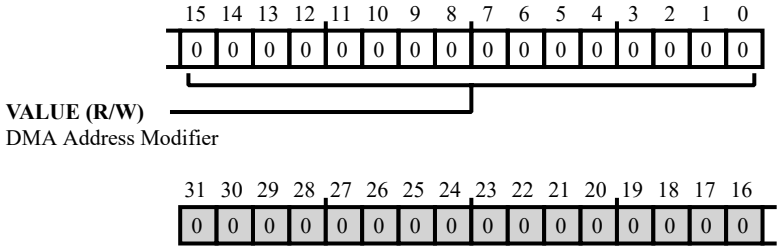


Figure 35-18: EMDMA_MOD0 Register Diagram

Table 35-22: EMDMA_MOD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	DMA Address Modifier. The EMDMA_MOD0.VALUE bit field is the DMA address modifier.

External Modifier Register

The EMDMA_MOD1 register contains the external (channel 1 DMA) address modifier.

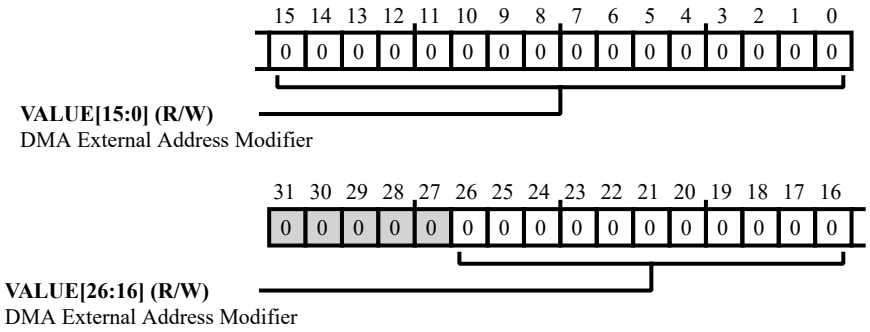


Figure 35-19: EMDMA_MOD1 Register Diagram

Table 35-23: EMDMA_MOD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
26:0 (R/W)	VALUE	DMA External Address Modifier. The EMDMA_MOD1.VALUE bit field is the DMA external address modifier.

Delay Line Tap Count Register

The EMDMA_TCNT register is the tap count register for delay line DMA. This register holds the length of the tap list (the number of taps). The total number of words read from the delay line is equal to the EMDMA_TCNT (tap count) multiplied by the EMDMA_CNT1 (read block size).

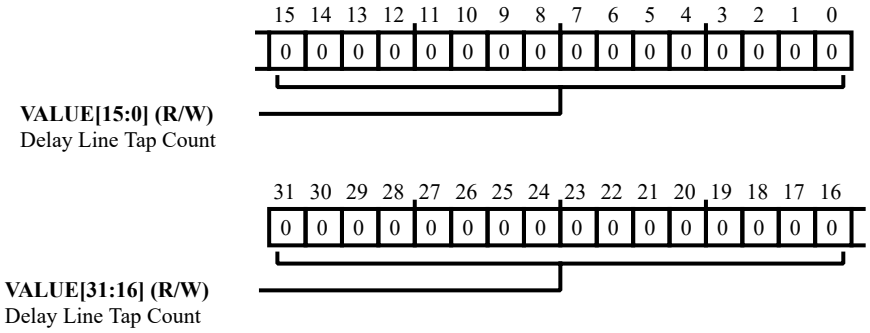


Figure 35-20: EMDMA_TCNT Register Diagram

Table 35-24: EMDMA_TCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Delay Line Tap Count. The EMDMA_TCNT.VALUE bit field is the tap count register for delay line DMA.

Tap List Pointer Register

The EMDMA_TPTR register holds the address of an array in memory which holds offsets to be used when accessing a delay line in system memory. The offset represents the first address of each read block.

Note: the lower 30-bits of this register are to be written with 30 MSBs of the word-aligned byte address of the array.

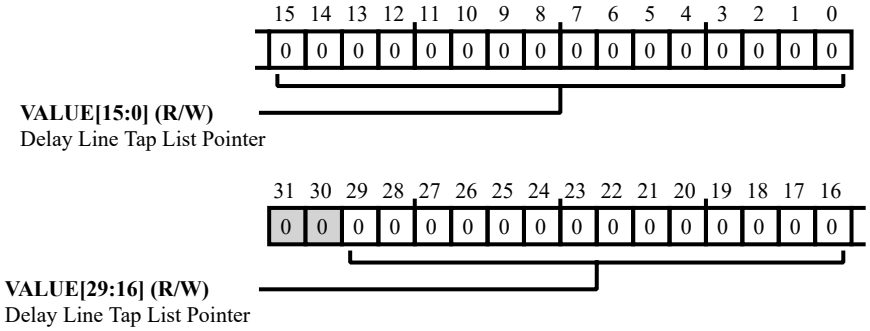


Figure 35-21: EMDMA_TPTR Register Diagram

Table 35-25: EMDMA_TPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Delay Line Tap List Pointer. The EMDMA_TPTR.VALUE bit field contains the offsets to be used when accessing a delay line in system memory.

36 Cyclic Redundancy Check (CRC)

The CRC peripheral performs the cyclic redundancy check (CRC) of the block of data that is presented to the peripheral. The peripheral provides a means to verify periodically the integrity of the system memory, the contents of memory-mapped registers (MMRs), or communication message objects. It is based on a CRC32 engine that computes the signature of 32-bit data presented to the peripheral.

The dedicated hardware compares the calculated signature of the operation to a pre-loaded expected signature. If the two signatures fail to match, the peripheral generates an error.

The source channel of the memory-to-memory DMA channels can provide data. The CRC optionally forwards data to memory through the destination DMA channel. Alternatively, the peripheral supports data presented by any qualified controller of the CRC peripheral bus.

The CRC peripheral implements a reduced table-look-up algorithm to compute the signature of the data. The CRC uses a programmable 32-bit CRC polynomial to generate the look-up table (LUT) contents automatically.

More CRC peripheral modes allow for initializing large memory sections with a constant value, or for verifying that sections of memory are equal to a constant value.

NOTE: CRC is supported by MDMA0, MDMA1, MDMA4, and MDMA5 channels only.

CRC Features

The CRC peripheral supports a number of key features.

- Memory scan modes for memory verification
- Memory transfer modes for on-the-fly CRC calculations while transferring data from one memory to another
- A programmable 32-bit CRC polynomial with automatic LUT generation
- Data mirroring options

The CRC module also includes the following features.

- CRC checksum computation and comparison modes
- 32-bit programmable CRC polynomial with bit reverse option
- Automatic look-up table (LUT) generation

- Data mirroring options for endian and reflected polynomial cases
- Automatic clear and preset of results
- Fault and error interrupt reporting
- DMA and MMR based operation

Because the CRC module is closely tied to memory-to-memory DMA (MDMA) channel pairs, the use cases include the following features.

- Memory scan mode with CRC compute or compare
- Memory transfer mode with CRC compute or compare
- Memory fill operation with 32-bit data patterns
- Memory verify operation
- MMR write access to FIFO of destination DMA
- MMR read access to FIFO of source DMA
- Profiting from advanced DMA features, like descriptor mode and bandwidth control or monitor

CRC Functional Description

The CRC peripheral supports a number of modes of operation that allow for the initialization and verification of regions of memory. The peripheral supports efficient memory-fill and verification operations on regions of memory with or against a constant value. These modes of operation do not require the CRC engine to calculate a signature. Other modes of operation allow for the calculation of CRC signature and verification for a memory region. The modes allow for on-the-fly CRC calculation when performing memory-to-memory DMA transfers from one memory region to another.

To minimize the need for core accesses, the peripheral interfaces with one or more (depending on processor features) memory-to memory DMA (MDMA) channels. This connectivity permits flexible configuration, in which data can be written-to or read-from the peripheral using DMA transactions, core transactions, or a combination of both.

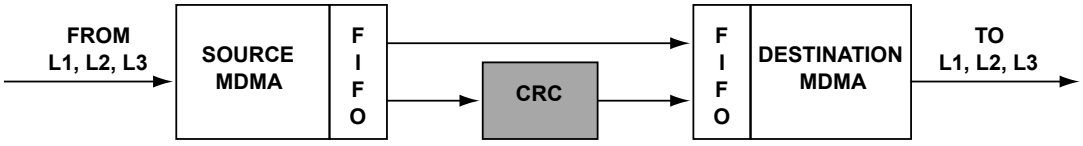


Figure 36-1: Memory Flow

ADSP-2184x CRC Register List

The Cyclic Redundancy Check (CRC) unit includes the data comparison, polynomial operation, and look up table generation features needed for CRC operation. The CRC provides CRC protection as specified by many functional safety requirements. This unit facilitates the system software's ability to periodically check the correctness

of the code/data available in memory. A set of registers govern CRC operations. For more information on CRC functionality, see the CRC register descriptions.

Table 36-1: ADSP-2184x CRC Register List

Name	Description
CRC_COMP	Data Compare Register
CRC_CTL	Control Register
CRC_DCNT	Data Word Count Register
CRC_DCNTCAP	Data Count Capture Register
CRC_DCNTRLD	Data Word Count Reload Register
CRC_DFIFO	Data FIFO Register
CRC_FILLVAL	Fill Value Register
CRC_INEN	Interrupt Enable Register
CRC_INEN_CLR	Interrupt Enable Clear Register
CRC_INEN_SET	Interrupt Enable Set Register
CRC_POLY	Polynomial Register
CRC_RESULT_CUR	CRC Current Result Register
CRC_RESULT_FIN	CRC Final Result Register
CRC_STAT	Status Register

ADSP-2184x CRC Interrupt List

Table 36-2: ADSP-2184x CRC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
34	CRC0_ERR	CRC0 Error	Level	
35	CRC1_ERR	CRC1 Error	Level	
40	CRC0_DCNTEXP	CRC0 Datacount Expiration	Level	
41	CRC1_DCNTEXP	CRC1 Datacount Expiration	Level	
42	CRC2_ERR	CRC2 Error	Level	
43	CRC3_ERR	CRC3 Error	Level	
48	CRC2_DCNTEXP	CRC2 Datacount Expiration	Level	
49	CRC3_DCNTEXP	CRC3 Datacount Expiration	Level	

CRC Definitions

To make the best use of the CRC, it is useful to understand the following terms.

CRC

Acronym for Cyclic Redundancy Check. An error detection code that can detect changes within a block of data.

CRC Polynomial

The 32-bit polynomial used by the CRC engine to generate the look-up table required for the CRC implementation

LUT

Acronym for the Look-up Table. The look-up table is automatically generated from the supplied 32-bit CRC polynomial.

DMA

Acronym for Direct Memory Access. Used to describe a data transfer that takes place through a DMA channel allowing data distribution around a system without intervention from the core.

MDMA

Acronym for Memory-To-Memory DMA transfer that often requires the use of two DMA channels to transfer data from one memory region to another memory region. One DMA channel is configured as a source channel and the second as a destination channel.

CRC Block Diagram

The ***CRC Block Diagram*** shows the functional block diagram of the CRC. The following sections describe the blocks.

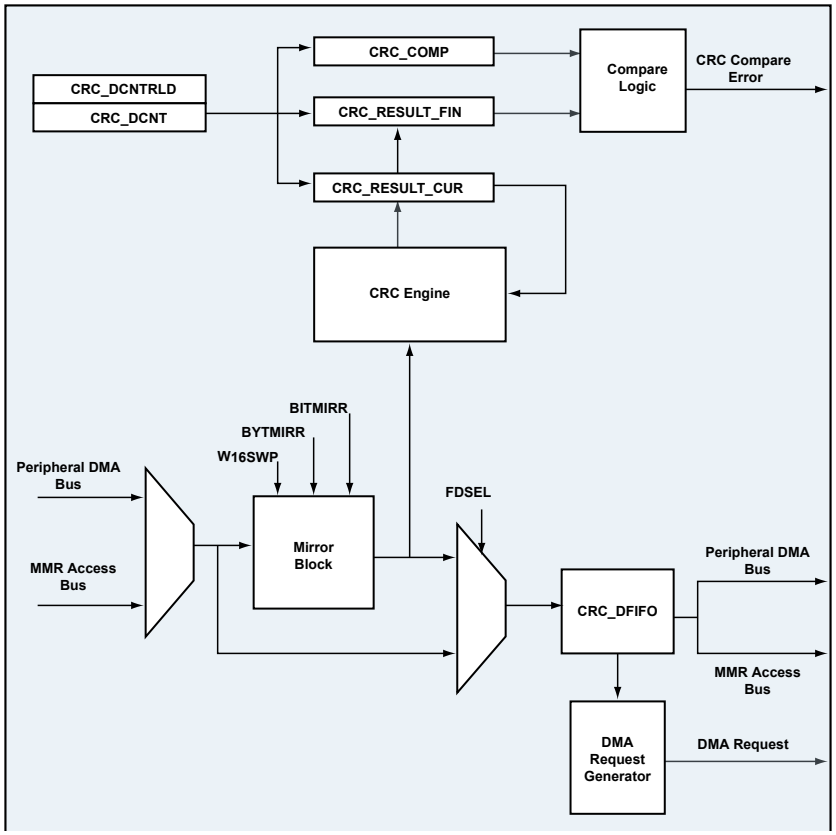


Figure 36-2: CRC Block Diagram

Peripheral DMA Bus

The CRC peripheral provides both an incoming and outgoing datapath to the peripheral DMA bus. The MDMA source channel is interfaced to the incoming datapath providing data to the CRC peripheral. For memory transfer and data fill modes, the CRC uses the MDMA destination channel to either output the data from the CRC FIFO or use the data for the fill operation.

MMR Access Bus

The core uses the MMR access bus to access all the memory-mapped registers of the peripheral for configuration, status, and debug purposes. The core can also use the MMR access bus to feed data to the CRC peripheral or read data from the FIFO of the CRC peripheral. The CRC operation is an alternative to the DMA channel operation to read data from the FIFO.

Data received by MMR writes can transfer to destination DMA. Similarly, data received by source DMA can be output through the MMR interface. Optionally, intermediate results can be made available to the MMR interface.

Mirror Block

The mirror block individually controls bit-reversing of the polynomial, the computation results, and the expected result. Bit mirroring, byte mirroring, word swapping, and any combination of these operations can control endian and the reflection of processed data.

Data FIFO

The CRC data FIFO is a 32-bit-wide 4-entry FIFO. The FIFO is accessible to both the peripheral DMA bus and the MMR access bus. The FIFO status is accessible from the CRC_STAT register.

DMA Request Generator

The DMA request generator is responsible for granting incoming DMA requests from the source DMA channel and issuing outgoing DMA requests to the destination DMA channel.

CRC Engine

The CRC engine is a 32-bit CRC engine that implements the reduced table look-up scheme. The CRC engine provides support for a user-programmable 32-bit polynomial that the CRC uses to load the look-up table parameters required for the CRC calculation. The CRC engine is a single cycle implementation operating on 32 bits of data per cycle.

Compare Logic

The compare logic takes the final CRC signature and compares it to the expected CRC signature, generating a CRC compare error when the signatures do not match. A compare error can flag a system fault.

CRC Architectural Concepts

A 32-bit polynomial is required before calculation of the CRC signature can occur. The CRC uses the polynomial to generate the contents of an internal look-up table that the reduced table look-up implementation requires. The look-up table is automatically generated when the polynomial is written. It must be initialized prior to any operation that requires the use of the CRC engine.

The mirror block logic can manipulate the data presented to the CRC engine before the CRC uses the data in the calculation of the CRC signature. The data mirror operation is configurable to allow for bit reversing, byte reversing, and 16-bit word swapping operations on the incoming data. For memory transfer compute-and-compare operations, programs can configure the peripheral to output the data in the same form in which it was received. Or, the operation can output the mirrored data in the same manner that it is presented to the CRC engine.

While the CRC peripheral is in operation, the status of the FIFO is continually updated and reflected in the CRC_STAT register. The FIFO status is required for core-based accesses to the CRC peripheral. The status indicates when:

- The CRC peripheral can receive data

- Data is available for reading from the FIFO
- The result of the CRC_RESULT_CUR register has been updated

The status of the CRC_RESULT_CUR register indicates that the current CRC calculation has completed and the result is available.

Look-up Table

The look-up table consists of a set of sixteen 32-bit registers that hardware populates automatically when a write access takes place to the CRC_POLY register. 16 clock cycles are required to generate all 16 look-up table entries. The status of the process for generating the look-up table is reflected in CRC_STAT.LUTDONE allowing for software to poll on the completion of the event or for generation of an interrupt.

NOTE: Hardware must populate the look-up table before any operation using the CRC peripheral can take place, even if the operation does not use the CRC engine. The peripheral does not issue any data requests until the table generation process is complete. In addition, the CRC_STAT.IBR field, that indicates the input buffer status as required for core-based transfers, is only valid upon completion of the process for generating the look-up table.

Data Mirroring

The data mirror block can be configured to manipulate the incoming data before the data passes to the CRC engine and, optionally, to the FIFO. This configuration allows the peripheral to handle various forms of endianness and to function with reflected polynomials.

There are three configuration bits that control the data mirroring process: CRC_CTL.BITMIRR, CRC_CTL.BYTMIRR, and CRC_CTL.W16SWP. The ***Data Mirroring Options*** table details how these options affect the incoming data and the output generated by the mirror block.

Table 36-3: Data Mirroring Options

W16SWP	BYTMIRR	BITMIRR	Output Data
0	0	0	Dout[31:0] = Din[31:0]
0	0	1	Dout[31:0] = Din[24:31],Din[16:23],Din[8:15],Din[0:7]
0	1	0	Dout[31:0] = Din[7:0],Din[15:8],Din[23:16],Din[31:24]
0	1	1	Dout[31:0] = Din[0:7],Din[8:15],Din[16:23],Din[24:31]
1	0	0	Dout[31:0] = Din[15:0], D[31:16]
1	0	1	Dout[31:0] = Din[8:15],Din[0:7], Din[24:31],Din[16:23]
1	1	0	Dout[31:0] = Din[23:16],Din[31:24], Din[7:0],Din[15:8]
1	1	1	Dout[31:0] = Din[16:23],Din[24:31], Din[0:7],Din[8:15]

When the CRC is configured to operate in the memory transfer compute-and-compare mode, the bit-reversed output data can be written to the FIFO. This feature is controlled through the CRC_CTL.FDSEL field.

In addition to providing bit swapping and mirror options to the incoming data, the CRC peripheral also supports bit mirroring on the following registers.

- CRC_RESULT_CUR and CRC_RESULT_FIN, controlled through the CRC_CTL.RSLTMIRR field. When mirroring is enabled, the values to be written to these registers are fully bit-reversed before the write operation occurs.
- CRC_POLY, controlled through the CRC_CTL.POLYMIRR field. When mirroring is enabled, the 32-bit polynomial is fully bit-reversed before the write operation to the register occurs.
- CRC_COMP, controlled through the CRC_CTL.CMPMIRR field. When mirroring is enabled, the contents to be loaded to this register are fully bit-reversed before the write operation occurs.

FIFO Status and Data Requests

The CRC peripheral provides indication of the input and output buffer status through CRC_STAT.IBR and CRC_STAT.OBR respectively. For core-based operations, software must monitor these status fields prior to writing to or reading from the CRC FIFO. No write to the CRC FIFO can occur while CRC_STAT.IBR indicates that the buffer is not ready to accept data. Similarly, the CRC FIFO cannot be read until CRC_STAT.OBR indicates that data is available.

The memory scan modes of operation only require the monitoring of the input buffer status. The memory transfer, compute-and-compare mode uses both input and output buffer status. If the current result of the CRC computation is required, then software must verify that the current operation has completed and that the intermediate result is ready. The CRC_STAT.IRR indicates the status.

NOTE: The memory transfer fill mode of operation requires the use of a DMA channel. The CRC does not support core reads from the CRC FIFO for this mode of operation.

Memory transfer, compute-and-compare mode uses burst transactions to make the most efficient use of the available resources. In this mode, when the FIFO is initially empty and the peripheral is enabled, the CRC_STAT.IBR bit indicates that the CRC is ready to accept data. The peripheral generates data requests to the source DMA channel (if the CRC uses DMA). While the number of words remaining in the CRC_DCNT register is greater than the FIFO depth, the peripheral issues data requests or accepts incoming data in bursts. The peripheral continues until the CRC FIFO becomes full.

Once full, the CRC_STAT.IBR and CRC_STAT.OBR bits are updated, and then the CRC issues outgoing data requests. Only when the FIFO is empty can the peripheral accept further incoming data, and the CRC_STAT.IBR and CRC_STAT.OBR bits are updated once again.

Once CRC_DCNT is decremented such that the number of words waiting for processing is less than the number required to fill the FIFO, the burst mode of operation is disabled. Incoming data is accepted when the FIFO is not full. Outgoing data is available when the FIFO is not empty. Therefore, there are no restrictions requiring the word count to be a multiple of the FIFO depth.

All other CRC modes of operation indicate that incoming data can be accepted when the FIFO is not full. Outgoing data is available when the FIFO is not empty.

The CRC_CTL.OBRSTALL and CRC_CTL.IRRSTALL bit configurations also influence how the CRC generates data requests and status bits. The following list describes the bits.

- The CRC_CTL.OBRSTALL bit can be configured such that the CRC peripheral stalls as soon as there is output data available in the FIFO. Use this mode of operation only in memory transfer, compute-and-compare mode. This mode results in the processing of one 32-bit word at a time. The peripheral does not request or accept incoming data until the current value being processed is read from the peripheral.
- The CRC_CTL.IRRSTALL bit can be configured so that the CRC peripheral stalls all further incoming data requests until the CRC_RESULT_CUR register is read after being updated. Use this mode of operation for CRC signature generation. It is not applicable to memory transfer data-fill mode or memory scan data-verify mode of operation.

CRC Operating Modes

The following sections describe the various operating modes of the CRC interface.

Data Transfer Modes

The CRC peripheral supports two main categories of operation involving data transfers:

- Memory scan mode
- Memory transfer mode

Memory scan modes are read-only operations that allow the contents of memory to be read into the peripheral and verified for correctness. There are two forms of memory scan mode:

- CRC compute-and-compare performs a CRC calculation on data presented to the peripheral and compares the CRC result with a pre-determined and pre-loaded result. An error is generated when the results differ.
- Data verify compares each 32-bit data word presented to the CRC peripheral to a pre-loaded 32-bit value and generates an error when the data differs.

Both modes of operation require at the most, a single DMA channel to read the data from memory into the peripheral. No data is forwarded to the data output or destination DMA. The CRC can also use core-driven transfers for either of these modes of operation.

The memory transfer modes involve memory write or memory read-and-write operations allowing for memory to be initialized or transferred from one region of memory to another. There are two forms of memory transfer mode:

- CRC compute-and-compare performs a full data transfer from one memory region to another memory region. The CRC generates a signature on the data presented and compares it with a pre-determined and pre-loaded result. An error is generated when the results differ.
- Data fill initializes a region of memory with a pre-loaded 32-bit constant value.

The CRC compute-and-compare mode of operation requires both incoming and outgoing data channels. The operation occurs either using DMA channels, using core-driven write or read operations to and from the FIFO

or using a combination of both. The data fill mode of operation requires only a memory write DMA destination channel—this mode does not support core driven operations.

Memory Scan Compute-and-Compare Mode

In this mode of operation, the CRC engine of the peripheral is enabled. The mode is configured through the CRC_CTL.OPMODE field and the CRC engine performs a 32-bit CRC operation on the incoming data stream.

The length of the data stream is configured through the CRC_DCNT register. The accumulated result of the CRC operation is contained in the CRC_RESULT_CUR register. As the CRC engine processes each 32-bit word, the CRC_DCNT register is decremented and CRC_RESULT_CUR is updated.

Once CRC_DCNT decrements to zero, the contents of the CRC_RESULT_CUR register are copied to CRC_RESULT_FIN and CRC_STAT.DCNTEXP is updated accordingly. The CRC uses the CRC_COMP register to store the expected result of the operation. After the CRC calculation, CRC_COMP is compared with CRC_RESULT_FIN and CRC_STAT.CMPERR is updated to reflect the status of the compare operation. CRC_STAT.CMPERR must be cleared before the next CRC operation is performed.

The CRC peripheral also contains the CRC_DCNTRLD register. The CRC uses this register to reload CRC_DCNT upon completion of the CRC operation in preparation for the next transfer.

The initial seed of the CRC computation can be configured through CRC_CTL.AUTOCLRZ and CRC_CTL.AUTOCLRF. This configuration provides a way to reset CRC_RESULT_CUR to 0x00000000, 0xFFFFFFFF or to leave the current register contents untouched for the next operation.

The peripheral can be configured to allow for the compare error and data expiration events to generate an interrupt.

Memory Scan Data Verify

In this mode of operation, the CRC engine of the peripheral is not required. The mode is enabled through the CRC_CTL.OPMODE field. Each 32-bit word of the data stream is compared with a constant value that is stored in the CRC_COMP register. The CRC_DCNT register contains the number of words for comparison. The CRC_DCNT register is decremented upon receiving a new 32-bit word from the data stream. If the compare operation fails, the CRC_STAT.CMPERR bit is updated and the contents of CRC_DCNT are captured in the CRC_DCNTCAP register. This information can be used to identify the location in the data stream where the error occurred. Clear the CRC_STAT.CMPERR field to re-enable capturing of further errors.

Once CRC_DCNT decrements to zero, CRC_STAT.DCNTEXP is updated accordingly to signal the end of the operation. The peripheral can be configured to allow for the compare error and data expiration events to generate an interrupt.

Memory Transfer Compute-and-Compare Mode

In this mode of operation, the CRC engine of the peripheral is enabled. The mode is configured through the CRC_CTL.OPMODE field and the CRC engine performs a 32-bit CRC operation on the incoming data stream.

The length of the data stream is configured through the CRC_DCNT register. The accumulated result of the CRC operation is contained in the CRC_RESULT_CUR register. As the CRC engine processes each 32-bit word, the CRC_DCNT register is decremented and CRC_RESULT_CUR is updated.

Once CRC_DCNT decrements to zero, the contents of the CRC_RESULT_CUR register are copied to CRC_RESULT_FIN and CRC_STAT.DCNTEXP is updated accordingly. The CRC uses the CRC_COMP register to store the expected result of the operation. Upon completion of the CRC calculation, CRC_COMP is compared with CRC_RESULT_FIN and CRC_STAT.CMPERR is updated to reflect the status of the compare operation. Clear CRC_STAT.CMPERR before the next CRC operation is performed.

The CRC peripheral also contains CRC_DCNTRLD register. The CRC uses this register to reload CRC_DCNT upon completion of the CRC operation in preparation for the next transfer.

The initial seed of the CRC computation can be configured through CRC_CTL.AUTOCLRZ and CRC_CTL.AUTOCLRF. This configuration provides a means to reset CRC_RESULT_CUR to 0x00000000, 0xFFFFFFFF or to leave the current register contents untouched for the next operation.

The peripheral can be configured to allow for the compare error and data expiration events to generate an interrupt.

Memory Transfer Data Fill Mode

In this mode of operation, the CRC engine of the peripheral is not required. The mode is enabled through the CRC_CTL.OPMODE field. The CRC_FILLVAL register is written with a 32-bit value. The CRC uses this value to initialize a block memory through the memory-to-memory DMA destination channel. When the CRC peripheral and the DMA destination channel are enabled, the contents of the CRC_FILLVAL register is written to the DMA channel to initialize the memory region. The CRC_DCNT register contains the number of words for the write operation.

Once CRC_DCNT decrements to zero, CRC_STAT.DCNTEXP is updated accordingly to signal the end of the operation. The peripheral can be configured to allow for the data expiration event to generate an interrupt.

CRC Event Control

The CRC peripheral can enable certain CRC status operations to generate an interrupt event to the system event controller. There, a CRC error can be qualified as a system fault.

Interrupt Signals

The CRC peripheral can generate two interrupt requests that are enabled as events to SEC1 for fault operations. One is a CRC status interrupt and the other is a CRC error interrupt.

The CRC_STAT.CMPERR status bit can be configured as an interrupt and is signaled through the CRC error interrupt signal. The CRC_STAT.CMPERR status field is set whenever the CRC peripheral performs a compare operation that fails. This status can be the result of a failed memory scan data-verify operation that compares the contents of a memory range with a constant 32-bit value. Or, it can be the result of a CRC signature calculated for a memory region that does not match the expected pre-programmed result for a memory-compare operation.

The CRC_STAT.DCNTEXP status bit is set when the CRC_DCNT register has decremented to zero. The status indicates that the CRC peripheral has now processed all the data requested for the current CRC operation. The CRC can also use this signal to generate an interrupt. The interrupt is signaled on the CRC status interrupt signal.

Both these status bits can be configured to generate and interrupt through the CRC_INEN register. The CRC_INEN register also has bit set, CRC_INEN_SET, and bit clear CRC_INEN_CLR equivalent registers that the CRC uses for the enabling and disabling of these interrupt sources.

The CRC_STAT register has two write one to clear (W1C) fields for clearing the two interrupt sources.

NOTE: Disabling the CRC peripheral through the CRC_CTL.BLKEN bit does not result in the clearing of interrupt sources. Clear the interrupt sources using a W1C operation to the CRC_STAT register.

CRC Programming Model

It is important to note the following restrictions when using the CRC peripheral with the DMA channels:

1. When enabling the CRC peripheral and the DMA channels, enable the CRC peripheral prior to enabling the DMA channels.
2. When disabling the CRC peripheral and the DMA channels, disable the DMA channels prior to disabling the CRC peripheral.

CRC Mode Configuration

Describes a number of tasks showing the various operation modes of the CRC peripheral.

- Look-up Table Generation
- Core Driven Memory Scan Compute-and-Compare Mode
- DMA Driven Memory Scan Compute-and-Compare Mode
- Core Driven Memory Scan Data Verify Mode
- DMA Driven Memory Scan Data Verify Mode
- Core Driven Memory Transfer Compute-and-Compare Mode
- DMA Driven Memory Transfer Compute-and-Compare Mode
- DMA Driven Memory Transfer Data Fill Mode

Look-up Table Generation

Describes the steps required to initialize the CRC peripheral LUT.

1. Write the 32-bit CRC polynomial of choice to the CRC_POLY register.

   ADDITIONAL INFORMATION: This operation results in the CRC peripheral starting the LUT initialization process. The CRC_STAT.LUTDONE bit is updated to reflect the operation is in progress.

2. Poll the CRC_STAT.LUTDONE bit until the status bit indicates that the operation is completed.

The CRC peripheral has completed initialization of all the LUT registers and is now ready for data operations. The CRC_STAT.LUTDONE bit remains in the current state until the CRC_POLY register is written again, or the peripheral or processor are reset.

Core Driven Memory Scan Compute-and-Compare Mode

Performs CRC signature calculation and verification for a region of memory using core transactions. The CRC peripheral is configured such that it operates in burst mode due to the stalling options configured through disabling the CRC_CTL register.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per CRC_CTL.BLKEN

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final compare operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute-and-compare mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.
   - Disable the CRC_CTL.OBRSTALL and CRC_CTL.IRRSTALL bit options for this task example.
   - Configure all mirroring and bit reversal options.

- Configure CRC auto-clear options.

The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Write memory region data to the CRC peripheral.
   a. While CRC_STAT.IBR bit indicates that the input buffer is ready, write the CRC_DFIFO register with 32-bit data.

      ADDITIONAL INFORMATION: Repeat this step until all data has been written.
8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures that all the data has been processed.
9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

   ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.
10. Write to the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear of these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The integrity check of the memory through the expected CRC signature has completed. The final result is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled.

Clear any W1C CRC status bits before performing more CRC operations.

DMA Driven Memory Scan Compute-and-Compare Mode

Performs CRC signature calculation and verification for a region of memory using DMA transactions. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured through disabling CRC_CTL.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit.

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute compare mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   - Disable the CRC_CTL.OBRSTALL and CRC_CTL.IRRSTALL bit options for this task example.
   - Configure all mirroring and bit reversal options.
   - Configure all CRC auto clear options.

   The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer from the memory region and writes the data to the CRC peripheral.

8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if the counter expired interrupt is disabled. Polling ensures all the data has been processed.

9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

   ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.

10. Write the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The integrity check of the memory through the expected CRC signature has completed and the final result indicated is through CRC_STAT.CMPERR and the corresponding interrupt, when enabled.

Clear any W1C CRC status bits before performing a further CRC operation. Clear any W1C status bits of the memory-to-memory source DMA channel before the next CRC operation.

Core Driven Memory Scan Data Verify Mode

Reads a region of memory using core transactions and performs a compare operation on each 32-bit word against a single pre-loaded 32-bit constant. The compare error interrupt is enabled to capture and log the location of any compare errors.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per CRC_CTL.BLKEN

The interrupt service routine for the compare error interrupt reads and stores the contents of CRC_DCNTCAP register to a buffer before clearing the compare error interrupt.

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the 32-bit constant that the memory region is expected to be filled with. Each 32 bit of data presented to the peripheral is compared with this value.

4. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts. If enabled, ensure that the corresponding interrupt handlers are also configured.

5. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan data verify mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

6. Write memory region data to the CRC peripheral.

a. Poll the CRC_STAT.IBR bit until input buffer is ready.

b. Write the CRC_DFIFO register with 32-bit data.

ADDITIONAL INFORMATION: Repeat these two steps until the entire memory region has been written to the CRC peripheral.

7. Poll the CRC_INEN_SET.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures all the data has been processed.

8. Check if the buffer used to capture the CRC_DCNTCAP register upon a compare error has any new entries.

   ADDITIONAL INFORMATION: The values captures in the buffer provide a means to locate where in the memory region the failures occurred.

9. Write to the CRC_STAT to clear both the CRC_INEN_SET.DCNTEXP and CRC_INEN.CMPERR bits.

   ADDITIONAL INFORMATION: If interrupts were enabled, the clear these status bits within the interrupt handlers for the respective interrupts.

   The CRC memory scan-verify operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The result of the integrity check of the memory with the 32-bit constant is indicated through the CRC_INEN.CMPERR bit and the corresponding interrupt, when enabled. Each comparison error is traceable due to the logging of CRC_DCNTCAP from within the compare error interrupt handler.

Clear any W1C CRC status bits before performing a further CRC operation.

DMA Driven Memory Scan Data Verify Mode

The memory scan data verify mode reads a region of memory using DMA transactions and performs a compare operation on each 32-bit word against a single pre-loaded 32-bit constant. The compare error interrupt is enabled to capture and log the location of any compare errors.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit

The interrupt service routine for the compare error interrupt reads and stores the contents of the CRC_DCNTCAP register to a buffer before clearing the compare error interrupt.

1. Initialize the CRC_DCNT register.

ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: The CRC module uses this register to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the 32-bit constant that the memory region is expected to be filled with. Each 32 bit of data presented to the peripheral is compared with this value.

4. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

5. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan data verify mode and CRC_CTL.BLKEN configured to enable the CRC peripheral.

   The CRC peripheral is now enabled and ready for the core or DMA channel to write the data.

6. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer from the memory region and writes the data to the CRC peripheral.

7. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures all the data has been processed.

8. Check if the buffer used to capture the CRC_DCNTCAP register upon a compare error has any new entries.

   ADDITIONAL INFORMATION: The values captures in the buffer provide a means to locate where in the memory region the failures occurred.

9. Write the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

   ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

   The CRC memory scan-verify operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The result of the integrity check of the memory with the 32-bit constant is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled. Each comparison error is traceable due to the logging of the CRC_DCNTCAP register from within the compare error interrupt handler.

Clear any W1C CRC status bits and DMA status bits before performing a further CRC operation.

Core Driven Memory Transfer Compute-and-Compare Mode

The memory transfer compute-and-compare mode performs CRC signature calculation and verification for a region of memory using core transactions while copying the contents to another memory region. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured through disabling the CRC_CTL register.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of the current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final compare operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute-and-compare mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   a. Disable the CRC_CTL.OBRSTALL bit and the CRC_CTL.IRRSTALL bit options for this task example.

   b. Configure all mirroring and bit reversal options.

c. Configure CRC auto clear options

The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Write memory region data to the CRC peripheral and read it back to the new destination.
   a. While the CRC_STAT.IBR bit indicates that the input buffer is ready, write the CRC_DFIFO register with 32-bit data.
   b. While the CRC_STAT.OBR bit indicates that the output buffer is ready, read the CRC_DFIFO register and store data to new destination.

      ADDITIONAL INFORMATION: Repeat these two steps until all required data has been processed through the CRC peripheral and copied to the new destination.
8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if the counter expired interrupt is disabled. Polling ensures all the data has been processed.
9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

   ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.
10. Write the CRC_STAT register to clear both CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation. The memory region has also been copied to its new destination.

The memory region has been copied to a new location and an integrity check of the memory through the expected CRC signature has also completed. The final result is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled.

Clear any W1C CRC status bits before performing a further CRC operation.

DMA Driven Memory Transfer Compute-and-Compare Mode

The memory transfer compute-and-compare mode performs CRC signature calculation and verification for a region of memory using DMA transactions. The memory region is also copied to another memory region using memory-to-memory DMA transfers. The CRC peripheral is configured such that it operates in burst mode due to the stalling options configured through disabling CRC_CTL.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)

- The CRC block is disabled per the CRC_CTL.BLKEN register.

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final compare operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute compare mode and CRC_CTL.BLKEN configured to enable the CRC peripheral.

   a. Disable the CRC_CTL.OBRSTALL and the CRC_CTL.IRRSTALL bit options for this task example.

   b. Configure all mirroring and bit reversal options

   c. Configure CRC auto clear options

   The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode and destination DMA channel for memory write STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer from one memory region to another through the memory-to-memory DMA channels and the CRC peripheral.

8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures all the data has been processed.

9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.

10. Write the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and the CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation. The memory region has also been copied to its new destination.

The integrity check of the memory through the expected CRC signature has completed and the final result is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled. The memory region has also been copied to its final destination.

Clear any W1C CRC status bits before performing a further CRC operation. Also, clear any W1C status bits of the memory-to-memory source and destination DMA channels before the next CRC operation.

DMA Driven Memory Transfer Data Fill Mode

This mode initializes a region of memory to a constant 32-bit value using DMA transactions.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_FILLVAL register.

   ADDITIONAL INFORMATION: This register contains the 32-bit constant that the CRC module uses to fill the memory region.

4. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of block completion. Configure these interrupts as required. If enabled, ensure that the corresponding interrupt handlers are also configured.

5. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory transfer fill mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   The CRC peripheral is now enabled and is ready for the DMA channel to write data.

6. Configure and enable the memory-to-memory destination DMA channel for memory write STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer taking the constant 32-bit value from the CRC peripheral and writing the data to the DMA channel.

7. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures that all the data has been processed.

8. Write the CRC_STAT register to clear the CRC_STAT.DCNTEXP bit.

   ADDITIONAL INFORMATION: If interrupts were enabled, then clear this status bit within the interrupt handlers for the respective interrupts.

   The CRC memory transfer fill operation is complete, and the CRC peripheral is ready to be configured for the next CRC operation.

The memory region is filled with the constant data and the CRC peripheral is ready to be configured for a new operation.

Clear any W1C CRC status bits and DMA status bits before performing a further CRC operation.

ADSP-2184x CRC Register Descriptions

Cyclic Redundancy Check Unit (CRC) contains the following registers.

Table 36-4: ADSP-2184x CRC Register List

Name	Description
CRC_COMP	Data Compare Register
CRC_CTL	Control Register
CRC_DCNT	Data Word Count Register
CRC_DCNTCAP	Data Count Capture Register
CRC_DCNTRLD	Data Word Count Reload Register
CRC_DFIFO	Data FIFO Register
CRC_FILLVAL	Fill Value Register
CRC_INEN	Interrupt Enable Register
CRC_INEN_CLR	Interrupt Enable Clear Register
CRC_INEN_SET	Interrupt Enable Set Register

Table 36-4: ADSP-2184x CRC Register List (Continued)

Name	Description
CRC_POLY	Polynomial Register
CRC_RESULT_CUR	CRC Current Result Register
CRC_RESULT_FIN	CRC Final Result Register
CRC_STAT	Status Register

Data Compare Register

The CRC_COMP register contains the value corresponding to the expected CRC result or signature for the current data stream. At the end of the operation, the content of this register is used to compare against the result produced by the CRC operation. In data verify mode, each incoming data value is compared with the content of this register.

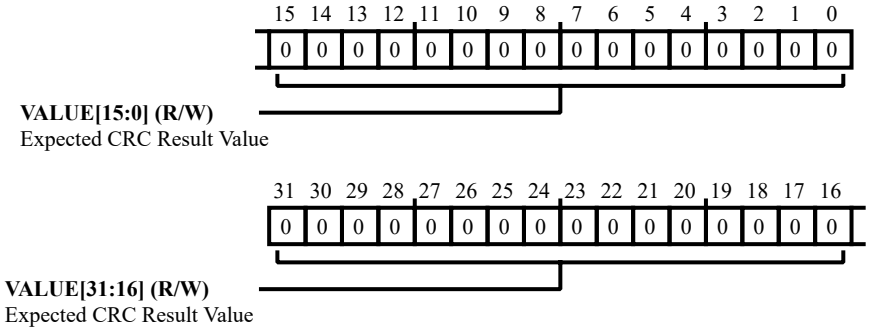


Figure 36-3: CRC_COMP Register Diagram

Table 36-5: CRC_COMP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Expected CRC Result Value. The CRC_COMP.VALUE bit field contains the value corresponding to the expected CRC result or signature for the current data stream.

Control Register

The CRC_CTL register configures the operation modes and settings for the CRC.

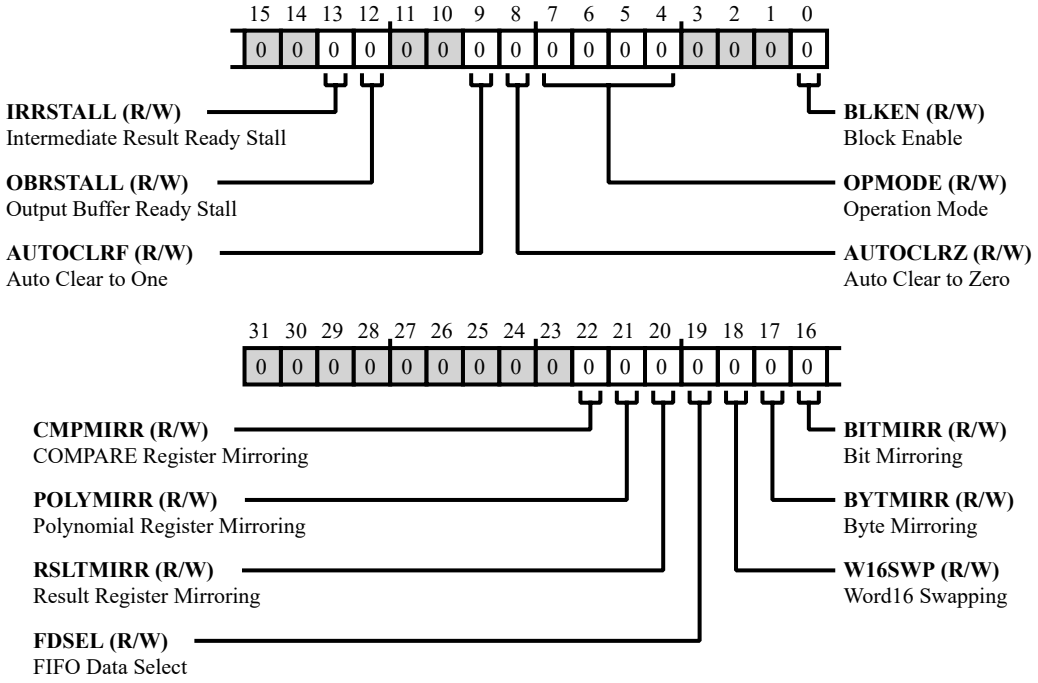


Figure 36-4: CRC_CTL Register Diagram

Table 36-6: CRC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	CMPMIRR	COMPARE Register Mirroring. The CRC_CTL.CMPMIRR bit enables data mirroring for the CRC_COMP compare register. When enabled, the 32-bit value in this register is fully bit mirrored (reversed). The bit-reversed value is used for comparison with the CRC_RESULT_FIN register.	
		0	Disable compare mirroring
		1	Enable compare mirroring
21 (R/W)	POLYMIRR	Polynomial Register Mirroring. The CRC_CTL.POLYMIRR bit enables data mirroring for the CRC_POLY polynomial register. When enabled, the 32-bit value in this register is fully bit mirrored (reversed). The bit-reversed value is used for CRC computations.	
		0	Disable polynomial mirroring
		1	Enable polynomial mirroring

Table 36-6: CRC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	RSLTMIRR	Result Register Mirroring. The CRC_CTL.RSLTMIRR bit enables data mirroring for the CRC_RESULT_CUR and CRC_RESULT_FIN result registers. When enabled, the 32-bit values in these registers are fully bit mirrored (reversed).	
		0	Disable result mirroring
		1	Enable result mirroring
19 (R/W)	FDSEL	FIFO Data Select. The CRC_CTL.FDSEL bit selects whether the CRC writes modified or unmodified data to the FIFO in memory transfer mode. If enabled, the data written is affected by the state of the data mirroring selections (CRC_CTL.BITMIRR, CRC_CTL.BYTMIRR, and CRC_CTL.W16SWP) before being written to the FIFO.	
		0	Write unmodified data to FIFO
		1	Write modified data to FIFO
18 (R/W)	W16SWP	Word16 Swapping. The CRC_CTL.W16SWP bit enables the CRC's data mirror block to swap the upper and lower 16-bit words within the 32-bit input data, before further processing.	
		0	Disable word16 swapping
		1	Enable word16 swapping
17 (R/W)	BYTMIRR	Byte Mirroring. The CRC_CTL.BYTMIRR bit enables the CRC's data mirror block to mirror the bytes within the 32-bit input data, before further processing.	
		0	Disable byte mirroring
		1	Enable byte mirroring
16 (R/W)	BITMIRR	Bit Mirroring. The CRC_CTL.BITMIRR bit enables the CRC's data mirror block to mirror the bits within each byte of the 32-bit input data, before further processing.	
		0	Disable bit mirroring
		1	Enable bit mirroring
13 (R/W)	IRRSTALL	Intermediate Result Ready Stall. The CRC_CTL.IRRSTALL bit enables stalling the state machine for input data when there is a valid intermediate result to be read in the CRC_RESULT_CUR register. This feature should be used only in CRC computation modes (for example, CRC_CTL.OPMODE =1 or =3).	
		0	Do not stall
		1	Stall on IRR

Table 36-6: CRC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	OBRSTALL	Output Buffer Ready Stall. The CRC_CTL.OBRSTALL bit enables stalling the state machine for input data when there is valid data in the output buffer. This feature should be used only in memory-to-memory transfer modes (for example, CRC_CTL.OPMODE =1).	
		0	Do not stall
		1	Stall on OBR
9 (R/W)	AUTOCLRF	Auto Clear to One. The CRC_CTL.AUTOCLRF bit enables auto clear to one when the CRC is in intermediate results ready stall mode (CRC_CTL.IRRSTALL=1) and the CRC data count expires (CRC_DCNT=0). Note that the CRC_CTL.AUTOCLRZ bit must be disabled, or the CRC_CTL.AUTOCLRF bit has no effect.	
		0	No auto clear
		1	Auto clear
8 (R/W)	AUTOCLRZ	Auto Clear to Zero. The CRC_CTL.AUTOCLRZ bit enables auto clear to zero when the CRC is in intermediate results ready stall mode (CRC_CTL.IRRSTALL=1) and the CRC data count expires (CRC_DCNT=0). Note that CRC_CTL.AUTOCLRF must be disabled, or the CRC_CTL.AUTOCLRZ has no effect.	
		0	No auto clear
		1	Auto clear
7:4 (R/W)	OPMODE	Operation Mode. The CRC_CTL.OPMODE bit field selects the memory transfer or scan mode.	
		0	Reserved
		1	CRC compute/compare memory transfer
		2	Data fill memory transfer
		3	CRC compute/compare memory scan
		4	Data verify memory scan
0 (R/W)	BLKEN	Block Enable. The CRC_CTL.BLKEN bit enables and disables the CRC operation.	
		0	Disable
		1	Enable

Data Word Count Register

The CRC_DCNT register holds the word count that is used for the CRC operation. On transfer of every 32-bit word, the CRC decrements by 1 the content of this register. When the count decrements to zero, this event triggers a CRC compare action, and the CRC_DCNT register is automatically loaded from the CRC_DCNTRLD register for the next CRC operation.

Note that the initial value programmed into the CRC_DCNT register may be different from what is programmed in the CRC_DCNTRLD register.

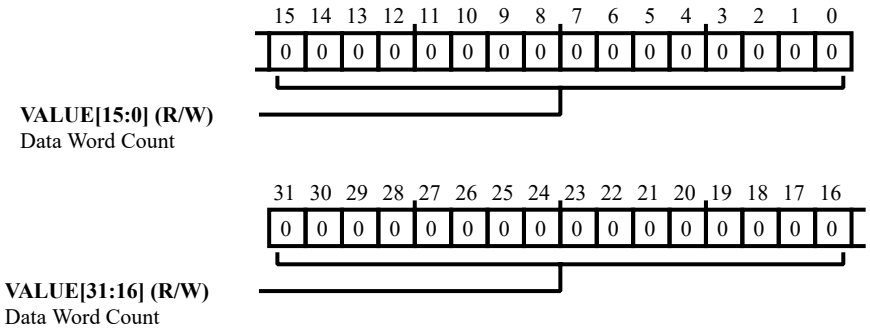


Figure 36-5: CRC_DCNT Register Diagram

Table 36-7: CRC_DCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Data Word Count. The CRC_DCNT.VALUE bit field holds the word count that is used for the CRC operation.

Data Count Capture Register

The CRC_DCNTCAP register captures the CRC_DCNT value when a compare operation fails in data verify mode. This capture can be used to track the position of an error in the data stream. The capture operation is enabled only if the CRC_STAT.CMPERR bit indicates no compare error. After an error occurs and the data count is captured, no further errors are logged until the CRC_STAT.CMPERR bit is cleared. To obtain the position of an error in the data stream, subtract the CRC_DCNTCAP register value from the initial CRC_DCNT.

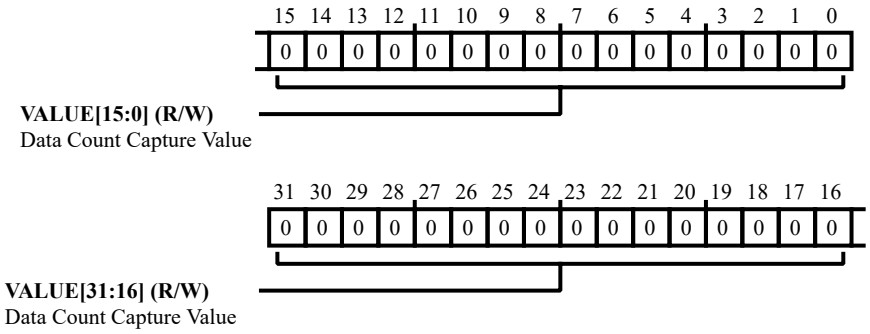


Figure 36-6: CRC_DCNTCAP Register Diagram

Table 36-8: CRC_DCNTCAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Data Count Capture Value. The CRC_DCNTCAP.VALUE bit field contains the CRC_DCNT value when a compare operation fails in data verify mode.

Data Word Count Reload Register

The CRC_DCNTRLD register holds the value that the CRC automatically loads into CRC_DCNT when the CRC_DCNT decrements to 0. At startup, the value programmed in CRC_DCNT and the CRC_DCNTRLD register could be different. So, for the first iteration, the CRC operation happens for the count initially programmed in the CRC_DCNT register. While for subsequent CRC operations, the count is taken from the CRC_DCNTRLD register.

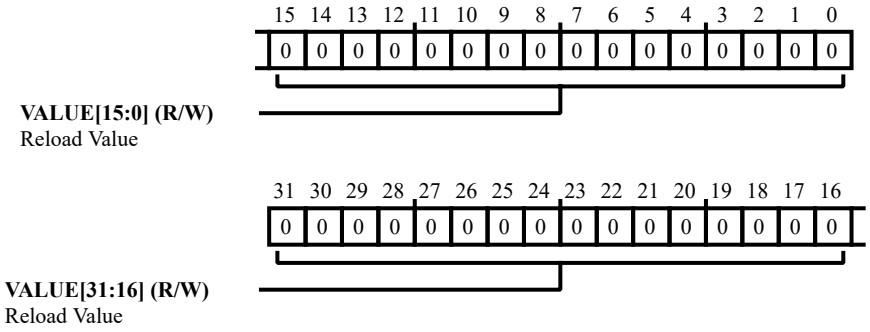


Figure 36-7: CRC_DCNTRLD Register Diagram

Table 36-9: CRC_DCNTRLD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Reload Value. The CRC_DCNTRLD.VALUE bit field holds the value that automatically loads into CRC_DCNT when the CRC_DCNT decrements to 0.

Data FIFO Register

In memory transfer mode (non-data fill mode), the data from the DMA or processor core buses is written into the CRC_DFIFO on each input data grant (DMA grant or core write). Data is read from this FIFO on each output data grant (DMA grant or core read). FIFO status information is available in the CRC_STAT register. Whenever, the FIFO has valid data, output data requests are generated.

Note that in non-memory transfer mode and in data fill mode, the input data does not get written into this FIFO. So, this register should not be read in these modes.

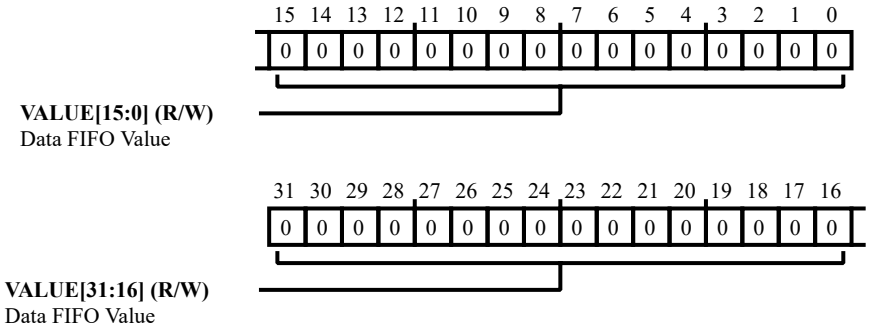


Figure 36-8: CRC_DFIFO Register Diagram

Table 36-10: CRC_DFIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Data FIFO Value. The CRC_DFIFO.VALUE bit field is the data from the DMA or processor core buses.

Fill Value Register

The CRC_FILLVAL register holds the value that the CRC uses for the memory fill operation. In data fill mode, the value programmed in this register is used for the memory fill operation.

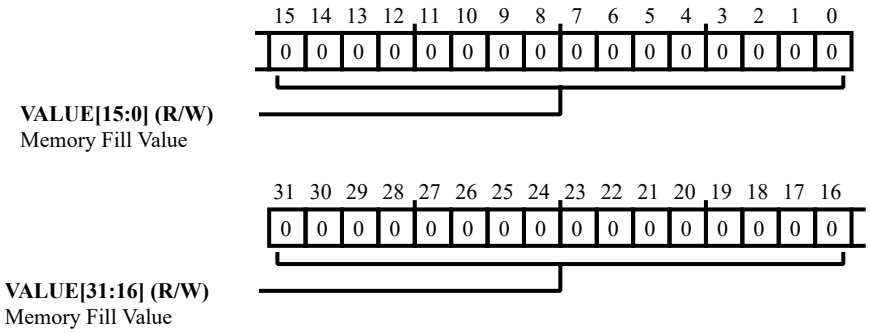


Figure 36-9: CRC_FILLVAL Register Diagram

Table 36-11: CRC_FILLVAL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Memory Fill Value. The CRC_FILLVAL.VALUE bit field holds the value that the CRC uses for the memory fill operation.

Interrupt Enable Register

The CRC_INEN register unmasks (enables) or masks (disables) interrupt requests generated in the CRC from going to the processor core.

Note that CRC interrupts are not disabled when the CRC is disabled (CRC_CTL.BLKEN =0).

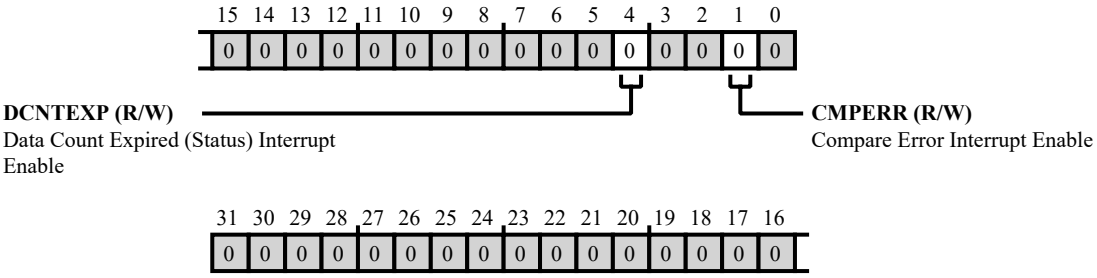


Figure 36-10: CRC_INEN Register Diagram

Table 36-12: CRC_INEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	DCNTEXP	Data Count Expired (Status) Interrupt Enable. The CRC_INEN.DCNTEXP enables (unmasks) the data count expired (CRC status) interrupt.	
		0	Disable (mask) interrupt
		1	Enable (unmask) interrupt
1 (R/W)	CMPERR	Compare Error Interrupt Enable. The CRC_INEN.CMPERR enables (unmasks) the data compare interrupt, which is generated when CRC data comparison fails.	
		0	Disable (mask) interrupt
		1	Enable (unmask) interrupt

Interrupt Enable Clear Register

The CRC_INEN_CLR register permits clearing individual bits in the CRC_INEN register without affecting other bits in the register.

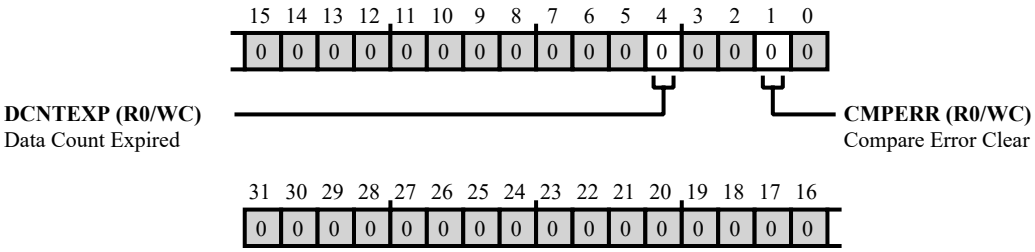


Figure 36-11: CRC_INEN_CLR Register Diagram

Table 36-13: CRC_INEN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/WC)	DCNTEXP	Data Count Expired. The CRC_INEN_CLR.DCNTEXP bit clears the data count expired (status) interrupt.	
		0	No effect
		1	Clear bit
1 (R0/WC)	CMPERR	Compare Error Clear. The CRC_INEN_CLR.CMPERR bit clears the compare error interrupt.	
		0	No effect
		1	Clear bit

Interrupt Enable Set Register

The CRC_INEN_SET register permits setting individual bits in the CRC_INEN register without affecting other bits in the register.

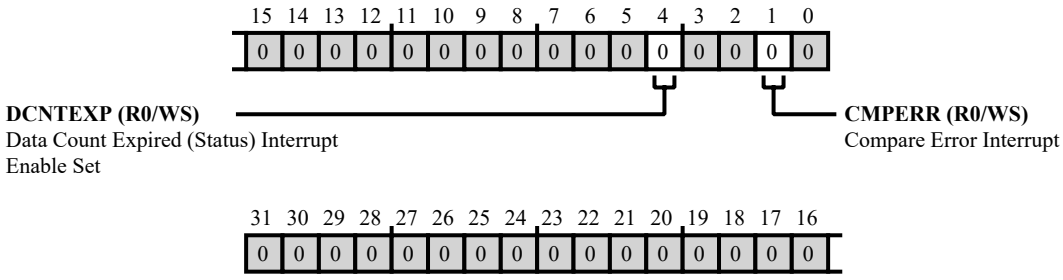


Figure 36-12: CRC_INEN_SET Register Diagram

Table 36-14: CRC_INEN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/WS)	DCNTEXP	Data Count Expired (Status) Interrupt Enable Set. The CRC_INEN_SET.DCNTEXP bit sets the data count expired (status) interrupt.	
		0	No effect
		1	Set bit
1 (R0/WS)	CMPERR	Compare Error Interrupt. The CRC_INEN_SET.CMPERR bit sets the compare error interrupt.	
		0	No effect
		1	Set bit

Polynomial Register

The CRC_POLY register holds a 32-bit polynomial for CRC operations. Bit 31 corresponds to the coefficient of x31 of the CRC polynomial, bit 30 corresponds to the coefficient of x30, and so on through bit 0. A coefficient of x32 is assumed to be "1" for any polynomial that is selected. Based on the polynomial in the CRC_POLY register, the CRC generates a look-up table (LUT), which is used to compute the CRC of the incoming data stream.

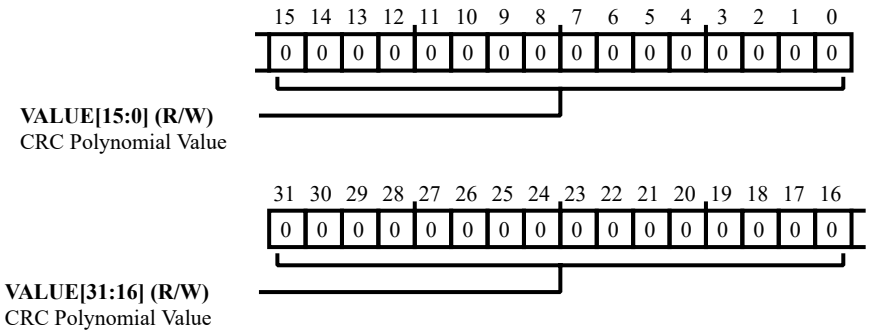


Figure 36-13: CRC_POLY Register Diagram

Table 36-15: CRC_POLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	CRC Polynomial Value. The CRC_POLY.VALUE bit field holds the 32-bit polynomial for CRC operations.

CRC Current Result Register

The CRC_RESULT_CUR register holds the current or intermediate CRC result. It is updated when new data is written into the CRC. Each time the CRC_DCNT expires, the CRC loads the value from this register into the CRC_RESULT_FIN register. The CRC_RESULT_CUR register may be set to auto clear to zero or auto clear to ones when CRC_DCNT expires by configuring the CRC_CTL.AUTOCLRZ and CRC_CTL.AUTOCLRF bits. Before starting a CRC operation, the CRC_RESULT_CUR register should be programmed to the desired value.

Note that this register can be read by the processor core at any time.

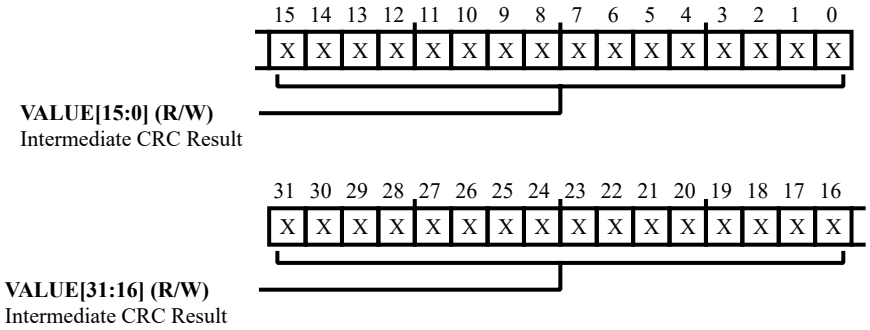


Figure 36-14: CRC_RESULT_CUR Register Diagram

Table 36-16: CRC_RESULT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Intermediate CRC Result. The CRC_RESULT_CUR.VALUE bit field holds the current or intermediate CRC result.

CRC Final Result Register

The CRC_RESULT_FIN register holds the final CRC computed for a data stream. A data stream is a DMA of CRC_DCNT number of words into the CRC. When CRC_DCNT decrements to zero for each data stream, the CRC loads the CRC_RESULT_FIN register with the value from the CRC_RESULT_CUR register.

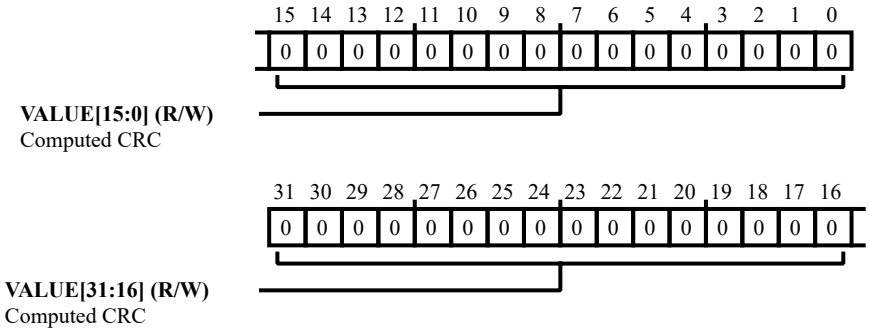


Figure 36-15: CRC_RESULT_FIN Register Diagram

Table 36-17: CRC_RESULT_FIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Computed CRC. The CRC_RESULT_FIN.VALUE bit field holds the final CRC computed for a data stream.

Status Register

The CRC_STAT register indicates the status for CRC operations and interrupt generation.

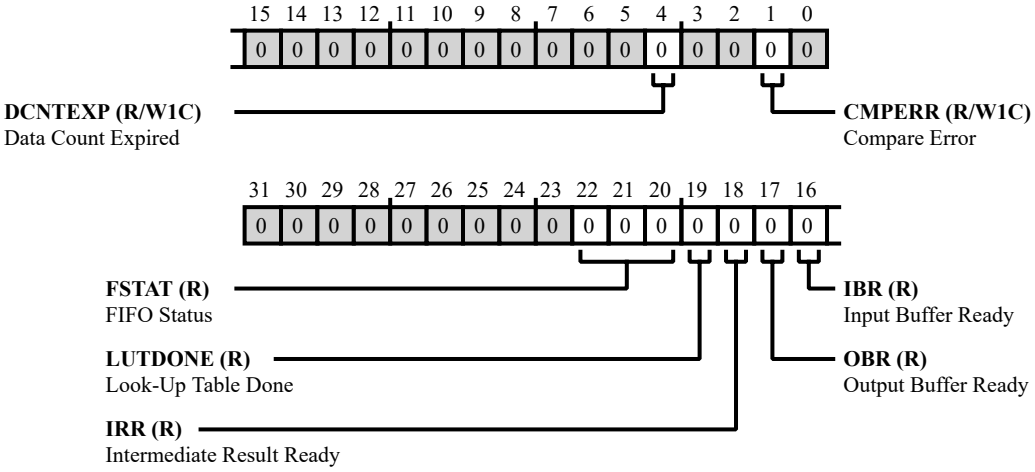


Figure 36-16: CRC_STAT Register Diagram

Table 36-18: CRC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22:20 (R/NW)	FSTAT	FIFO Status. The CRC_STAT.FSTAT indicates the current FIFO status. This field is read-only.	
		0	FIFO empty
		1	FIFO has 1 data
		2	FIFO has 2 data
		3	FIFO has 3 data
		4	FIFO has 4 data (full)
19 (R/NW)	LUTDONE	Look-Up Table Done. The CRC_STAT.LUTDONE bit indicates that the CRC has generated the look-up table for the current polynomial. This read-only bit is cleared at reset and cleared when the CRC_POLY is written.	
		0	No status
		1	LUT generation done

Table 36-18: CRC_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	IRR	Intermediate Result Ready. The CRC_STAT.IRR bit indicates that the CRC has updated the CRC_RESULT_CUR register with intermediate CRC results for the new data written to the CRC. The processor core should read from the CRC_RESULT_CUR register only after detecting CRC_STAT.IRR =1. This read-only bit is cleared by CRC hardware and is valid when the CRC_CTL.IRRSTALL bit is enabled.	
		0	No status
		1	Intermediate results ready
17 (R/NW)	OBR	Output Buffer Ready. The CRC_STAT.OBR bit indicates that the CRC has data ready for the processor core to read. The processor core should read from the CRC only after detecting CRC_STAT.OBR =1. This read-only bit is cleared by CRC hardware.	
		0	No status
		1	Output buffer ready
16 (R/NW)	IBR	Input Buffer Ready. The CRC_STAT.IBR bit indicates that the CRC is ready to accept a processor core write. The processor core should write to the input register only after detecting that CRC_STAT.IBR =1. This read-only bit is cleared by CRC hardware.	
		0	No status
		1	Input buffer ready
4 (R/W1C)	DCNTEXP	Data Count Expired. The CRC_STAT.DCNTEXP bit indicates that the CRC_DCNT has expired. This W1C bit is not automatically cleared when the CRC is disabled (CRC_CTL.BLKEN =0). When the CRC sets this bit on CRC_DCNT expiry, the CRC generates the CRC_INEN.DCNTEXP interrupt.	
		0	No status
		1	Data counter expired
1 (R/W1C)	CMPERR	Compare Error. The CRC_STAT.CMPERR bit indicates that a CRC mismatch or data mismatch has been detected. This W1C bit is not automatically cleared when the CRC is disabled (CRC_CTL.BLKEN =0). When the CRC sets this bit on detecting a mismatch, the CRC generates the CRC_INEN.CMPERR interrupt. While this bit is set, the CRC_DCNTCAP register is disabled from capturing the data count values.	
		0	No status
		1	Compare error

37 System Security

The requirements to protect content, keys, intellectual property, and other sensitive information have become increasingly prevalent. The processor contains several modules and system elements that contribute to creating a secure operating environment for trusted code to execute.

The modules and system elements for security are:

- Boot Kernel
- Secure Core
- System Protection Unit (SPU)
- Security Packet Engine (PKTE)
- Public Key Accelerator (PKA)
- Public Key Interrupt Controller (PKIC)
- System Memory Protection Unit (SMPU)
- DEBUG through the Test Access Port Controller (TAPC)
- One-Time Programmable (OTP) memory
- Cryptographic Accelerators (optionally) and Hardware Security Module (HSM)

NOTE: This product includes security features for protecting embedded non-volatile memory contents and preventing execution of non-authorized code. Enabling security on this device (either by the ordering party or the subsequent receiving parties), limits the ability of Analog Devices to conduct failure analysis on returned devices. Contact Analog Devices, Inc. for details on the failure analysis limitations for this device.

Security Features

The security infrastructure in the system provides the following features:

- Secure operating environment for secure code execution
- Protecting sensitive IP from theft by malicious users or competitors

- Protecting sensitive data (for example cipher keys)
- Allowing debugging while still maintaining security

Security Functional Description

In order to provide a secure operating environment in a system, it requires the involvement of multiple elements.

Boot Kernel

The boot kernel is the root of trust for a secure system. Since the boot kernel is developed by Analog Devices, Inc., is stored in ROM, and can't be changed, it can be trusted. The boot kernel validates the authenticity of the application binary image that needs to be booted in. It also handles decrypting the binary image if it's encrypted.

Verifying the authenticity of the application binary asserts that

1. The image is not tampered with or altered
2. The image came from a trusted developer

If the image is encrypted, it ensures confidentiality since the boot image is stored on an external storage device and can be more easily read or stolen than in the part.

Once the boot kernel can verify and optionally decrypt the boot image, it can be loaded into the processor for execution. At this point, the *chain of trust* is continued with the verified application.

Secure Booting

Secure booting is when the boot kernel uses cryptographic algorithms to perform checks on the application binary and to decrypt it. When security is enabled, signature verification and optional decryption on the application binary are both performed by secure boot. See Security Mode Configuration.

Secure Core

In a system with multiple requester and completer resources, not everything is considered secure. There are secure and non-secure peripherals and secure and non-secure segments of memory.

For a system to be considered secure, a secure core that can access and execute instructions (verified application) from secure memory must be configured. Typically, in a single core processor, either the core security is hardwired, or the core can switch between secure and non-secure modes.

System Protection Unit (SPU)

The SPU in the system serves two functions. First, it acts as a gatekeeper, guarding against non-secure accesses to secure resources (peripherals). Second, it is used to define which resources in the system are secure or non-secure requesters and which resources in the system are secure or non-secure completers.

NOTE: Though the SPU can be configured by secure or non-secure requester, the first steps that the verified secure application must perform are:

1. Configure the SPU as a secure completer itself so only other secure requesters can configure it
2. Define and configure the secure requesters and completers using the SPU

This way, once the SPU is secure and other secure requesters and completers are configured, non-secure requesters cannot tamper with the security privileges of secure requesters and completers nor can non-secure resources be changed to a secure resource.

System Memory Protection Unit (SMPU)

Similar to the SPU protecting the MMR address range for peripherals, the SMPU can guard memory ranges or pages. Memory pages can be configured as secure or non-secure. Again, like in the case of the SPU, the SMPU can guard against non-secure transaction attempts to secure memory.

NOTE: A verified application should reside in memory configured as secure. By default the SMPU has all memory configured as secure. When the program needs to update the memory protections using the SMPU, the application and its data should remain in secure memory.

Debug

The typical way of accessing a system is through the debug port via a JTAG or serial wire interface. When these access points are not secure, sensitive IP such as cipher keys can be exposed and code can be changed to disable other security settings. To guard against this type of attack or security hole and still provide debugging capabilities for a developer, a debug unit with security features is used.

When the developer first receives the part, they can define a JTAG/DEBUG key that is programmed into OTP memory. Once security is enabled, the debug unit compares the key sent from the host debugger with the key inside the system. If a match occurs, debug access to secure resources are allowed.

One-Time-Programmable (OTP) Memory

Customer programmable OTP memory is used to safely and securely store sensitive information such as cipher keys.

In a public key algorithm (for example, ECDSA which is used in secure boot), the public key is used to verify the digital signature that accompanies the application binary. The private key is used by the developer on the host development machine to create the digital signature. The public and private key pair is unique, plus the public key needs to be stored in non-volatile, one-time-programmable memory. When the public key can be changed then users can generate their own public/private key pairs and successfully boot in malicious code. This can either re-purpose the part or change security configurations to allow easier access to sensitive information stored elsewhere in the OTP memory.

Cryptographic Accelerators

Cryptographic algorithms are mathematical tools to help provide security. Hardware engines provide some advantages but are not necessarily required. For computationally expensive operations like those used in Elliptic Curve

Cryptography, like ECDSA used in secure boot, the operations can be accelerated while the core performs other tasks. Also, it's less likely that the hardware engine can be hacked to change the results.

Hardware Security Module (HSM)

While cryptographic accelerators primarily optimize performance, the Hardware Security Module (HSM) can also serve as the dedicated root of trust for the SoC. The HSM is an autonomous subsystem that includes its own processor, isolated memory, and cryptographic engines. Its primary function is to manage the lifecycle of high-value assets—such as encryption keys and certificates—within a secure boundary that is inaccessible to the main application cores. By performing sensitive operations (like key generation and signing) internally, the HSM ensures that secret keys are never exposed to the host operating system, significantly reducing the attack surface against software-based vulnerabilities.

Security Mode Configuration

Security as a feature does not necessarily be employed by the user. There are no steps to disable security, if security protection is not required. The part does not have security enabled.

To use the security features using cryptographic accelerators, perform the following steps:

- Generate the public/private key pair on the host development machine.[1]
- Program OTP memory with the public key.[2]
- Program OTP memory with the decryption key if the application binary needs to be encrypted.[2]
- Program OTP memory with the debug/JTAG key.[2]
- Develop the application and sign it, creating the digital signature with the private key.[1]
- If confidentiality is required, encrypt the application binary before signing it.
- Set the LOCK bit in OTP memory to enable security. After this, subsequent boots are secure boots.[2, 3]

[1] Software tools are provided with developments tools to generate keys, sign boot streams, and also perform encryption. Refer to the development tools manuals for details on how to use.

[2]Refer to the OTP Chapter for programming the OTP and other related information.

[3]Refer to the Booting Chapter for more information on Secure Booting and other related information.

HSM Assisted Secure Boot Flow

In this implementation, the SoC retains the role of the primary boot master upon coming out of reset. The secure boot sequence is a staged process where the SoC first provisions the security subsystem before delegating sensitive tasks.

1. HSM Initialization and Staging—The SoC loads the HSM application firmware from external storage into the intermediate memory. Once the image is staged, the SoC releases the HSM from reset. The HSM then boots from the intermediate memory, initializing its isolated environment and authenticating the HSM firmware integrity.
2. Runtime Security Services (SIC Protocol)—Once operational, the HSM transitions to a service-provider role, communicating with the SoC via the Secure Inter-Processor Communication (SIC) protocol. For subsequent boot stages, specifically the incoming Secure Image, the SoC offloads all cryptographic validation to the HSM. Tasks such as ECDSA signature verification and AES decryption, previously handled by direct crypto accelerator access, are now executed within the secure boundary of the HSM, ensuring that the verification keys and decrypted code remain isolated from the main application core.

All the relevant asset specific to HSM boot are programmed in HSM OTP space.

Status and Error Signals

In a fully functional secure system, non-secure resources should not even attempt to access a secure resource. When this happens, either the code has been altered or replaced with malicious code or the system contains a coding error.

Errors or error events are dependent on the configuration of the SPU and SMPU, the protection units which guard against security violations. In the case of the SMPU, the error can simply be captured, captured and interrupt generated, or the access prevented without capturing any error. It is the developer's responsibility to:

1. Determine when there was an error because of a blocked access.
2. Determine how to handle a blocked access. For example fix the bug (offline) or try to use a different resource (run time).

38 System Protection Unit (SPU)

In a system with multiple system MMR requesters, peripheral configurations can be changed unintentionally, leading to bad data or even system malfunctions. The peripherals are shared resources in the system. The SPU restricts access to certain MMRs, similar to the functionality of a semaphore.

The SPU also protects peripherals based on security settings. It is part of the overall security infrastructure of the processor.

SPU Features

The SPU has the following features:

- Write-protect system MMR from certain system requesters and core requesters.
- Simultaneously lock multiple peripheral configuration registers through a global lock mechanism.
- Write-protect and block access to its own write-protection registers from other system requesters.
- Defined security privileges to peripherals and system resources.
- Security protection to guard secure peripheral MMRs against non-secure accesses.

SPU Functional Description

The following sections provide information on the function of the SPU.

ADSP-2184x SPU Register List

The System Protection Unit (SPU) provides a set of registers that can protect system resources from errant writes. The protection categories are global lock (protects configuration registers) and write protect register lock (protects the write protect register). For more information on SPU functionality, see the SPU register descriptions.

Table 38-1: ADSP-2184x SPU Register List

Name	Description
SPU_CTL	Control Register
SPU_SECURECHK	Secure Check Register

Table 38-1: ADSP-2184x SPU Register List (Continued)

Name	Description
SPU_SECURECTL	Secure Control Register
SPU_SECUREC[n]	Secure Core Registers
SPU_SECUREP[n]	Secure Peripheral Register
SPU_STAT	Status Register
SPU_WP[n]	Write Protect Register n

ADSP-2184x SPU Interrupt List

Table 38-2: ADSP-2184x SPU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
306	SPU0_INT	SPU0 Interrupt	Level	

Peripheral Register Write Protection

The SPU has a write-protection register (SPU_WP[n]) associated with each peripheral. Each of these write-protection registers has the exact same bits that correspond to a particular SMMR requester (for example, Core 0, MDMA). When the bits are set, the SPU locks the corresponding SMMR requesters from accessing the register address space of the associated peripheral. The bits in the register can be cleared to allow access to the registers of the peripheral again. When the SPU initializes the write-protection register, any writes that are in-progress complete before the SPU blocks subsequent writes.

In the ***SPU Write Protect Registers*** figure, each write-protect register in the SPU is associated with a particular peripheral.

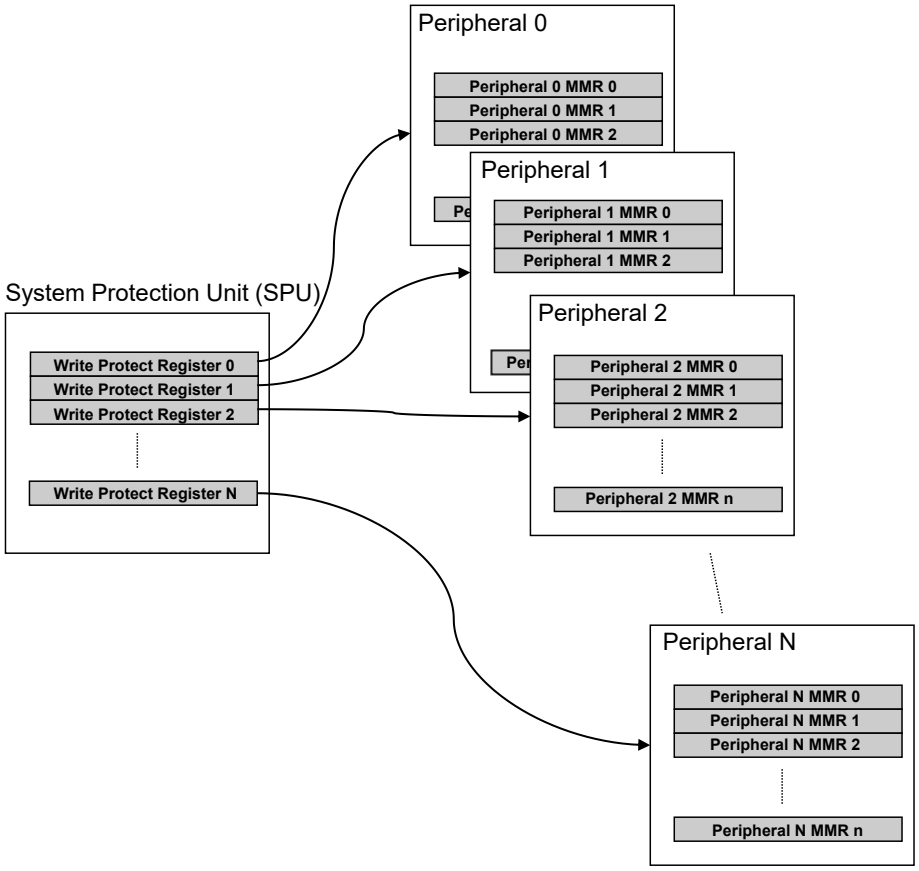


Figure 38-1: SPU Write Protect Registers

In the figure, a write-protect register in the SPU module blocks write-attempts to the MMR space of the associated peripheral. The bits in the write-protect register specify from which requesters to block write-access.

NOTE: A SPU write protection register (`SPU_WP[n]`) exists for the SPU alone. If all defined bits are set in this register for the SPU, any configurations in the SPU are locked and cannot be changed. Only a system reset can restore access to the SPU.

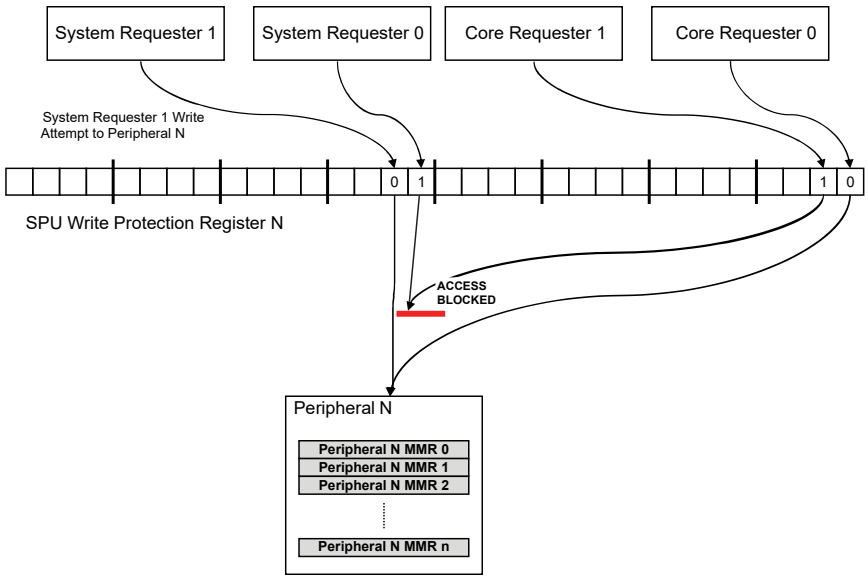


Figure 38-2: SPU Write-Protect Register Blocking Access from System Requester 0 and Core Requester 1

Global Locking

The SPU also has global locking capability. When enabled by setting `SPU_CTL.GLCK` bit field to a value other than 0xAD, a system-wide global lock signal is active. Some peripherals have a lock enable bit in their control register. When this bit is set, the peripheral recognizes the global lock signal and blocks further write-accesses to its own control register. Access to the configuration register of the peripheral is enabled when the global lock is turned off in the SPU.

The ***Global Locking*** figure is a conceptual diagram. The diagram shows how the SPU module (or any peripheral) blocks any write attempts to its control register when:

- The global lock signal from the SPU is active, and
- The global lock enable bit is set in the control register of the peripheral

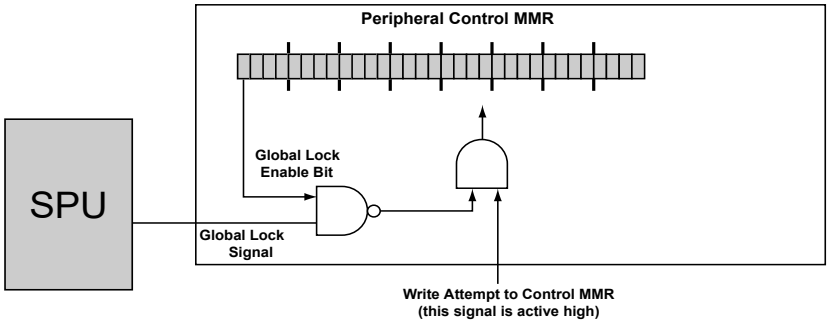


Figure 38-3: Global Locking

The SPU can write-protect its own registers. When the `SPU_CTL.WPLCK` bit is set and global locking is enabled, the SPU blocks accesses to the SPU write-protection registers. To enable write access to the write-protection registers in the SPU, disable the global locking.

SPU Block Diagram

The ***SPU System-Level Block Diagram*** shows a system-level block diagram of where the SPU is located in the system. It resides between the SMMR interface and the system crossbar. Depending on the configuration of the SPU write-protect registers, it can block access to some peripherals from certain SMMR requesters.

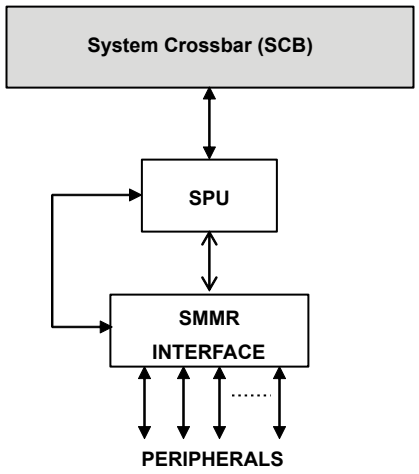


Figure 38-4: SPU System-Level Block Diagram

SPU Architectural Concepts

As shown in the block diagram, the SPU sits between the system crossbar (SCB) and the SMMR interface to the peripherals. The SPU gates any MMR access to any peripheral from any requester that comes through the SCB. Depending on the configuration of the write-protection registers in the SPU, the SPU does or does not allow the MMR write to go through.

The SPU also checks whether the transaction is a secure or non-secure transaction and blocks it according to the configured security setting for the target destination. A secure requester can generate secure read or secure write transactions which can access secure or non-secure completers. A non-secure requester can generate non-secure read or non-secure write transactions and can only access non-secure completers.

SPU Event Control

The system protection unit provides write-protection against MMRs peripherals and its own write-protect registers. If a write attempt is made to any locked MMR peripheral the SPU has write-protected, it blocks the write. The SPU generates a bus error to the requester that attempted the write. That requester does or does not generate an event, based on the returned error.

The SPU can be configured to generate an interrupt for the write-protection violation by setting the `SPU_CTL.PINTEN` bit. The SPU can also be configured to generate an interrupt for a security violation by setting the `SPU_SECURECTL.SINTEN` bit. If either one or both bits is triggered, the `SPU_STAT.VIRQ` bit is set.

The SPU can also lock its own registers from write attempts. If a write-attempt is made to a locked register in the SPU, the SPU blocks it and records it as an error in the SPU_STAT.LWERR bit. Again, the SPU generates a bus error to the requester that attempted the write.

The requester does or does not generate an event, based on the returned error.

The SPU does not generate an event for a blocked write access to an SPU register. If the SPU_CTL.PINTEN bit is set, the SPU triggers an interrupt for this blocked access attempt.

The global lock is enabled by setting the SPU_CTL.GLCK bit to something other than 0xAD. If the lock bit is set in that same configuration register, a peripheral can block write access to its configuration register. When the SPU blocks a write attempt, the peripheral logs and reports the failed attempt. The SPU is unaware and therefore does not provide any indication of a failed write attempt to the configuration register of the peripheral.

SPU Programming Model

The system protection unit (SPU) consists of write-protect and access-protect registers. Each one corresponds to a different peripheral instance. Bits in the write-protect registers correspond to system requesters that can modify the MMR contents of the peripherals. By writing to these write-protect registers, the corresponding memory-mapped registers of the peripheral are write-protected against requesters whose bits in the write-protect register are set.

The SPU globally locks the control register of the peripheral. Peripherals that support this feature have a lock enable bit in their control register. The peripheral blocks any additional write attempts to its control register from any requester when:

- The global lock signal is active from the SPU, and
- The lock enable bit of the peripheral is set

If the lock enable bit of a peripheral is not set and the global lock signal is active, access to that control register of the peripheral is still allowed. To grant access again, disable the global lock signal from the SPU by writing the value 0xAD into the SPU_CTL.GLCK bit field.

Another protection mechanism that the SPU offers is write-protection against the write-protection registers. If the write protect register lock bit (SPU_CTL.WPLCK) is set and the global lock signal is active, writes to the write-protect registers of the SPU are blocked. To re-enable access to the write-protect registers in the SPU, deactivate the global lock signal by writing 0xAD into the SPU_CTL.GLCK bit field.

For security, the SPU provides a set of SPU_SECUREC[n] registers (one for each processor core from Analog Devices) to configure their security settings. The SPU also provides a set of SPU_SECUREP[n] registers (one for each peripheral instance) to configure their security settings.

Enabling and Disabling the SPU

The SPU is always operating. There are no bits to enable or disable the SPU. The SPU configuration can be updated at any time. Any ongoing transactions finish before a new configuration is in effect. By default, the SPU does not write-protect any of the MMRs.

Write-Protecting the SPU

The SPU is treated like any other peripheral in the system. As such, the SPU also has an associated write-protection register. If this write-protection register is configured to block all writes from all requesters, any SPU configuration remains the same until the next system reset.

Checking the Security State

In some cases while running a peripheral, an application system requester does not know whether they are a secure requester generating secure transactions or not. The SPU provides a means for checking the security state of the requester through the SPU_SECURECHK register. When read by a secure requester, the register reads 0xFFFFFFFF and when read by a non-secure requester, the value is 0x00000000.

SPU Mode Configuration

The SPU can provide address range-wide protection by write-protecting the peripherals MMR address range from system MMR requesters. It can also provide register wide protection using global locking. Peripherals that support this feature can enable it in their respective configuration register. When the SPU enables the global lock signal, all subsequent writes to the configuration register of the peripheral are blocked until the global lock signal is deasserted. Similarly, the write-protection registers of the SPU can be write-protected using the global lock signal as well. The SPU uses all these modes of operation together.

Locking Write-Protect Registers

Use the following steps to lock (write-protect) a register.

1. Set the SPU_CTL.WPLCK bit and configure the SPU_CTL.GLCK field to something other than 0xAD.

The SPU write-protect registers are blocked from further write accesses.

Protecting a Peripheral

Use the following procedure to protect a peripheral.

1. Determine which peripheral needs protection and locate the corresponding write-protect register (SPU_WP[n]) in the SPU. See the Write-Protect and Secure Peripheral Registers section.
2. Determine the SMMR requesters from which the peripheral needs protection. Then, set the corresponding bit or bits in the write-protect register (SPU_WP[n]) for the peripheral. See the Write-Protect and Secure Peripheral Registers section.

After setting the write-protect register for the particular peripheral, the identified SMMR requesters are blocked from writing to any MMR in the address space of the peripheral. This block remains in place until the bits in the write-protect register are cleared.

Configuring Security Privileges of a Peripheral

Use the following procedure to configure the security privileges of a peripheral.

1. Determine the peripheral and its corresponding secure peripheral register (SPU_SECUREP[n]) in the SPU. See the Write-Protect and Secure Peripheral Registers section.
2. If the peripheral is to be a secure completer that only accepts secure transactions, set bit 0 (SPU_SECUREP[n].SSEC).
3. If the peripheral is to be a secure requester that generates secure transactions (keeping in mind not all peripherals can be requesters), set bit 1 (SPU_SECUREP[n].MSEC).

This procedure sets the security privileges of a peripheral.

NOTE: Only a secure requester can set security privileges, keeping the chain of trust intact. If a non-secure requester configures the security privileges, it can undermine security protection.

ADSP-2184x SPU Register Descriptions

System Protection Unit (SPU) contains the following registers.

Table 38-3: ADSP-2184x SPU Register List

Name	Description
SPU_CTL	Control Register
SPU_SECURECHK	Secure Check Register
SPU_SECURECTL	Secure Control Register
SPU_SECUREC[n]	Secure Core Registers
SPU_SECUREP[n]	Secure Peripheral Register
SPU_STAT	Status Register
SPU_WP[n]	Write Protect Register n

Control Register

The SPU control register (SPU_CTL) provides a global lock for configuration registers as well as control for locking the write protect (SPU_WP[n]) registers. It also controls the generation of an interrupt to report blocked accesses.

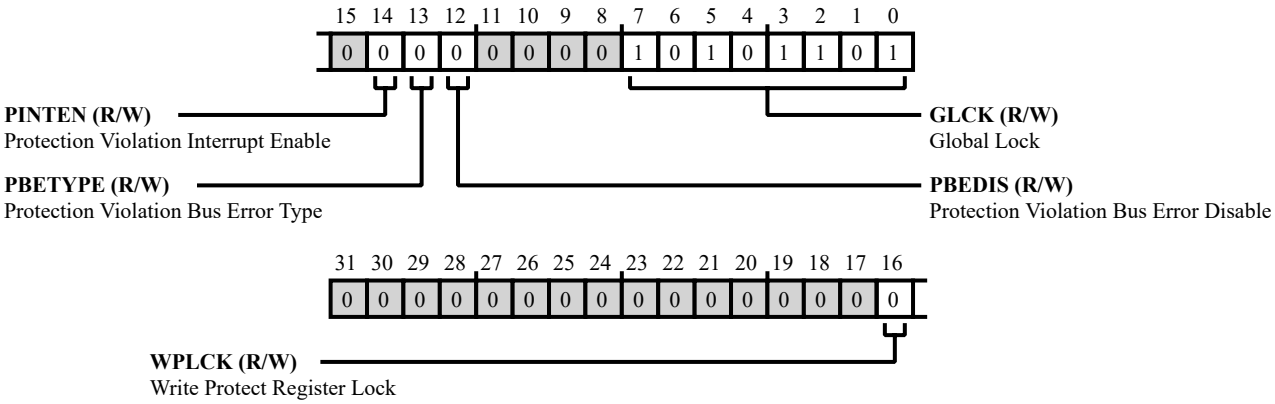


Figure 38-5: SPU_CTL Register Diagram

Table 38-4: SPU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	WPLCK	Write Protect Register Lock. When the SPU_CTL.WPLCK bit is set in combination with the SPU_CTL.GLCK bit, writes to the SPU's write protect registers are blocked and return an error.	
		0	Disable
		1	Enable
14 (R/W)	PINTEN	Protection Violation Interrupt Enable. When the SPU_CTL.PINTEN bit is set (=1), a block of any transaction according to the configured settings produces an interrupt.	
		0	Disable
		1	Enable
13 (R/W)	PBETYPE	Protection Violation Bus Error Type.	
		0	Reserved
		1	Reserved
12 (R/W)	PBEDIS	Protection Violation Bus Error Disable.	
		0	Reserved
		1	Reserved

Table 38-4: SPU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	GLCK	Global Lock. The `SPU_CTL.GLCK` controls the global lock signal. The global lock signal provides register-based write protection. Writing 0xAD to this field disables the lock, and writing any other value enables the lock.

Secure Check Register

The SPU_SECURECHK register reads by secure requesters return 0xFFFFFFFF. Reads by non-secure requesters return 0x00000000.

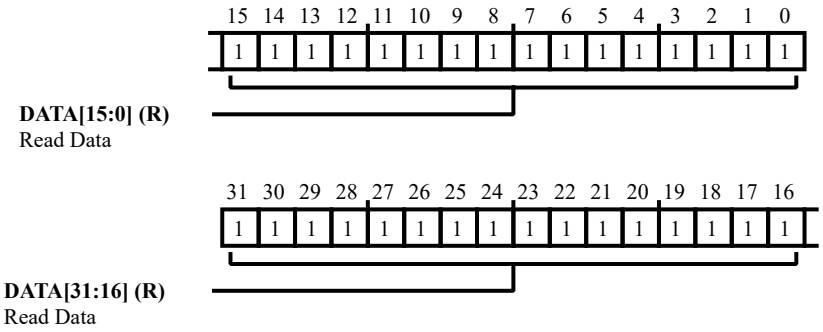


Figure 38-6: SPU_SECURECHK Register Diagram

Table 38-5: SPU_SECURECHK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Read Data. The SPU_SECURECHK.DATA bit field performs reads. Reads by secure requesters return 0xFFFFFFFF. Reads by non-secure requesters return 0x00000000.

Secure Control Register

The SPU Secure Control Register (SPU_SECURECTL) allows the user to lock write access to all the SPU_SECUREC[n] and SPU_SECUREP[n] registers as well as configure the interrupt generation in an event of a security error. It also allows bulk clear of the SSEC bits and/or MSEC bits in the SPU_SECUREP[n] registers.

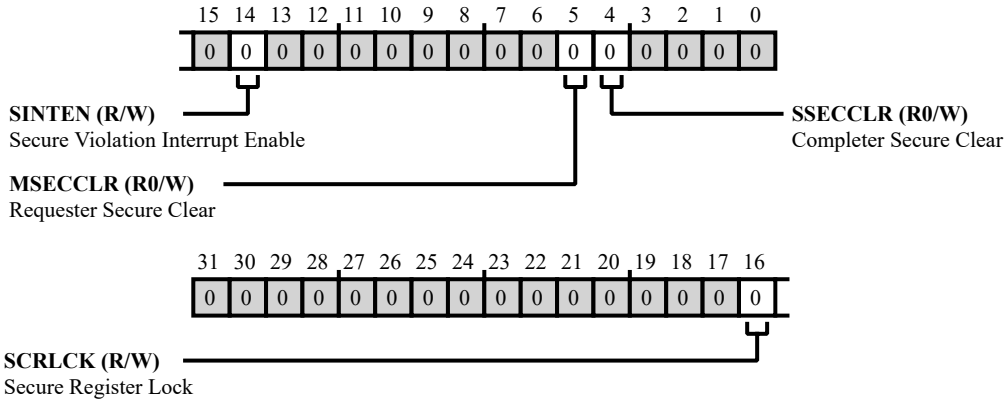


Figure 38-7: SPU_SECURECTL Register Diagram

Table 38-6: SPU_SECURECTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	SCRLCK	Secure Register Lock. When the SPU_SECURECTL.SCRLCK bit is set in combination with the SPU_CTL.GLCK bit, writes to the Security Configuration registers (SPU_SECUREC[n] and SPU_SECUREP[n]) are blocked and return an error which is captured in the SPU_STAT.LWERR bit.	
		0	Disable
		1	Enable
14 (R/W)	SINTEN	Secure Violation Interrupt Enable. The SPU_SECURECTL.SINTEN bit generates an interrupt if a security violation was captured. Interrupt status is provided in the SPU_STAT.VIRQ bit.	
		0	Disable
		1	Enable
5 (R0/W)	MSECCLR	Requester Secure Clear. When the SPU_SECURECTL.MSECCLR bit is set, the SPU_SECUREP[n].MSEC bits in all SPU_SECUREP[n] registers are cleared. The SPU_SECURECTL.MSECCLR bit always reads back as a 0.	
		0	No Action
		1	Clear All Requester Secure Control Bits

Table 38-6: SPU_SECURECTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/W)	SSECCLR	Completer Secure Clear. When the SPU_SECURECTL.SSECCLR bit is set, the SPU_SECUREP[n].SSEC bits in all SPU_SECUREP[n] registers are cleared. The SPU_SECURECTL.SSECCLR bit always reads back as a 0.	
		0	No Action
		1	Clear All Completer Secure Control Bits

Secure Core Registers

A SPU register exists for every DSP core in the system. The SEC field enables or disables security for features in the core.

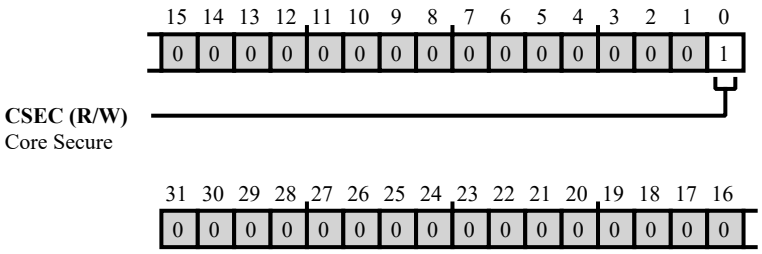


Figure 38-8: SPU_SECUREC[n] Register Diagram

Table 38-7: SPU_SECUREC[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	CSEC	Core Secure. The `SPU_SECUREC[n].CSEC` bit controls whether non-secure accesses are allowed to L1 memory of the processor core. When =1, the core (as a completer) is set as secure meaning only secure transactions are allowed to L1.	
		0	Disable
		1	Enable

Secure Peripheral Register

In the system, each SPU_SECUREP[n] register is assigned to a specific MMR address range associated with one peripheral. Each SPU_SECUREP[n] has a Completer Secure (SSEC) bit and a Requester Secure (MSEC) bit. When the SSEC bit is set, the SPU will only allow secure requesters generating secure transactions to access the peripheral's MMR address space. When the MSEC bit is set, the associated peripheral will be secure and will generate secure transactions.

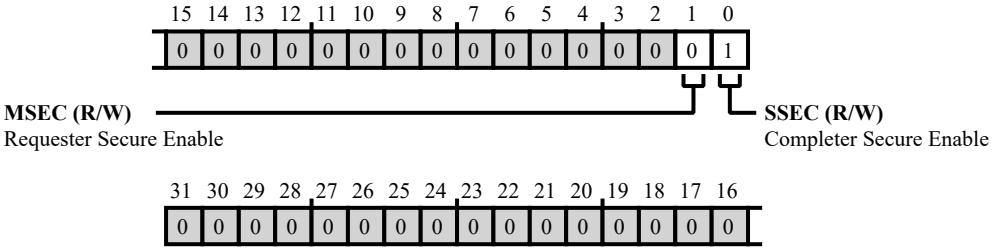


Figure 38-9: SPU_SECUREP[n] Register Diagram

Table 38-8: SPU_SECUREP[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	MSEC	Requester Secure Enable. The SPU_SECUREP[n].MSEC bit controls whether the peripheral generates secure transactions as a requester. When clear (=0), the peripheral generates non-secure transactions as a requester (if applicable). When set (=1), the peripheral generates secure transactions as a requester.	
		0	Disable
		1	Enable
0 (R/W)	SSEC	Completer Secure Enable. The SPU_SECUREP[n].SSEC bit controls whether the peripheral is protected from non-secure transactions. When clear (=0), the security status of the transaction is ignored. When set (=1), only secure transactions are allowed to access the address space of the peripheral and non-secure transactions are blocked.	
		0	Disable
		1	Enable

Status Register

The SPU_STAT register indicates if there have been any errors, active interrupts and global lock status.

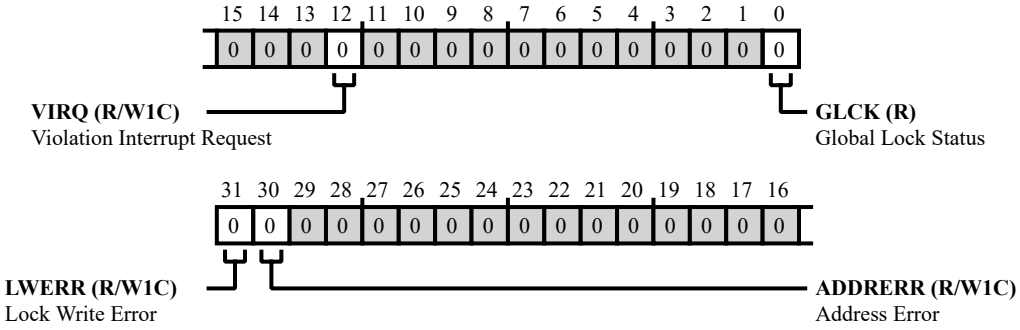


Figure 38-10: SPU_STAT Register Diagram

Table 38-9: SPU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Lock Write Error. The SPU_STAT.LWERR indicates whether there has been an attempted write to a register in the SPU with its lock bit (SPU_CTL.WPLCK or SCRLCK) set while SPU_CTL.GLCK was asserted. This bit is W1C.	
		0	Inactive
		1	Active
30 (R/W1C)	ADDRERR	Address Error. The SPU_STAT.ADDRERR indicates whether there has been an attempted write to a read-only register or an access an invalid address in the SPU MMR address range. This bit is W1C.	
		0	Inactive
		1	Active
12 (R/W1C)	VIRQ	Violation Interrupt Request. The SPU_STAT.VIRQ bit indicates that a security and/or protection violation has been detected and interrupt asserted. This is a W1C bit.	
		0	Inactive
		1	Active
0 (R/NW)	GLCK	Global Lock Status. The SPU_STAT.GLCK indicates whether the global lock is enabled or disabled.	
		0	Disabled (global_lock=0)
		1	Enabled (global_lock=1)

Write Protect Register n

In the system, each SPU_WP[n] register is assigned to a specific MMR address range associated with one peripheral. When the appropriate bits are set, writes to the peripheral from a specific requester are blocked and an error is returned to the requester. For more information, see the processor specific additional information for the SPU_WP[n] register.

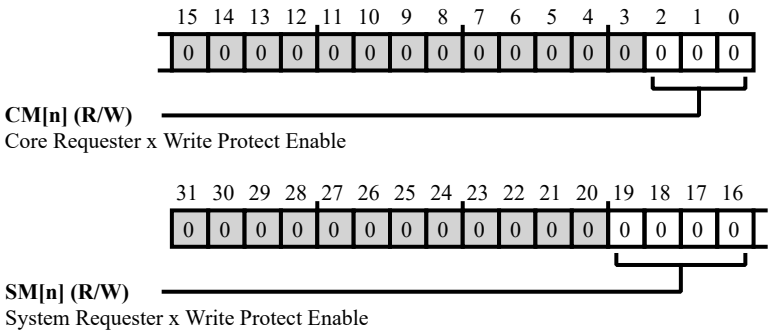


Figure 38-11: SPU_WP[n] Register Diagram

Table 38-10: SPU_WP[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	SM[n]	System Requester x Write Protect Enable. The SPU_WP[n].SM[n] bits correspond to different system requesters in the system. When a particular bit is set in this field, the corresponding system requester cannot write to the corresponding peripheral's MMR address space. The write attempt is blocked by the SPU.
2:0 (R/W)	CM[n]	Core Requester x Write Protect Enable. The SPU_WP[n].CM[n] bits correspond to different cores in the system. When a particular bit is set in this field, the corresponding core cannot write to the corresponding peripheral's MMR address space. The write attempt is blocked by the SPU.

Write-Protect, Secure Peripheral, and Secure Core Registers

The SPU consists of a collection of write-protect registers each of which are associated with a specific peripheral or target. The SPU also has a collection of secure peripheral registers which are also associated with specific peripherals. The table gives the write-protect register and secure peripheral number for each of the peripherals that are provided with write protection and security through the SPU. The SPU for the processor is configured with 136 write-protect registers and also 136 secure peripheral registers. The number corresponding to a peripheral correlates to both the Write-Protect register and the Secure Peripheral register.

For each processor, there are different numbers of controllers that are able to access the SMMR space

The SPU_WP[n] register shows which bits enable the protection against which controller.

For each peripheral, there is a corresponding write-protect register, SPU_WP[n], and secure peripheral register, SPU_SECUREP[n]. The table shows the write-protect register and secure peripheral number for each peripheral.

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
0	Reserved			
1	DMC0_PRG_CDC_BRIDGE	WP1	SECURE1	N/A
2	SPIF GPV SPACE	WP2	SECURE2	N/A
3	LP0	WP3	SECURE3	N/A
4	LP1	WP4	SECURE4	N/A
5	LP0 DDE	WP5	SECURE5	Yes
6	LP1 DDE	WP6	SECURE6	Yes
7	TWI3	WP7	SECURE7	N/A
8	TWI4	WP8	SECURE8	N/A
9	TWI5	WP9	SECURE9	N/A
10	TWI0	WP10	SECURE10	N/A
11	TWI1	WP11	SECURE11	N/A
12	TWI2	WP12	SECURE12	N/A
13	SPORT0A	WP13	SECURE13	N/A
14	SPORT0B	WP14	SECURE14	N/A
15	SPORT1A	WP15	SECURE15	N/A
16	SPORT1B	WP16	SECURE16	N/A
17	SPORT2A	WP17	SECURE17	N/A
18	SPORT2B	WP18	SECURE18	N/A
19	SPORT3A	WP19	SECURE19	N/A
20	SPORT3B	WP20	SECURE20	N/A
21	SPORT4A	WP21	SECURE21	N/A
22	SPORT4B	WP22	SECURE22	N/A
23	SPORT5A	WP23	SECURE23	N/A
24	SPORT5B	WP24	SECURE24	N/A
25	SPORT6A	WP25	SECURE25	N/A
26	SPORT6B	WP26	SECURE26	N/A
27	SPORT7A	WP27	SECURE27	N/A
28	SPORT7B	WP28	SECURE28	N/A
29	UART0	WP29	SECURE29	N/A

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
30	UART1	WP30	SECURE30	N/A
31	UART2	WP31	SECURE31	N/A
32	PORTA	WP32	SECURE32	N/A
33	PORTB	WP33	SECURE33	N/A
34	PORTC	WP34	SECURE34	N/A
35	PORTD	WP35	SECURE35	N/A
36	PORTE	WP36	SECURE36	N/A
37	PORTF	WP37	SECURE37	N/A
38	PORTG	WP38	SECURE38	N/A
39	PORTH	WP39	SECURE39	N/A
41	PADS	WP41	SECURE41	N/A
42	PINT1	WP42	SECURE42	N/A
43	PINT1	WP43	SECURE43	N/A
44	PINT2	WP44	SECURE44	N/A
45	PINT3	WP45	SECURE45	N/A
46	PINT4	WP46	SECURE46	N/A
47	PINT5	WP47	SECURE47	N/A
48	PINT6	WP48	SECURE48	N/A
49	PINT7	WP49	SECURE49	N/A
50	SMC0	WP50	SECURE50	N/A
51	SMPU_SMC	WP51	SECURE51	N/A
52	WDT0	WP52	SECURE52	N/A
53	WDT1	WP53	SECURE53	N/A
54	WDT2	WP54	SECURE54	N/A
55	WDT3	WP55	SECURE55	N/A
56	CNT0	WP56	SECURE56	N/A
57	OTP MMR	WP57	SECURE57	N/A
58	SMPU-OTP	WP58	SECURE58	N/A
59	SWU SMC	WP59	SECURE59	N/A
60	HADC0	WP60	SECURE60	N/A
61	TMU0	WP61	SECURE61	N/A
62	TMR	WP62	SECURE62	N/A

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
63	xSPI	WP63	SECURE63	Yes
64	SPORT0A DDE	WP64	SECURE64	Yes
65	SPORT0B DDE	WP65	SECURE65	Yes
66	SPORT1A DDE	WP66	SECURE66	Yes
67	SPORT1B DDE	WP67	SECURE67	Yes
68	SPORT2A DDE	WP68	SECURE68	Yes
69	SPORT2B DDE	WP69	SECURE69	Yes
70	SPORT3A DDE	WP70	SECURE70	Yes
71	SPORT3B DDE	WP71	SECURE71	Yes
72	SPORT4A DDE	WP72	SECURE72	Yes
73	SPORT4B DDE	WP73	SECURE73	Yes
74	SPORT5A DDE	WP74	SECURE74	Yes
75	SPORT5B DDE	WP75	SECURE75	Yes
76	SPORT6A DDE	WP76	SECURE76	Yes
77	SPORT6B DDE	WP77	SECURE77	Yes
78	SPORT7A DDE	WP78	SECURE78	Yes
79	SPORT7B DDE	WP79	SECURE79	Yes
80	UART0RX DDE	WP80	SECURE80	Yes
81	UART0TX DDE	WP81	SECURE81	Yes
82	UART1RX DDE	WP82	SECURE82	Yes
83	UART1TX DDE	WP83	SECURE83	Yes
84	UART2RX DDE	WP84	SECURE84	Yes
85	UART2TX DDE	WP85	SECURE85	Yes
86	Core MBIST	WP86	SECURE86	Yes
87	MEMREPAIR	WP87	SECURE87	Yes
88	SPI0TX DDE	WP88	SECURE88	Yes
89	SPI0RX DDE	WP89	SECURE89	Yes
90	SPI1TX DDE	WP90	SECURE90	Yes
91	SPI1RX DDE	WP91	SECURE91	Yes
92	SPI2TX DDE	WP92	SECURE92	Yes
93	SPI2RX DDE	WP93	SECURE93	Yes
94	SPI5TX DDE	WP94	SECURE94	Yes

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
95	SPI5RX DDE	WP95	SECURE95	Yes
96	SPI0	WP96	SECURE96	N/A
97	SPI1	WP97	SECURE97	N/A
98	SPI2	WP98	SECURE98	N/A
99	SPI5	WP99	SECURE99	N/A
100	EMAC0	WP100	SECURE100	Yes
101	CAN0	WP101	SECURE102	N/A
102	CAN1	WP102	SECURE102	N/A
103	DDR PFB	WP103	SECURE103	N/A
104	L2CTL0	WP104	SECURE104	N/A
105	L2CTL1	WP105	SECURE105	N/A
107	SMPU L2CTL0 CL2 0	WP107	SECURE107	N/A
108	SMPU L2CTL0 DL2 0	WP108	SECURE108	N/A
109	SMPU L2CTL0 CL2 1	WP109	SECURE109	N/A
110	SMPU L2CTL0 DL2 1	WP110	SECURE110	N/A
111	SMPU L2CTL0 CL2 2	WP111	SECURE111	N/A
112	SEC0	WP112	SECURE112	N/A
113	TRU0	WP113	SECURE113	N/A
114	SPU0	WP114	SECURE114	N/A
115	RCU0	WP115	SECURE115	N/A
116	CGU0	WP116	SECURE116	N/A
117	CGU1	WP117	SECURE117	N/A
118	CDU0	WP118	SECURE118	N/A
119	DPM0	WP119	SECURE119	N/A
120	PLL0	WP120	SECURE120	N/A
121	PLL1	WP121	SECURE121	N/A
122	SWU_L2CTL0_CL2_0	WP122	SECURE122	N/A
123	SWU_L2CTL0_DL2_0	WP123	SECURE123	N/A
124	SWU_L2CTL0_CL2_1	WP124	SECURE124	N/A
125	SWU_L2CTL0_DL2_1	WP125	SECURE125	N/A
126	SWU SMMR	WP126	SECURE126	N/A
127	SWU_L2CTL0_CL2_2	WP127	SECURE127	N/A

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
128	Enhanced BW MDMA	WP128	SECURE128	Yes
129	Max BW MDMA	WP129	SECURE129	Yes
130	Max BW MDMA 1	WP120	SECURE130	Yes
131	MLB0	WP131	SECURE131	Yes
132	SWU DMC0	WP132	SECURE132	N/A
133	SMPU DMC0	WP133	SECURE133	N/A
134	SMPU SPI2	WP134	SECURE134	N/A
135	MEC0	WP135	SECURE135	N/A
136	MEC1	WP136	SECURE136	N/A
137	MEC2	WP137	SECURE137	N/A
138	CRC0	WP138	SECURE138	N/A
139	CRC1	WP139	SECURE139	N/A
140	MDMA0 DDE0 CRC0	WP140	SECURE140	Yes
141	MDMA0 DDE1 CRC0	WP141	SECURE141	Yes
142	MDMA1 DDE0 CRC1	WP142	SECURE142	Yes
143	MDMA1 DDE1 CRC1	WP143	SECURE143	Yes
144	MDMA4 DDE0 CRC2	WP144	SECURE144	Yes
145	MDMA4 DDE1 CRC2	WP145	SECURE145	Yes
146	MDMA5 DDE0 CRC3	WP146	SECURE146	Yes
147	MDMA5 DDE1 CRC3	WP147	SECURE147	Yes
148	SWU SPI2	WP148	SECURE148	N/A
149	MISC REG	WP149	SECURE149	N/A
150	CRC2	WP150	SECURE150	N/A
151	CRC3	WP151	SECURE151	N/A
152	Enhanced BW MDMA 1	WP152	SECURE152	Yes
153	DAI0	WP153	SECURE153	N/A
154	DAI1	WP154	SECURE154	N/A
155	CRYPTO SPE	WP155	SECURE155	Yes
156	CRYPTO PKP	WP156	SECURE156	N/A
157	DLMDMA0	WP157	SECURE157	Yes
158	DAP ROM	WP158	SECURE158	N/A
159	SHARC-FX0 CTI	WP159	SECURE159	N/A

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
160	SHARC-FX0 TRAX	WP160	SECURE160	N/A
161	SHARC-FX0 PMU	WP161	SECURE161	N/A
162	SHARC-FX0 OCD	WP162	SECURE162	N/A
163	SHARC-FX0 Misc	WP163	SECURE163	N/A
164	CSTF	WP164	SECURE164	N/A
165	ETF	WP165	SECURE165	N/A
166	ETR	WP166	SECURE166	N/A
167	TPIU	WP167	SECURE167	N/A
168	CTI Trace	WP168	SECURE168	N/A
169	CTI System	WP169	SECURE169	N/A
170	TSGEN	WP170	SECURE170	N/A
171	A55 ROM	WP173	SECURE173	N/A
172	A55 PE0 Debug	WP171	SECURE171	N/A
173	A55 PE1 Debug	WP172	SECURE172	N/A
174	A55 PE0 PMU	WP174	SECURE174	N/A
175	A55 PE1 PMU	WP175	SECURE175	N/A
176	A55 PE0 CTI	WP176	SECURE176	N/A
177	A55 PE1 CTI	WP177	SECURE177	N/A
178	A55 PE0 ETM	WP178	SECURE178	N/A
179	A55 PE1 ETM	WP179	SECURE179	N/A
180	M85 MISC	WP180	SECURE180	N/A
181	M85 PFB	WP181	SECURE181	N/A
182	M85 SMPU	WP182	SECURE182	N/A
183	XSPI2_PFB	WP183	SECURE183	N/A
184	XSPI2	WP184	SECURE184	MSEC184
185	Reserved			
186	EMMC	WP186	SECURE186	MSEC186
187	HSM	WP187	SECURE187	N/A
188	M85_2 MISC	WP188	SECURE188	N/A
189	M85_2 PFB	WP189	SECURE189	N/A
190	M85_2 SMPU	WP190	SECURE190	N/A
191	FRAC-PLL2	WP191	SECURE191	N/A

No.	Peripheral/Block Name	SWPn Mapping	SECUREPn Mapping	Controller Capable
192	LPDDR4 Ctrl	WP192	SECURE192	N/A
193	LPDDR4 Perf	WP193	SECURE193	N/A
194	HSM_MISC	WP194	SECURE194	N/A
195	LPDDR4 PHY	WP195	SECURE195	N/A

Processor-Specific Information

Global Locking

The global lock signal from the SPU along with the peripheral lock bit can be used to provide lock functionality for the control MMR of the peripheral. The Global Lock (`SPU_CTL.GLCK`) field determines whether global lock is enabled or not. Global Lock is disabled if the `SPU_CTL.GLCK` field is 0xAD (default value), otherwise it is enabled. The following is a list of peripherals that have the global lock bit in their control MMR.

- General-Purpose IO (GPIO)
- System Event Controller (SEC0)
- Trigger Routing Unit (TRU0
- Clock Generation Unit (CGU0)
- Clock Generation Unit1 (CGU1)
- Clock Distribution Unit (CDU0)
- Dynamic Power Management (DPM)
- Reset Control Unit (RCU0)
- System Protection Unit (SPU0)
- L2 Memory Controller (L2CTL0)

39 Security Packet Engine (PKTE)

The PKTE is a security packet engine designed to off-load the host processor to improve the speed of applications requiring cryptographic processing. The packet engine contains a set of modules for encryption and decryption, hashing, and pseudo-random number generation.

PKTE Features

The PKTE has the following features.

- Hardware assisted processing for the cryptographic ciphers, hashes, and pseudo-random number generation
- Header and trailer processing for Internet security protocols
- DMA capability to move data in and out of the engine efficiently and to allow the engine to run autonomously while moving data
- Interrupt controller to signal module status and errors
- Clock manager for enabling or disabling different features to save power

NOTE: Not all algorithms, decrypt, and hash functions and extra features are available on all product models. For complete information on included features, see the product-specific data sheet.

NOTE: This packet engine provides support for various network security protocols by processing headers and trailers as well as accelerating cryptographic functions. Not all processors have direct support for Ethernet. As such, the packet engine can still be used if Ethernet is indirectly used.

PKTE Functional Description

The packet engine contains a set of modules for encryption and decryption, hashing, and pseudo-random number generation. The following sections describe these functional blocks.

ADSP-2184x PKTE Register List

The Security Packet Engine (PKTE) provides security-related features. A set of registers governs PKTE operations. For more information on PKTE functionality, see the PKTE register descriptions.

Table 39-1: ADSP-2184x PKTE Register List

Name	Description
PKTE_ARC4STATE_ADDR	Packet Engine ARC4 State Record Address
PKTE_ARC4STATE_BUF	Starting Entry of 256-byte ARC4 State Buffer
PKTE_BUF_PTR	Packet Engine Buffer Pointer Register
PKTE_BUF_THRESH	Packet Engine Buffer Threshold Register
PKTE_CDRBASE_ADDR	Packet Engine Command Descriptor Ring Base Address
PKTE_CDSC_CNT	Packet Engine Command Descriptor Count Register
PKTE_CDSC_INCR	Packet Engine Command Descriptor Count Increment Register
PKTE_CFG	Packet Engine Configuration Register
PKTE_CLK_CTL	PE Clock Control Register
PKTE_CONT	PKTE Continue Register
PKTE_CTL_STAT	Packet Engine Control Register
PKTE_DATAIO_BUF	Starting Entry of 256-byte Data Input/Output Buffer
PKTE_DEST_ADDR	Packet Engine Destination Address
PKTE_DMA_CFG	Packet Engine DMA Configuration Register
PKTE_ENDIAN_CFG	Packet Engine Endian Configuration Register
PKTE_HLT_CTL	Packet Engine Halt Control Register
PKTE_HLT_STAT	Packet Engine Halt Status Register
PKTE_IMSK_DIS	Interrupt Mask Disable Register
PKTE_IMSK_EN	Interrupt Mask Enable Register
PKTE_IMSK_STAT	Interrupt Masked Status Register
PKTE_INBUF_CNT	Packet Engine Input Buffer Count Register
PKTE_INBUF_INCR	Packet Engine Input Buffer Count Increment Register
PKTE_INT_CFG	Interrupt Configuration Register
PKTE_INT_CLR	Interrupt Clear Register
PKTE_INT_EN	Interrupt Enable Register
PKTE_IUMSK_STAT	Interrupt Unmasked Status Register
PKTE_LEN	Packet Engine Length Register
PKTE_OUTBUF_CNT	Packet Engine Output Buffer Count Register
PKTE_OUTBUF_DECR	Packet Engine Output Buffer Count Decrement Register
PKTE_PE_ALT_KEY_STATUS	PE Alternative Key stat register
PKTE_PE_CACHE_CTRL_0	Packet Engine Cache Control 0

Table 39-1: ADSP-2184x PKTE Register List (Continued)

Name	Description
PKTE_PE_CACHE_CTRL_1	Packet Engine Cache Control 1
PKTE_RDRBASE_ADDR	Packet Engine Result Descriptor Ring Base Address
PKTE_RDSC_CNT	Packet Engine Result Descriptor Count Registers
PKTE_RDSC_DECR	Packet Engine Result Descriptor Count Decrement Registers
PKTE_RING_CFG	Packet Engine Ring Configuration
PKTE_RING_PTR	Packet Engine Ring Pointer Status
PKTE_RING_STAT	Packet Engine Ring Status
PKTE_RING_THRESH	Packet Engine Ring Threshold Registers
PKTE_SA_ADDR	Packet Engine SA Address
PKTE_SA_ARC4IJPTR	ARC4 i and j Pointer Register
PKTE_SA_CMD0	SA Command 0
PKTE_SA_CMD1	SA Command 1
PKTE_SA_IDIGEST[n]	SA Inner Hash Digest Registers
PKTE_SA_KEY[n]	SA Key Registers
PKTE_SA_NONCE	SA Initialization Vector Register
PKTE_SA_ODIGEST[n]	SA Outer Hash Digest Registers
PKTE_SA_RDY	SA Ready Indicator
PKTE_SA_SEQNUM[n]	SA Sequence Number Register
PKTE_SA_SEQNUM_MSK[n]	SA Sequence Number Mask Registers
PKTE_SA_SPI	SA SPI Register
PKTE_SRC_ADDR	Packet Engine Source Address
PKTE_STAT	Packet Engine Status Register
PKTE_STATE_ADDR	Packet Engine State Record Address
PKTE_STATE_BYTE_CNT[n]	State Hash Byte Count Registers
PKTE_STATE_IDIGEST[n]	State Inner Digest Registers
PKTE_STATE_IV[n]	State Initialization Vector Registers
PKTE_USERID	Packet Engine User ID

ADSP-2184x PKTE Interrupt List

Table 39-2: ADSP-2184x PKTE Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
226	PKTE0_IRQ	PKTE0 Interrupt	Level	

PKTE Definitions

Command Descriptor

An 8-word structure that is either written directly into the packet command MMR set or is placed in a Command Descriptor Ring (CDR) in the processor memory. The packet engine sequentially processes the structure. The command descriptor contains the information that varies for every packet. This information includes pointers to the SA record, the state information, the source packet, and the destination packet.

Command Descriptor Ring (CDR)

A circular contiguous portion of memory which is used to manage one or more command descriptions for the packet engine.

Result Descriptor

When the packet engine completes the processing of a packet, it writes a result descriptor with the state information. The result descriptor can be read directly from the result register set or from the Result Descriptor Ring (RDR) in the processor memory.

Result Descriptor Ring (RDR)

A circular contiguous portion of memory which holds the mirror or copy of the CDR but contains the result descriptors. The RDR and CDR can be overlaid on top of each other.

Security Association (SA) Record

A structure that contains the remainder of the information the packet engine requires to process a packet. Most of the information fields in the SA record such as the key and encryption mode are static for the lifetime of the association. The fields do not require frequent manipulation by the processor core. The SA record non-static fields are the sequence number and sequence number mask. The SA record can have a corresponding state record for saving results from the current operations that are useful for future operations. The state record can hold the IV, the hash byte count, and the intermediate hash digest.

Cipher

A method or algorithm to encrypt or decrypt information

Hash

A cryptographic hash is a function that takes an arbitrary block of data and returns a fixed-size bit string. Four main properties define the function:

- It is easy to compute a hash value for any given input
- It is infeasible to generate the original input from a given hash
- It is infeasible to modify the input without changing the resulting hash
- It is infeasible to find two different inputs that result in the same hash

Autonomous Ring Mode (ARM)

Mode of operation in which most of the parameters as well as the data are set up in memory and moved to the engine for configuration and processing through DMA.

Target Command Mode (TCM)

Mode of operation where some parameters are set up in memory and moved into the packet engine through DMA while the other parameters are directly written to the registers. DMA moves the input and output data in and out of the engine.

Direct Host Mode (DHM)

Mode of operation that does not use DMA. All parameters are directly written to and read from the MMRs. The input and output are written to and read from the FIFO buffers.

Cipher Module

The cipher module does the symmetric encrypt or decrypt operations for:

- Data Encryption Standard (DES)
- Triple-DES
- ARC4
- Advanced Encryption Standard (AES) algorithms

The cipher module supports standard modes for DES and AES that include Electronic Code Book (ECB) and Cipher Block Chaining (CBC). The key size for DES is 56 bits, for Triple-DES is 168 bits. The AES module also provides support for AES counter-modes for IPsec and SRTP. All AES modes can use key sizes of 128 bits and 192/256 bits. Key scheduling is automatic and done in parallel with the encrypt or decrypt operation.

Hash Module

The hash module is tightly coupled with the encrypt or decrypt module and provides hardware accelerated one-way hash functions. Operations that combine both hash and encrypt or decrypt functions are provided to reduce processing time for data that needs both applied. For hash-then-decrypt operations, the packet engine performs parallel execution of both functions from the input buffer. For encrypt-then-hash operations, the processing is pipelined from the input buffer to provide minimum latency. An offset can be specified between the start of hashing and the start of encryption to support protocols such as IPsec or SRTP. The HMAC keyed hashing mechanism is supported for MD5, SHA-1, SHA-2-224 and SHA-2-256. The SSL-MAC is supported for MD5 and SHA-1.

Pseudo-Random Number Generator

Cipher algorithms that operate in CBC mode or counter-mode, require an IV. This IV must not be secret; however the IV must be unpredictable and unique for each execution of the encryption process. Pseudo-random number generators are deterministic algorithms that output statistically independent and unbiased numbers. True random number generators are non-deterministic and use the randomness that occurs in a physical process. The packet engine incorporates an ANSI X9.31 compliant Pseudo Random Number Generator (PRNG) that it can use to generate unique IVs using strong encryption. The ANSI X9.31 PRNG is defined as part of the ANSI X9 standards that are used to secure financial transactions. The function can also be used for pseudo-random number generation as part of an implementation of the digital signature standard described in NIST FIPS PUB 186-2.

The PRNG function, as defined by ANSI X9.31, is based on the AES cipher. This section describes the function to promote understanding of the different inputs and outputs of the PRNG function itself.

NOTE: The PRNG in the packet engine is only based on the AES cipher with 128-bit keys. Other ciphers and key lengths are not supported for the PRNG based on ANSI X9.31.

Let $e \times K(Y)$ represent the AES encryption of Y under the key K.

The PRNG function uses three inputs:

- K, a 128-bit key
- V, a 128-bit seed value
- DT, a 128-bit date/ time vector which is updated on each iteration

The intermediate value I is the result of an AES encryption of the data and time vector under key K.

$I = e \times K(DT)$

That value I is then XOR-ed with the seed V and AES encrypted under key K. The result R is the output of the PRNG function.

$R = e \times K(I \text{ XOR } V)$

A new seed value V is generated from the AES encryption of the result R XOR'ed with the intermediate value I under the key K.

$V = e \times K(R \text{ XOR } I)$

The PRNG function is deeply integrated inside the datapath of the packet engine. The function is controlled indirectly through the PRNG mode bits in the PKTE_CTL_STAT.PRNGMD bit field of the command descriptor and the IV source selection bits in the PKTE_SA_CMD0.IVSRC bit field of the SA record.

The PKTE module supports four different modes.

1. Load IV from PRNG for the current operation: PKTE_CTL_STAT.PRNGMD = 0b00 and PKTE_SA_CMD0.IVSRC = 0b11.
2. PRNG init mode initializes the PRNG with a key, seed, and date/time value: PKTE_CTL_STAT.PRNGMD = 0b01.

   Before the PRNG function can be used, it must be initialized with a key, seed, and date/time value. At initialization, the key, seed, and date/time values are programmed. Other PRNG operations do not change the key, however the seed, and date/time values are updated (not re-programmed). The date/time is updated every 128 system clock cycles.

   The date/time is a 128-bit value with randomly distributed number of ones and zeros. It must not be all zeros.

   The ***SA Record for PRNG Init Operation*** table shows how the key, seed and date/time values are loaded into the PKTE registers for initialization.

Table 39-3: SA Record for PRNG Init Operation

Parameter	SA Field	Description
K	SA_KEY0	PRNG key [127:96]
	SA_KEY1	PRNG key [95:64]
	SA_KEY2	PRNG key [63:32]
	SA_KEY3	PRNG key [31:0]
V	SA_IDIGEST0	PRNG seed [127:96]
	SA_IDIGEST1	PRNG seed [95:64]
	SA_IDIGEST2	PRNG seed [63:32]
	SA_IDIGEST3	PRNG seed [31:0]
DT	SA_ODIGEST0	PRNG date/time [127:96]
	SA_ODIGEST1	PRNG date/time [95:64]
	SA_ODIGEST2	PRNG date/time [63:32]
	SA_ODIGEST3	PRNG date/time [31:0]

3. PRNG generate mode generates pseudo-random data on initialized key, seed, and date/time value: PKTE_CTL_STAT.PRNGMD = 0b10.

   The PRNG function can be used to generated pseudo-random data for other purposes than IVs. For this mode, the PRNG must have been initialized once with the PRNG init mode.

The PRNG generate mode uses the initialized key and unique changing date/time value as inputs. The pseudo-random data is output to the output buffer of the packet engine.

The LEN field in the command descriptor indicates the amount of pseudo random data that is generated in multiples of 16 bytes. The maximum is 255 × 16 = 4080 bytes.

In autonomous ring mode, the output data is copied to the host memory at the destination address in the command descriptor. In direct host mode, the host must read the data directly from the output buffer. No SA record is used for this function.

Directly after the PRNG generate mode, a new pseudo-random number is generated and available for the next operation that uses the option PKTE_SA_CMD0.IVSRC = PRNG.

4. PRNG test mode generates pseudo-random data on initialized key, seed, and input (test) data: PKTE_CTL_STAT.PRNGMD = 0b11.

   The PRNG test mode can be used to test the correctness of the PRNG function. This mode is similar to the PRNG generate mode, except that the data is read from the input buffer of the packet engine, instead of the date/time value.

   For this mode, the PRNG must have been initialized once with the PRNG Init mode.

   The LEN field in the command descriptor indicates the amount of pseudo-random data to be generated in multiples of 16 bytes. The maximum is limited to the LEN field in bytes.

   In autonomous ring mode, the output data is copied to the host memory at the destination address in the command descriptor. In the direct host mode, the host must read the data directly from the output buffer. No SA record is used for this function.

   Directly after the PRNG test mode, a new pseudo-random number is generated and available for the next operation that uses the option PKTE_SA_CMD0.IVSRC = PRNG.

Packet Engine Processing Details

This section describes data processing through the packet engine. It describes padding and supported algorithms for each protocol.

A valid Security Association (SA) must be created before packet processing can start. A formatted SA record must reside in memory and be accessible to the packet engine. The host processor application is responsible for these tasks.

Crypto Padding

Padding is the process of adding data to fill-out a fixed-size plain text data structure. Three factors determine when to use a pad field:

1. If a block cipher encryption algorithm is used, a pad field is used to expand the plain text to a multiple of the block size.
2. Padding can be used to ensure that the cipher text terminates on an n-byte boundary.

3. Padding can conceal the actual length of the payload.

To facilitate peak encrypt or decrypt performance, the packet engine supports the following most commonly used padding functions in hardware:

1. Pad generation and insertion of pad bytes to the end of plain text prior to encryption, for outbound operations.
2. Pad verification to check for correct padding after decrypting a packet for inbound operations.
3. Pad consumption to strip the pad bytes from the plain text data after decrypting a packet, for inbound operations.

Pad Generation and Insertion

The pad type and quantity of bytes the packet engine inserts depends on the plain text length and the value of the following fields:

- `PKTE_SA_CMD0.PADTYPE` and `PKTE_SA_CMD0.EXTPAD` defines the type of padding
- `PKTE_CTL_STAT.PADVAL` defines a value that is inserted in the pad
- `PKTE_SA_CMD0.CIPHER` enforces a certain pad alignment
- `PKTE_SA_CMD1.CIPHERMD` enforces a certain pad alignment
- `PKTE_SA_CMD0.SCPAD` allows stream ciphers to be padded
- `PKTE_CTL_STAT.PADCTLSTAT` controls the pad alignment

The `PKTE_CTL_STAT.PADCTLSTAT` bit field of the result descriptor returns the total number of inserted pad bytes.

Pad Types

The pad type bit field (`PKTE_SA_CMD0.PADTYPE`) and the extended pad bit (`PKTE_SA_CMD0.EXTPAD`) select the pad type for the extended protocol group. The packet engine can generate different pad types in hardware as described in the ***Pad Examples*** table.

The `PKTE_CTL_STAT.PADVAL` bit field, together with the number of pad bytes, defines the value that is inserted in the pad. The format of the pad and the use of this field is best explained in an example (see the ***Pad Examples*** table).

For the IPsec pad type, this field holds the value that is inserted into the next header field (in the ESP trailer) of the innermost operation's header. For the Constant pad type or the Constant SSL pad type, this field holds the inserted fixed constant pad value. For all other pad types, this field is not used and must be zero.

Table 39-4: Pad Types

Pad Type	Value	Description
IPSec	0b000	Append 0 to 255 pad bytes, followed by a pad length byte and a next header byte. The first pad byte appended to the plain text is numbered 1, with subsequent pad bytes making up a monotonically increasing sequence: 1, 2, 3 and up. Append the pad length field that indicates the number of pad bytes (0–255), where a value of zero indicates no pad bytes present. Append the next header byte as specified in the `PKTE_CTL_STAT.PADVAL` field of the command descriptor. A minimum of 2 bytes are appended; zero pad bytes plus the pad length byte plus the next header byte, in which case the `PKTE_CTL_STAT.PADCTLSTAT` field in the result descriptor returns 0x02. A maximum of 257 bytes can be appended, in which case the `PKTE_CTL_STAT.PADCTLSTAT` field in the descriptor returns 0x01.
PKCS#7	0b001	Appends 1–128 pad bytes with a pad byte value equal to the pad length (1–128).
Constant	0b010	Appends 0–255 pad bytes of a user-specified character to the plain text data. This character is specified in the `PKTE_CTL_STAT.PADVAL` field of the command descriptor.
Zero	0b011	Appends 0–255 pad bytes of 0x00 to the plain text data.
Constant SSL	0b110	Appends 0–255 pad bytes of a user-specified character to the plain text data, followed by a 'pad length' byte (0–255). This character is specified in the `PKTE_CTL_STAT.PADVAL` field of the command descriptor. A total of 256 bytes can be appended, in which case the pad field returns 0x00.

For example, the ***Pad Examples*** table shows the appended pad for any of the pad types for an outbound (encrypt) operation. The table shows a plain text input of 2 bytes using the 8-byte block cipher crypto-algorithm DES-ECB and a `PKTE_CTL_STAT.PADVAL` field value of 0xAA.

Table 39-5: Pad Examples

Pad Type	Pad field (extended to Plain Text)	`PKTE_CTL_STAT`
IPSec	0x01, 0x02, 0x03, 0x04, 0x04, 0xAA	0x06
PKCS#7	0x06, 0x06, 0x06, 0x06, 0x06, 0x06	0x06
Constant	0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA	0x06
Zero	0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x06
Constant SSL	0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0x05	0x06

Pad Length

The ***Pad Alignment*** table lists the alignment (boundaries) to which the packet engine will pad, based on:

- The selected crypto algorithm
- Crypto mode
- The value of the pad stream cipher bit
- The value of pad control

The minimum number of inserted pad bytes depends on the cipher algorithm, selected using the PKTE_SA_CMD0.CIPHER bit field, and the cipher mode, selected using the PKTE_SA_CMD1.CIPHERMD bit field.

For block ciphers, the plain text data is always (as a minimum) padded to the next block boundary. More pad bytes beyond the algorithm or protocol alignment requirements can be inserted using the pad control (PKTE_CTL_STAT.PADCTLSTAT) in the command descriptor. This feature can be used for *traffic flow security* to conceal the number of plain text bytes in an encrypted packet.

Encrypt operations that use block ciphers have minimum pad requirements based on their block size. The packet engine enforces a minimum pad alignment for block ciphers according to the ***Pad Alignment*** table. For ESP outbound operations, the minimum pad alignment is forced to 4 bytes.

For stream ciphers and null crypto the data is never padded when the stream cipher pad bit (PKTE_SA_CMD0.SCPAD) =0. When PKTE_SA_CMD0.SCPAD =1 the plain text data is padded to the length as defined by the PKTE_CTL_STAT.PADCTLSTAT bits in the command descriptor.

NOTE: The SSL protocol does not allow padding to exceed the ciphers block length. This length is 8 bytes for DES/Triple-DES and 16 bytes for AES. For SSL, the packet engine does not enforce this pad alignment value. The host processor must ensure that the PKTE_CTL_STAT.PADCTLSTAT bit field is configured correctly.

Table 39-6: Pad Alignment

Pad Control **PADCTLSTAT = PKTE_CTL_STAT[31:24]**			**0x00**	**0x01*1**	**0x02**	**0x04**	**0x08**	**0x10**	**0x20**	**0x40**	**0x80*2**
Crypto Algorithm	**Crypto Mode**	**Pad Stream Ciphers**	**%8**	**%1**	**%4**	**%8**	**%16**	**%32**	**%64**	**%128**	**%256**
DES	ECB, CBC	N/A	8	8	8	8	16	32	64	128	256
AES	ECB, CBC	N/A	16	16	16	16	16	32	64	128	256
	CTR, ICM with ESP	N/A	8	4	4	8	16	32	64	128	256
	CTR, ICM no ESP	no	0	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256
NULL	with ESP	N/A	8	4	4	8	16	32	64	128	256
	no ESP	no	8	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256
ARC4	with ESP	N/A	8	4	4	8	16	32	64	128	256
	no ESP	no	0	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256

*1 When PKTE_CTL_STAT.PADCTLSTAT is configured for no padding (0x01), it does not mean that no padding bytes are inserted. When PKCS#7 padding is selected, a pad length field with a value =1 is inserted. When SSL or TLS padding is selected, a pad length field with a value =0 is inserted. When IPsec padding is selected, a pad length field is forced to (PKTE_CTL_STAT.PADCTLSTAT=0x20). When zero pad and constant pad are selected, no pad bytes are inserted.

*2 Pad type PKCS#7 supports a maximum length of 128 pad bytes, so the packet engine overrules a 256-byte alignment (PKTE_CTL_STAT.PADCTLSTAT=0x80) to a 128-byte boundary (PKTE_CTL_STAT.PADCTLSTAT=0x40).

The packet engine does not constrain the pad type that is used for an operation; any pad type can be used for each operation. The user must be aware that some protocol specifications only allow specific pad types. The SRTP specification does not have padding defined as padding performed by RTP. The host must pad the RTP packet.

The host software can implement padding that is not supported in hardware. In this case, the host must select the *zero pad* type and set the PKTE_CTL_STAT.PADCTLSTAT bit field in the packet descriptor to zero (no padding). The hardware padding engine does not add any bytes. When using a block cipher, the host must insert pad bytes. Then, the data to be encrypted (plain text and pad bytes) are a multiple of the block ciphers boundary. For stream ciphers, any number of pad bytes can be added.

Pad Verification and Consumption

The packet engine can validate a pad type against the expected values. The value of the following bits controls the pad verify function:

- PKTE_SA_CMD0.PADTYPE and PKTE_SA_CMD0.EXTPAD define the type of padding
- PKTE_SA_CMD0.CIPHER enforces a certain pad alignment
- PKTE_SA_CMD1.CIPHERMD enforces a certain pad alignment
- PKTE_SA_CMD0.SCPAD allows stream ciphers to be padded

When packet processing is complete, the status byte in the first word of the result descriptor reports the pad verification status. Refer to the Table 39-28 Extended Error Codes - Status Encoding table.

The PKTE_CTL_STAT.PADCTLSTAT bits in the first word of the result descriptor return the total number of detected pad bytes and returns zero for a pad verify error.

When the IPsec pad type is selected, the PKTE_CTL_STAT.PADVAL bit field of the result descriptor returns the next header field. For IPsec ESP outbound operations, this field returns the decimal value 50. For IPsec inbound operations and basic inbound operations that use the IPsec pad mode, the packet engine returns the next header field it detects on the header of the innermost operation. This value is typical for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination option headers, another header value could be seen. For all other inbound operations, the returned pad value is zero.

Pad verification is performed for inbound (decrypt) operations that use IPsec, TLS/DTLS or PKCS#7 pad type:

- In combination with a block cipher algorithm: DES-ECB, DES-CBC, AES-ECB, AES-CBC,
- In combination with a stream cipher and stream cipher padding PKTE_SA_CMD0.SCPAD enabled: AES-CTR, AES-ICM and ARC4
- In combination with null-crypto (no encryption).

Pad Types

The PKTE_SA_CMD0.PADTYPE bit field and the extended pad PKTE_SA_CMD0.EXTPAD bit selects the pad type for the extended protocol group pad type. The packet engine can verify the different pad types in hardware as described in the ***Pad Types*** table.

The constant and zero pad types are not verified since they do not include a pad length field. The SSL pad type is not verified since it does not have a defined pattern.

Table 39-7: Pad Types

Pad Type	SA_CMD0[18, 7:6]	Description
IPSec	0b000	Verify that the pad field includes 0–255 pad bytes, followed by a correct pad length and a next header byte. Verify that pad bytes appended to the plain text are an incremental count, starting at one. Verify that the pad length field is the number of pad bytes (0–255), where a value of zero indicates no pad bytes present. Verify that a next header byte is present as the last byte of the packet, the value is not verified. This is after removal of the ICV. The total number of detected pad bytes is returned in the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor. **NOTE:** A minimum of 2 bytes must be present, zero pad bytes plus the pad length byte plus the next header byte. In this case the PKTE_CTL_STAT.PADCTLSTAT field in the descriptor returns 0x02. A maximum of 257 bytes can be present, in which case the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor returns 0x01. The value of the next header byte is returned in the PKTE_CTL_STAT.PADVAL field in the result descriptor.
PKCS#7	0b001	Verify that the pad field includes 1–128 pad bytes, with a pad byte value equal to the pad length (1–128).

When a block cipher is used and the payload is not padded to the nearest block size boundary, as required by the protocol, a *block size error* is generated for all pad types.

Pad Consumption

The packet engine can optionally consume the decrypted pad bytes for an inbound operation that uses the IPsec, SSL, TLS/DTLS, or PKCS#7 pad types. The pad types constant and zero are not consumed since they do not include a pad length field.

Pad consumption (or stripping) is selected on a flow-by-flow basis in the SA-record with the PKTE_SA_CMD1.CPYPAD bit. When this bit is set, the length returned in the LEN field of the result descriptor is the total length of the plain text including the pad. When the PKTE_SA_CMD1.CPYPAD is disabled, the detected pad length, as returned in the PKTE_CTL_STAT.PADCTLSTAT bits, is subtracted from the total length and then returned in the LEN field of the result descriptor.

The pad is always written to the result packet buffer in memory. When the PKTE_SA_CMD1.CPYPAD bit is disabled, only the result length is corrected.

Crypto and Hash Algorithms

The packet engine supports a wide range of crypto and hash algorithms to accelerate basic operations and protocol operations. These algorithms are:

- Basic Encrypt and Basic Decrypt Operations
- Basic Hash Operations
- Basic Encrypt-Hash and Basic Hash-Decrypt Operations
- IPSec ESP Operations
- SRTP Operations

The following tables provide allowed algorithm combinations. Those algorithms not listed in the tables are invalid and can give unexpected results.

NOTE: Not all crypto and hash algorithms are available on all product models. For information on algorithm availability, see the product-specific data sheet.

Table 39-8: Algorithms for Basic Encrypt and Basic Decrypt Operations

Crypto Algorithm	Crypto Mode
DES, Triple-DES	ECB, CBC
AES	ECB, CBC, CRT, ICM
NULL	—

Table 39-9: Algorithms for Basic Encrypt and Basic Decrypt Operations

Crypto Algorithm	Crypto Mode
SHA-1	Basic hash, HMAC, SSL-MAC
SHA-224	Basic hash, HMAC
SHA-256	Basic hash, HMAC
NULL	—

Table 39-10: Algorithms for Basic Encrypt-Hash and Basic Hash-Decrypt Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	ECB, CBC	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC
		NULL	—

Table 39-10: Algorithms for Basic Encrypt-Hash and Basic Hash-Decrypt Operations (Continued)

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	ECB, CBC, CRT, ICM	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC
		NULL	—
NULL	—	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC

Table 39-11: Algorithms for IPsec ESP Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
AES	CBC, CTR	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
NULL	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	—

Table 39-12: Algorithms for Basic SSL and Extended SSL Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—

Table 39-12: Algorithms for Basic SSL and Extended SSL Operations (Continued)

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	CBC	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—
ARC4	Stateful	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—
NULL	—	SHA-1	SSL-MAC
	—	MD5	SSL-MAC
	—	NULL	—

Table 39-13: Algorithms for Basic TLS, Extended TLS and DTLS Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
AES	CBC, CTR	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
ARC4[1]	Stateful	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
NULL	—	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—

1 Only for Basic TLS and Extended TLS

Table 39-14: Algorithms for SRTP Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	ICM	SHA-1	HMAC
NULL	—	SHA-1	HMAC

IV Processing

An initialization vector (IV) is necessary to start a cipher stream or a block cipher in any of the streaming modes of operation. The IV must be unique for each packet. The IV ensures that all cipher texts are unique even if produced by the same encryption key. This functionality prevents every packet from needing a unique encryption key.

Depending on the packet engine operation, the IV can be loaded from different sources. The IV format depends on the algorithm and the source of the IV. The ***Format of the IV*** table provides an overview of all IV formats.

Table 39-15: Format of the IV

Algorithm	IV Source (`PKTE_SA_CMD0.IVSRC`)	Format	IV Offset (`PKTE_SA_CMD1.HSHCOFFST`)
DES/Triple-DES (CBC)	Previous Result of IV	Internal IV register [63:0]	0x00
	Input Buffer	Input buffer [63:0]	0x02
	Saved IV	State Record Saved IV [63:0]	0x00
	Automatic	PRNG output [63:0]	0x00
AES (CBC)	Previous Result of IV	Internal IV register [127:0]	0x00
	Input Buffer	Input Buffer [127:0]	0x04
	Saved IV	State Record Saved IV [127:0]	0x00
	Automatic	PRNG output [127:0]	0x00
AES (ICM) for Basic and SRTP	Input Buffer	Input Buffer [127:0]	0x04
	Saved IV	State Record Saved IV [127:0]	0x00
AES (CTR) for Basic and IP-sec	Input Buffer	SA_NONCE || Input Buffer[63:0] || 0x00000001	0x02
	Saved IV	State Record Saved IV [127:0]	0x00
	Automatic	SA_NONCE || PRNG output [95:32] || 0x00000001	0x00

Notes:

1. The `PKTE_SA_CMD1.HSHCOFFST` bit field provides the IV offset. The offset is only applicable for basic hash or encrypt operations. For protocol operations, the offset is automatically enforced.

2. AES-CTR: The Nonce value as described in RFC 3686, is mapped to the PKTE_SA_NONCE register. This Nonce value remains constant for the lifetime of the security association.
3. The host processor controls the IV update using the save-IV bit (PKTE_SA_CMD0.SVIV). When part of a packet is processed using a stream cipher and the encrypt or decrypt data is not an integer multiple of the block size, the saved IV is invalid. It must not be used to resume processing the packet.
4. The packet engine supports automatic IV generation for outbound operations. A new IV is generated for every packet with the internal PRNG. This automatic IV generation can be used for all DES, Triple-DES, and AES modes that use an IV, except for AES-ICM mode.
5. For outbound operations, automatic IV generation is the most efficient. No additional I/O is required, and the host processor does not need to provide the IV. When the saved IV option supplies the IV, the IV must be changed for each packet sent. This activity happens when the packet engine writes back the IV to the state record after processing.
6. Outbound IPsec operations put the IV explicitly at the front of the packet. For an inbound IPsec operation, loading from the input buffer is most efficient and always used.
7. CBC processing must not use a predictable IV. Do not use the saved IV and previous result IV options for CBC processing. Refer to RFC 3602 for more details.

ARC4 Processing

The ARC4 algorithm supports two modes of operation: stateless and stateful. For SSL, TLS and DTLS operations that use the ARC4 algorithm, the mode must be set to stateful.

Stateless Mode. Each packet is processed with a newly initialized ARC4 key taken from the key field of the SA record. In this mode, the state information from the SA is never to be read.

CAUTION: When an ARC4 operation in stateless mode is interrupted by an *Interface Error* and the ARC4 state building process is aborted, the next packet fails. The ARC4 is not reset between two packets. For the next packet the ARC4 proceeds from the internal state that it was aborted. A soft reset *does not* reset the ARC4 internal state, a hardware reset is required.

Stateful Mode. When the PKTE_CTL_STAT.INITARC4 bit =1, the ARC4 algorithm initializes using the key specified in the SA record. When the PKTE_CTL_STAT.INITARC4 bit =0, the ARC4 context is read from the ARC4 state field of the SA record and the*i* and *j* pointer field of the SA record. The encrypt and decrypt processing continues from this algorithm state.

The packet engine supports ARC4 key sizes of 40 bits to 128 bits. Longer keys can be used, but they cannot be made inside the packet engine.

The host processor applies ARC4 key scheduling function to the s-box and puts the 256-byte result into the ARC4 state record. It writes the *i* and *j* pointers in the SA (initial $i = 1$, $j = 0$). The host programs the packet engine and specifies stateful operation to continue the ARC4 algorithm.

Hash State Loading

The hash state can be loaded from various sources, depending on the selected protocol and hash algorithm. The ***Different Sources for Loading the Hash State*** table provides a list of all the options.

Table 39-16: Different Sources for Loading the Hash State

Hash Algorithm **PKTE_SA_CMD0.HASHSRC =**	From SA 0b00	RESERVED 0b01	From State 0b10	No Load 0b11
SHA-1	yes	-	yes	Yes
SHA-224	Yes	-	Yes	Yes
SHA-256	Yes	-	Yes	Yes
Null hash	x	x	x	x

Sequence Number Processing

The packet engine supports sequence number generation and verification for IPsec and extended SSL, TLS, and DTLS protocol operations.

A Sequence Number (SN) is an unsigned number that a sender must implement, and a receiver can use to support anti-replay service (replay attacks) for a specific SA. This processing includes detection of the same packet that arrives more than once and detection of packets that arrive in an incorrect sequence and is outside an accepted level of order relative to the last received packet.

The packet engine supports the following options.

- Sequence number loaded from SA for outbound operations and is retrieved from the input packet for inbound operations.

The sequence number and sequence number mask fields are part of the SA record.

Sequence Number Processing in Extended SSL/TLS

SSL and TLS use an implicit sequence number of 64 bits that is not send in the packet but part of the hash.

For inbound operations, no sequence number verification is performed; instead, an incorrect sequence number results in an authentication error (`PKTE_CTL_STAT.AUTHERR`).

For outbound operations, with `PKTE_SA_CMD0.HDRPROC` enabled and `PKTE_SA_CMD1.ENSQNCHK` enabled, the packet engine generates a sequence number error when the 64-bit sequence number counter overflows (counter is 2^{64}-1 and increments to 0). The host processor must not send the packet.

For both outbound and inbound operations, with `PKTE_SA_CMD0.HDRPROC` enabled and `PKTE_SA_CMD1.ENSQNCHK` enabled, the packet engine reads the sequence number from the `PKTE_SA_SEQNUM[n]` registers in the SA. When the operation is finished, the packet engine stores the incremented sequence number in the same SA fields. The sequence number mask `PKTE_SA_SEQNUM_MSK[n]` registers in the SA are not used.

For both outbound and inbound operations, with PKTE_SA_CMD0.HDRPROC disabled, the packet engine does not increment the sequence number. It expects the host to provide the sequence number through the input buffer as part of the header.

Sequence Number Processing in DTLS

DTLS uses an explicit sequence number of 64 bits that is sent in the packet. The DTLS sequence number is composed of the epoch (16 bits) and packet sequence number (48 bits) that together form the 64-bit number, like the TLS sequence number. The epoch is incremented after each *Change Cipher Spec* message. The packet sequence number is incremented per packet starting from zero after each change cipher spec message.

For outbound operations, with PKTE_SA_CMD0.HDRPROC enabled, the packet engine reads the sequence number from the PKTE_SA_SEQNUM[n] fields in the SA, then inserts the sequence number in the packet. Then the packet engine stores the incremented sequence number in the PKTE_SA_SEQNUM[n] fields in the SA. The sequence number mask PKTE_SA_SEQNUM_MSK[n] fields in the SA are not used.

For outbound operations, with PKTE_SA_CMD0.HDRPROC enabled and PKTE_SA_CMD1.ENSQNCHK enabled, the packet engine generates a sequence number error when the 48-bit sequence number counter overflows (counter is $2^{48} - 1$ and increments to 0). The host must not send the packet.

For outbound operations, with PKTE_SA_CMD0.HDRPROC disabled or PKTE_SA_CMD1.ENSQNCHK disabled, the packet engine does not increment and verify a sequence number counter overflow, and therefore never generates a sequence number error. With PKTE_SA_CMD0.HDRPROC disabled the Packet Engine does not update the sequence number and sequence number mask fields in the SA and expects the host to provide the sequence number through the input buffer as part of the header.

For inbound operations, with PKTE_SA_CMD0.HDRPROC enabled and PKTE_SA_CMD1.ENSQNCHK enabled, the packet engine verifies the PKTE_SA_SEQNUM[n] fields against the sequence number in the packet using the PKTE_SA_SEQNUM_MSK[n] from the SA. Three situations can occur:

1. If the received sequence number falls outside and above the 64-bit sequence number mask, the mask is shifted. The packet engine updates the SA with the received sequence number and the shifted sequence number mask.
2. If the received sequence number falls inside the 64-bit sequence number mask and is not a duplicate sequence number (the same as received before), the corresponding bit in the mask is set. The packet engine updates the SA with the received sequence number and the updated sequence number mask.
3. If the received sequence number falls outside the 64-bit sequence number mask or matches and earlier received number a sequence number error is generated. The packet engine does not update the sequence number and sequence number mask fields in the SA. The host must discard the packet.

For inbound operations, with PKTE_SA_CMD0.HDRPROC disabled or PKTE_SA_CMD1.ENSQNCHK disabled, the packet engine does not verify the sequence number against the sequence number in the packet and therefore never generates a sequence number error.

With PKTE_SA_CMD0.HDRPROC disabled the packet engine does not update the sequence number and sequence number mask fields in the SA and expects the host to provide the sequence number through the input buffer as part of the header.

The following tables provide details of sequence number processing.

Table 39-17: Sequence Number Generation and Verification Control (Outbound)

Header Processing SA_CMD0[19]	Anti-Replay Service SA_CMD1[29]	Description
Outbound		
1	1	Sequence number generation (increment)
1	1	Sequence number overflow check (2^{32}–1) to zero for IPsec, (2^{48}–1) to zero for DTLS and (2^{64}–1) to zero for Extended SSL/TLS.
1	0/1	Sequence number update in SA
0/1	0/1	Extended sequence number update in SA

Table 39-18: Sequence Number Generation and Verification Control (Inbound)

Header Processing SA_CMD0[19]	Anti-Replay Service SA_CMD1[29]	Description
Inbound		
1	1	Sequence number verification (check against sequence number and sequence number mask in SA). Applicable only for IPsec and DTLS.
1	0/1	Sequence number update in SA, except on authentication error
1	1	Sequence number mask update in SA. Applicable only for IPsec and DTLS.
0/1	0/1	Extended sequence number update in SA

Table 39-19: Header Processing Enabled, Anti-replay Service Enabled

	IPsec ESP	Ext. SSL	Ext. TLS	DTLS
Header processing enabled, anti-replay service enabled				
Inbound				
Initial value in the SA SA_SEQNUM1[31:0] SA_SEQNUM0[31:0] SA_SEQNUM_MSK	B3 B0 0x00000000 0x00000000 not used	B3 B0 0x00000000 0x00000000 not used	B3 B0 0x00000000 0x00000000 not used	B3 B0 0xAA550000 0x00000000 not used
Value in the first packet or hashbyte stream, highest byte is B0	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0x00000000 B4 B7 0x00000000	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0xAA550000 B4 B7 0x00000001

Table 39-19: Header Processing Enabled, Anti-replay Service Enabled (Continued)

	IPsec ESP	Ext. SSL	Ext. TLS	DTLS
Header processing enabled, anti-replay service enabled				
Inbound				
Initial value in the SA after first packet SA_SEQNUM1[31:0] SA_SEQNUM0[31:0] SA_SEQNUM_MSK	B3 B0 0x00000000 0x00000001 not used	B3 B0 0x00000000 0x00000001 not used	B3 B0 0x00000000 0x00000001 not used	B3 B0 0xAA550000 0x00000001 not used
Initial value in the SA SA_SEQNUM1[31:0] SA_SEQNUM0[31:0] SA_SEQNUM_MSK	B3 B0 0xFEDCBA98 0x76543210 not used	B3 B0 0xFEDCBA98 0x76543210 not used	B3 B0 0xFEDCBA98 0x76543210 not used	B3 B0 0xAA55BA98 0x76543210 not used
Value in the first packet or hashbyte stream, highest byte is B0	B0 B3 0xFEDCBA98 B4 B7 0x76543211	B0 B3 0xFEDCBA98 B4 B7 0x76543211	B0 B3 0xFEDCBA98 B4 B7 0x76543211	B0 B3 0xAA55BA98 B4 B7 0x76543211
Initial value in the SA after the packet SA_SEQNUM_1[31:0] SA_SEQNUM_0[31:0] SA_SEQNUM_MASK	B3 B0 0xFEDCBA98 0x76543211 not used	B3 B0 0xFEDCBA98 0x76543211 not used	B3 B0 0xFEDCBA98 0x76543211 not used	B3 B0 0xAA55BA98 0x76543211 not used
Highest value before overflow SA_SEQNUM_1[31:0] SA_SEQNUM_0[31:0] SA_SEQNUM_MASK	B0 B3 0xFFFFFFFF 0xFFFFFFFF not used	B0 B3 0xFFFFFFFF 0xFFFFFFFF not used	B0 B3 0xFFFFFFFF 0xFFFFFFFF not used	B0 B3 0xAA55FFFF 0xFFFFFFFF not used
Value in the packet or hashbyte stream, highest byte is B0	B0 B3 0x00000000B4 B7 0x00000000	B0 B3 0xFFFFFFFF B4 B7 0xFFFFFFFF	B0 B3 0xFFFFFFFF B4 B7 0xFFFFFFFF	B0 B3 0xAA55FFFF B4 B7 0xFFFFFFFF
Value after overflow in SA SA_SEQNUM_1[31:0] SA_SEQNUM_0[31:0] SA_SEQNUM_MASK	B0 B3 0x00000000 0x00000000 not used	B0 B3 0x00000000 0x00000000 not used	B0 B3 0x00000000 0x00000000 not used	B0 B3 0xAA550000 0x00000000 not used

Table 39-19: Header Processing Enabled, Anti-replay Service Enabled (Continued)

	IPsec ESP	Ext. SSL	Ext. TLS	DTLS
Header processing enabled, anti-replay service enabled				
Inbound				
Initial value in SA	B3 B0	B3 B0	B3 B0	B3 B0
SA_SEQNUM1[31:0]	0x00000000	0x00000000	0x00000000	0xAA550000
SA_SEQNUM0[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
SA_SEQNUM_MSK1[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
SA_SEQNUM_MSK0[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
Expected value in the first packet or hash-byte stream, highest byte is B0	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0xAA550000 B4 B7 0x00000001
Value in SA after first packet	B3 B0	B3 B0	B3 B0	B3 B0
SA_SEQNUM1[31:0]	0x00000000	0x00000000	0x00000000	0xAA550000
SA_SEQNUM0[31:0]	0x00000001	0x00000001	0x00000001	0x00000001
SA_SEQNUM_MSK1[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
SA_SEQNUM_MSK0[31:0]	0x00000001	0x00000001	0x00000001	0x00000001

PKTE Block Diagram

The ***PKTE Block Diagram*** shows the functional blocks within the PKTE.

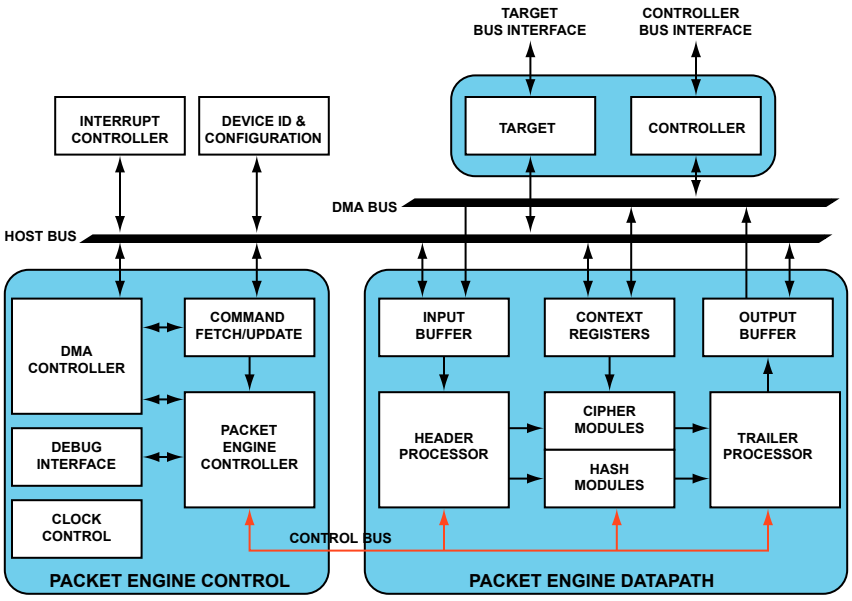


Figure 39-1: PKTE Block Diagram

PKTE Architectural Concepts

The following descriptions provide details on the functional blocks within the security packet engine.

Packet Engine

The packet engine contains symmetric cipher and hash engines. It is optimized to off-load intensive cryptographic operations from the host processor. It can perform parallel and pipelined encryption and hashing operations, reducing the latency, and processing time for packets that need both operations applied. The processor core provides the command information and packet data for the packet engine. The packet engine can run autonomously, using its local DMA controller to perform DMA transfers across the main bus to access the memory of the processor core. The DMA process incorporates flow-control to guarantee proper data flow. Two elements provide the command information that defines the processing for each packet:

- Command descriptor
- SA record

When the packet engine finishes the operation, it updates the SA record, when needed, and provides a result descriptor.

The packet engine has four different modes of operation. The modes give the processor core various levels of control over the command information and packet data transfers to and from the packet engine.

- Autonomous Ring Mode (ARM)
- Target Command Mode (with and without result descriptor ring) (TCM)
- Direct Host Mode (DHM)

Input/Output FIFO Buffers

The data for the packet engine is buffered at both input and output. These buffers decouple the DMA I/O process from the cipher and hash modules inside the packet engine. This functionality enables large DMA burst sizes and allows the crypto engines to process data during I/O latency periods. Data moves automatically from the input buffer through the encryption and hash engines to the output buffer. When the output buffer is full, the process stops until the data is read and space is available in the output buffer. Each buffer is a 256-byte dual-port RAM.

Parallel Operations

The hash functionality and encrypt or decrypt functionality are tightly coupled. Operations that combine both hash and encrypt or decrypt functions are available to reduce processing time for data that must apply both. For hash-then-decrypt operations, the packet engine performs parallel execution of both functions from the input buffer. For encrypt-then-hash operations, the processing is pipelined from the input buffer to provide minimum latency. An offset can be specified between the start of the hashing and the start of the encryption to support protocols such as IPSec and SRTP.

DMA Controller

The packet engine uses a high-performance DMA controller for autonomous data transfers for:

- Command descriptor reads
- SA record and state record reads
- Packet data read
- Result packet writes
- SA record and state record writes
- Result descriptor writes

Interrupt Controller

The packet engine includes an interrupt controller that, under programmable configuration control, can generate an interrupt on completion of certain operations. Individual interrupts can be masked and cleared. The interrupt registers show both the raw and masked interrupt status of the internal interrupts. The processor core can use interrupts, together with their associated threshold settings, to optimize the overall packet processing in the system. One interrupt can inform the processor core that the input side of the packet engine is almost empty to avoid a stall-on-empty condition. One interrupt can inform the host that the output side is almost full to avoid a stall-on-full condition. The controller uses several interrupts to inform the processor core about errors inside the packet engine. All available interrupts are combined into a single output port as either a level- or edge-active programmable interrupt output.

Clock Controller

The packet engine includes a clock controller that generates clock enable signals. A clock manager external to the packet engine uses the clock enable signals to switch the clocks to modules in the packet engine, reducing power consumption. The power saving can be significant, depending on the crypto-operation and the idle time of the packet engine. The clock controller generates the clock enable signals dynamically depending on the current crypto-operation. A clock control register provides the processor core the possibility to override this dynamic process.

PKTE Operating Modes

The packet engine can be configured in one of three command modes. For all modes the packet engine can generate an interrupt at completion of packet processing:

- Autonomous Ring Mode (ARM). The core prepares descriptors in the CDR and then initiates a descriptor fetch by triggering the packet engine. When a packet operation is complete, the packet engine writes the result descriptor out into a ring in host memory using the system requester bus interface.
- Target Command Mode (TCM). The core directly writes the command descriptors to the packet command register set to initiate a packet operation. This process eliminates an extra DMA transfer to fetch a descriptor, but requires the core to synchronously initiate packet processing. This mode can be configured both without RDR or with RDR. In the latter case, the packet engine writes the result descriptor out into the RDR in host memory using the system requester bus interface.

- Direct Host Mode (DHM). The core has full control over the packet engine and uses the system completer bus interface. The core provides all the command descriptors, SA record and state record, and packet input data. When the packet engine completes processing, the core must read the packet output data, the SA record, the state record, and the result descriptors.

Autonomous Ring Mode (ARM)

The *Autonomous Ring Mode* allows the packet engine and the host processor to operate asynchronously. A queue of multiple packets in the host processor memory can be processed continuously to provide the highest possible throughput. The packet engine autonomously fetches the command descriptor, the SA record, and optionally the state record and the input data from host processor memory. After the packet engine finishes processing, it autonomously writes the output data, updates the SA and state record, and writes the result descriptor in the host processor memory. It accesses the host processor memory through DMA read transfers across the system bus requester.

This mode uses both command descriptor ring (CDR) and result descriptor ring (RDR).

Physically the CDR, RDR, SA record, source packet, and result packet can all be in different memories depending on the system memory architecture. The host processor writes command descriptors to the CDR in host processor memory. Then, it writes to the `PKTE_CDSC_INCR` register with the number of command descriptors that it prepared in the CDR. This write to the `PKTE_CDSC_INCR` register is the trigger for the packet engine to fetch the command descriptors sequentially from the CDR. When a command descriptor is fetched and written to the internal packet command register set, the descriptor is validated. If the ownership bits are set for the packet engine and the command is valid, processing starts. If not, that command is discarded, and a result descriptor is written to the RDR with the error code *invalid command descriptor*. In this mode, the host processor can set a threshold on the CDR and enable an associated interrupt. The packet engine generates an interrupt when the number of command descriptors in the CDR is equal or below the threshold value.

The SA record and state record that contains the crypto context information are stored in a memory area. The packet engine autonomously accesses the memory area through DMA transfers across the system bus requester. Also, the source packet and result packet are stored in a memory area that the packet engine autonomously accesses through the same bus requester interface.

After decoding the command descriptor, the packet engine fetches the SA record and then, optionally, the state record.

Then, the source packet is fetched and stored in the input buffer. Packets less than the size of the input buffers are fetched entirely at once. Larger packets are fetched in parts that completely fill the input buffer. The packet engine initiates a new fetch each time the number of empty spaces in the input buffer reaches its threshold value. When the first packet data is available in the input buffer, the crypto engines start processing the data. After processing, the crypto engines write the result packet to the output buffer.

The packet engine writes the result packet from the output buffer to host processor memory when the number of bytes in the output buffer reaches its threshold value. Packets less than the threshold value are written entirely at once. Larger packets are written in parts that completely empty the output buffer

The source packet data fetching, data processing, and result packet data writing are parallel processes that continue until the last result packet is written to host processor memory. Then, the packet engine optionally writes the SA record and the state record to update the crypto context information. As a final step, the packet engine writes the result descriptor to the RDR. The host processor must either poll the RDR or wait for an interrupt from the packet engine to determine when packet processing is complete.

Target Command Mode (TCM)

This mode provides a synchronous interface between the processor core and the packet engine. The Command Descriptor Ring (CDR) is disabled and the processor core initiates packet processing by writing the command descriptor directly to the internal command descriptor MMRs of the packet engine. The Result Descriptor Ring (RDR) is optional.

- For `PKTE_CFG.MODE` =01, the RDR is disabled and the processor core reads the result descriptor directly from the internal result descriptor register set for the packet engine.
- For `PKTE_CFG.MODE` =10, the RDR is enabled and stored in a memory area that the packet engine can access through its requester bus interface as in autonomous ring mode.

In target command mode, the packet engine autonomously fetches the SA record, state record, source packet data as in autonomous ring mode. Also, as in ARM, after processing, the packet engine updates state fields of the SA record and state record in the host processor memory.

Direct Host Mode (DHM)

This mode provides a synchronous interface between the processor core and the packet engine. The packet engine is under full control of the processor core. The host processor writes the command descriptors, SA record, and state record directly to the packet engine registers. Then, the processor core writes the source packet data into the input buffer. When processing is complete, the processor core reads back the result packet data from the output buffer. Finally, it reads the result descriptor, the updated SA record, and state record directly from the packet engine registers.

PKTE Event Control

The following section provides information about interrupts in the PKTE module.

PKTE Interrupt Signals

The Packet Engine has an internal Interrupt Controller with 9 interrupt sources. There are 7 registers associated with the interrupt controller:

1. Interrupt Unmasked Status - `PKTE_IUMSK_STAT`
2. Interrupt Mask Status - `PKTE_IMSK_STAT`
3. Interrupt Clear Register - `PKTE_INT_CLR`
4. Interrupt Enable Register - `PKTE_INT_EN`

5. Interrupt Mask Disable - PKTE_IMSK_DIS
6. Interrupt Mask Enable - PKTE_IMSK_EN
7. Interrupt Configuration - PKTE_INT_CFG

The ***Packet Engine Interrupt Controller Block Diagram*** shows the blocks of the interrupt controller.

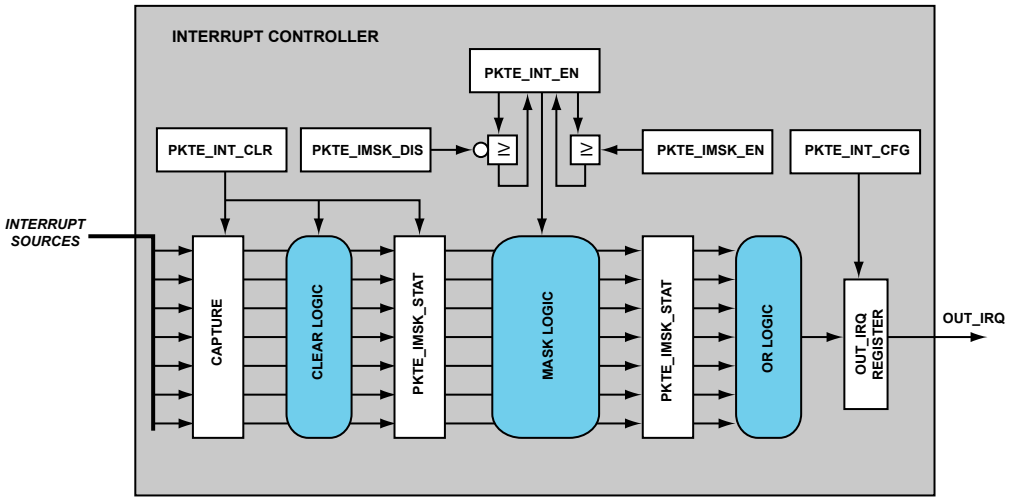


Figure 39-2: Packet Engine Interrupt Controller Block Diagram

All of the interrupt sources are pulse or level events in their native form.

These interrupts are captured and stored at their unmasked and masked status in their respective PKTE_IUMSK_STAT and PKTE_IMSK_STAT registers. This allows the host processor to read the status of any interrupt source either before or after the mask is applied.

The PKTE_INT_EN register provides a mask to select what interrupt source are enabled to the output interrupt request. Writing a one to the PKTE_INT_CLR register resets both the masked and unmasked interrupt.

The PKTE_IMSK_EN and PKTE_IMSK_DIS registers can be set to enable and disable individual interrupts respectively in the PKTE_INT_EN register. This avoids the need for read-modify-write operations from the host processor.

Ring Interrupts

Two interrupts are provided for efficient ring management: The CDR threshold interrupt (*cdrthrs*h) and the RDR threshold interrupt (*rdrthrsh*).

Command Descriptor Ring

The CDR threshold interrupt (cdrthrsh) is a level-based interrupt and connects to the threshold value in the PKTE_RING_THRESH.CDRTHRSH bit field. It enables the host processor to efficiently fill the CDR. The host processor writes command descriptors to the CDR with this interrupt masked until the CDR is full. Then the host processor enables the CDR threshold interrupt. When the interrupt is activated, the host processor clears the interrupt and it is guaranteed that it can put CDR threshold number of descriptors in the CDR.

Example Configuration:

```
PKTE_RING_CFG.RINGSZ=256,
PKTE_RING_THRESH.CDRTHRSH=224,
PKTE_INT_EN.CDRTHRSH=0 /*(IRQ disabled)*/
```

1. The host writes 8 Command Descriptors at once, then writes the PKTE_CDSC_INCR register to 8.
2. This is repeated until there are less than 8 empty entries in the CDR.
3. The fill level is now equal to the threshold (224)
4. The host enables the CDR threshold IRQ (PKTE_INT_EN.CDRTHRSH=1)
5. The packet engine processes packets and the fill level (PKTE_CDSC_CNT register) decreases.
6. Then the CDR threshold IRQ is activated as the fill level equals 224.
7. The host handles the interrupt, clears it, and continues with step #1.

Result Descriptor Ring

The RDR threshold interrupt (rdrthrsh) is a level-based interrupt and connects to the threshold value in the PKTE_RING_THRESH.RDRTHRSH bit field and the timeout value in the RD timeout (PKTE_RING_THRESH.RDTO) bit field. It enables the host processor to efficiently empty the RDR. The timeout reminds the host processor that when the threshold kicks in result descriptors stay long in the RDR and must be processed to reduce latency. The timeout counts when the ring is not empty, regardless of the fill level and restarts when the host processor writes the PKTE_RDSC_CNT register. Initially the host processor enables the RDR threshold interrupt. When the interrupt is activated the host processor reads result descriptors until the RDR is empty or contains less than the PKTE_RING_THRESH.RDRTHRSH number of descriptors.

Example Configuration:

```
PKTE_RING_CFG.RINGSZ=256,
PKTE_RING_THRESH.RDRTHRSH=32, timeout=1ms
PKTE_INT_EN.RDRTHRSH=1     /*(IRQ enabled)*/
```

1. The Packet Engine writes the Result Descriptor, timeout counter starts.
2. The Packet Engine writes 32 more Result Descriptors, fill level (PKTE_RDSC_CNT register) increases to 33.
3. The fill level exceeds threshold within 1ms, the RDR threshold IRQ is activated.
4. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. The write to the PKTE_RDSC_DECR register restarts the timeout counter. The fill level is now under the threshold but there are still 25 descriptors left.
5. The Packet Engine writes 8 more Result Descriptors, fill level increases to 33.
6. The fill level exceeds threshold within 1 ms, the RDR threshold IRQ is activated.

7. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. The write to the PKTE_RDSC_DECR register restarts the timeout counter. The fill level is now under the threshold but there are still 25 descriptors left.
8. After 1 ms, the timeout counter interrupt is activated.
9. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. This is repeated until there are less than 8 full entries in the RDR. The fill level is now under the threshold and the RDR threshold IRQ interrupt is inactive. Each write to the PKTE_RDSC_DECR register restarts the timeout counter.

PKTE Programming Model

The host processor must always follow a pre-defined sequence of five phases required by the packet engine on a per packet basis when using direct host mode. The following sections describe the five phases.

Phase One. Write the Command Descriptor

1. Write the first command descriptor word with status and control information to the PKTE_CTL_STAT register.
2. Optionally, write the user ID to the PKTE_USERID register.
3. Write the last descriptor word to the PKTE_LEN register.
4. Write the value 0x1 to the PKTE_CDSC_CNT register. This operation triggers the packet engine to validate the command descriptor. When the command descriptor is invalid, an error is generated. (See the PKTE_CTL_STAT section in the Register Descriptions). When the command descriptor is valid, the packet engine waits for an PKTE_SA_RDY register write.

Phase Two. Write the State Registers, (ARC4 Buffer) and SA Registers

All required fields of the SA record and state record must be written. The fields required depend on the operation. The last field to be written is the PKTE_SA_RDY register. This register triggers the packet engine to start processing.

1. Write the required state record fields.
   - ARC4 state
   - IV
   - Digest count
   - State digest
2. Write the required SA record data.
3. To complete the SA record and state record, write the PKTE_SA_RDY register.

Phase Three. Write the Source Packet Data and Read Result Packet Data

The packet engine has input and output buffers. If a source packet is smaller than the size of the input buffer, then the packet can be written in one part. Otherwise, it must be written in multiple parts. The same applies to the output data. If the result packet size is smaller than the size of the output buffer, then the packet can be read in one part. Otherwise, it must be read in multiple parts.

NOTE: An outbound packet that is smaller than the size of the input buffer can increase in size due to padding and does not always fit in the output buffer. Conversely, an inbound packet that is larger than the size of the input buffer can decrease in size, and due to de-padding, can fit in the output buffer. If the input buffer becomes empty or the output buffer becomes full, the engine stalls.

Two following steps describe different situations:

- Source packet smaller than the size of the input buffer, start at step #1.
- Source packet larger than the size of the input buffer, start at step #3.

The host processor must follow these steps:

1. Write the source packet data. Write the full source packet to the input buffer. Go to step #4.
2. Write the input buffer count register (PKTE_INBUF_CNT) with the number of valid bytes that are written to the input buffer. This value must correspond to the value in the PKTE_LEN.TOTLEN field of the command descriptor rounded up to the next 4-byte multiple. Go to step #5 to check the packet engine status.
3. Write part of the source packet data. The PKTE_STAT.IBUFEMPTYCNT field indicates the amount of free space in the input buffer. Programs write the number of bytes determined by the setting in the PKTE_BUF_THRESH register. Write the (partial) source packet to the input buffer. The host processor must resume where it ended the previous write operation. Do not write more than the buffer size at once. Go to step 4.
4. Write the PKTE_INBUF_CNT register with the number of valid bytes written to the input buffer. Go to step 5 to check the packet engine status.
5. Check packet engine status. Wait for an interrupt or poll the PKTE_STAT register for any of the following conditions:
   - Condition 1–an error interrupt or any of the bits [7:5] in the PKTE_STAT register becomes active to indicate a packet processing error. Depending on the type of error, the host processor must take appropriate action. Usually, the result packet is not valid after a processing error has occurred. Go to phase 5 to read the result descriptor.
   - Condition 2–an operation done interrupt or the PKTE_STAT.OPDN bit becomes active (the packet engine completed processing). Go to step 8 to read the remaining output data.
   - Condition 3–an output buffer threshold interrupt or the PKTE_STAT.OBUFREQ bit becomes active. Go to step #6 to read a block of output data.

- Condition 4–an input buffer threshold interrupt or the PKTE_STAT.IBUFREQ bit becomes active. Go to step #3 to write a block of input data.

6. Read part of the output data. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of bytes in the output buffer. This value is rounded up to full words. Programs read the number of bytes indicated in the PKTE_BUF_THRESH register. Read the (partial) output packet from the output buffer. The host processor must resume where it ended the previous read operation. Do not read more than the input buffer size at once. Go to step #7.
7. Write the PKTE_OUTBUF_CNT register with the number of valid bytes read from the output buffer. Go to step #5 to check the packet engine status.
8. Read the remaining output data. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of bytes in the output buffer. This value is rounded up to words. Read the (partial) packet output data from the output buffer. The host processor must resume where it ended the previous read operation. Go to step #9.
9. Write the PKTE_OUTBUF_CNT register with the number of valid bytes read from the output buffer. Go to phase four.

Phase Four. Read the Result Descriptor

1. Read the first result descriptor word from the PKTE_CTL_STAT.
2. Optionally read the user ID from the PKTE_USERID register.
3. Read the last result descriptor word from the PKTE_LEN register.
4. Write the value 0x1 to the PKTE_RDSC_DECR register. This operation allows the packet engine to accept new command descriptors. Go to phase five.

Phase Five. Read the SA Record and State Record

Depending on the operation, the SA record or state record is updated. Check the bit fields [23:16] in the PKTE_CTL_STAT register for the following conditions:

- Condition 1 - At least one error bit in the bit fields [23:16] of the PKTE_CTL_STAT register is set. Do not update the local host processor maintained version of the SA record and state record but take any required action.
- Condition 2 - None of the error bits in the bit fields [23:16] of the PKTE_CTL_STAT register is set, the packet is processed normally (without errors). Update the host processor maintained version of the SA record and state record with the result read from the packet engine registers:
  - ARC4 state
  - Sequence number
  - Sequence number mask
  - Result IV

- Result digest count
- Result digest

PKTE Mode Configuration

Before using the packet engine, it must be configured. The mode of the packet engine must be defined and the PRNG (if used) must be initialized.

Configure the packet engine in one of three command modes:

- Autonomous Ring Mode: `PKTE_CFG.MODE` =b'11
- Target Command Mode: `PKTE_CFG.MODE` =b'10
- Direct Host Mode: `PKTE_CFG.MODE` =b'00

PKTE Programming Concepts

The following sections provide conceptual information for programming the PKTE.

Packet Engine Descriptor

IMPORTANT: Depending on the mode, ARM, TCM, or DHM, the descriptor is either:

- in the memory of the host processor in the command descriptor ring, or
- written directly to the descriptor registers in the packet engine

References to descriptor registers are for either the register that is mirrored in the descriptor structure in memory or for the actual register itself.

Command descriptors are host-supplied commands that control the real-time operation of the packet engine. The packet engine returns result descriptors at the end of an operation that provide the status information to the host. The ***Command Descriptor Structure*** and the ***Result Descriptor Structure*** tables show these descriptors.

Table 39-20: Command Descriptor Structure

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
0	Pad Control	—		Next Header/ Pad Value	Control	0x000
1	Source Address					0x004
2	Destination Address					0x008
3	SA Address					0x00C
4	SA State Address					0x010
5	Reserved/ARC4 State Address					0x014

Table 39-20: Command Descriptor Structure (Continued)

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
6	User ID					0x018
7	Bypass (words)	Control	Reserved	Input Packet Length (bytes)		0x01C

Table 39-21: Result Descriptor Structure

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
0	Pad Status	Status		Next Header/ Pad Value	Control	0x000
1	Source Address					0x004
2	Destination Address					0x008
3	SA Address					0x00C
4	SA State Address					0x010
5	Reserved/ARC4 State Address					0x014
6	User ID					0x018
7	Bypass (words)	Control	Reserved	Input Packet Length (bytes)		0x01C

When the packet engine is configured for autonomous ring mode, command descriptors and result descriptors reside in a ring in host memory. Command descriptors are automatically fetched from the Command Descriptor Ring (CDR) through DMA into the command descriptor registers. When an operation is complete, the result descriptors are automatically read from the packet engine and through DMA to the Result Descriptor Ring (RDR).

When the packet engine is configured for direct host mode, the host processor manually writes the command descriptor directly to the internal command descriptor MMR set. When an operation is complete, the host processor manually reads the result descriptor directly from the result descriptor MMR set.

The target command mode is a combination of the direct host mode and the autonomous ring mode. The host processor writes the command descriptor directly to the internal command descriptor register set. When an operation is complete there are two options.

1. The host processor can read the result descriptor directly from the result descriptor registers.
2. The result descriptors reside in a ring in host memory and the result descriptors are automatically DMA'd from the packet engine to the RDR.

When the host processor writes a command descriptor to the command descriptor registers, the packet engine is triggered when the host processor updates the `PKTE_CDSC_CNT` register. This functionality guarantees that all fields in the command descriptor are valid before the command is executed.

Descriptor Processing

This section describes the functional steps of the packet engine while processing the command descriptors.

Descriptor Ring Configuration

At initialization, the host processor specifies the size of the Command Descriptor Ring (CDR). The Result Descriptor Ring (RDR) has the same size.

NOTE: In some configurations, these two rings overlay each other with the results written on top of the command descriptors. This configuration is called overlaid ring mode.

When the packet engine is configured and enabled, it fetches the descriptors from the CDR using system bus requester reads.

Descriptor Ring Processing

To validate the descriptor exchange between the host processor and the packet engine, the ownership bits PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY are used. One pair of ownership bits is in the first word of the descriptor (PKTE_CTL_STAT), and one pair is in the last word (PKTE_LEN). The 'consumer' of a descriptor must verify that both ownership pairs match to ensure that a race condition did not occur between one party writing and the other party reading the descriptor. A race condition can occur when a memory locking scheme is not used.

Each pair [PKTE_CTL_STAT.PERDY, PKTE_CTL_STAT.HOSTRDY] of ownership bits provide 3 states:

- b'00 = idle or null descriptor
- b'01 = host processor has written a descriptor in the CDR and passed ownership to the packet engine
- b'10 = packet engine processing complete: packet engine has written the descriptor in the RDR and passed ownership back to the host.
- b'11 = Reserved

At initialization, the host sets the entire CDR memory area to zero, when the CDR is used.

1. The host processor writes one or more command descriptors to the CDR. The host processor must set the PKTE_CTL_STAT.HOSTRDY bit to 1 and the PKTE_CTL_STAT.PERDY bit to 0 to indicate that ownership has passed to the packet engine. These bits are mirrored in the PKTE_LEN descriptor word.
2. The host processor must write the PKTE_CDSC_INCR register with the number of new valid command descriptors in the CDR.
3. The packet engine reads and validates one command descriptor.
4. The packet engine reads the SA record and state record, processes the packet, and updates the SA record and state record.

5. If the rings are not overlaid and the PKTE_CFG.ENCDRUPDT bit is 1, the packet engine writes the result descriptor to the CDR with the PKTE_CTL_STAT.HOSTRDY bit set to 0 and PKTE_CTL_STAT.PERDY bit set to 1. These bits are mirrored in the PKTE_LEN descriptor word.
6. The packet engine writes the result descriptor to the RDR. The packet engine sets the PKTE_CTL_STAT.PERDY bit to 1 and the PKTE_CTL_STAT.HOSTRDY bit to 0 to indicate that the ownership has passed to the host. These bits are mirrored in the PKTE_LEN descriptor word.
7. The packet engine decrements the value in the PKTE_CDSC_CNT register. If the value is not zero, the packet engine reads with the next command descriptor (step 3).
8. The packet engine increments the value in the PKTE_RDSC_CNT register.
9. The host processor reads one or more result descriptors from the RDR and processes the results.
10. The host processor must write the PKTE_RDSC_DECR register with the number of processed result descriptors in the RDR.

Descriptor Ownership

The ownership of the command descriptor and result descriptor is set by ownership bits in the first and last word of the respective descriptor, in the PKTE_CTL_STAT register and the PKTE_LEN register. For the command descriptor, it is the processor core that sets the ownership to the packet engine. For the result descriptor, it is the packet engine that sets the ownership to the host processor.

The packet engine reads the ownership bits before processing, irrespective of the mode of the packet engine. The ownership bits are used to validate and identify the descriptor. When two separate rings are used, the packet engine can be programmed to clear the ownership bits of the command descriptors in the CDR, so the host processor knows which descriptors have been processed.

NOTE: This update of the ownership bits can be disabled in the PKTE_CFG register when the host processor actively counts the number of descriptors in the CDR to prevent ring wrapping.

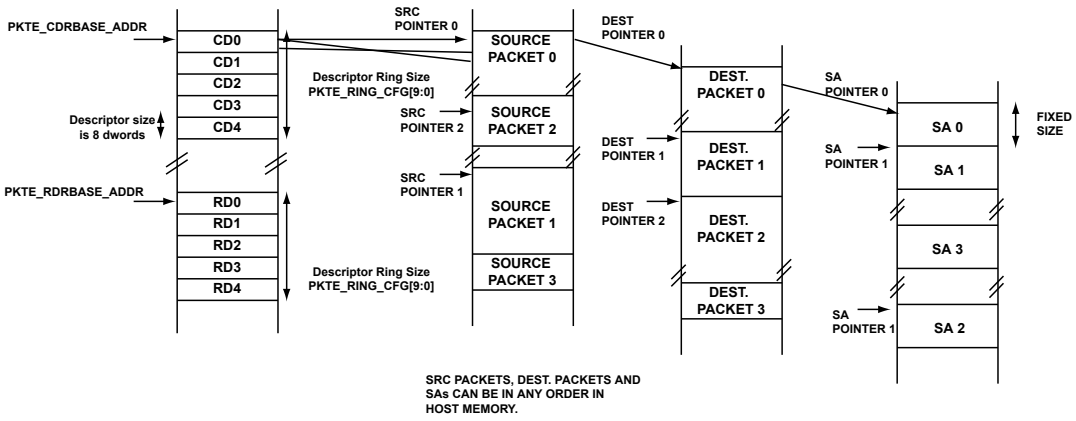


Figure 39-3: Descriptor Rings in Autonomous Ring Mode

SA Record and State Record Structure

The SA record is a packed structure that contains the remainder of the information needed by the packet engine to process a packet. Most of the information fields in the SA record, such as the key and encryption mode, are static for the lifetime of the association. The fields do not require frequent manipulation by the host processor. The SA record non-static fields are the sequence number and sequence number mask.

The SA record can have a corresponding state record that is used to save results from the current operations that can be used for future operations. The state record can hold the IV, the hash byte count, and the intermediate hash digest.

If an SA record is used for operations that use ARC4 processing in stateful mode, it has a corresponding ARC4 state record that holds the ARC4 State.

In this manual, the state record and the ARC4 state record are referred to as state record.

There is no practical limit to how many SA records and corresponding state records the packet engine can support.

In the autonomous ring mode and target command mode, once the packet engine has validated a command descriptor, it automatically fetches the SA record and optional state record. After processing, the packet engine updates the stateful fields in the SA record and state record in the host processor memory.

In direct host mode, after the descriptor is validated, the host must write the SA record directly into the internal registers of the packet engine. After processing, the host reads the stateful fields from the SA registers of the packet engine and saves them back to the SA record in the host processor memory.

SA Record Structure

The ***SA Record Structure*** table shows the structure for an SA Record. When using direct host mode, the corresponding elements are accessed directly with the registers. When using autonomous ring mode or target command mode, the SA Record is defined, configured, and accessed in host memory.

Table 39-22: SA Record Structure

Word Offset	Description (name)	Use
0	PKTE_SA_CMD0[31:0]	SA Control word 0 (all operations)
1	PKTE_SA_CMD1[31:0]	SA Control word 1 (all operations)
2	PKTE_SA_KEY0[31:0]	Key word (DES, Triple-DES, AES-128/192/256, ARC4,)
3	PKTE_SA_KEY1[63:32]	Key word (DES, Triple-DES, AES-128/192/256, ARC4,)

Table 39-22: SA Record Structure (Continued)

Word Offset	Description (name)	Use
4	PKTE_SA_KEY2[95:64]	Key word (Triple-DES, AES-128/192/256, ARC4,)
5	PKTE_SA_KEY3[127:96]	Key word (Triple-DES, AES-128/192/256, ARC4,)
6	PKTE_SA_KEY4[159:128]	Key word (Triple-DES, AES-192/256)
7	PKTE_SA_KEY5[191:160]	Key word (Triple-DES, AES-192/256)
8	PKTE_SA_KEY6[223:192]	Key word (AES-256)
9	PKTE_SA_KEY7[255:224]	Key word (AES-256)
10	PKTE_SA_IDIGEST0[31:0]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
11	PKTE_SA_IDIGEST1[63:32]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
12	PKTE_SA_IDIGEST2[95:64]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
13	PKTE_SA_IDIGEST3[127:96]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
14	PKTE_SA_IDIGEST4[159:128]	Inner Hash digest (Basic Hash and HMAC with SHA-1, SHA-224, SHA-256)
15	PKTE_SA_IDIGEST5[191:160]	Inner Hash digest (Basic Hash and HMAC with SHA-224, SHA-256)
16	PKTE_SA_IDIGEST6[223:192]	Inner Hash digest (Basic Hash and HMAC with SHA-224, SHA-256)

Table 39-22: SA Record Structure (Continued)

Word Offset	Description (name)	Use
17	PKTE_SA_IDIGEST7[255:224]	Inner Hash digest (Basic Hash and HMAC with SHA-256)
18	PKTE_SA_ODIGEST0[31:0]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
19	PKTE_SA_ODIGEST1[63:32]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
20	PKTE_SA_ODIGEST2[95:64]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
21	PKTE_SA_ODIGEST3[127:96]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
22	PKTE_SA_ODIGEST4[159:128]	Outer Hash digest (HMAC with SHA-1, SHA-224, SHA-256)
23	PKTE_SA_ODIGEST5[191:160]	Outer Hash digest (HMAC with SHA-224, SHA-256)
24	PKTE_SA_ODIGEST6[223:192]	Outer Hash digest (HMAC with SHA-224, SHA-256)
25	PKTE_SA_ODIGEST7[255:224]	Outer Hash digest (HMAC with SHA-256)
26	PKTE_SA_SPI[31:0]	SPI (IPsec), Type[23:16] / Version [15:0] (SSL, TLS, DTLS)
27	PKTE_SA_SEQNUM0[31:0]	Sequence Number (IPsec, SSL, TLS, DTLS with Header Processing)
28	PKTE_SA_SEQNUM1[63:32]	
29	PKTE_SA_SEQNUM_MSK0[31:0]	Sequence Number Mask (IPsec, DTLS inbound with Header Processing)
30	PKTE_SA_SEQNUM_MSK1[63:32]	
31	PKTE_SA_NONCE[31:0] / PKTE_SA_READY	Nonce value (AES-CTR, AES-ICM)/ARC4, i and

Table 39-22: SA Record Structure (Continued)

Word Offset	Description (name)	Use
		j pointers (ARC4,)/SA ready indicator (Direct Host Mode)

Some of these fields may be updated by the packet engine. These include:

- PKTE_SA_SEQNUM0
- PKTE_SA_SEQNUM1
- PKTE_SA_SEQNUM_MSK0
- PKTE_SA_SEQNUM_MSK1

All the other fields remain unchanged.

SA State Structure

The security association state structure contains information that may be updated after each packet, such as the IV and the intermediate hash result. The ***SA State Structure*** table shows the SA state structure and usage. In direct host mode, the elements are accessed directly using the PKTE registers. In target command mode and autonomous ring mode, this structure is defined and updated in host memory.

Table 39-23: SA State Structure

Word Offset	Description (name)	Use
0	PKTE_STATE_IV0[31:0]	Initialization Vector (DES, Triple DES, AES)
1	PKTE_STATE_IV1[63:32]	Initialization Vector (DES, Triple DES, AES)
2	PKTE_STATE_IV2[95:64]	Initialization Vector (AES)
3	PKTE_STATE_IV3[127:96]	Initialization Vector (AES)
4	PKTE_STATE_BYTE_CNT0[31:0]	Current hash byte count (MD5, SHA-1, SHA-224, SHA-256)
5	PKTE_STATE_BYTE_CNT1[63:32]	Current hash byte count (MD5, SHA-1, SHA-224, SHA-256)
6	PKTE_STATE_IDIGEST0[31:0]	Inner Hash digest (mirror of PKTE_SA_IDIGEST0)
7	PKTE_STATE_IDIGEST1[63:32]	Inner Hash digest (mirror of PKTE_SA_IDIGEST1)
8	PKTE_STATE_IDIGEST2[95:64]	Inner Hash digest (mirror of PKTE_SA_IDIGEST2)
9	PKTE_STATE_IDIGEST3[127:96]	Inner Hash digest (mirror of PKTE_SA_IDIGEST3)
10	PKTE_STATE_IDIGEST4[159:128]	Inner Hash digest (mirror of PKTE_SA_IDIGEST4)
11	PKTE_STATE_IDIGEST5[191:160]	Inner Hash digest (mirror of PKTE_SA_IDIGEST5)
12	PKTE_STATE_IDIGEST6[223:192]	Inner Hash digest (mirror of PKTE_SA_IDIGEST6)
13	PKTE_STATE_IDIGEST7[255:224]	Inner Hash digest (mirror of PKTE_SA_IDIGEST7)

ARC4 State Structure

The *ARC4 State Structure* table describes the state structure used with ARC4. When using the PKTE in direct host mode, these fields are accessed with the registers, starting at the value in the PKTE_ARC4STATE_BUF register. When using the PKTE in autonomous ring mode or target command mode, these fields are defined and accessed in a structure in host memory.

Table 39-24: ARC4 State Structure

Word Offset	Description (name)*1	Use
0	PKTE_ARC4_STATE0[3:0]*2	ARC4 (Basic, SSL and TLS)
1	PKTE_ARC4_STATE1[7:4]	
...	...	
62	PKTE_ARC4_STATE2[251:248]	
63	PKTE_ARC4_STATE3[255:252]	

*1 The indices in these fields indicate bytes.

*2 There are no corresponding named registers for these fields. PKTE_ARC4_STATE0 corresponds to PKTE_ARC4STATE_BUF and PKTE_ARC4_STATE1 corresponds to the 32-bit register following PKTE_ARC4STATE_BUF and so on.

Configuring Operations in the PKTE

The operation (cipher, hash function, and others) that the PKTE performs is configured primarily in the PKTE_SA_CMD0 register. The following sections include a series of tables to help configure the least significant 16 bits of the PKTE_SA_CMD0 register. These fields include:

- The operation code field (PKTE_SA_CMD0.OPCD)
- The direction field (PKTE_SA_CMD0.DIR)
- The operation group field (PKTE_SA_CMD0.OPGRP)
- The padding type (PKTE_SA_CMD0.PADTYPE)
- The cipher selection (PKTE_SA_CMD0.CIPHER)
- The hash selection (PKTE_SA_CMD0.HASH)

Basic Operations and Decoding

Table 39-25: Basic Operation Decoding

Outbound				Inbound			
OpGroup	Dir	OpCode	Operation	OpGroup	Dir	OpCode	Operation
0b00	0	0b000	Encrypt	0b00	1	0b000	Decrypt
0b00	0	0b001	Encrypt - Hash	0b00	1	0b001	Hash - Decrypt

Table 39-25: Basic Operation Decoding (Continued)

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b00	0	0b010	Reserved	0b00	1	0b010	Reserved
0b00	0	0b011	Hash	0b00	1	0b011	Hash
0b00	0	0b100... 0b110	Reserved	0b00	1	0b100... 0b110	Reserved
0b00	0	0b111	PRNG	0b00	1	0b111	Reserved

Table 39-26: Protocol Operation Decoding

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b01	0	0b000	ESP Outbound	0b01	1	0b000	ESP Inbound
0b01	0	0b001... 0b011	Reserved	0b01	1	0b001	Reserved
0b01	0	0b100	Basic SSL Outbound	0b01	1	0b010	Basic SSL Inbound
0b01	0	0b101	Basic TLS Outbound	0b01	1	0b011	Basic TLS Inbound
0b01	0	0b110	Reserved	0b01	1	0b100... 0b110	Reserved
0b01	0	0b111	SRTP Outbound	0b01	1	0b111	SRTP Inbound

NOTE: For SSL/TLS and SRTP, no header processing is performed in hardware.

Table 39-27: Extended Protocol Operation Decoding

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b11 0	0	0b000	Reserved	0b11	1	0b000	Reserved
0b11 0	0	0b001	DTLS Outbound	0b11	1	0b001	DTLS Inbound
0b11 0	0	0b010... 0b011	Reserved	0b11	1	0b010... 0b011	Reserved

Table 39-27: Extended Protocol Operation Decoding (Continued)

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b11 0	0	0b100	Ext. SSL Out-bound	0b11	1	0b100	Ext. SSL In-bound
0b11 0	0	0b101	Ext. TLS v1.0 Outbound	0b11	1	0b101	Ext. TLS v1.0 Inbound
0b11 0	0	0b110	Ext. TLS v1.1 Outbound	0b11	1	0b110	Ext. TLS v1.1 Inbound
0b11 0	0	0b111	Reserved	0b11	1	0b111	Reserved

Error Code Description

The `PKTE_CTL_STAT` register is used to configure the packet engine for processing in Direct Host Mode (DHM) or Target Command Mode (TCM). The `PKTE_CTL_STAT` structure element in memory is used when the packet engine is configured for Autonomous Ring Mode (ARM). In both cases, when an operation is started, errors are reported in the status field (bits [23:16]) of this register or structure element. The ***Extended Error Codes - Status Encoding*** table provides a guide on how to decipher the meaning of the bits that are set when an error occurs.

Extended Error Codes

The following table provides information about the extended errors associated with the PKTE module.

Table 39-28: Extended Error Codes - Status Encoding

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b0000_0000	0x00	NA	Successful completion. No errors occurred during processing of the packet.	Packet fully processed
0b----_---1	0x-1	NA	Authentication Error. For an inbound IPsec ESP operation, the Integrity Check Value (ICV) does not match the computed value. For an inbound SRTP operation, the authentication tag does not match the computed value. For a basic SSL/TLS, Extended SSL/TLS or DTLS operation the Message Authentication Code (MAC) does not match the computed value.	Packet fully processed

Table 39-28: Extended Error Codes - Status Encoding (Continued)

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b----_--1-	0x-2	NA	Pad Verify Error. For inbound operations that use pad type Constant TLS, IPsec or PKCS#7, the decrypted pad does not match the expected values for the selected pad type.	Packet fully processed
0b----_-1--	0x-4	NA	Sequence Number Error. For an inbound IPsec or DTLS operation, there was a fault in the Anti-Replay Sequence Number. For an outbound IPsec packet, the sequence number overflows; count is 2^{32}– 1 and increments to 0. For an outbound DTLS operation, the sequence number overflows; count is 2^{48}– 1 and increments to 0. For an outbound SSL or TLS operation, the sequence number overflows; count is 2^{64}– 1 and increments to 0.	Packet fully processed
0b0000_1---	0x08	1	System Bus error. The requester bus interface generates an error due to ERROR response from system completer. The completer bus interface generates an error due to request for non-word (32-bit) access.	Packet is stopped. The host must reject the packet and apply a hardware reset to the system.
0b0001_1---	0x18	2	Invalid Command Descriptor Error. The ownership bits in the command descriptor are not set to the packet engine, after the PKTE_CDSC_CNT register is incremented.	Command descriptor is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0010_1---	0x28	3	Invalid Crypto Operation Error. A reserved operation is selected.	The SA record is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0011_1---	0x38	4	Invalid Crypto Algorithm Error. A reserved cipher is selected, refer to PKTE_SA_CMD0.CIPHER. A reserved hash is selected, refer to PKTE_SA_CMD0.HASH.	The SA record is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0100_1---	0x48	5	SPI Error. On an inbound packet, the 32-bit SPI value in the packet does not match the value in the SA while header processing is enabled. Note: A failure caused by an SPI mismatch, in general should not occur because the host checks the SPI and does not send an incorrect SPI to the packet engine.	Packet is fully processed. The host must reject the packet.

Table 39-28: Extended Error Codes - Status Encoding (Continued)

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b0101_1---	0x58	3	Zero Length Error. The packet length defined in the command descriptor PKTE_LEN.TOTLEN is zero, which is illegal.	Packet command is ignored, no packet processed. The host must reject the packet.
0b0110_1xxx	0x68	6	Invalid Packet Length Error. For Basic Encrypt-Hash and Hash-Decrypt operations: PKTE_LEN.TOTLEN < Hash/Encrypt Offset For IPsec ESP inbound operations: PKTE_LEN.TOTLEN < ICV length or PKTE_LEN.TOTLEN is non-4 byte aligned For SRTP inbound operations: PKTE_LEN.TOTLEN ≤ IV (opt.) + Bypass Offset + ROC For SSL inbound operations: PKTE_LEN.TOTLEN ≤ 1 or packet length > 65535 bytes (SSL packet-bypass length) For TLS and DTLS inbound operations: PKTE_LEN.TOTLEN ≤ 13 or payload length > 65535 bytes (data to be hashed) Note: For IPsec ESP the ICV is stripped before the length is checked.	Packet processing is stopped Result packet length is zero. The host must reject the packet and apply a software reset of the packet engine.
0b0111_1xxx	0x78	7	Block Size Error. The length of the inbound packet defined in the Command Descriptor PKTE_LEN.TOTLEN is not a multiple of the DES or AES block cipher length. For outbound packets the size is always automatically aligned (padded) to the correct block size. The hashed packet length is not a multiple of the hash block size for intermediate hash operation. For a final hash operation no error is generated. Note: For IPsec ESP operations the ICV is stripped before the block size is checked.	Packet is fully processed. The host must reject the packet.
0b1000_1xxx	0x88	8	Processing Error. The number of bytes in the input buffer is more than defined in the PKTE_LEN.TOTLEN field. The number of bytes written to the output buffer is less than processed in the datapath.	Packet processing stops. Result packet length is zero. The host must reject the packet and apply a software reset.
0b1010_1xxx 0b1111_1xxx	Reserved			

Number Format

When dealing with cryptographic functions, data and keys are large vectors. For instance, AES supports keys of sizes 128, 192, and 256 bits. When a key needs to be loaded or read, multiple 32-bit key registers are used, namely PKTE_SA_KEY[n] registers. The first key register PKTE_SA_KEY[n]0 holds bits 31:0, and PKTE_SA_KEY[n]1 holds the next thirty two bits 63:32, and so on.

Generally, for large vectors defined as Byte0, Byte1, Byte2, Byte3 and so on, the values are stored in the PKTE registers as PKTE_REG0 = Byte3Byte2Byte1Byte0, followed by PKTE_REG1 = Byte7Byte6Byte5Byte4 and so on.

PKTE Programming Examples

Use these examples to extend your understanding of PKTE features, operating modes, event control, and programming modes.

Calculating SHA in Direct Host Mode

This section describes how to configure the packet engine to calculate a hash digest using one of supported SHA algorithms in direct host mode. This configuration follows the procedure outlined in the *PKTE Programming Model* section.

1. Configure the packet engine for direct host mode by setting the PKTE_CFG.MODE bit =0
2. Set the ownership back to the packet engine to process a command descriptor by setting the PKTE_CTL_STAT.PERDY bit =0 and the PKTE_CTL_STAT.HOSTRDY bit =1.
3. Set the PKTE_CTL_STAT.HASHFINAL bit to indicate this command descriptor handles all the input data for the hash calculation. This configuration is needed for the packet engine because the last block requires special handling (see FIPS 180-4 for details).
4. Set size of the input data in bytes in the PKTE_LEN.TOTLEN bit field.
5. Also set the PKTE_LEN.PEDONE bit =0 and the PKTE_LEN.HSTRDY bit =1. These bits must be the same as the PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY bits to guarantee ownership.
6. Set the PKTE_CDSC_CNT register =1 to trigger the packet engine to start validating the command descriptor. In this case, the PKTE_CTL_STAT, PKTE_LEN and PKTE_CDSC_CNT registers are the only command descriptor registers modified.
7. Configure the PKTE_SA_CMD0 and PKTE_SA_CMD1 registers to define the operation. For an SHA, set the PKTE_SA_CMD0.OPCD bit field =0b011 for hash operation and the PKTE_SA_CMD0.OPGRP bit field =0b00 for basic operation.
8. Select the specific SHA function using the PKTE_SA_CMD0.HASH bit field as follows.
   - For SHA-1, PKTE_SA_CMD0.HASH =0b0001
   - SHA-224, PKTE_SA_CMD0.HASH =0b0010
   - for SHA-256, PKTE_SA_CMD0.HASH =0b0011

9. Depending on the SHA selected, the appropriate digest length must be chosen for the PKTE_SA_CMD0.DIGESTLEN bit field as follows.
   - For SHA-1, PKTE_SA_CMD0.DIGESTLEN =0b0101 (5 words)
   - For SHA-224, PKTE_SA_CMD0.DIGESTLEN =0b0111 (7 words)
   - For SHA-256, PKTE_SA_CMD0.DIGESTLEN =0b1000 (8 words)
10. The SHA specifies initial constants. These constants can be pre-loaded or read from memory. In this example, by setting the PKTE_SA_CMD0.HASHSRC bit field =0b11, the packet engine provides the correct initial constants depending on the SHA chosen.
11. Next, set the PKTE_SA_CMD1.CPYDGST bit =1 and PKTE_SA_CMD1.CPYPAD bit =1 to move the result to the output buffer of the packet engine at the PKTE_DATAIO_BUF location.
12. At this point, write to the PKTE_SA_RDY register with any value to trigger the operation.
13. Start writing the input to the data buffer of the packet engine starting at the PKTE_DATAIO_BUF location.
14. Write the PKTE_INBUF_CNT register with the length of the input rounded up to the next multiple of 4. For example, if the input length is 30 bytes, set this register to 32.
15. Poll the PKTE_STAT register to see if any errors occurred or if the operation completed without errors.

Once the operation is done, the digest is available in the packet engine data I/O buffer.

NOTE: The input data or message is input into the packet engine data buffer in big endian format while the result or digest is little endian format.

Performing AES Decryption in Direct Host Mode

This section describes how to configure the packet engine to decrypt using AES-128 in direct host mode. This configuration follows the procedure outlined in the *PKTE Programming Model* section.

1. Configure the packet engine for direct host mode by setting the PKTE_CFG.MODE bit =0
2. Start configuring the command descriptor registers. Set the ownership back to the packet engine to process a command descriptor by setting the PKTE_CTL_STAT.PERDY bit =0 and the PKTE_CTL_STAT.HOSTRDY bit =1.
3. Next, configure the PKTE_LEN.TOTLEN bit field with the size of the packet or message to decrypt. If the entire input message (cipher text) fits into the 256-byte data I/O buffer of the packet engine, the process can be done in one shot.
4. Set the PKTE_LEN.PEDONE bit =0 and the PKTE_LEN.HSTRDY bit =1. These bits must have the same setting as the PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY bits to guarantee ownership.

5. Set the PKTE_CDSC_CNT register =1 to trigger the packet engine to start validating the command descriptor. In this case, the PKTE_CTL_STAT, PKTE_LEN, and PKTE_CDSC_CNT registers are the only command descriptor registers modified.
6. Next, configure the PKTE_SA_CMD0 and PKTE_SA_CMD1 registers to define the operation.
   - For a AES decrypt inbound cipher operation, set the PKTE_SA_CMD0.OPCD bit field =0b000 and the PKTE_SA_CMD0.DIR bit field =0b1.
   - Set thePKTE_SA_CMD0.OPGRP bit field =0b00 for basic operation.
   - To choose the AES cipher, set the PKTE_SA_CMD0.CIPHER bit field =0b0011. Set the PKTE_SA_CMD0.HASH bit field to 0b1111 to choose the NULL function.
7. Next, set the PKTE_SA_CMD1.AESKEYLEN bit field to select the appropriate key length. In this case, setting it to 0b10 select 128 bits. Also, set PKTE_SA_CMD1.CIPHERMD bit field to select the mode. In this case, setting it to 0b01 select CBC mode.
8. Continue configuring the Security Association (SA) record by loading the key in the PKTE_SA_KEY[n] registers.
9. Next load the initialization vector in the SA state registers (PKTE_STATE_IV[n]).
10. Finally, write anything in to the PKTE_SA_RDY register to trigger the operation.
11. The input data can now be written into the data I/O buffer starting at PKTE_DATAIO_BUF.
12. After the data is written, write the length (or next multiple of 4) into PKTE_INBUF_CNT register.
13. Poll PKTE_STAT to see if any errors occurred or if the operation completed without errors.

Once the operation is done, the result can be found in the same data I/O buffer.

ADSP-2184x PKTE Register Descriptions

Security Packet Engine (PKTE) contains the following registers.

Table 39-29: ADSP-2184x PKTE Register List

Name	Description
PKTE_ARC4STATE_ADDR	Packet Engine ARC4 State Record Address
PKTE_ARC4STATE_BUF	Starting Entry of 256-byte ARC4 State Buffer
PKTE_BUF_PTR	Packet Engine Buffer Pointer Register
PKTE_BUF_THRESH	Packet Engine Buffer Threshold Register
PKTE_CDRBASE_ADDR	Packet Engine Command Descriptor Ring Base Address
PKTE_CDSC_CNT	Packet Engine Command Descriptor Count Register
PKTE_CDSC_INCR	Packet Engine Command Descriptor Count Increment Register

Table 39-29: ADSP-2184x PKTE Register List (Continued)

Name	Description
PKTE_CFG	Packet Engine Configuration Register
PKTE_CLK_CTL	PE Clock Control Register
PKTE_CONT	PKTE Continue Register
PKTE_CTL_STAT	Packet Engine Control Register
PKTE_DATAIO_BUF	Starting Entry of 256-byte Data Input/Output Buffer
PKTE_DEST_ADDR	Packet Engine Destination Address
PKTE_DMA_CFG	Packet Engine DMA Configuration Register
PKTE_ENDIAN_CFG	Packet Engine Endian Configuration Register
PKTE_HLT_CTL	Packet Engine Halt Control Register
PKTE_HLT_STAT	Packet Engine Halt Status Register
PKTE_IMSK_DIS	Interrupt Mask Disable Register
PKTE_IMSK_EN	Interrupt Mask Enable Register
PKTE_IMSK_STAT	Interrupt Masked Status Register
PKTE_INBUF_CNT	Packet Engine Input Buffer Count Register
PKTE_INBUF_INCR	Packet Engine Input Buffer Count Increment Register
PKTE_INT_CFG	Interrupt Configuration Register
PKTE_INT_CLR	Interrupt Clear Register
PKTE_INT_EN	Interrupt Enable Register
PKTE_IUMSK_STAT	Interrupt Unmasked Status Register
PKTE_LEN	Packet Engine Length Register
PKTE_OUTBUF_CNT	Packet Engine Output Buffer Count Register
PKTE_OUTBUF_DECR	Packet Engine Output Buffer Count Decrement Register
PKTE_PE_ALT_KEY_STATUS	PE Alternative Key stat register
PKTE_PE_CACHE_CTRL_0	Packet Engine Cache Control 0
PKTE_PE_CACHE_CTRL_1	Packet Engine Cache Control 1
PKTE_RDRBASE_ADDR	Packet Engine Result Descriptor Ring Base Address
PKTE_RDSC_CNT	Packet Engine Result Descriptor Count Registers
PKTE_RDSC_DECR	Packet Engine Result Descriptor Count Decrement Registers
PKTE_RING_CFG	Packet Engine Ring Configuration
PKTE_RING_PTR	Packet Engine Ring Pointer Status
PKTE_RING_STAT	Packet Engine Ring Status

Table 39-29: ADSP-2184x PKTE Register List (Continued)

Name	Description
PKTE_RING_THRESH	Packet Engine Ring Threshold Registers
PKTE_SA_ADDR	Packet Engine SA Address
PKTE_SA_ARC4IJPTR	ARC4 i and j Pointer Register
PKTE_SA_CMD0	SA Command 0
PKTE_SA_CMD1	SA Command 1
PKTE_SA_IDIGEST[n]	SA Inner Hash Digest Registers
PKTE_SA_KEY[n]	SA Key Registers
PKTE_SA_NONCE	SA Initialization Vector Register
PKTE_SA_ODIGEST[n]	SA Outer Hash Digest Registers
PKTE_SA_RDY	SA Ready Indicator
PKTE_SA_SEQNUM[n]	SA Sequence Number Register
PKTE_SA_SEQNUM_MSK[n]	SA Sequence Number Mask Registers
PKTE_SA_SPI	SA SPI Register
PKTE_SRC_ADDR	Packet Engine Source Address
PKTE_STAT	Packet Engine Status Register
PKTE_STATE_ADDR	Packet Engine State Record Address
PKTE_STATE_BYTE_CNT[n]	State Hash Byte Count Registers
PKTE_STATE_IDIGEST[n]	State Inner Digest Registers
PKTE_STATE_IV[n]	State Initialization Vector Registers
PKTE_USERID	Packet Engine User ID

Packet Engine ARC4 State Record Address

The PKTE_ARC4STATE_ADDR register holds the start address of the SA ARC4 state record.

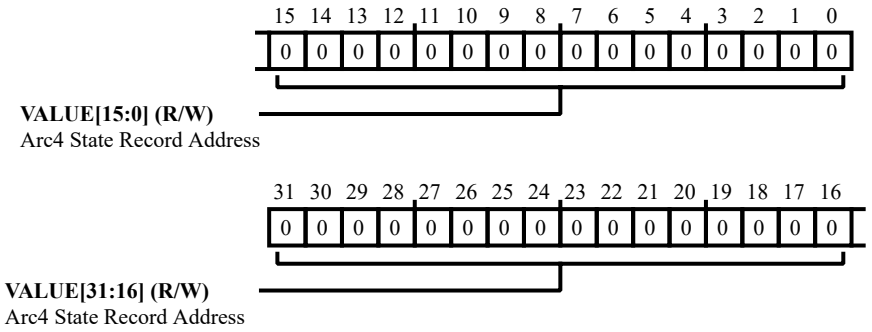


Figure 39-4: PKTE_ARC4STATE_ADDR Register Diagram

Table 39-30: PKTE_ARC4STATE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Arc4 State Record Address.

Starting Entry of 256-byte ARC4 State Buffer

The PKTE_ARC4STATE_BUF register is used to store the pre-processed key that initializes the ARC4 module. In direct host mode, before processing starts, the Host must write the ARC4 state, starting from the base address and increment the address pointer for each write. When processing completes the Host must read the ARC4 state and copy it to the local Host maintained state record. After a reset, a read from any address in the address range of the ARC4 buffer returns an undefined value.

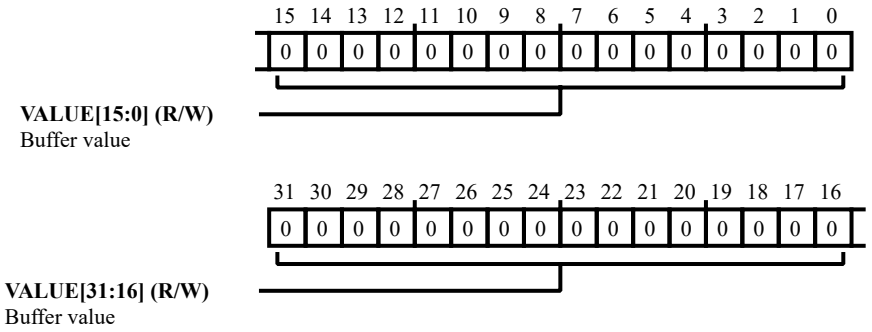


Figure 39-5: PKTE_ARC4STATE_BUF Register Diagram

Table 39-31: PKTE_ARC4STATE_BUF Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Buffer value. The PKTE_ARC4STATE_BUF.VALUE bit field stores the pre-processed key that initializes the ARC4 module.

Packet Engine Buffer Pointer Register

The PKTE_BUF_PTR register contains the offset of the next buffer address (entry) to be read or written by the packet engine. This register is used in direct host mode only.

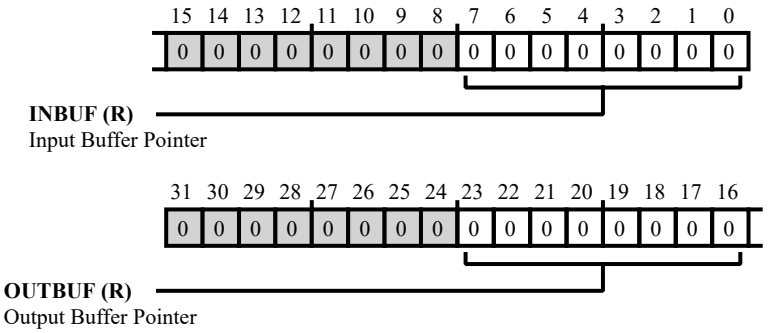


Figure 39-6: PKTE_BUF_PTR Register Diagram

Table 39-32: PKTE_BUF_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/NW)	OUTBUF	Output Buffer Pointer. The PKTE_BUF_PTR.OUTBUF bit field indicates the offset of the next address (entry) in the output buffer that will be written next by the packet engine. This bit field is reset to zero after starting up and decremented by 4 at every output buffer write operation. Pointers wrap around; the maximum value this field can have equals the output buffer size minus 4.
7:0 (R/NW)	INBUF	Input Buffer Pointer. The PKTE_BUF_PTR.INBUF bit field indicates the offset of the next address (entry) in the input buffer that will be read next by the packet engine. The bit field is reset to zero after starting up and incremented by 4 at every input buffer read operation. Pointers wrap around; the maximum value this field can have equals the input buffer size minus 4.

Packet Engine Buffer Threshold Register

When in autonomous ring mode or target command mode, the PKTE_BUF_THRESH register defines the high- and low-level value at which the packet engine starts to transfer packet data in or out of the internal packet buffers. These parameters can be used to control the DMA burst size for packet data input and output from the packet engine. In direct host mode, this register contains both threshold values to reduce the amount of packet engine interrupts.

The input buffer threshold (ibufthrsh) interrupt indicates that the input buffer counter is less than or equal to the input buffer threshold value set in this register - this interrupt can be used to wake up a process that stalled on a full input buffer.

The output buffer threshold (obufthrsh) interrupt indicates that the output buffer counter exceeds the output buffer threshold setting. The output buffer interrupt remains active until the output buffer counter is decremented to zero again.

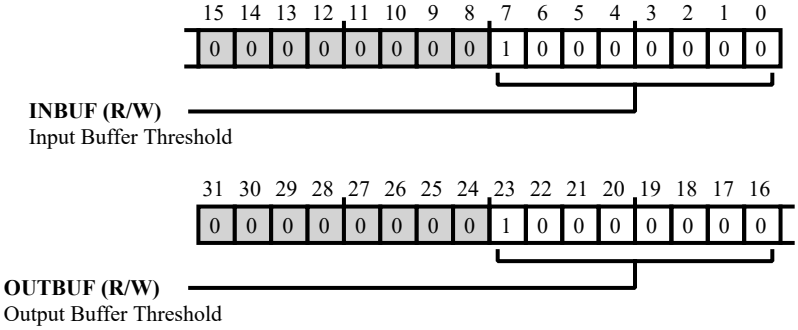


Figure 39-7: PKTE_BUF_THRESH Register Diagram

Table 39-33: PKTE_BUF_THRESH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	OUTBUF	Output Buffer Threshold. The PKTE_BUF_THRESH.OUTBUF bit field specifies how many bytes must be available in the packet engine output buffer before an output transfer starts. Valid values range from 0 to 252, in multiples of 4. In autonomous ring mode, a value of 128 generally gives a good performance, but the optimal value depends on the system and application. In direct host mode, the output buffer threshold (obufthrsh) interrupt activates when the output buffer counter for the output buffer exceeds the value set in this field. A value of 128 generally gives a good performance, but the optimal value depends on the system and application.

Table 39-33: PKTE_BUF_THRESH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	INBUF	Input Buffer Threshold. The `PKTE_BUF_THRESH.INBUF` bit field specifies how many bytes must be free in the packet engine input buffer before an input transfer starts. Valid values range from 0 to 252, in multiples of 4. In autonomous ring mode, a value of 128 generally gives a good performance, but the optimal value depends on the system and application. In direct host mode, the input buffer threshold (ibufthrsh) interrupt activates when the input buffer counter for the input buffer is below or equal the value set in this field. A value of 128 generally gives a good performance, but the optimal value depends on the system and application.

Packet Engine Command Descriptor Ring Base Address

The PKTE_CDRBASE_ADDR register holds the command descriptor ring base address in host memory. It is only applicable in autonomous ring mode. The PKTE_CDRBASE_ADDR register is ignored for all other modes when command descriptors are directly written into the descriptor registers.

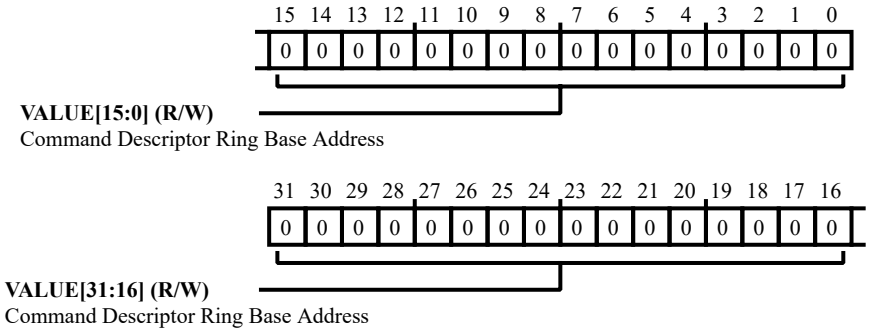


Figure 39-8: PKTE_CDRBASE_ADDR Register Diagram

Table 39-34: PKTE_CDRBASE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Descriptor Ring Base Address. The PKTE_CDRBASE_ADDR.VALUE bit field specifies the base location of the command descriptor ring in the host memory space.

Packet Engine Command Descriptor Count Register

The PKTE_CDSC_CNT register holds the counter for the number of descriptors in the Command Descriptor Ring (CDR). It is decremented by the packet engine each time a valid descriptor is read from the CDR and processed.

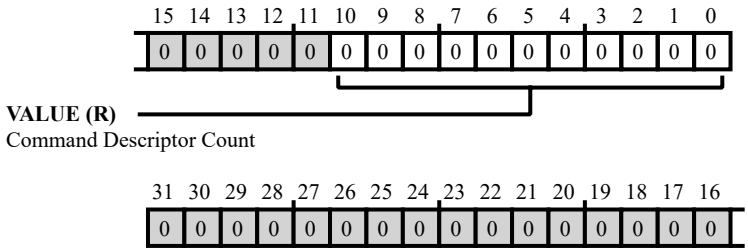


Figure 39-9: PKTE_CDSC_CNT Register Diagram

Table 39-35: PKTE_CDSC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Command Descriptor Count. The PKTE_CDSC_CNT.VALUE bit field provides the number of command descriptors in the command descriptor ring. The packet engine decrements the counter when a valid command descriptor is read from the CDR and processed.

Packet Engine Command Descriptor Count Increment Register

The PKTE_CDSC_INCR register is accessible by the host connected through the system completer bus. The host can increment the command descriptor counter by writing a value between 1 and 255 to the lowest byte of this register.

In autonomous ring mode, the host must prepare 1 to 255 valid command descriptors in the CDR and then write this register with a value between 1 and 255. The write triggers the packet engine to fetch the command descriptors from the CDR. In direct host mode or target command mode, the host must write one valid command descriptor to the internal descriptor registers and then write this register with the value 1, to indicate that one valid descriptor is available.

A CDR threshold interrupt is activated when the command descriptor counter is less than or equal to the threshold value set in the PKTE_RING_THRESH register. This interrupt can be used to wake up a process that stalled on a full CDR.

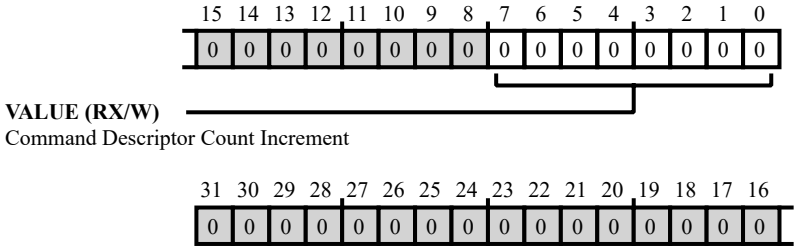


Figure 39-10: PKTE_CDSC_INCR Register Diagram

Table 39-36: PKTE_CDSC_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (RX/W)	VALUE	Command Descriptor Count Increment. The value written to the PKTE_CDSC_INCR.VALUE bit field is added to the command descriptor counter. The counter is protected against overflow (see the PKTE_RING_STAT register description). Note that bits[10:8] should be written with zeros.

Packet Engine Configuration Register

The PKTE_CFG register is used to select static settings that control the packet-processing path. This register is typically the last one to be written during the initialization sequence. These settings are typically set at initialization and not changed again.

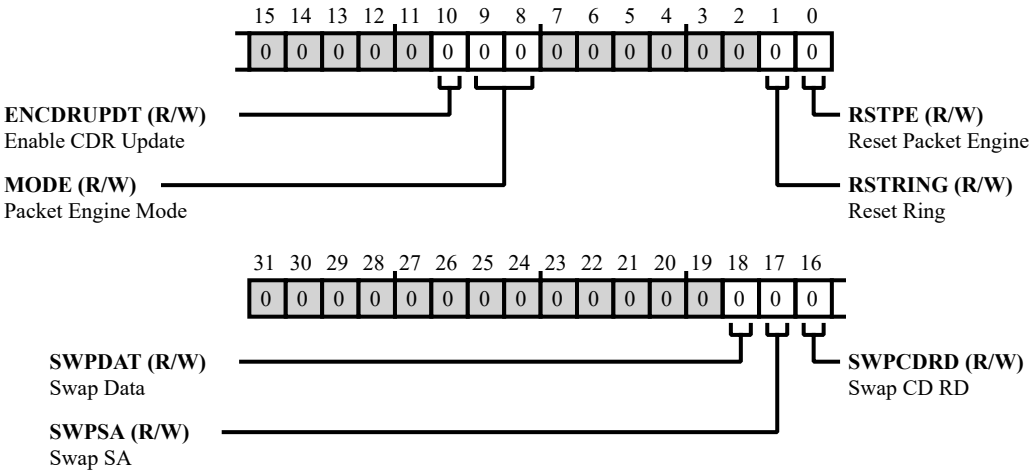


Figure 39-11: PKTE_CFG Register Diagram

Table 39-37: PKTE_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	SWPDAT	Swap Data. The PKTE_CFG.SWPDAT bit enables endian swap for packet data as configured in the PKTE_ENDIAN_CFG.MBSWAP bits for the packet data DMA read and write.	
		0	No Endian Swap
		1	Apply Endian Swap
17 (R/W)	SWPSA	Swap SA. The PKTE_CFG.SWPSA bit enables endian swap for a SA record as configured in the PKTE_ENDIAN_CFG.MBSWAP bits for the SA record and state record DMA read and write. If the PKTE_ENDIAN_CFG.MBSWAP bits specify "no endian swap", this bit is ignored.	
		0	No Endian Swap
		1	Apply Endian Swap

Table 39-37: PKTE_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	SWPCDRD	Swap CD RD. The PKTE_CFG.SWPCDRD bit enables endian swap for descriptors as configured in the PKTE_ENDIAN_CFG.MBSWAP bits for the command descriptor DMA read and result descriptor DMA write. If the PKTE_ENDIAN_CFG.MBSWAP bits specify "no endian swap", this bit is ignored.	
		0	No Endian Swap
		1	Apply Endian Swap
10 (R/W)	ENCDRUPDT	Enable CDR Update. The PKTE_CFG.ENCDRUPDT bit enables the packet engine to update, (clear the ownership bits) in the command descriptor in the CDR.	
		0	Do Not Clear Ownership Bits. The packet engine does not clear the ownership bits in the command descriptor when it completes an operation. The host application must clear the ownership bits in "old descriptors" before the packet engine is allowed to wrap around the CDR to re-encounter these "old descriptors". This setting has the advantage of eliminating a separate DMA write to the CDR.
		1	Clear Ownership Bits. The packet engine clears (set to zero) the ownership bits in the current command descriptor in the CDR. This prevents the packet engine from re-processing an "old descriptor" when it wraps around the CDR.
9:8 (R/W)	MODE	Packet Engine Mode. The PKTE_CFG.MODE bit field selects how the packet engine receives commands.	
		0	Direct Host Mode.
		1	Target Command Mode with Result Descriptor Ring Disabled.
		2	Target Command Mode with Result Descriptor Ring Enabled.
		3	Autonomous Ring Mode

Table 39-37: PKTE_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	RSTRING	Reset Ring. The PKTE_CFG.RSTRING bit resets the internal counters for the CDR and RDR, PKTE_CDSC_CNT and PKTE_RDSC_CNT registers) to zero. Resets the PKTE_RING_PTR register to the base address. After the reset the rings are empty. This bit must be written with a '1' to reset the descriptor ring manager and then re-written with a '0' to release the reset. Note that this bit can remain in the reset state if the CDR ring is disabled (PKTE_CFG.MODE is not 0b11). Note that this reset must be coordinated with the 'owner' of the descriptor ring to ensure that the pointers are in sync after the reset.	
		0	Release the Descriptor Ring Manager Reset
		1	Reset the Descriptor Ring Manager
0 (R/W)	RSTPE	Reset Packet Engine. The PKTE_CFG.RSTPE bit resets the packet engine and the state machine logic that drives header processing, DMA, and context management. The PKTE_CFG.RSTPE bit resets the PKTE_CTL_STAT and PKTE_LEN internal registers. This bit must be written with a 1 to reset the packet engine and then re-written with a 0 to release the reset. Note that this bit should not be used by a typical application. It is provided to use during development testing or to recover from critical errors. Note that the PKTE_CTL_STAT.PADVAL and PKTE_CTL_STAT.PADCTLSTAT bit fields are only reset when in autonomous ring mode, but the PKTE_CTL_STAT.PRNGMD bit is not reset. Halt mode is not affected by this reset as well. When exiting out of halt mode, a HW reset is required or a write to the PKTE_CONT register.	
		0	Release the Packet Engine Reset
		1	Reset the Packet Engine

PE Clock Control Register

The PKTE_CLK_CTL register controls the clock enable signals. This register can be used to enable the clock for read and write access to SA registers or to enable the required clock signals for certain crypto functions. The setting of this register overrides the packet engine dynamic clock enable.

In autonomous ring mode and target command modes, this register can be all zeros; the packet engine dynamically requests the external clock manager to activate the module clocks. This register can be used in combination with the debugging interface for internal register access.

In direct host mode, the clock enable bits for the packet engine (PKTE_CLK_CTL.ENPECLK) and for ARC4 (PKTE_CLK_CTL.ENARC4CLK) must be enabled to write and read the SA record and state record registers. All module clocks that are required for the current operation must be enabled during processing.

Note that all the clocks are enabled by default to reset all the registers within the packet engine. After a system reset the host can program this register to disable clocks for power reduction.

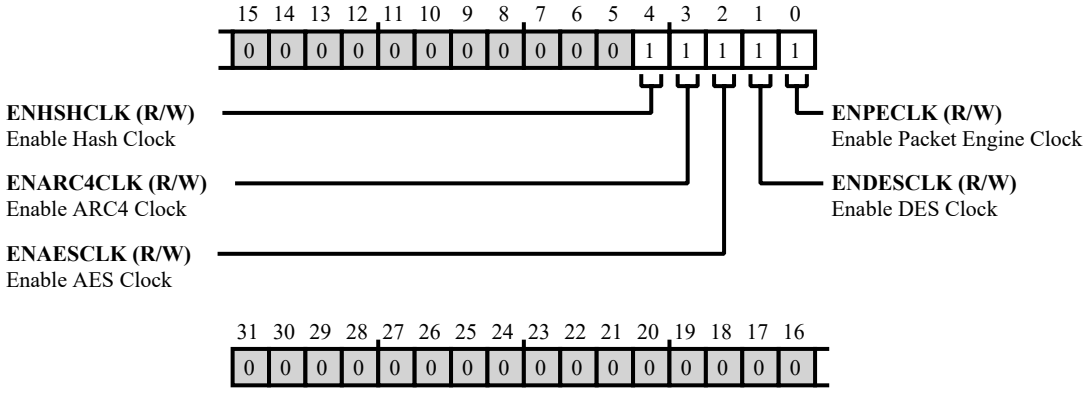


Figure 39-12: PKTE_CLK_CTL Register Diagram

Table 39-38: PKTE_CLK_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	ENHSHCLK	Enable Hash Clock. The PKTE_CLK_CTL.ENHSHCLK bit enables the clock to the hash functions.	
		0	Do not enable the hash clock
		1	Enable the hash clock
3 (R/W)	ENARC4CLK	Enable ARC4 Clock. The PKTE_CLK_CTL.ENARC4CLK bit enables the clock to the ARC4 function.	
		0	Do not enable the ARC4 clock
		1	Enable the ARC4 clock

Table 39-38: PKTE_CLK_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	ENAESCLK	Enable AES Clock. The `PKTE_CLK_CTL.ENAESCLK` bit enables the clock to the AES encrypt/decrypt function.	
		0	Do not enable the AES clock
		1	Enable the AES clock
1 (R/W)	ENDESCLK	Enable DES Clock. The `PKTE_CLK_CTL.ENDESCLK` bit enables the clock to the DES function.	
		0	Do not enable the DES clock
		1	Enable the DES clock
0 (R/W)	ENPECLK	Enable Packet Engine Clock. The `PKTE_CLK_CTL.ENPECLK` bit enables the clock in the PKTE data path.	
		0	Do not enable the PKTE data path clock
		1	Enable the PKTE data path clock

PKTE Continue Register

A write to the PKTE_CONT register (with any value) releases the packet engine from a halt state when in halt mode.

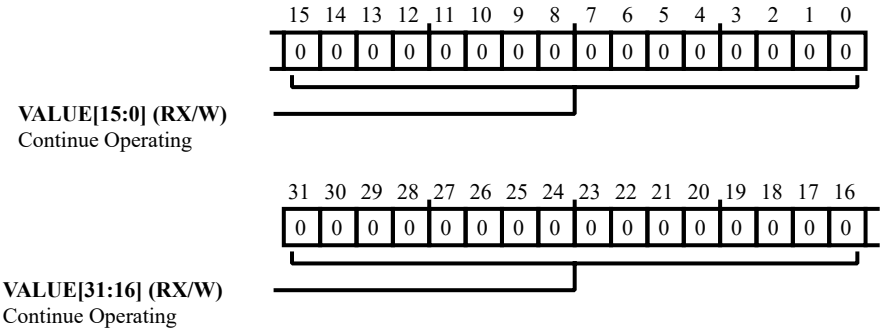


Figure 39-13: PKTE_CONT Register Diagram

Table 39-39: PKTE_CONT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Continue Operating. The PKTE_CONT.VALUE bit field releases the packet engine from a halt state when written with any value in halt mode.

Packet Engine Control Register

The PKTE_CTL_STAT register has a dual function. Together with the data in the SA, this register provides the basic command information for the packet engine to process a packet. When the packet engine successfully or unsuccessfully completes an operation, the packet engine control/status register provides the result status for the host.

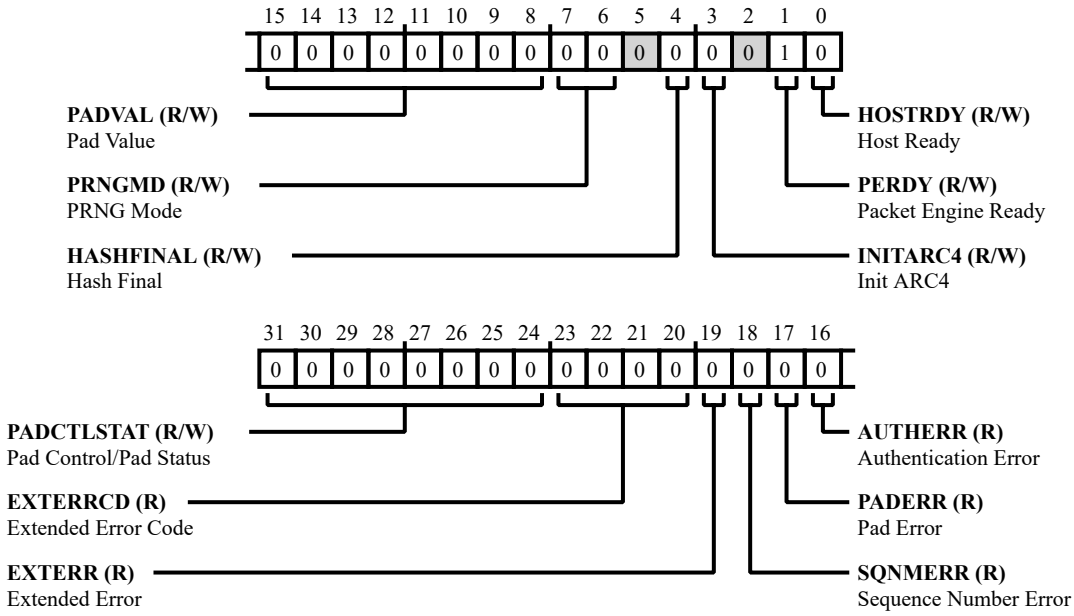


Figure 39-14: PKTE_CTL_STAT Register Diagram

Table 39-40: PKTE_CTL_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	PADCTLSTAT	Pad Control/Pad Status. The PKTE_CTL_STAT.PADCTLSTAT bit field is used to control the pad boundary for pad insertion (outbound) and after processing returns the number of inserted (outbound) or detected (inbound) pad bytes. For the command descriptor, the enumerations below provide the codes for the pad boundary for the outbound operations. This can be used for "traffic flow security" to conceal the number of payload bytes in an encrypted packet. For the result descriptor inbound operations that use pad types SSL, TLS, IPsec or PKCS#7, it returns the number of detected pad bytes. For all other inbound operations, it returns zero since the other pad modes do not allow implicit determination of pad count. If a "pad verify failure" occurs, it returns zero. For an outbound operation, it returns the number of inserted pad bytes for all pad types. The pad value includes added bytes such as the pad length and the next header field in an IPsec ESP pad type.	
		0	Align packet end to modulo 8-byte boundary

Table 39-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	Align packet end to modulo 1-byte boundary
		2	Align packet end to modulo 4-byte boundary
		4	Align packet end to modulo 8-byte boundary
		8	Align packet end to modulo 16-byte boundary
		16	Align packet end to modulo 32-byte boundary
		32	Align packet end to modulo 64-byte boundary
		64	Align packet end to modulo 128-byte boundary
		128	Align packet end to modulo 256-byte boundary
23:20 (R/NW)	EXTERRCD	Extended Error Code. The PKTE_CTL_STAT.EXTERRCD bit field represents an encoded error condition.	
19 (R/NW)	EXTERR	Extended Error. The PKTE_CTL_STAT.EXTERR bit field provides an extended error code.	
		0	No Extended Error
		1	Extended Error
18 (R/NW)	SQNMERR	Sequence Number Error. The PKTE_CTL_STAT.SQNMERR bit indicates that for an inbound operation, there was a fault in the anti-replay sequence number. For an outbound operation, there was a sequence number overflow condition.	
		0	No Sequence Number Error
		1	Sequence Number Error
17 (R/NW)	PADERR	Pad Error. The PKTE_CTL_STAT.PADERR bit indicates that for an inbound operation the decrypted pad does not match the expected values.	
		0	No Pad Error
		1	Pad Error
16 (R/NW)	AUTHERR	Authentication Error. The PKTE_CTL_STAT.AUTHERR bit indicates that for an inbound operation the authentication value in the packet does not match the computed value.	
		0	No Authentication Error
		1	Authentication Error

Table 39-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/W)	PADVAL	Pad Value. The PKTE_CTL_STAT.PADVAL bit field is used to pass the pad value between the host and the packet engine. Command Descriptor: (write-only) For outbound operations that use pad type IPsec, the host must populate this field with the value that is to be inserted into the next header field. For the IPsec ESP operation, this next header is part of the ESP trailer of the innermost operation's header and the value must be 50 decimal. For outbound encrypt operations that use the pad type constant or constant SSL, the host must specify the fixed constant value in this field. For all other outbound and inbound operations, this field is not used. Result Descriptor: (read only) For inbound operations that use pad type IPsec, the packet engine returns the next header field that it detects. For IPsec ESP inbound operations, this is the next header field in the innermost operation's header, which will typically be the value for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination option headers, another header value could be seen. For all other outbound operations, the packet engine will not update this field. For all other inbound operations, the returned pad value is zero.	
7:6 (R/W)	PRNGMD	PRNG Mode. The PKTE_CTL_STAT.PRNGMD bits select the pseudo-random number generator mode.	
		0	Operation does not use the PRNG function. Operation does not use the PRNG function.
		1	PRNG Init. PRNG is initialized with a SEED, KEY and an LFSR value as defined in the SA.
		2	PRNG Generate. Pseudo-random data is generated with the LFSR as input value. Before this mode can be used, the PRNG must be initialized with a valid SEED, KEY and LFSR using PRNG Init (PKTE_CTL_STAT.PRNGMD=b'01).
		3	PRNG Test. It can be used to test the PRNG function with custom input data. Before this mode can be used, the PRNG must be initialized once with a valid SEED using PRNG Init (PKTE_CTL_STAT.PRNGMD=b'01).

Table 39-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	HASHFINAL	Hash Final. When the PKTE_CTL_STAT.HASHFINAL bit is zero, the data to be hashed must be a multiple of the hash block size, 64 bytes for SHA-1, MD5, SHA-224, SHA-256. This bit is only applicable for Basic Hash, Basic Encrypt-Hash and Basic Hash-Decrypt operations that use the SHA-1, MD5, SHA-224, SHA-256 hash algorithm. The PKTE_CTL_STAT.HASHFINAL bit is overruled for HMAC operations that always completes the hash and always returns the last written value on a read by the host.	
		0	Perform Intermediate Hash Operation. The packet engine performs an intermediate hash operation by generating an intermediate hash digest on the data presented on the input. No hash pad is applied.
		1	Perform Final Hash Operation. The packet engine appends the required final hash pad and generates the final hash digest on the data presented on the input. This completes the hash operation.
3 (R/W)	INITARC4	Init ARC4. The PKTE_CTL_STAT.INITARC4 bit initializes the ARC4 crypto algorithm with a new key. This bit always returns the last written value on a read by the host. This bit is only applicable for operations that use the ARC4 algorithm and must be zero for all other operations.	
		0	Load ARC4 State and ARC4 i/j pointer from the SA The ARC4 State and ARC4 i/j pointer are loaded from the SA to continue the encrypt/decrypt processing from the previous algorithm state.
		1	Read ARC4 key from the SA-record and initialize ARC4 S-boxes using this key The ARC4 key is read from the SA-record, the ARC4 S-boxes are initialized using this key, prior to the encryption/decryption of data. This bit overrules Stateful mode as defined in bits [9:8] of PKTE_SA_CMD1.
1 (R/W)	PERDY	Packet Engine Ready. The PKTE_CTL_STAT.PERDY bit indicates that the packet engine has completed processing the command descriptor and returns the result descriptor with ownership set to the host. This bit can be reset to 0 by the host and the packet engine, but only the packet engine can set this bit. When the packet engine is idle (not processing), this bit always returns '1' on a read by the host.	

Table 39-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	HOSTRDY	Host Ready. The `PKTE_CTL_STAT.HOSTRDY` bit indicates that the host has populated the command descriptor. This bit can be reset to 0 by the host and the packet engine, but only the host can set this bit. When the packet engine is idle (not processing), this bit always returns '0' on a read by the host.

Starting Entry of 256-byte Data Input/Output Buffer

When in direct host mode, the source packet data is written here to be transferred to the packet engine. The host can monitor the available space in the input buffer through the PKTE_STAT register. This is also the location in the packet engine from where output data is read when in direct host mode. The host can monitor the available bytes in the output buffer through the PKTE_STAT register.

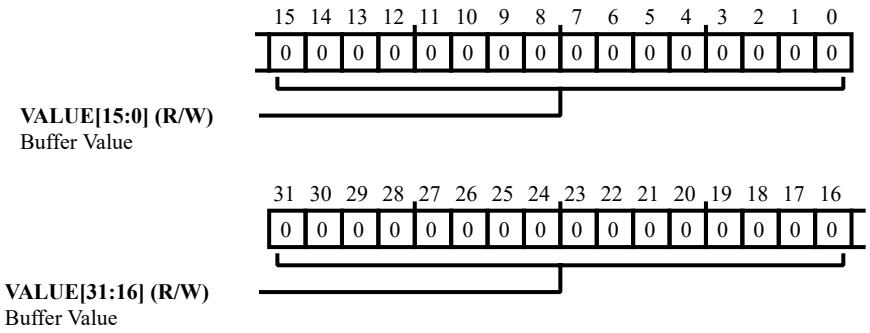


Figure 39-15: PKTE_DATAIO_BUF Register Diagram

Table 39-41: PKTE_DATAIO_BUF Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Buffer Value.

Packet Engine Destination Address

The PKTE_DEST_ADDR register holds the starting (byte) address to write the result packet from the requested operation.

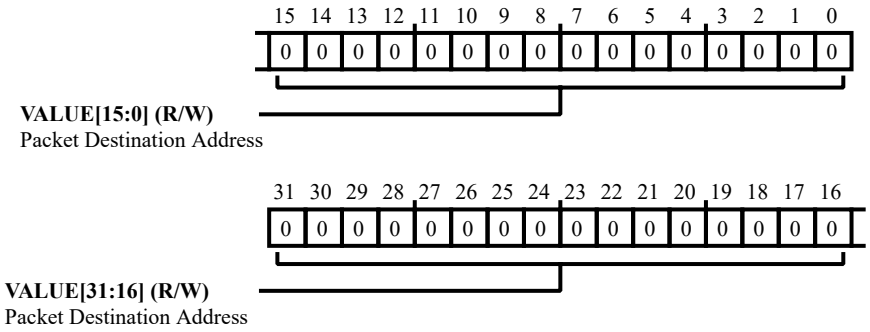


Figure 39-16: PKTE_DEST_ADDR Register Diagram

Table 39-42: PKTE_DEST_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Packet Destination Address.

Packet Engine DMA Configuration Register

The PKTE_DMA_CFG register configures the maximum burst transfer size, enables incremental transfers, and insertions IDLE cycles between two bus transfers.

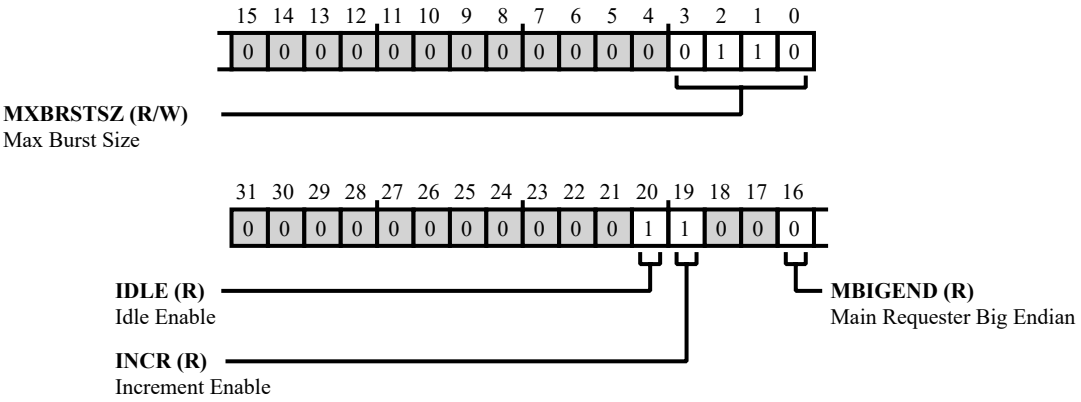


Figure 39-17: PKTE_DMA_CFG Register Diagram

Table 39-43: PKTE_DMA_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	IDLE	Idle Enable. The PKTE_DMA_CFG.IDLE bit allows the peripheral bus requester to insert one additional IDLE transfer between two successive peripheral bus requester burst operations. This provides the arbiter one additional cycle to hand over the grant to another peripheral bus requester.	
		0	The peripheral bus requester inserts no IDLE cycle between two successive burst operations
		1	The peripheral bus requester inserts one additional IDLE transfer between two successive burst operations

Table 39-43: PKTE_DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	INCR	Increment Enable. The PKTE_DMA_CFG.INCR bit lets the peripheral bus requester generate INC4, INC8 and INC16 type of burst transfers. By default, the peripheral bus requester generates the largest possible incremental burst of unspecified length (INCR) with a maximum length (in bytes) as configured by the PKTE_DMA_CFG.MXBRSTSZ bit field. In case there are less than 4 bytes of data available or the 1kB boundary will be crossed using a burst operation, then a single transfer of size byte is generated. When the PKTE_DMA_CFG.INCR bit is set, the peripheral bus requester generates one or more incremental burst of specified length (INC4, INC8, INC16). In case there is less data available then the smallest possible burst (INC4) or the 1kB boundary will be crossed using a burst operation, then an unspecified length burst or a single transfer of size byte is generated.	
		0	The bus requester will generate only INCR burst types
		1	The bus requester will generate INC4, INC8 and INC16 burst types
16 (R/NW)	MBIGEND	Main Requester Big Endian. The PKTE_DMA_CFG.MBIGEND bit determines whether the engine is used in a little or big endian system.	
		0	Little endian
		1	Big endian
3:0 (R/W)	MXBRSTSZ	Max Burst Size. The PKTE_DMA_CFG.MXBRSTSZ bit field configures the maximum size of an unspecified length burst (INC) at the bus in bytes. When there is less data available than the PKTE_DMA_CFG.MXBRSTSZ bit field setting or the 1kB boundary will be crossed using a burst operation, then the length of the burst can be less than PKTE_DMA_CFG.MXBRSTSZ. Any requested transfers larger than this size are broken up in to multiple burst transfers of this size or less.	

Packet Engine Endian Configuration Register

The packet engine incorporates a powerful interface specific endian handler. This endian handler allows byte lane swapping in each direction for data passing through the host interface.

The PKTE_ENDIAN_CFG register configures the byte order function for the peripheral bus requester and peripheral bus completer interface. The bits for the peripheral bus requester are combined in four sets of two bits; each group configures a byte swap function for a particular DMA transfer. The same applies for the peripheral bus completer interface.

The PKTE_ENDIAN_CFG register also defines the endian swapping that occurs for host-initiated target transfers and for packet engine requester DMA read and write transfers. Individual endian swap enable bits in the configuration (PKTE_CFG) register can enable the endian swap for various transaction types: command descriptors and result descriptors, SA records and state records, packet data.

In direct host mode, only target operations are supported. Only the target endian configuration of this register is applicable.

Note: This register is typically programmed once during the initialization phase, although software is allowed to dynamically change the setting in this register just before initiating a data transfer. The developer will have to analyze the benefit of the cycles needed to write the endian register dynamically versus handling endian swapping for some data structures in the host system (most modern processors support a byte swap in zero cycles). Certainly the endian swap should be set correctly for the packet data, since this represents the majority of the data transferred.

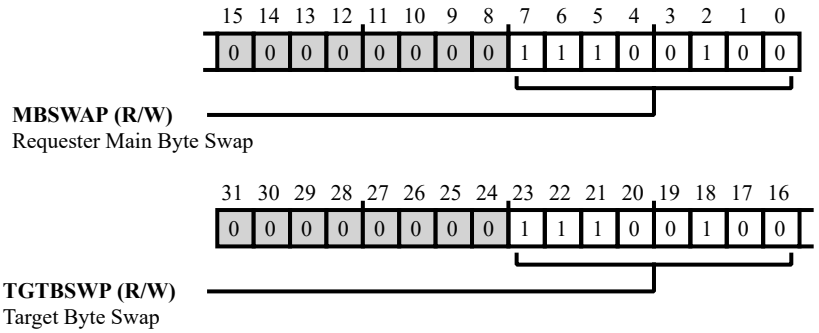


Figure 39-18: PKTE_ENDIAN_CFG Register Diagram

Table 39-44: PKTE_ENDIAN_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	TGTBSWP	Target Byte Swap. The PKTE_ENDIAN_CFG.TGTBSWP bit field configures the byte swap for peripheral bus target transfers. Note that only target word transfers are supported. Each double-bit field in this register specifies the source of the indicated byte lane. The field values are interpreted as follows: 00 = byte 0, 01 = byte 1, 10 = byte 2, 11 = byte 3. Note: Setting the value 0xE4 defines no swap (little endian) and setting value 0x1B defines a full byte swap within a 32-bit word (big endian).
7:0 (R/W)	MBSWAP	Requester Main Byte Swap. The PKTE_ENDIAN_CFG.MBSWAP bit field configures the byte swap for peripheral bus requester multi-byte transfers, including command descriptors, result descriptors, SA records, state records and packet data. Separate controls in the PKTE_CFG register can enable this swap individually for each of the 4 types of data. Each double-bit field in this register specifies the source of the indicated peripheral bus byte lane. The field values are interpreted as follows: 00=byte 0, 01=byte 1, 10=byte 2, 11=byte 3. Note: Setting the value 0xE4 defines no swap (little endian) and setting value 0x1B defines a full byte swap within a 32-bit word (big endian).

Packet Engine Halt Control Register

Th PKTE_HLT_CTL register controls the packet engine halt mode. This register can be used for debugging purposes while processing in autonomous ring mode or target command mode. During the halt mode, the host can read all internal registers for examination without side-effects. When halted, the host should not write to any registers. To continue packet engine operation, the host must write to the PKTE_CONT (PKTE continue) register.

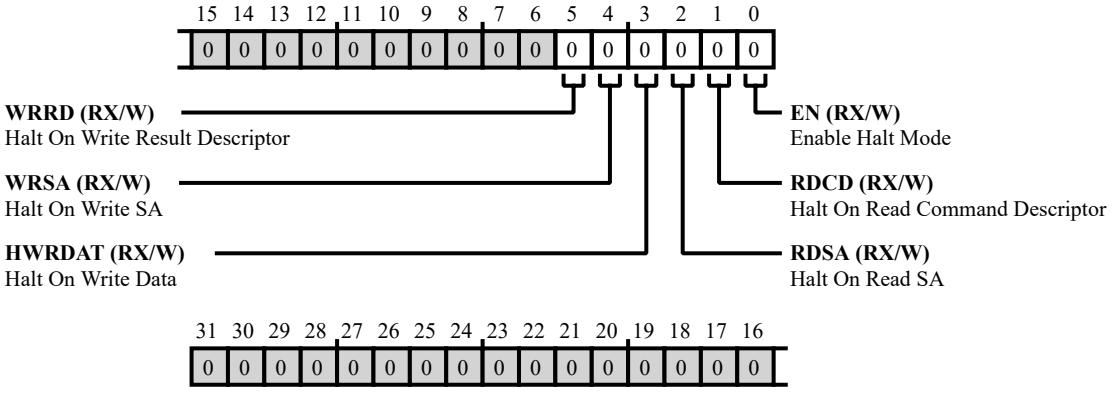


Figure 39-19: PKTE_HLT_CTL Register Diagram

Table 39-45: PKTE_HLT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	WRRD	Halt On Write Result Descriptor. The PKTE_HLT_CTL.WRRD bit halts the packet engine in the HALT_WRITE_STATUS state after it completes a result descriptor write operation to the result descriptor ring. The host can use this bit to examine the result descriptor that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
4 (RX/W)	WRSA	Halt On Write SA. The PKTE_HLT_CTL.WRSA bit halts the packet engine in the HALT_WRITE_SA state after it completes an SA write operation to the host memory. The host can use this bit to examine the security context that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation

Table 39-45: PKTE_HLT_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (RX/W)	HWRDAT	Halt On Write Data. The PKTE_HLT_CTL.HWRDAT bit halts the packet engine in the HALT_DATA state after it completes writing the result packet data to the host memory. The host can use this bit to examine the result packet that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
2 (RX/W)	RDSA	Halt On Read SA. The PKTE_HLT_CTL.RDSA bit halts the packet engine in the HALT_READ_SA state after it completes an SA read operation from the host memory. The host can use this bit to examine the security context that is currently in the SA registers.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
1 (RX/W)	RDCD	Halt On Read Command Descriptor. The PKTE_HLT_CTL.RDCD bit halts the packet engine in the HALT_READ_DESCR state after it completes a command descriptor read operation from the command descriptor ring. It will halt whether the descriptor is valid or invalid. The host can use this bit to examine the command descriptor that is currently in the internal command descriptor registers.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
0 (RX/W)	EN	Enable Halt Mode. The PKTE_HLT_CTL.EN bit enables halt mode where the packet engine can halt processing at any processing state as indicated by bits [5:1]. When halted, the packet engine continues operation on a write to the PKTE_CONT register.	
		0	Do not enable halt mode
		1	Enable halt mode

Packet Engine Halt Status Register

The PKTE_HLT_STAT register reflects the status of the packet engine in halt mode. This register can be used for debugging purposes while processing in autonomous ring mode or target command mode. When the packet engine is halted, the host can read all internal registers for examination without side effects. The host should not write to any registers.

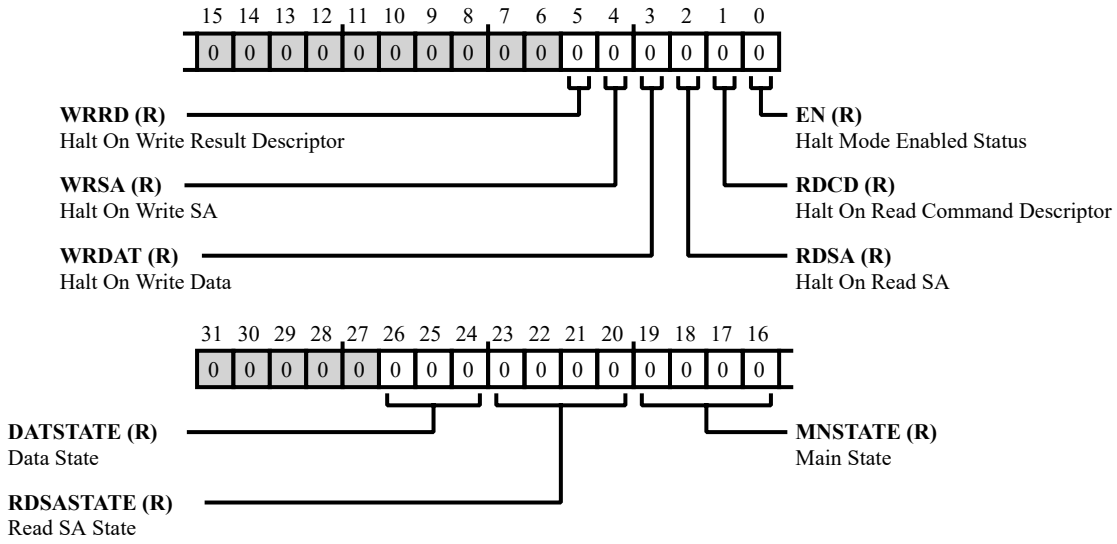


Figure 39-20: PKTE_HLT_STAT Register Diagram

Table 39-46: PKTE_HLT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/NW)	DATSTATE	Data State. The PKTE_HLT_STAT.DATSTATE bit field indicates the state of the packet engine read data FSM.	
		0	DATA_IDLE, no operation
		1	DATA_READ
		2	DATA_WRITE
		3	DATA_WAIT
		5	DATA_PAD_READ
		6	DATA_BYP_READ
		7	RESERVED
23:20 (R/NW)	RDSASTATE	Read SA State. The PKTE_HLT_STAT.RDSASTATE bit field indicates the state of the packet engine read SA FSM.	

Table 39-46: PKTE_HLT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	SA_IDLE, no operation
		1	SA_READ_CMD
		2	SA_READ_STATE_IV
		3-5	RESERVED
		6	SA_READ_ARC4_STATE
		7	SA_READ_WAIT
		8	RESERVED
		9	SA_WRITE_PROT_HDR
		10	RESERVED
		11	SA_WRITE_IV
		12	SA_WRITE_DIGEST
		13	SA_WRITE_ARC4_IJ_PNTR
		14	SA_WRITE_ARC4_STATE
		15	SA_WRITE_WAIT
19:16 (R/NW)	MNSTATE	Main State. The `PKTE_HLT_STAT.MNSTATE` bit field indicates the state of the packet engine main FSM.	
		0	MAIN_IDLE, no operation
		1	MAIN_READ_CD, reading command descriptor
		2	MAIN_READ_SA, reading SA
		3	MAIN_DATA, processing data
		4	MAIN_WRITE_SA, writing SA
		5	MAIN_WRITE_STATUS, writing status
		6	MAIN_WRITE_CD, updating command descriptor
		7	MAIN_WRITE_RD, updating result descriptor
		8	MAIN_INIT_WAIT, wait single clock
		9	MAIN_HALT_READ_CD, halt after read command descriptor
		10	MAIN_HALT_READ_SA, halt after read SA
		11	MAIN_HALT_DATA, halt after processing data
		12	MAIN_HALT_WRITE_SA, halt after write SA

Table 39-46: PKTE_HLT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		13	MAIN_WAIT_FOR_CLOCK, wait for clocks to be active
		15	MAIN_HALT_WRITE_RD, halt after write result descriptor
5 (R/NW)	WRRD	Halt On Write Result Descriptor. The PKTE_HLT_STAT.WRRD bit reflects the value in the PKTE_HLT_CTL.WRRD bit.	
4 (R/NW)	WRSA	Halt On Write SA. The PKTE_HLT_STAT.WRSA bit reflects the value in the PKTE_HLT_CTL.WRSA bit.	
3 (R/NW)	WRDAT	Halt On Write Data. The PKTE_HLT_STAT.WRDAT bit reflects the value in the PKTE_HLT_CTL.HWRDAT bit.	
2 (R/NW)	RDSA	Halt On Read SA. The PKTE_HLT_STAT.RDSA bit reflects the value in the PKTE_HLT_CTL.RDSA bit.	
1 (R/NW)	RDCD	Halt On Read Command Descriptor. The PKTE_HLT_STAT.RDCD bit reflects the value in the PKTE_HLT_CTL.RDCD bit.	
0 (R/NW)	EN	Halt Mode Enabled Status.	
		0	Halt mode not enabled
		1	Halt mode enabled

Interrupt Mask Disable Register

The host can use the PKTE_IMSK_DIS register to clear individual bits in the PKTE_INT_EN register for the host interrupt. This register is a bitmap for each of the possible interrupt sources: A 1 clears the interrupt enable bit, a 0 does not affect the interrupt enable bit in the PKTE_INT_EN register. Clearing the enable bits through this register avoids the time-consuming read-modify-write operation on the host.

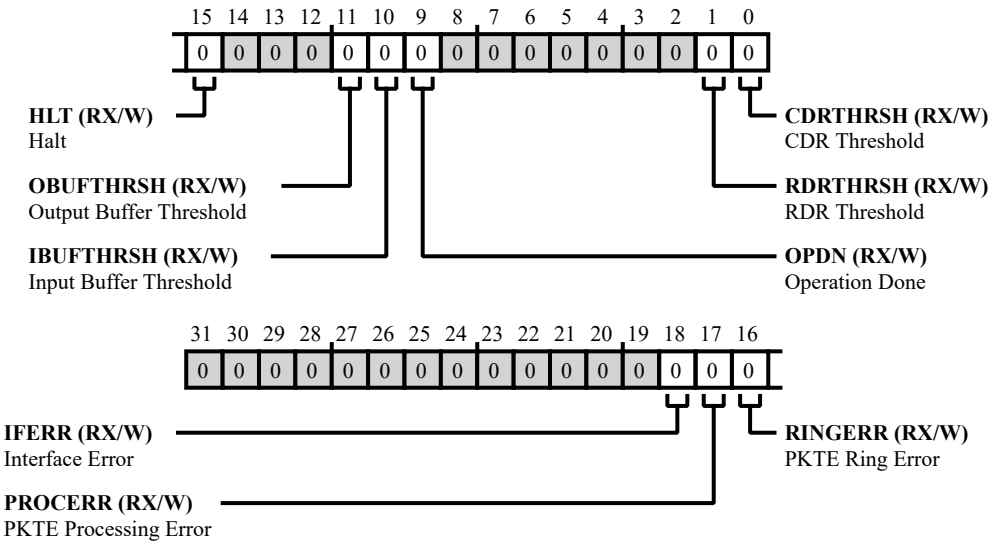


Figure 39-21: PKTE_IMSK_DIS Register Diagram

Table 39-47: PKTE_IMSK_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. Write the PKTE_IMSK_DIS.IFERR bit to clear when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. Write the PKTE_IMSK_DIS.PROCERR bit to clear an extended error that occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error. Write the PKTE_IMSK_DIS.RINGERR bit to clear a CDR overflow or an RDR underflow.
15 (RX/W)	HLT	Halt. Write the PKTE_IMSK_DIS.HLT bit to clear when the packet engine is in the halt state.

Table 39-47: PKTE_IMSK_DIS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. Write the `PKTE_IMSK_DIS.OBUFTHRSH` bit to clear the output buffer counter exceeds the output buffer threshold value defined in `PKTE_BUF_THRESH.OUTBUF` bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. Write the `PKTE_IMSK_DIS.IBUFTHRSH` bit to clear when the input buffer counter is less than or equal to the input buffer threshold value defined in `PKTE_BUF_THRESH.INBUF` bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. Write the `PKTE_IMSK_DIS.RDRTHRSH` bit to clear when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the `PKTE_RING_THRESH.RDRTHRSH` bit, or the RD counter for the RDR in the `PKTE_RDSC_CNT` register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. Write the `PKTE_IMSK_DIS.CDRTHRSH` bit to clear when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the `PKTE_RING_THRESH.CDRTHRSH` bit.

Interrupt Mask Enable Register

The host can use the PKTE_IMSK_EN register to set individual bits in the PKTE_INT_EN register for the host interrupt. This register is a bitmap for each of the possible interrupt sources: A 1 sets the interrupt enable bit, a 0 does not affect the interrupt enable bit in the PKTE_INT_EN register. Setting the enable bits through this register avoids the time-consuming read-modify-write operation on the host.

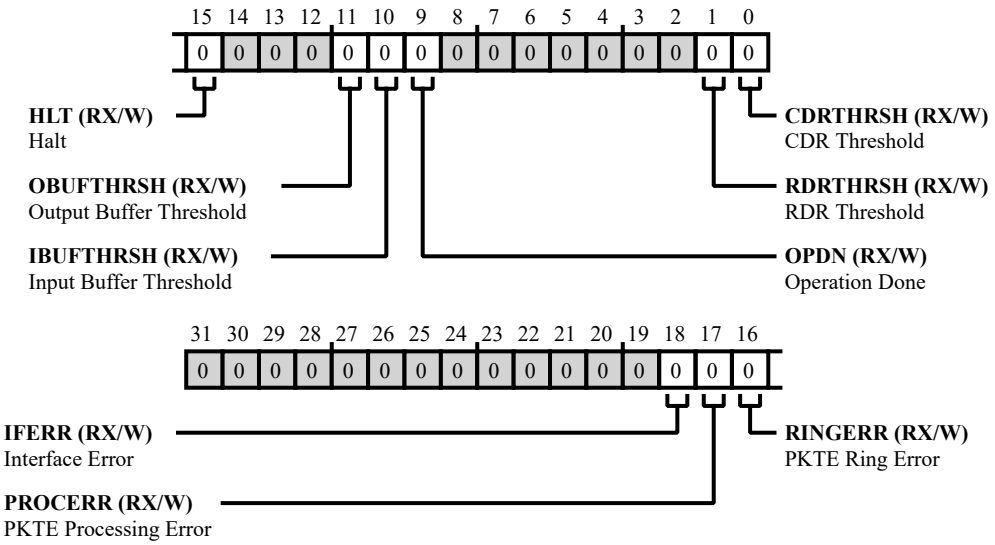


Figure 39-22: PKTE_IMSK_EN Register Diagram

Table 39-48: PKTE_IMSK_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. Set the PKTE_IMSK_EN.IFERR bit to indicate a host request for a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. Set the PKTE_IMSK_EN.PROCERR bit to indicate an extended error occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error.
15 (RX/W)	HLT	Halt. Set the PKTE_IMSK_EN.HLT bit to indicate when the packet engine is in the halt state.

Table 39-48: PKTE_IMSK_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. Set the PKTE_IMSK_EN.OBUFTHRSH bit to indicate that the output buffer counter exceeds the output buffer threshold value defined in the PKTE_BUF_THRESH.OUTBUF bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. Set the PKTE_IMSK_EN.IBUFTHRSH bit to indicate the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. Set the PKTE_IMSK_EN.RDRTHRSH bit to indicate when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. Set the PKTE_IMSK_EN.CDRTHRSH bit to indicate when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Masked Status Register

The PKTE_IMSK_STAT register provides interrupt status visibility to the host, after the interrupt mask is applied. This lets the host view the selected sources of interrupts that are directed to the interrupt output signal, that is connected to the system interrupt controller. As with the unmasked status register, all interrupt bits are latched and must be cleared using the PKTE_INT_CLR register in order to capture a subsequent event. A 1 indicates that the associated interrupt is present.

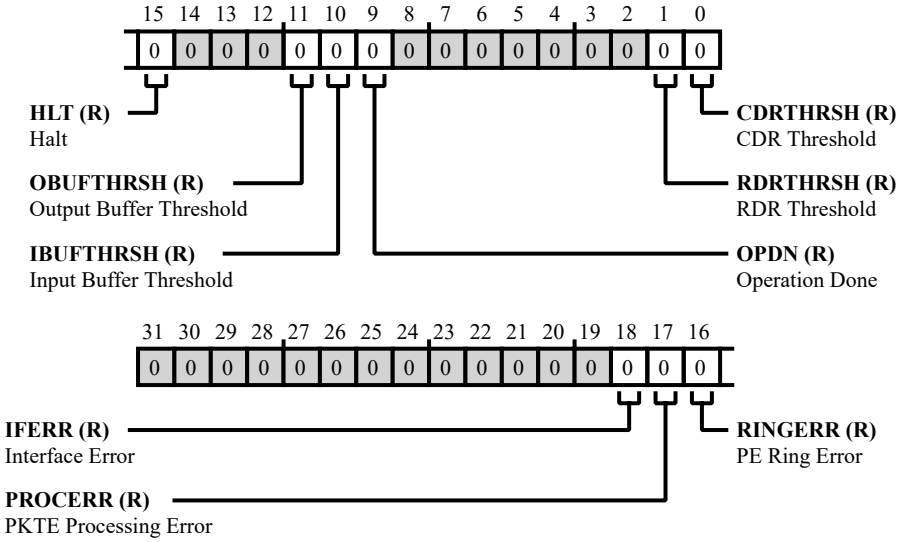


Figure 39-23: PKTE_IMSK_STAT Register Diagram

Table 39-49: PKTE_IMSK_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/NW)	IFERR	Interface Error. The PKTE_IMSK_STAT.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (R/NW)	PROCERR	PKTE Processing Error. The PKTE_IMSK_STAT.PROCERR bit is when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (R/NW)	RINGERR	PE Ring Error. The PKTE_IMSK_STAT.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (R/NW)	HLT	Halt. The PKTE_IMSK_STAT.HLT bit is set when the packet engine is in the HALT state.

Table 39-49: PKTE_IMSK_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/NW)	OBUFTHRSH	Output Buffer Threshold. The PKTE_IMSK_STAT.OBUFTHRSH bit is set when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (R/NW)	IBUFTHRSH	Input Buffer Threshold. The PKTE_IMSK_STAT.IBUFTHRSH bit is set when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (R/NW)	OPDN	Operation Done.
1 (R/NW)	RDRTHRSH	RDR Threshold. The PKTE_IMSK_STAT.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/NW)	CDRTHRSH	CDR Threshold. The PKTE_IMSK_STAT.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Packet Engine Input Buffer Count Register

The PKTE_INBUF_CNT register provides the number of bytes available in the input buffer. The PKTE_INBUF_CNT register is used in direct host mode only.

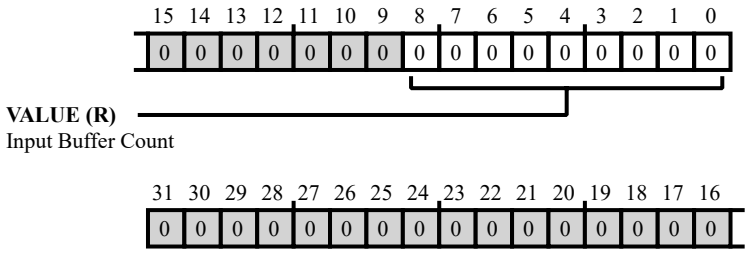


Figure 39-24: PKTE_INBUF_CNT Register Diagram

Table 39-50: PKTE_INBUF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	VALUE	Input Buffer Count. The PKTE_INBUF_CNT.VALUE bit field provides the number of bytes in the input buffer. The packet engine decrements the counter by 4 when a 32-bit word is read from the input buffer.

Packet Engine Input Buffer Count Increment Register

A host connected through the system completer bus can increment the input buffer counter by writing a value between 4 and 256, in multiples of 4, to the lowest bits of this register. The PKTE_INBUF_INCR register is used in direct host mode only.

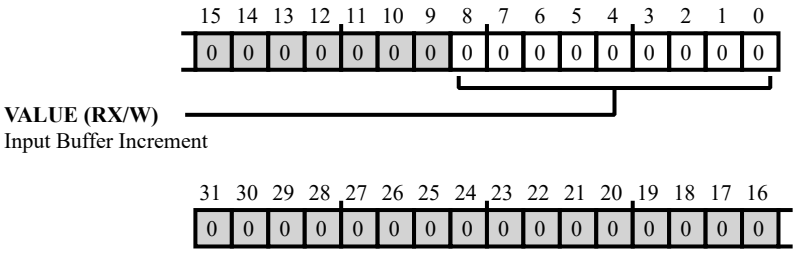


Figure 39-25: PKTE_INBUF_INCR Register Diagram

Table 39-51: PKTE_INBUF_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (RX/W)	VALUE	Input Buffer Increment. The value written is added to the input buffer counter. Valid values range from 4 to 256, in multiples of 4.

Interrupt Configuration Register

The PKTE_INT_CFG register configures the interrupt type that is sent to the interrupt line connected to the system interrupt controller. (Note that this only effects the final output of the interrupt subsystem).

Configuring the interrupt output type for pulse causes the interrupt signal to pulse low for two clock cycles when activated. When set for level, the interrupt signal is set low until cleared by the host (it follows the bit in the masked status register). For the host, this is typically set to level.

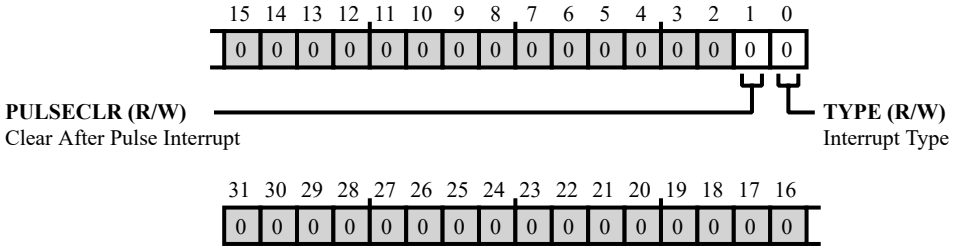


Figure 39-26: PKTE_INT_CFG Register Diagram

Table 39-52: PKTE_INT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PULSECLR	Clear After Pulse Interrupt. The PKTE_INT_CFG.PULSECLR bit clears the latched interrupt source after the pulse interrupt.	
		0	Manually clear pulse interrupt source. Do not automatically clear the interrupt sources after pulsing the interrupt output. Clear the source by writing to the PKTE_INT_CLR register.
		1	Automatically clear pulse interrupt source. After pulsing the interrupt output, automatically clear the sources.
0 (R/W)	TYPE	Interrupt Type. The PKTE_INT_CFG.TYPE bit selects the type, pulse or level, for the interrupt output to the system.	
		0	Level. The interrupt output is a level signal that is set low when an enabled interrupt is active until the interrupt is cleared.
		1	Pulse. The interrupt output is a two clock cycle low-active pulse, activated when an enabled interrupt is active.

Interrupt Clear Register

The PKTE_INT_CLR register allows the host processor to clear pending interrupts. A 1 written to a given bit in this register clears the corresponding interrupt. A 0 leaves the interrupt latch unchanged for that position.

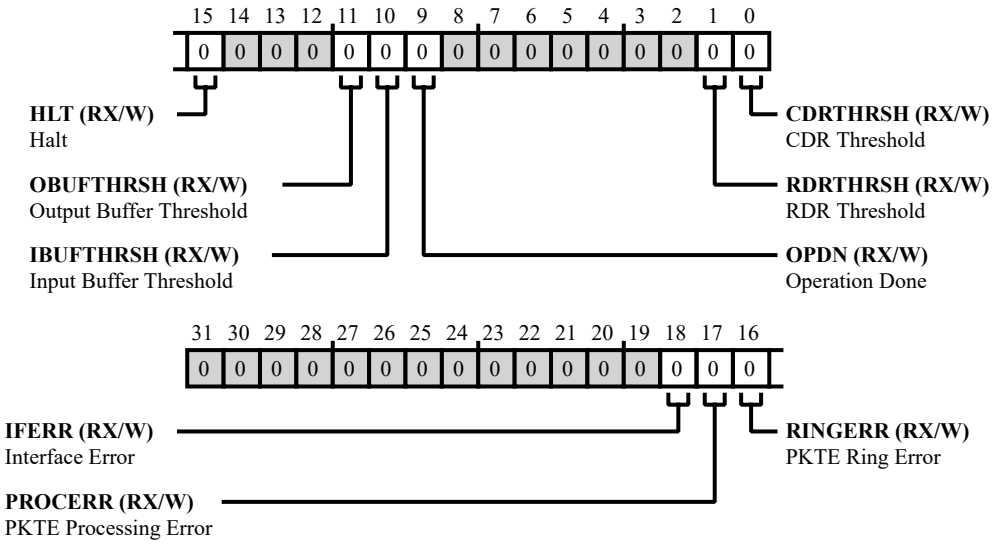


Figure 39-27: PKTE_INT_CLR Register Diagram

Table 39-53: PKTE_INT_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. The PKTE_INT_CLR.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. The PKTE_INT_CLR.PROCERR bit is set when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error. The PKTE_INT_CLR.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (RX/W)	HLT	Halt. The PKTE_INT_CLR.HLT bit is set when the packet engine is in the HALT state.
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. The PKTE_INT_CLR.OBUFTHRSH bit is set when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.

Table 39-53: PKTE_INT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. The PKTE_INT_CLR.IBUFTHRSH bit is set when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. The PKTE_INT_CLR.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. The PKTE_INT_CLR.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Enable Register

The PKTE_INT_EN register configures the interrupt mask for the host interrupt. This register is a bitmap for each of the possible interrupt sources. A 1 enables the interrupt source and a 0 disables the source. If an interrupt source is disabled, a cleared bit also clears the matching interrupt in the PKTE_IMSK_STAT register.

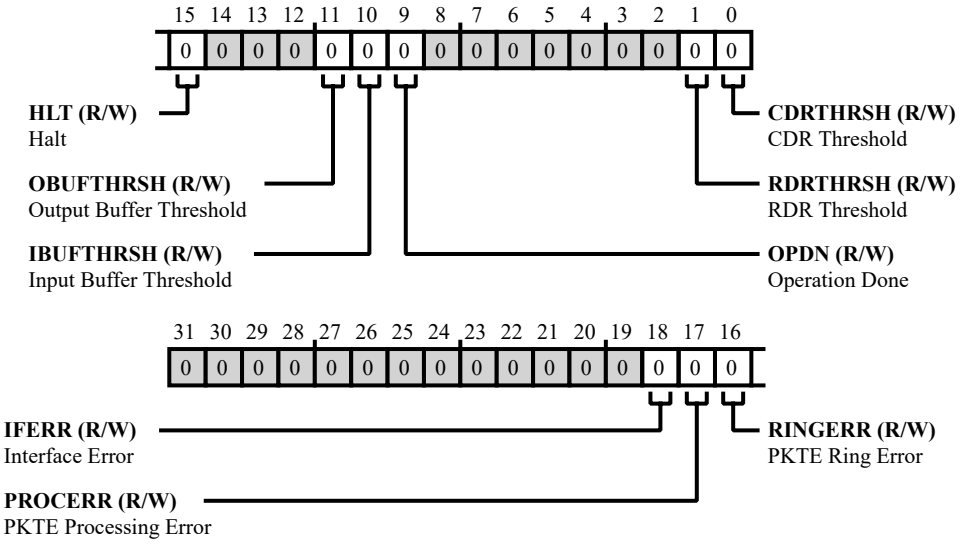


Figure 39-28: PKTE_INT_EN Register Diagram

Table 39-54: PKTE_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	IFERR	Interface Error. Set the PKTE_INT_EN.IFERR bit for host requests for a non 32-bit access to the packet engine interrupt or when the packet engine receives an error writing data back out to the host memory system.
17 (R/W)	PROCERR	PKTE Processing Error. Set the PKTE_INT_EN.PROCERR bit to enable the extended error occurred before, during or after processing the current packet in the packet engine interrupt.
16 (R/W)	RINGERR	PKTE Ring Error. Set the PKTE_INT_EN.RINGERR bit to enable the CDR overflow or RDR underflow interrupt.
15 (R/W)	HLT	Halt. Set the PKTE_INT_EN.HLT bit for when the packet engine is in the HALT state.
11 (R/W)	OBUFTHRSH	Output Buffer Threshold. Set the PKTE_INT_EN.OBUFTHRSH bit for to trigger an interrupt when the output buffer counter exceeds the output buffer threshold value defined in the PKTE_BUF_THRESH.OUTBUF bit.

Table 39-54: PKTE_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	IBUFTHRSH	Input Buffer Threshold. Set the PKTE_INT_EN.IBUFTHRSH bit for to trigger an interrupt when the input buffer counter is less than or equal to the input buffer threshold value defined in the PKTE_BUF_THRESH.INBUF bit.
9 (R/W)	OPDN	Operation Done.
1 (R/W)	RDRTHRSH	RDR Threshold. Set the PKTE_INT_EN.RDRTHRSH bit for to trigger an interrupt when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/W)	CDRTHRSH	CDR Threshold. Set the PKTE_INT_EN.CDRTHRSH bit for to trigger an interrupt when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Unmasked Status Register

The PKTE_IUMSK_STAT register provides interrupt status visibility to the host, prior to the interrupt mask being applied. Using this register, the host can view all potential sources of incoming interrupts. All of these sources, whether masked in or out, are latched in this register and must be cleared using the PKTE_INT_CLR register in order to capture a subsequent event. A 1 indicates that the associated interrupt is present.

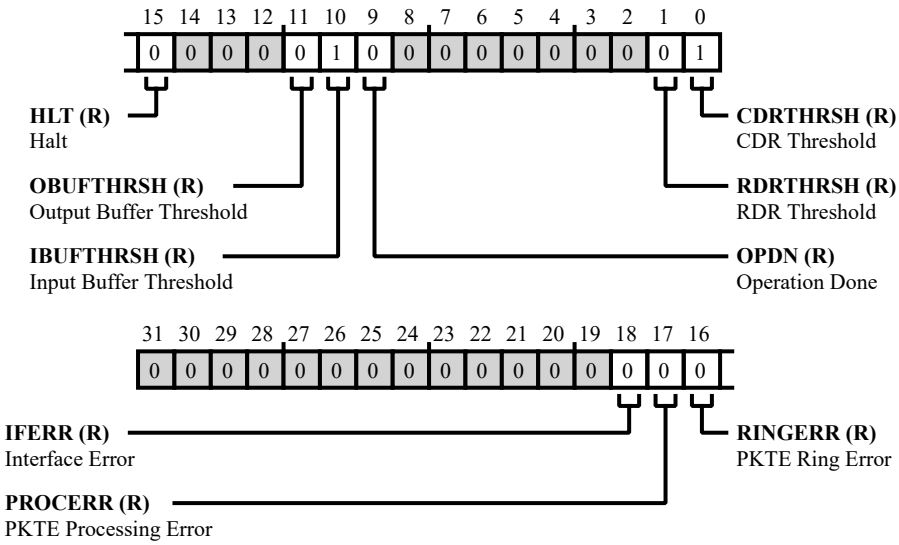


Figure 39-29: PKTE_IUMSK_STAT Register Diagram

Table 39-55: PKTE_IUMSK_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/NW)	IFERR	Interface Error. The PKTE_IUMSK_STAT.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (R/NW)	PROCERR	PKTE Processing Error. The PKTE_IUMSK_STAT.PROCERR bit is set when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (R/NW)	RINGERR	PKTE Ring Error. The PKTE_IUMSK_STAT.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (R/NW)	HLT	Halt. The PKTE_IUMSK_STAT.HLT bit is set when the packet engine is in the HALT state.

Table 39-55: PKTE_IUMSK_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/NW)	OBUFTHRSH	Output Buffer Threshold. The PKTE_IUMSK_STAT.OBUFTHRSH interrupt is triggered when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (R/NW)	IBUFTHRSH	Input Buffer Threshold. The PKTE_IUMSK_STAT.IBUFTHRSH interrupt is triggered when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (R/NW)	OPDN	Operation Done.
1 (R/NW)	RDRTHRSH	RDR Threshold. The PKTE_IUMSK_STAT.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/NW)	CDRTHRSH	CDR Threshold. The PKTE_IUMSK_STAT.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Packet Engine Length Register

The PKTE_LEN register gives the length of the packet, the bypass data and a second set of ownership bits.

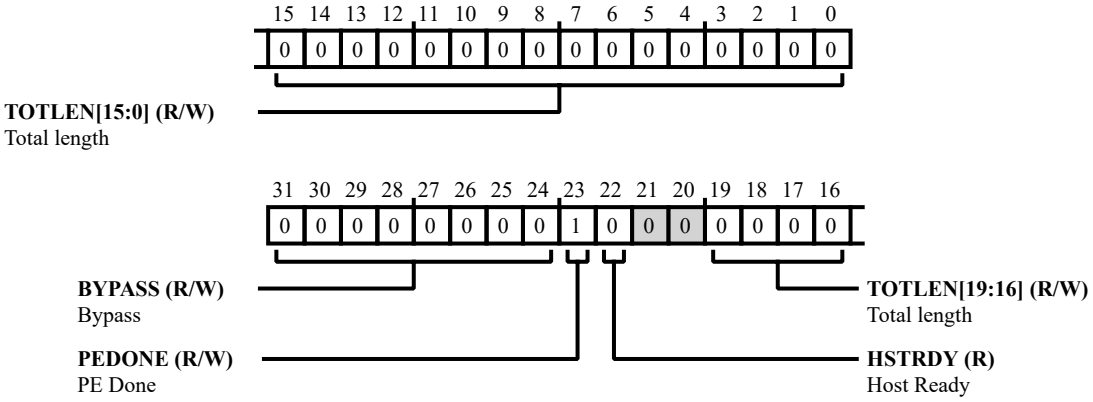


Figure 39-30: PKTE_LEN Register Diagram

Table 39-56: PKTE_LEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYPASS	Bypass. The PKTE_LEN.BYPASS bit field indicates the length of data in words that must bypass the packet engine and are directly copied from the source buffer to the destination buffer. The packet engine does not process this data. Valid bypass offsets range from 0 (0x00) to 255 (0xFF) words. For SRTP operations, this field specifies the offset in words between the hash and encrypt/decrypt data.
23 (R/W)	PEDONE	PE Done. The PKTE_LEN.PEDONE bit is a mirrored bit from the PKTE_CTL_STAT.PERDY bit. The bit is repeated here to guarantee ownership consistency between the first and last word. When the packet engine fetches a descriptor, these bits must match or the descriptor is discarded and fetched again.
22 (R/NW)	HSTRDY	Host Ready. The PKTE_LEN.HSTRDY bit is a mirrored bit of the PKTE_CTL_STAT.HOSTRDY bit. The bit is repeated here to guarantee ownership consistency between the first and last word. It should also be set along with the PKTE_CTL_STAT.HOSTRDY bit when the command descriptor is finished being populated. When the packet engine fetches a descriptor, these bits must match or the descriptor is discarded and fetched again.

Table 39-56: PKTE_LEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	TOTLEN	Total length. Command Descriptor: The PKTE_LEN.TOTLEN bit field indicates the total length (in bytes) of all data to be passed to the packet engine's input buffer for an operation. Exceptions are the PRNG init and PRNG generate operations. The PRNG init operation does not require any input data; this field must be zero. For the PRNG generate operation, this field indicates the number of pseudo-random bytes to be generated. Valid lengths range from 16 (0x00010) to 255*16 = 4080 (0x00FF0) bytes in multiples of 16 bytes. Valid lengths for the basic operation range from 1 (0x00001) to 1,048,575 (0xFFFFF) bytes. This is the length of the data to be encrypted or hashed and includes the bypass data and padding bytes. Valid lengths for IPsec ESP range from 1 (0x00001) to 65535 (0x0FFFF) bytes. This is the length of the IP payload. Valid lengths for SSL v3.0, TLS v1.x and DTLS range from 1 (0x00001) to 16383 (0x03FFF). This is the length of the payload. Valid lengths for SRTP range from 1 (0x00001) to 65535 (0x0FFFF). This is the length of the payload. Note: A length of zero bytes is illegal and will result in an error status code in the result descriptor. Result Descriptor: Upon completion of an operation, the PKTE_LEN.TOTLEN field indicates the result length of the result packet. Valid lengths range from 1 (0x001) to 1,048,575 (0xFFFFF) bytes. This includes the bypass data and padding bytes. Note: When an extended error (PKTE_CTL_STAT[18]=1) is reported in the result descriptor and no packet data is processed, this field returns zero.

Packet Engine Output Buffer Count Register

The PKTE_OUTBUF_CNT register provides the number of data bytes there are in the output buffer. The PKTE_OUTBUF_CNT register is used in direct host mode only.

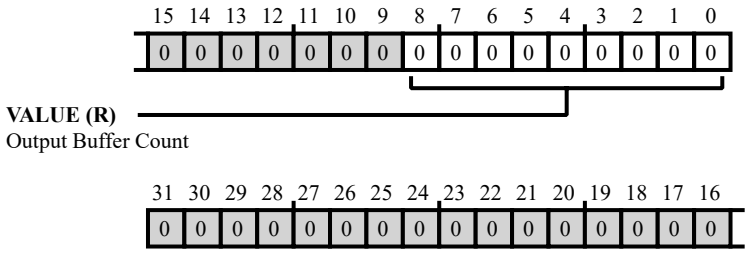


Figure 39-31: PKTE_OUTBUF_CNT Register Diagram

Table 39-57: PKTE_OUTBUF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	VALUE	Output Buffer Count. The PKTE_OUTBUF_CNT.VALUE bit field provides the number of bytes in the output buffer. The packet engine increments the counter by 4 when a 32-bit word is written to the output buffer.

Packet Engine Output Buffer Count Decrement Register

A host connected via the system completer bus can decrement the output buffer counter by writing a value between 4 and 256, in multiples of 4, to the lowest bits of this register. The PKTE_OUTBUF_DECR register is used in direct host mode only.

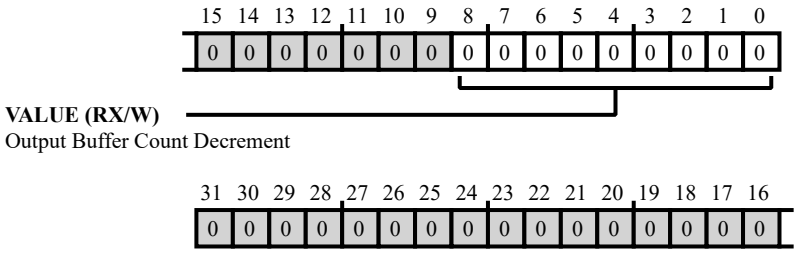


Figure 39-32: PKTE_OUTBUF_DECR Register Diagram

Table 39-58: PKTE_OUTBUF_DECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (RX/W)	VALUE	Output Buffer Count Decrement. The PKTE_OUTBUF_DECR.VALUE bit field is the value written is subtracted to the output buffer counter. Valid values range from 4 to 256, in multiples of 4.

PE Alternative Key stat register

The Key registers can be written in support of several cipher modes and key sizes.

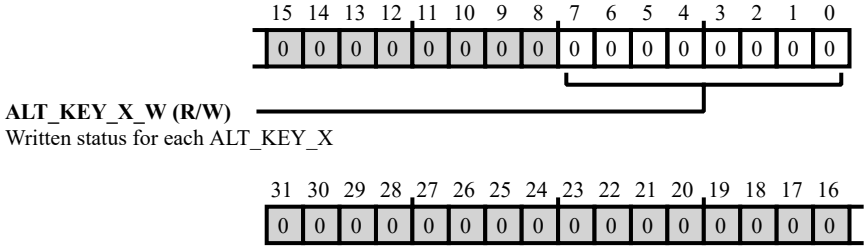


Figure 39-33: PKTE_PE_ALT_KEY_STATUS Register Diagram

Table 39-59: PKTE_PE_ALT_KEY_STATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	ALT_KEY_X_W	Written status for each ALT_KEY_X.

Packet Engine Cache Control 0

Packet Engine Cache Control 0

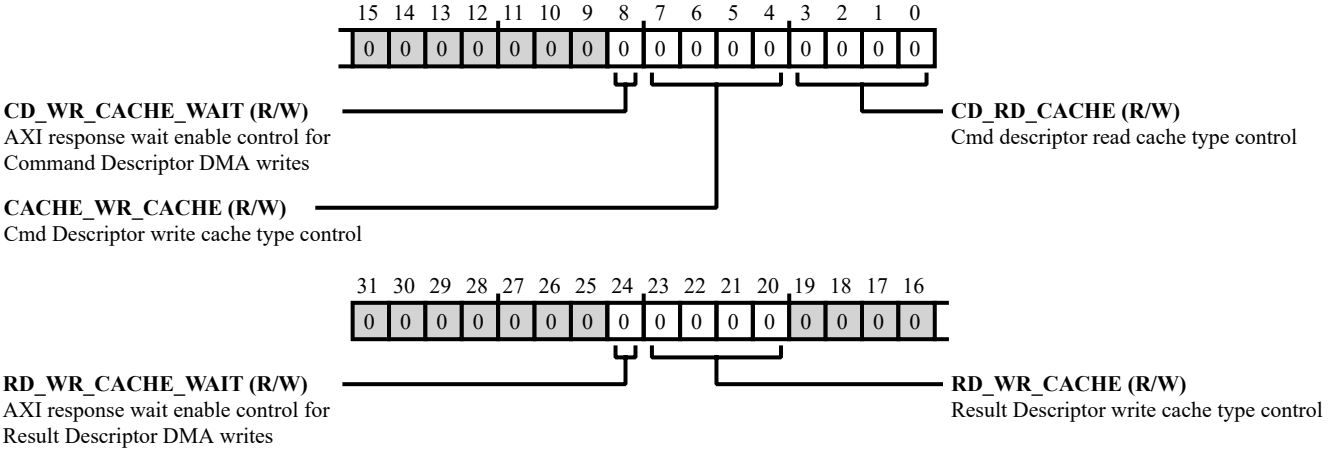


Figure 39-34: PKTE_PE_CACHE_CTRL_0 Register Diagram

Table 39-60: PKTE_PE_CACHE_CTRL_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	RD_WR_CACHE_WAIT	AXI response wait enable control for Result Descriptor DMA writes.
23:20 (R/W)	RD_WR_CACHE	Result Descriptor write cache type control.
8 (R/W)	CD_WR_CACHE_WAIT	AXI response wait enable control for Command Descriptor DMA writes.
7:4 (R/W)	CACHE_WR_CACHE	Cmd Descriptor write cache type control.
3:0 (R/W)	CD_RD_CACHE	Cmd descriptor read cache type control.

Packet Engine Cache Control 1

Packet Engine Cache Control 1

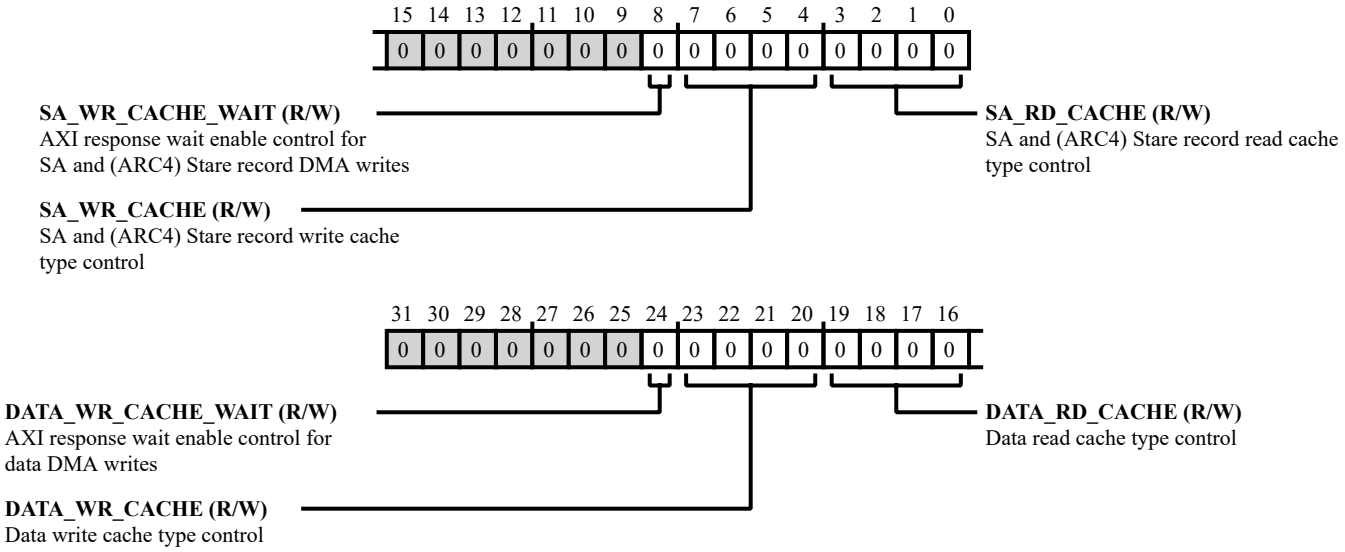


Figure 39-35: PKTE_PE_CACHE_CTRL_1 Register Diagram

Table 39-61: PKTE_PE_CACHE_CTRL_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	DATA_WR_CACHE_WAIT	AXI response wait enable control for data DMA writes.
23:20 (R/W)	DATA_WR_CACHE	Data write cache type control.
19:16 (R/W)	DATA_RD_CACHE	Data read cache type control.
8 (R/W)	SA_WR_CACHE_WAIT	AXI response wait enable control for SA and (ARC4) Stare record DMA writes.
7:4 (R/W)	SA_WR_CACHE	SA and (ARC4) Stare record write cache type control.
3:0 (R/W)	SA_RD_CACHE	SA and (ARC4) Stare record read cache type control.

Packet Engine Result Descriptor Ring Base Address

The PKTE_RDRBASE_ADDR register holds the result descriptor ring base address in host memory. It is only applicable in autonomous ring mode and target command mode with RDR enabled. Note that in target command mode, the CDR is not used, but the RDR must be configured when enabled so that the packet engine knows where to write the result descriptors.

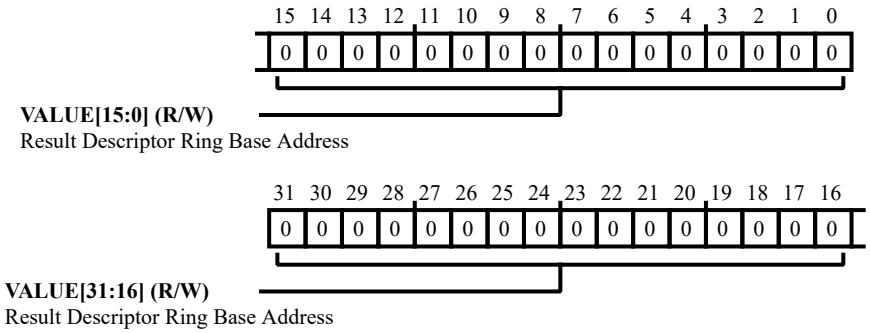


Figure 39-36: PKTE_RDRBASE_ADDR Register Diagram

Table 39-62: PKTE_RDRBASE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Result Descriptor Ring Base Address. The PKTE_RDRBASE_ADDR.VALUE bit field specifies the base location of the result descriptor ring in the host memory space.

Packet Engine Result Descriptor Count Registers

The PKTE_RDSC_CNT register holds the counter for the number of descriptors in the Result Descriptor Ring (RDR). It is incremented by the packet engine each time a valid result descriptor is written to the RDR.

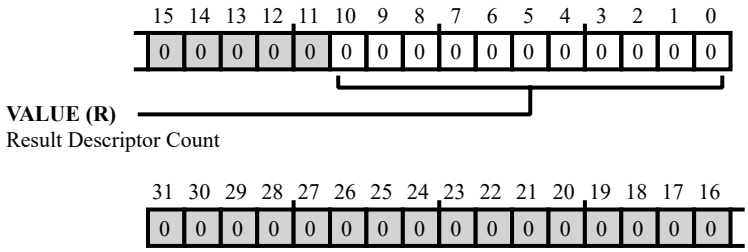


Figure 39-37: PKTE_RDSC_CNT Register Diagram

Table 39-63: PKTE_RDSC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Result Descriptor Count. The PKTE_RDSC_CNT.VALUE bit field provides the number of result descriptors in the result descriptor ring. The packet engine increments the counter when a valid result descriptor is written to the RDR.

Packet Engine Result Descriptor Count Decrement Registers

The PKTE_RDSC_DECR register is accessible by the host connected through the system completer bus can decrement the result descriptor counter by writing a value between 1 and 255 to the lowest byte of this register.

With an RDR enabled, this is the number of result descriptors that have been read by the host. With an RDR disabled, this indicates that the host has read one valid result descriptor.

In autonomous ring mode or target command mode with the RDR enabled, the host must process 1 to 255 result descriptors from the RDR and then write this register with the number of result descriptors that have been processed by the host.

In direct host mode or target command mode with the RDR disabled, the host must read one result descriptor from the internal descriptor registers and then write this register with the value 1, to indicate that one valid descriptor is read. An RDR threshold interrupt is activated when the result descriptor counter exceeds the threshold value set in the PKTE_RING_THRESH register. This interrupt can be used to wake up a process that stalled on an empty RDR.

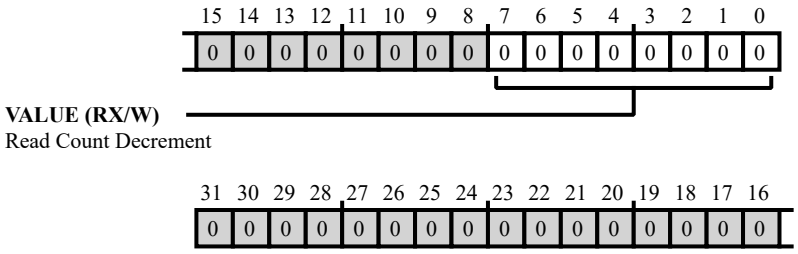


Figure 39-38: PKTE_RDSC_DECR Register Diagram

Table 39-64: PKTE_RDSC_DECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (RX/W)	VALUE	Read Count Decrement. The value written to the PKTE_RDSC_DECR.VALUE bit field is subtracted from the result descriptor counter. The counter is protected against underflow (See the PKTE_RING_STAT register). Note that bits [10:8] should be written with zeros.

Packet Engine Ring Configuration

The PKTE_RING_CFG register configures the size (in number of descriptor ring entries minus 1) for both the command descriptor ring and result descriptor ring in host memory. This register is only applicable for autonomous ring mode and target command mode with RDR enabled.

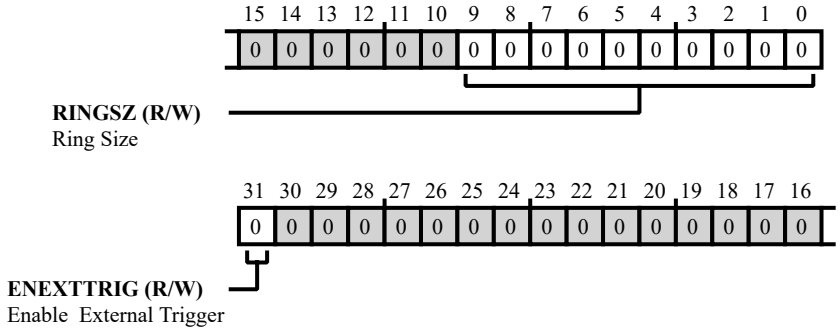


Figure 39-39: PKTE_RING_CFG Register Diagram

Table 39-65: PKTE_RING_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	ENEXTTRIG	Enable External Trigger. The PKTE_RING_CFG.ENEXTTRIG signal enables the increment of the PKTE_CDSC_CNT register through the external input pin ext_cd_cnt_incr and enables the decrement of the PKTE_RDSC_CNT fields through the external input pin ext_rd_cnt_decr.
9:0 (R/W)	RINGSZ	Ring Size. The PKTE_RING_CFG.RINGSZ bit field specifies the size of the command ring in number of descriptors, minus 1. Valid sizes range from 1 (for 2 descriptors) to 1023 (for 1024 descriptors). The accompanying result ring will have the same size.

Packet Engine Ring Pointer Status

The PKTE_RING_PTR register holds the pointers to the current entry of the Command Descriptor Ring (CDR) and Result Descriptor Ring (RDR).

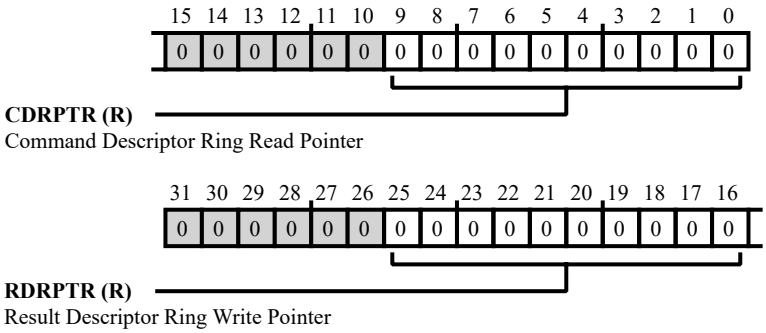


Figure 39-40: PKTE_RING_PTR Register Diagram

Table 39-66: PKTE_RING_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/NW)	RDRPTR	Result Descriptor Ring Write Pointer. The PKTE_RING_PTR.RDRPTR bit field indicates the entry number in the RDR that will be written next by the packet engine. The PKTE_RING_PTR.RDRPTR bit field is reset to zero after starting up and updated after every result descriptor write DMA operation. Pointers wrap around; the maximum value this field can have equals the contents of the ring size (PKTE_RING_CFG.RINGSZ) bit field.
9:0 (R/NW)	CDRPTR	Command Descriptor Ring Read Pointer. The PKTE_RING_PTR.CDRPTR bit field indicates the entry number in the CDR that will be read next by the packet engine. The PKTE_RING_PTR.CDRPTR bit field is reset to zero after starting up and updated after every command descriptor read DMA operation. Pointers wrap around; the maximum value this field can have equals the contents of the ring size (PKTE_RING_CFG.RINGSZ) field.

Packet Engine Ring Status

The PKTE_RING_STAT register gives indication of either a Command Descriptor Ring (CDR) overflow or a Result Descriptor Ring (RDR) underflow. A ring error (ringerr) interrupt in the interrupt controller is activated on a command descriptor ring overflow or a result descriptor ring underflow. This type of error can occur when the host and the packet engine get out-of-sync. The host can read this register to retrieve information on which ring is corrupted. The corrupted ring must be reset and reinitialized. See the PKTE_CFG.RSTRING bit.

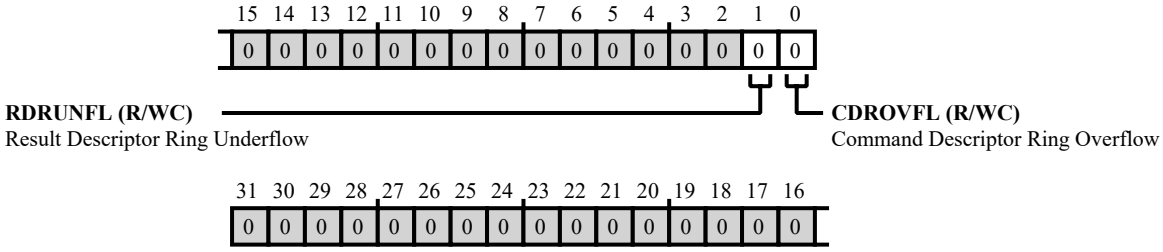


Figure 39-41: PKTE_RING_STAT Register Diagram

Table 39-67: PKTE_RING_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/WC)	RDRUNFL	Result Descriptor Ring Underflow. The PKTE_RING_STAT.RDRUNFL bit is set when the command descriptor count (PKTE_RDSC_CNT) register is decremented below zero. This bit is reset with a write of any value.
0 (R/WC)	CDROVFL	Command Descriptor Ring Overflow. The PKTE_RING_STAT.CDROVFL bit is set when the command descriptor count (PKTE_CDSC_CNT) register is incremented above the ring size (PKTE_RING_CFG.RINGSZ) bits. This bit is reset with a write of any value.

Packet Engine Ring Threshold Registers

To reduce the amount of packet engine result interrupts, the PKTE_RING_THRESH register contains threshold and time-out values.

The CDR threshold (cdrthrsh) interrupt indicates that the command descriptor counter is less than or equal to the CDR threshold (cdrthrsh) value set in this register. This interrupt can be used to wake up a process that stalled on a full CDR.

The RDR threshold (rdrthrsh) interrupt indicates that the result descriptor counter exceeds the result descriptor threshold set here, or that the result descriptor counter is non-zero for a time longer than the result descriptor time-out setting. The RDR result interrupt remains active until the result descriptor counter is decremented below the RDR threshold (rdrthrsh) value. In case the interrupt is the result of a time-out and the result descriptor counter is below the threshold value, the result descriptor counter must be decremented once before the interrupt can be cleared in the interrupt controller.

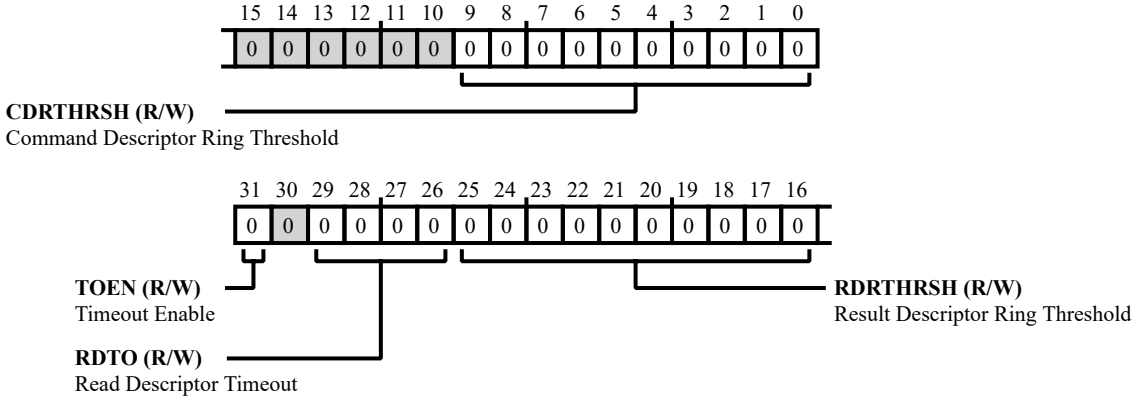


Figure 39-42: PKTE_RING_THRESH Register Diagram

Table 39-68: PKTE_RING_THRESH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	TOEN	Timeout Enable. A 1 in the PKTE_RING_THRESH.TOEN bit indicates the result descriptor timeout counter is enabled. This bit can be used to de-activate the timeout counter to save power.

Table 39-68: PKTE_RING_THRESH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:26 (R/W)	RDTO	Read Descriptor Timeout. The timeout enable (PKTE_RING_THRESH.TOEN) bit in this register must be set to activate this PKTE_RING_THRESH.RDTO result descriptor timeout counter. The rdrthrsh interrupt activates when the RD counter for the RDR is non-zero for more than $2^{(N+10)}$ internal system clock cycles, where 'N' is the value set in this field. Valid settings range from 0 to 15. The minimum time-out value for N=0 is 1024 clock cycles and the maximum time-out value for N=15 is 33554432 clock cycles. At 100 MHz, this is 5.12 us for N=0 and ~335.55 ms for N=15. Note: The time-out delay may not be exact - expect a variation on the order of 1024 system clock cycles (just more than one microsecond at 100 MHz system clock frequency).
25:16 (R/W)	RDRTHRSH	Result Descriptor Ring Threshold. The rdrthrsh interrupt activates when the RD counter for the RDR exceeds the value set in the PKTE_RING_THRESH.RDRTHRSH field. Valid settings range from 0 to 1023.
9:0 (R/W)	CDRTHRSH	Command Descriptor Ring Threshold. The cdrthrsh interrupt activates when CD counter for the CDR is below or equal the value set in the PKTE_RING_THRESH.CDRTHRSH field. Valid settings range from 0 to 1023.

Packet Engine SA Address

The PKTE_SA_ADDR register holds the start address of the SA record.

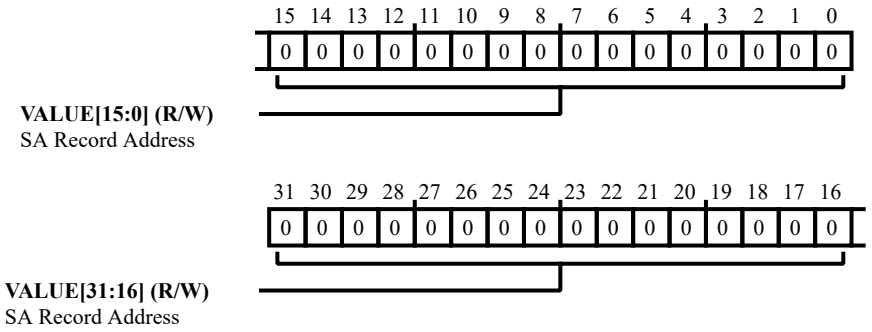


Figure 39-43: PKTE_SA_ADDR Register Diagram

Table 39-69: PKTE_SA_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Record Address. The PKTE_SA_ADDR.VALUE bit field holds the start address of the SA record.

ARC4 i and j Pointer Register

When starting a new ARC4 operation the PKTE_SA_ARC4IJPTR register contains the initialization value, which is zeros. After processing the ARC4 algorithm it contains the latest status of the ARC4_IJ_PNTR.

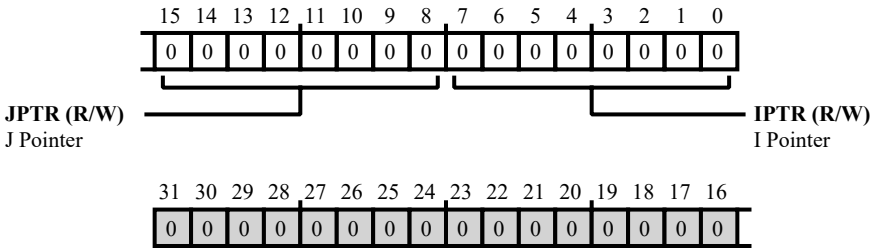


Figure 39-44: PKTE_SA_ARC4IJPTR Register Diagram

Table 39-70: PKTE_SA_ARC4IJPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	JPTR	J Pointer. The PKTE_SA_ARC4IJPTR.JPTR bit field contains the j pointer into s-box array for swapping bytes with i pointer.
7:0 (R/W)	IPTR	I Pointer. The PKTE_SA_ARC4IJPTR.IPTR bit field contains the i pointer into s-box array for swapping bytes with j pointer.

SA Command 0

The two SA command registers, PKTE_SA_CMD0 and PKTE_SA_CMD1, are used to control the cryptographic operation of the packet engine. The PKTE_SA_CMD0 register contains the major control bits to define an operation while the PKTE_SA_CMD1 register contains the minor control bits. In direct host mode, this is a write-only register.

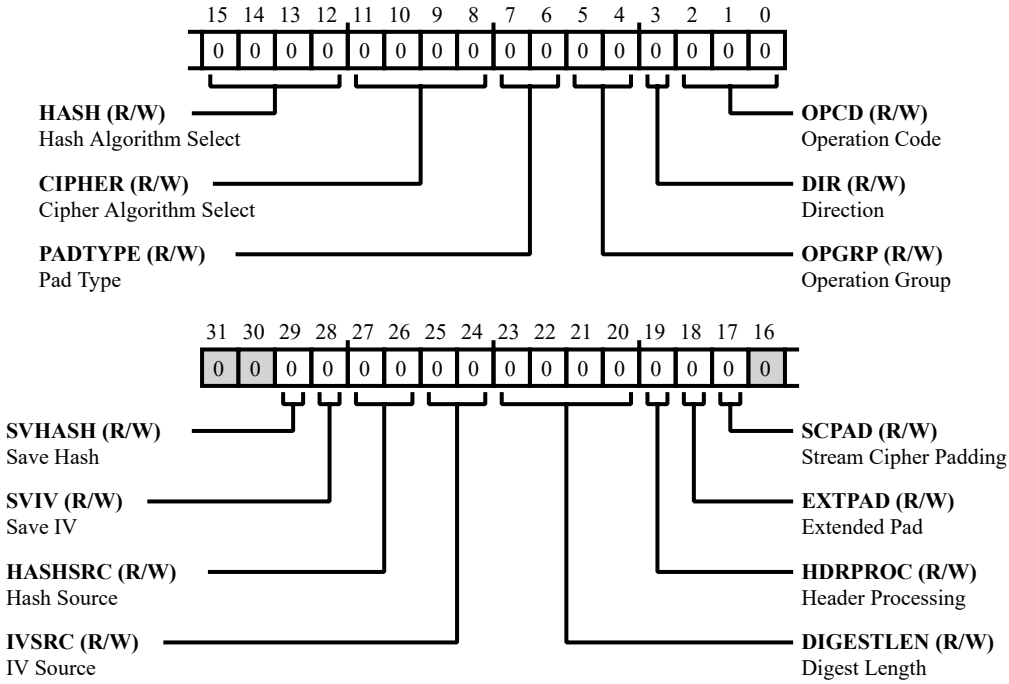


Figure 39-45: PKTE_SA_CMD0 Register Diagram

Table 39-71: PKTE_SA_CMD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SVHASH	Save Hash. The PKTE_SA_CMD0.SVHASH bit indicates that the Hash State is saved to the STATE_BYTE_CNT_X and STATE_IDIGEST_X fields in the SA record in memory after completion of a crypto operation.	
		0	Hash state is not saved
		1	Hash state is saved

Table 39-71: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	SVIV	Save IV. The `PKTE_SA_CMD0.SVIV` bit field indicates that for DES or the AES the Initialization Vector (IV) is saved to the STATE_IV_X fields in the state record , or the ARC4 state is saved to the ARC4 state record, after completion of the crypto operation.	
		0	ARC4 State is not saved.
		1	ARC4 State is not saved.
27:26 (R/W)	HASHSRC	Hash Source. The `PKTE_SA_CMD0.HASHSRC` bit field selects the source of the hash digest used by the algorithm.	
		0	From SA. Digest only hash byte count is forced to 0x40.
		1	Reserved
		2	From State. Read saved inner hash digest and saved hash byte count.
		3	No Load. Use the hash algorithm defined constants for the initial hash. Hash byte count is 0x00.
25:24 (R/W)	IVSRC	IV Source. The `PKTE_SA_CMD0.IVSRC` bit field selects the source of the initialization vector used by the crypto algorithm.	
		0	No load. Use previous result IV, not applicable for inbound data. This option should never be used for operations with DES-CBC or AES-CBC, (see RFC3602) or any AES counter modes
		1	From input buffer. The IV is provided as part of the input data stream.
		2	From State. Read STATE_IV_X, from the SA structure. Refer to inner hash digest register structure. Useful for resume operations.
		3	From internal PRNG. Not applicable for inbound operations.
23:20 (R/W)	DIGESTLEN	Digest Length. The `PKTE_SA_CMD0.DIGESTLEN` bit field defines the length of the hash digest in words as put in the output buffer.	
		0	3 Words (96-bit output)
		1	1 Word
		2	2 Words

Table 39-71: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	3 Words (IPsec)
		4	4 Words (MD5 and AES-based hash)
		5	5 Words (SHA-1)
		6	6 Words
		7	7 Words (SHA-224)
		8	8 Words (SHA-256)
		9	Reserved
		10	10 bytes (SRTP and TLS)
		11-15	Reserved
19 (R/W)	HDRPROC	Header Processing. The PKTE_SA_CMD0.HDRPROC bit enables header processing for protocol operations. There is no header-processing support for basic SSL, basic TLS and SRTP protocol operations as defined in the protocol group (see the Crypto and Hash Algorithms section). This bit must be zero for these operations; however, the protocol header must be supplied to the packet engine since it is part of the hash calculation. Refer to the protocol specifications for more information about header-processing support for a protocol.	
		0	No header processing
		1	Header processing; insert the protocol header for outbound operations, verify the protocol header for inbound operations.
18 (R/W)	EXTPAD	Extended Pad. The PKTE_SA_CMD0.EXTPAD bit extends the number of padding types. Used in combination with PKTE_SA_CMD0.PADTYPE.	
17 (R/W)	SCPAD	Stream Cipher Padding. The PKTE_SA_CMD0.SCPAD bit enables padding for stream ciphers algorithms.	
15:12 (R/W)	HASH	Hash Algorithm Select. The PKTE_SA_CMD0.HASH bit field selects the hash algorithm.	
		0	MD5
		1	SHA-1
		2	SHA-224
		3	SHA-256
		4-14	Reserved
		15	Null

Table 39-71: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/W)	CIPHER	Cipher Algorithm Select. The PKTE_SA_CMD0.CIPHER bit field selects the cipher algorithm to be used for encryption and decryption. Note: Each type of protocol operation supports different sets of crypto algorithms. Refer to the Crypto and Hash Algorithms general processing section for details of the supported algorithms.	
		0	DES
		1	Triple-DES
		2	ARC4
		3	AES
		4-14	Reserved
		15	Null
7:6 (R/W)	PADTYPE	Pad Type. The PKTE_SA_CMD0.PADTYPE bit field indicates the type of crypto that must be generated for outbound packets or checked for inbound packets.	
		0	Select IPSec operation (if Bit 18=0); Reserved (if Bit 18=1)
		1	PKCS#7 (if Bit 18=0); Select TLS/DTLS Pad, required for TLS/DTLS operation (if Bit 18=1)
		2	Constant pad (if Bit 18=0); Select Constant SSL Pad, required for SSL operation (if Bit 18=1)
		3	Zero pad (if Bit 18=0), Reserved (if Bit 18=1)
5:4 (R/W)	OPGRP	Operation Group. The PKTE_SA_CMD0.OPGRP bit field defines the operation groups. Refer to the Basic Operations and Decoding section for more information.	
		0	Basic operation group
		1	Protocol operation group
		2	Extended protocol operations group
		3	Reserved
3 (R/W)	DIR	Direction. The PKTE_SA_CMD0.DIR bit field selects the direction of operation.	
		0	Outbound operations
		1	Inbound operations

Table 39-71: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	OPCD	Operation Code. The `PKTE_SA_CMD0.OPCD` bit field selects the operation within the operation group.

SA Command 1

The PKTE_SA_CMD1 register contains the minor control bits that define an operation. In direct host mode, this is a write-only register.

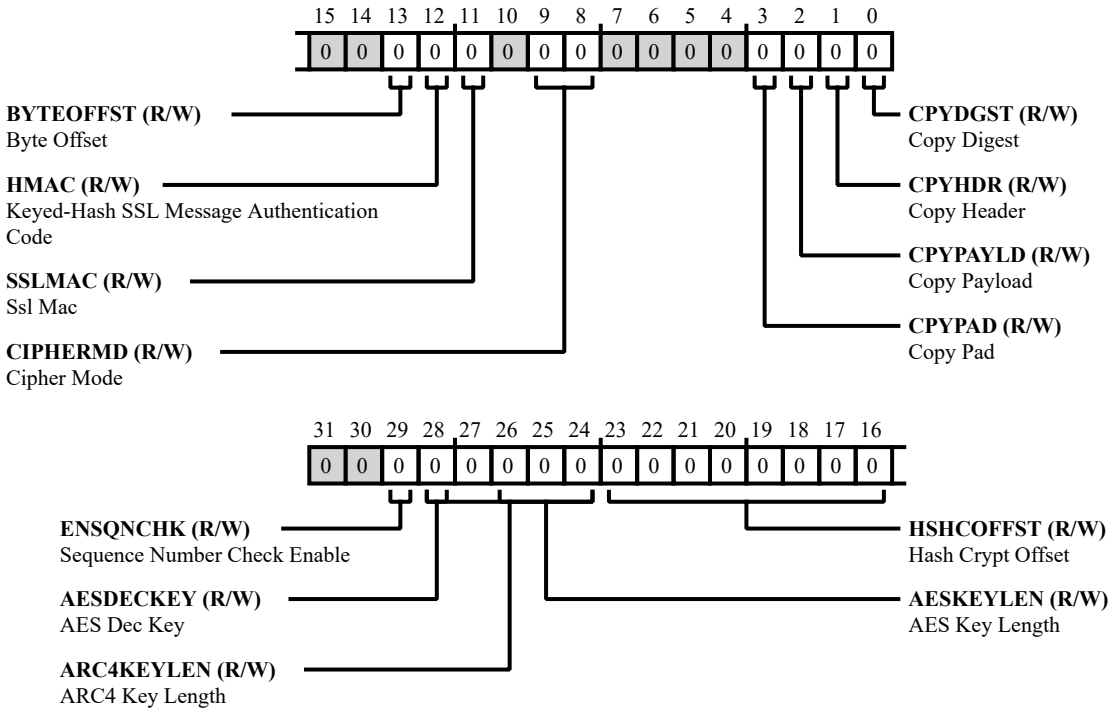


Figure 39-46: PKTE_SA_CMD1 Register Diagram

Table 39-72: PKTE_SA_CMD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	ENSQNCHK	Sequence Number Check Enable. The PKTE_SA_CMD1.ENSQNCHK bit defines that the key in the SA key field is an AES encrypt key or an AES decrypt key.	
		0	Disable sequence number check
		1	Enable sequence number check
28 (R/W)	AESDECKEY	AES Dec Key. If the PKTE_SA_CMD1.AESDECKEY bit is set, the key in loaded in the PKTE_SA_KEY[n] registers are expected to be the key from the last round from key expansion. If not set, the key loaded in the PKTE_SA_KEY[n] registers are expected to be the same key used during the encryption process.	
		0	AES key is an encrypt key.
		1	AES key is a decrypt key.

Table 39-72: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/W)	AESKEYLEN	AES Key Length. The PKTE_SA_CMD1.AESKEYLEN bit field select the size of the key used for the AES algorithm in increments of 64 bits.	
		0-1	Reserved
		2	128 Bits
		3	192 Bits
		4	256 Bits
		5-7	Reserved
28:24 (R/W)	ARC4KEYLEN	ARC4 Key Length.	
23:16 (R/W)	HSHCOFFST	Hash Crypt Offset. For Basic Encrypt-Hash and Basic Hash-Decrypt operations, the PKTE_SA_CMD1.HSHCOFFST bit field specifies the offset between the hash data and the encrypt/decrypt data. The data to be hashed is assumed to come first, with an offset to the beginning of encrypt/decrypt data. When PKTE_SA_CMD1.BYTEOFFST, bit 13, is zero, then the offset is defined in 32-bit words. When an initialization vector is loaded through the input buffer, valid values range from "IV size" to 255. In all other cases, valid values range from 0 to 255. When PKTE_SA_CMD1.BYTEOFFST, bit 13, is one, then the offset is defined in 8-bit bytes. When an initialization vector is loaded through the input buffer, valid values range from "IV size" to 255. In all other cases, valid values range from 4 to 255. (The IV size is two words for DES, Triple-DES and AES-CTR and four words for AES-CBC and AES-ICM operations). Other operations do not use these bits (a default value is applied by the packet engine).	
13 (R/W)	BYTEOFFST	Byte Offset. The PKTE_SA_CMD1.BYTEOFFST bit defines how the PKTE_SA_CMD1.HSHCOFFST, bits of this register are used.	
		0	HASH_CRYPT_OFFSET is defined in 32-bit words
		1	HASH_CRYPT_OFFSET is defined in 8-bit bytes
12 (R/W)	HMAC	Keyed-Hash SSL Message Authentication Code. For basic operations that include hashing, the PKTE_SA_CMD1.HMAC bit enables the HMAC processing, which calls for an extra outer hash operation.	
		0	Standard Hash
		1	HMAC Processing

Table 39-72: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	SSLMAC	Ssl Mac.	
		0	Standard Hash
		1	SSL-MAC processing
9:8 (R/W)	CIPHERMD	Cipher Mode. The `PKTE_SA_CMD1.CIPHERMD` bit field selects the crypto mode to be used for the cipher algorithm.	
		0	Electronic Code Book (ECB) used for DES and AES
		1	Cipher Block Chaining (CBC) used for DES and AES
		2	AES Counter Mode (CTR) for IPsec using a 32-bit counter
		3	AES Integer Counter Mode (ICM) for SRTP using a 16-bit counter. Note: This is implemented as a 32-bit counter, the Host must check for an overflow from the value 0xFF to zero and then refresh the IV.
3 (R/W)	CPYPAD	Copy Pad. The `PKTE_SA_CMD1.CPYPAD` bit indicates that the padding data for an inbound operation is copied to the output buffer and saved in memory.	
		0	Do not copy the padding to output
		1	Copy padding to output
2 (R/W)	CPYPAYLD	Copy Payload. The `PKTE_SA_CMD1.CPYPAYLD` bit indicates that the payload data is copied to the output buffer and saved in memory.	
		0	Do not copy the payload to output
		1	Copy payload to output
1 (R/W)	CPYHDR	Copy Header. The `PKTE_SA_CMD1.CPYHDR` bit indicates that the protocol header is copied to the output buffer and saved in memory. For Basic Encrypt-Hash and Basic Hash-Decrypt operations, the header is defined as the Hash/Crypt Offset data (authenticated only).	
		0	Do not copy the header to output
		1	Copy header to output

Table 39-72: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	CPYDGST	Copy Digest. The PKTE_SA_CMD1.CPYDGST bit copies the hash result is to the output buffer and saves in memory. The length of the hash result is defined by the PKTE_SA_CMD0.DIGESTLEN field.	
		0	Do not copy hash result to output
		1	Copy hash result to output, when the command descriptor PKTE_CTL_STAT.HASHFINAL bit is set.

SA Inner Hash Digest Registers

The PKTE_SA_IDIGEST[n] registers are a set of eight 32-bit read/write registers.

For MD5, SHA-1, SHA-224 and SHA-256, these read/write registers are used to enter a start hash state, and to read the interim or final hash digest.

For IPsec, TLS and DTLS operations that make use of MD5, SHA-1, SHA-224 or SHA-256 with basic hash or HMAC authentication with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the pre-computed inner hash digest. This is the hash of the hash-key padded with 0x36 hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-MD5 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the inner hash pre-compute; this is the hash of the MAC_WRITE_SECRET padded with 0x36 hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-SHA-1 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the MAC_WRITE_SECRET. Note that it is not possible to calculate a hash pre-compute for SHA-1 in combination with SSL-MAC (specification flaw). The packet engine appends the hash-key pad (0x36 hex) and sets the starting hash byte count automatically to 60 decimal / 0x3C hex (to indicate that 60 bytes have already been prepared for the hash).

The reset value for these registers is zero.

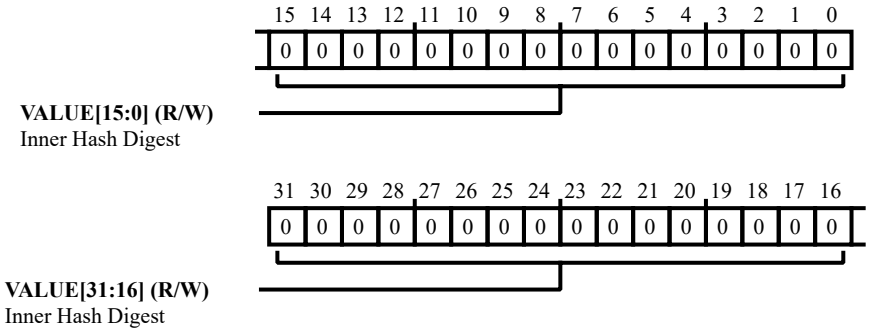


Figure 39-47: PKTE_SA_IDIGEST[n] Register Diagram

Table 39-73: PKTE_SA_IDIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Inner Hash Digest.

SA Key Registers

These are the PKTE_SA_KEY[n] registers for DES, Triple-DES, ARC4 and AES: A set of eight 32-bit write only registers. The reset value of these registers is zero.

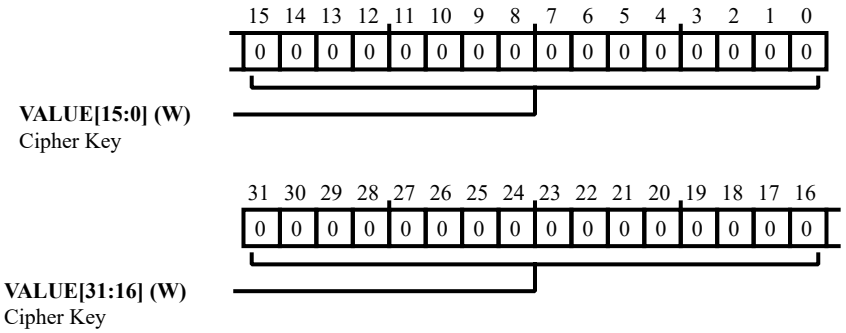


Figure 39-48: PKTE_SA_KEY[n] Register Diagram

Table 39-74: PKTE_SA_KEY[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Cipher Key.

SA Initialization Vector Register

The PKTE_SA_NONCE register is used for operations that make use of the IV value loaded from the SA record. This register is used both to enter a starting IV state, as well as for reading the interim or final IV. For IPsec outbound operations, it is recommended that the automatic IV insertion mode be used, this register is not needed. For IPsec inbound operations, the IV is extracted from the header of the packet.

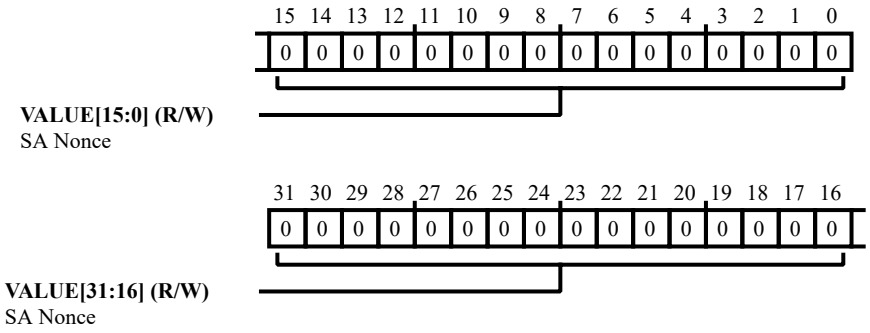


Figure 39-49: PKTE_SA_NONCE Register Diagram

Table 39-75: PKTE_SA_NONCE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Nonce.

SA Outer Hash Digest Registers

The PKTE_SA_ODIGEST[n] registers are a set of five eight 32-bit write-only registers.

For write operations, these registers contain the pre-computed outer hash digest for IPsec operations with basic HMAC operations with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA).

For MD5, SHA-1, SHA-224 and SHA-256, these read/write registers hold a start hash state, or the interim outer hash digest. They are only used for HMAC processing.

For IPsec, SSL, TLS, DTLS and SRTP operations that make use of MD5, SHA-1, SHA-224 or SHA-256 with HMAC authentication with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the pre-computed outer hash digest. This is the hash of the hash-key padded with 0x5C hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-MD5 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the outer hash pre-compute; this is the hash of the MAC_WRITE_SECRET padded with 0x5C hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-SHA-1 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the MAC_WRITE_SECRET. Note that it is not possible to calculate a hash pre-compute for SHA-1 in combination with SSL-MAC (specification flaw). The packet engine appends the required hash-key pad (0x5C hex) and sets the starting hash byte count automatically to 60 decimal / 0x3C hex (to indicate that 60 bytes have already been prepared for the hash).

The reset value for these registers is zero.

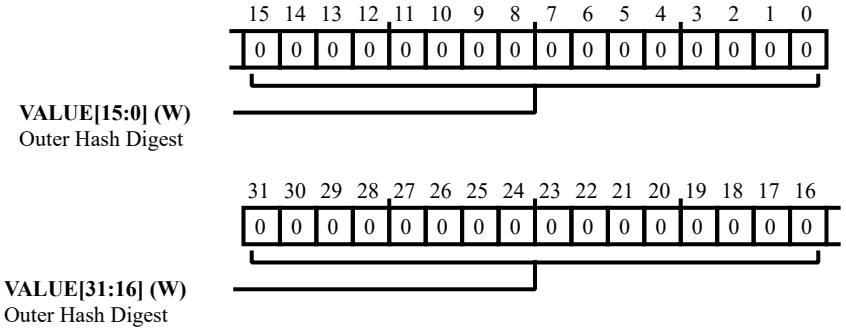


Figure 39-50: PKTE_SA_ODIGEST[n] Register Diagram

Table 39-76: PKTE_SA_ODIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Outer Hash Digest.

SA Ready Indicator

In direct host mode, a write to the PKTE_SA_RDY register triggers the packet engine to start processing using the command descriptor, SA record and state record in the packet engine registers. This register MUST be written for all direct host mode packet operations. It is intended that this register is written in sequence; as the entire SA record is written.

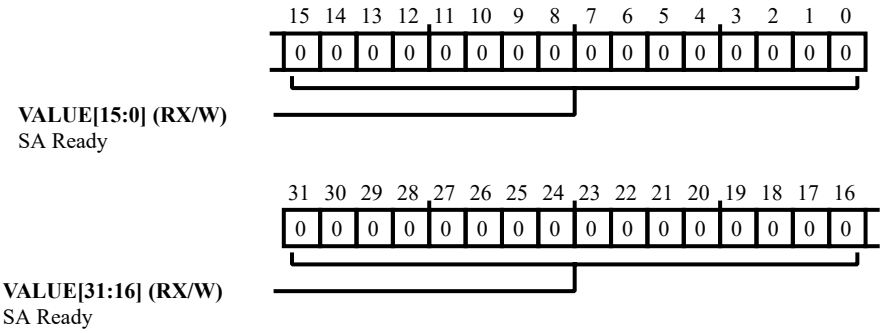


Figure 39-51: PKTE_SA_RDY Register Diagram

Table 39-77: PKTE_SA_RDY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	SA Ready.

SA Sequence Number Register

The PKTE_SA_SEQNUM[n] registers are a set of two read/write registers and are used for IPsec ESP, SSL, TLS, DTLS operations to specify the anti-replay sequence number value that is to be placed in the ESP header (outbound), or to be checked against for inbound packets. The packet engine manages this counter value for both inbound and outbound operations.

Outbound: The host writes the counter value stored in the SA record to this register to start an IPsec, SSL, TLS, DTLS operation. The packet engine automatically increments the count if header processing is selected. Upon successful completion, the host reads back this value and writes it to the SA record.

Inbound: The host writes the counter value stored in the SA record to this register to start an IPsec or DTLS operation. The packet engine automatically performs the specified inbound processing (per RFC 4303) as it processes the packet. As a result, the expected count value may or may not be updated during processing. Upon successful completion, the host should read back this value and write it to the SA record.

Note: The description is only for the direct host mode. The sequence number for autonomous ring mode and target command mode are updated by the packet engine.

The reset value of this register is zero.

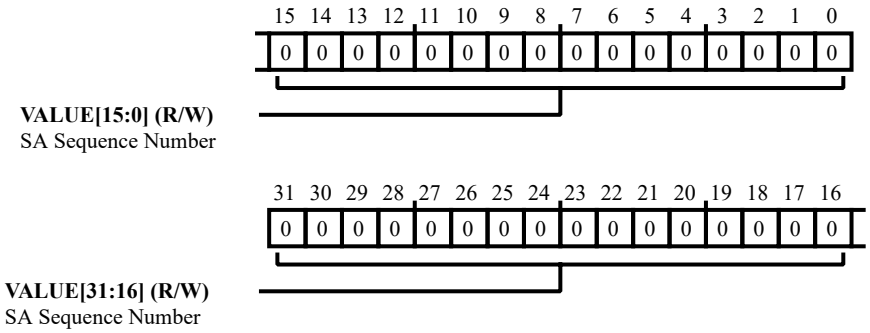


Figure 39-52: PKTE_SA_SEQNUM[n] Register Diagram

Table 39-78: PKTE_SA_SEQNUM[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Sequence Number.

SA Sequence Number Mask Registers

The PKTE_SA_SEQNUM_MSK[n] registers are a set of two read/write registers and are used for IPsec ESP and DTLS operations to specify the anti-replay sequence number mask value for inbound operations. The packet engine manages this counter value automatically.

Inbound: The host writes the counter value stored in the SA record into this register upon starting an IPsec, DTLS operation. The packet engine automatically performs the specified inbound processing (per RFC 4303) as it processes the packet. As a result, the new mask value may or may not be updated during processing. Upon successful completion, the host should read back this value and write it to the SA record.

Outbound: not used.

Note that the above description only applies to the direct host mode, for autonomous ring mode and target command mode the packet engine extracts the sequence number mask from the SA record.

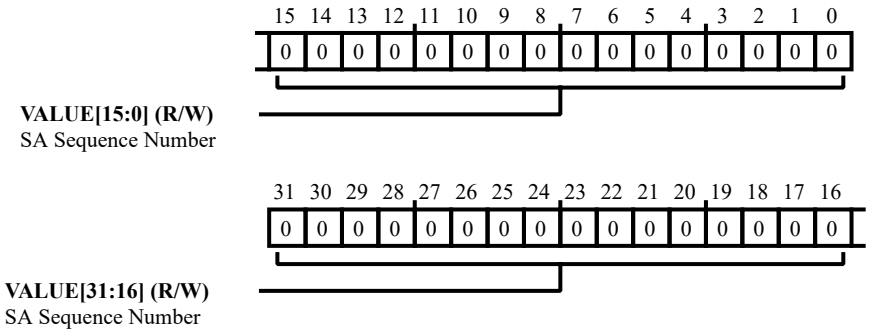


Figure 39-53: PKTE_SA_SEQNUM_MSK[n] Register Diagram

Table 39-79: PKTE_SA_SEQNUM_MSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Sequence Number.

SA SPI Register

For IPsec operations, the PKTE_SA_SPI register is written with the SPI (Security Parameters Index) associated with the inbound or outbound flow.

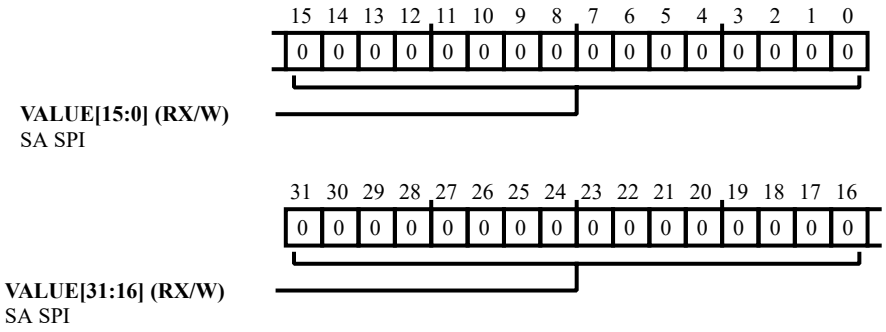


Figure 39-54: PKTE_SA_SPI Register Diagram

Table 39-80: PKTE_SA_SPI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	SA SPI. The PKTE_SA_SPI.VALUE bit field is used for IPsec ESP operations to specify the Security. Parameters Index (SPI) value that is to be placed in the ESP header. There is no need to read back this value at the end of an operation, since the Packet Engine does not change it. For SSL, TLS and DTLS this register stores the 8-bit TYPE field in bits [23:16], and the 16-bit Version field in bits [15:0] that are part of the protocol header.

Packet Engine Source Address

The PKTE_SRC_ADDR register holds the starting (byte) address for the packet to be processed.

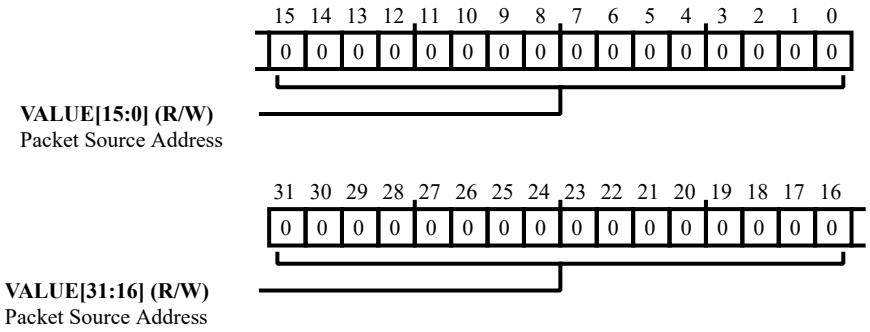


Figure 39-55: PKTE_SRC_ADDR Register Diagram

Table 39-81: PKTE_SRC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Packet Source Address. The PKTE_SRC_ADDR.VALUE bit field holds the starting (byte) address for the packet to be processed.

Packet Engine Status Register

The PKTE_STAT register is used to provide the status of the packet engine. This register is useful in the direct host mode to determine when data must be written to or read from the packet engine, or for debugging the software when errors occur. This register can be ignored in autonomous ring mode and target command mode where the DMA engine controls the packet data I/O. This is a read-only register. A write to any of the bits has no effect.

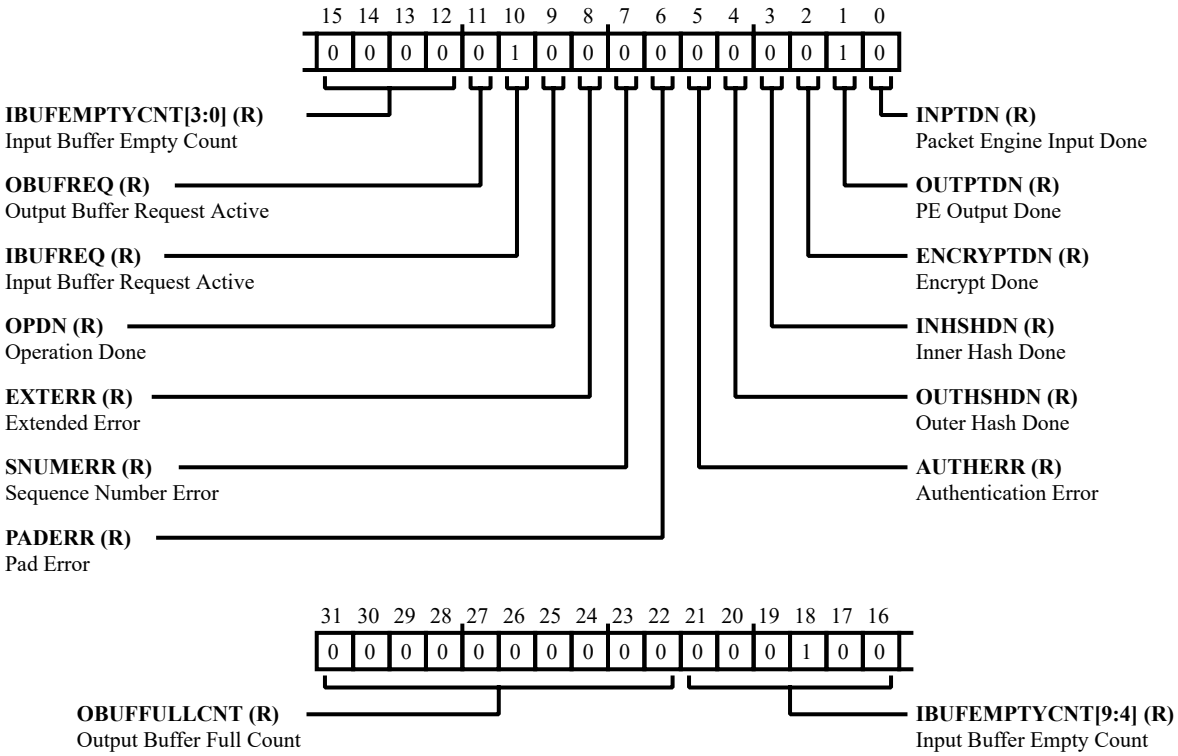


Figure 39-56: PKTE_STAT Register Diagram

Table 39-82: PKTE_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:22 (R/NW)	OBUFFULLCNT	Output Buffer Full Count. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of 32-bit words that are available in the packet engine output buffer. It works in conjunction with bit 11 from this register. When bit 11 is asserted, to indicate a request for output, the word count matches the specified output buffer threshold setting in the PKTE_BUF_THRESH register. For the last output for a given packet, any value from 1 dword to the full output buffer threshold can be seen. Transfers must be a multiple of full dwords. The application must read the PKTE_LEN field in the result descriptor to determine the exact byte-length of the result.

Table 39-82: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:12 (R/NW)	IBUFEMPTYCNT	Input Buffer Empty Count. The PKTE_STAT.IBUFEMPTYCNT bit field indicates the number of 32-bit empty spaces that are available in the packet engine input buffer. It works in conjunction with the PKTE_STAT.IBUFREQ bit (10) from this register. The value in the register is deducted from the specified packet length, so will never exceed the number of dwords that remain in the packet. For packets smaller than the buffer size, this register typically indicates that buffer space is available for the entire packet (rounded up to the nearest dword). For very large packets, these bits usually have a value around the maximum buffer size, indicating that the full input buffer is available.	
11 (R/NW)	OBUFREQ	Output Buffer Request Active. The PKTE_STAT.OBUFREQ bit indicates that the packet engine requests output data to be read from the output buffer.	
		0	No request for output data
		1	Request for output data
10 (R/NW)	IBUFREQ	Input Buffer Request Active. The PKTE_STAT.IBUFREQ bit indicates that the packet engine requests input data to be written to the input buffer.	
		0	No request for input data
		1	Request for input data
9 (R/NW)	OPDN	Operation Done. The PKTE_STAT.OPDN bit indicates that the packet engine has finished processing a packet when in direct host mode. This bit is zero in autonomous ring mode and target command mode.	
		0	Packet engine is idle
		1	Packet engine has finished processing a packet
8 (R/NW)	EXTERR	Extended Error. The PKTE_STAT.EXTERR bit indicates that an extended error occurred for this packet. For more information, refer to table Extended Error Codes - Status Encoding.	
		0	No extended error
		1	Extended error

Table 39-82: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SNUMERR	Sequence Number Error. For an inbound operation, the PKTE_STAT.SNUMERR bit indicates that there was a fault in the anti-replay sequence number. For an outbound operation, there was a sequence number overflow condition. For more information, refer to table Extended Error Codes - Status Encoding.	
		0	No sequence number error
		1	Input done, all bytes written to input buffer
6 (R/NW)	PADERR	Pad Error. The PKTE_STAT.PADERR bit indicates that an inbound crypto pad fault is detected. For more information about pad verification, refer to the Pad Verification and Consumption section.	
		0	No pad error
		1	Pad error
5 (R/NW)	AUTHERR	Authentication Error. The PKTE_STAT.AUTHERR bit indicates that an inbound ICV (for IPsec) or TAG (for SRTP) or MAC (for SSL/TLS/DTLS) fault is detected; the value carried within the packet did not match the value just computed.	
		0	No authentication error
		1	Authentication error
4 (R/NW)	OUTHSHDN	Outer Hash Done. The PKTE_STAT.OUTHSHDN bit indicates that the outer hash processing for this packet is finished.	
		0	Outer hash busy
		1	Outer hash done
3 (R/NW)	INHSHDN	Inner Hash Done. The PKTE_STAT.INHSHDN bit indicates that the inner hash processing for this packet is finished.	
		0	Inner hash busy
		1	Inner hash done
2 (R/NW)	ENCRYPTDN	Encrypt Done. The PKTE_STAT.ENCRYPTDN bit indicates that the encryption or decryption for this packet is finished.	
		0	Encryption or decryption busy
		1	Encryption or decryption done

Table 39-82: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	OUTPTDN	PE Output Done. The PKTE_STAT.OUTPTDN bit indicates that the output data for the current packet is read from the packet engine output buffer.	
		0	Output not done, more output bytes available
		1	Output done, all bytes read from the output buffer
0 (R/NW)	INPTDN	Packet Engine Input Done. The PKTE_STAT.INPTDN bit indicates that the number of bytes specified in the command descriptor PKTE_LEN field is written into the packet engine input buffer.	
		0	Input not done, more input bytes expected
		1	Input done, all bytes written to input buffer

Packet Engine State Record Address

The PKTE_STATE_ADDR register holds the start address of the SA state record.

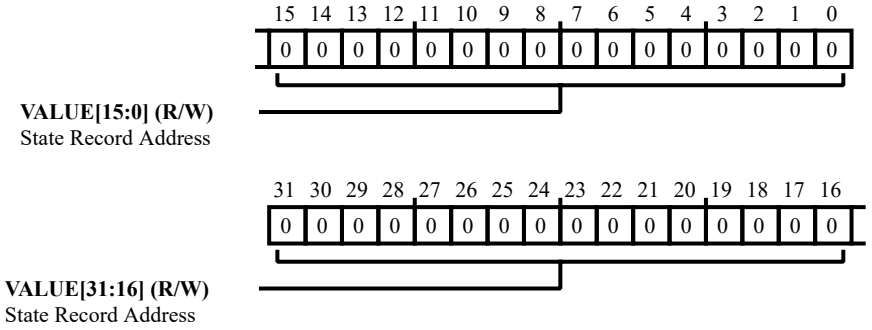


Figure 39-57: PKTE_STATE_ADDR Register Diagram

Table 39-83: PKTE_STATE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	State Record Address. The PKTE_STATE_ADDR.VALUE bit field holds the start address of the SA state record.

State Hash Byte Count Registers

The PKTE_STATE_BYTE_CNT[n] registers are used to enter a starting hash byte count, as well as to read the interim or final byte count.

For some hash operations, these registers are ignored and the byte count is internally set to 64 (0x40 hex) to indicate that the first 64 bytes (512 bits) hash block has been processed using a pre-computed hash state. These operations are:

All IPsec, SSL, TLS, DTLS and SRTP operations that use authentication; the "pre-computed" inner and outer hash digests are loaded from SA words 10 - 19.

Basic operations with PKTE_SA_CMD0.HASHSRC bits = 00 (from SA) specified. For Basic Hash with no HMAC, a pre-computed digest is loaded from SA words 10 - 14. For Basic Hash with HMAC, the inner and outer digests are loaded from SA words 10 - 19.

Note: Protocol operations can not be suspended in mid-packet and resumed later, therefore protocol operations do not use these registers.

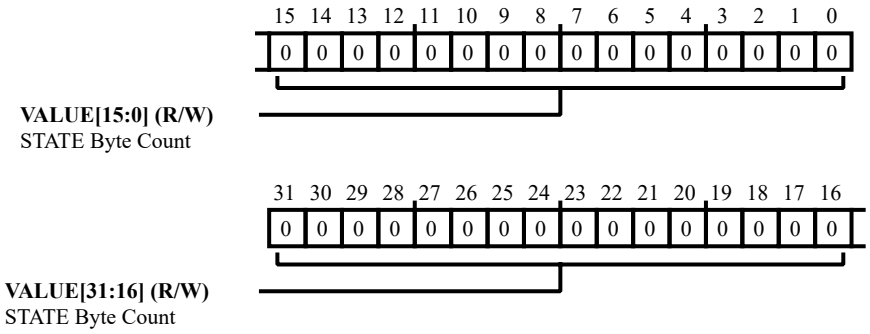


Figure 39-58: PKTE_STATE_BYTE_CNT[n] Register Diagram

Table 39-84: PKTE_STATE_BYTE_CNT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	STATE Byte Count.

State Inner Digest Registers

The PKTE_STATE_IDIGEST[n] registers consist of eight 32-bit registers. These read/write registers are used to read the interim or final hash digest. The PKTE_STATE_IDIGEST[n] registers are only used with basic operations involving basic hash, and are typically used for operations that must be suspended and resumed in the middle of a hash. The interim hash state can be read from these registers along with the hash byte-count from the previous register. Both can be restored when resuming the hash. The appropriate save hash state (PKTE_SA_CMD0.SVHASH=1) and load hash from state (PKTE_SA_CMD0.HASHSRC=0b10) settings must be used. These registers are a mirror of the SA record inner hash digest register.

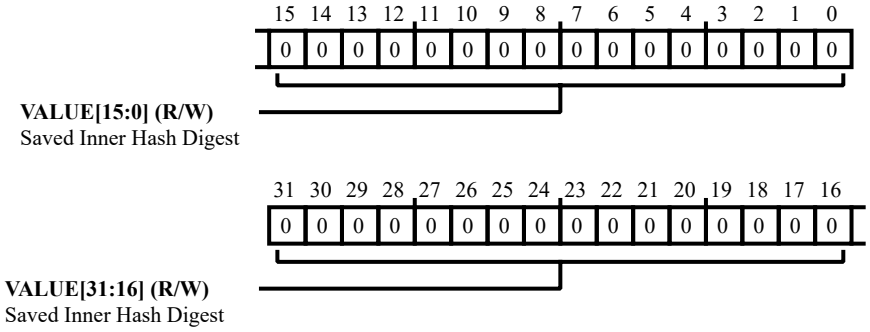


Figure 39-59: PKTE_STATE_IDIGEST[n] Register Diagram

Table 39-85: PKTE_STATE_IDIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Saved Inner Hash Digest.

State Initialization Vector Registers

The PKTE_STATE_IV[n] consists of four 32-bit registers. These registers are used to enter a starting IV state and to read the interim or final IV. PKTE_STATE_IV0 and PKTE_STATE_IV1 are used with DES/3DES cipher while PKTE_STATE_IV0 to PKTE_STATE_IV3 are used with AES cipher. The reset value of these registers is zero.

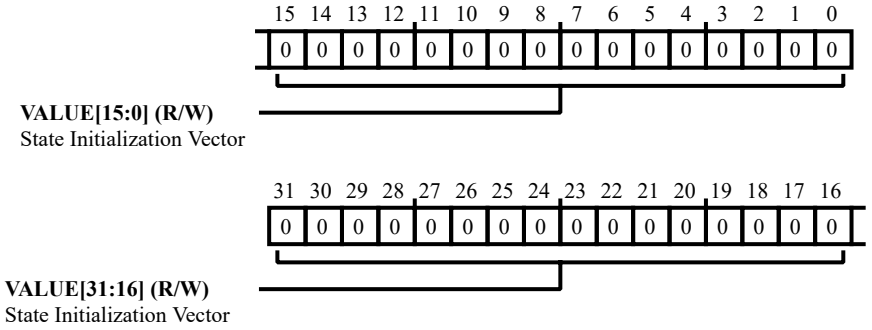


Figure 39-60: PKTE_STATE_IV[n] Register Diagram

Table 39-86: PKTE_STATE_IV[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	State Initialization Vector. The PKTE_STATE_IV[n].VALUE bit field is used to enter a starting IV state and to read the interim or final IV.

Packet Engine User ID

The PKTE_USERID register is a read/write register that gives identification to a command descriptor and the resultant result descriptor. The host is free to use this field for its own purpose. The host can write a unique identifier to the register in direct host mode or includes it as part of the command descriptor in autonomous ring mode. The PKTE_USERID register value passes though the packet engine without alteration to the result descriptor to be read back by the host.

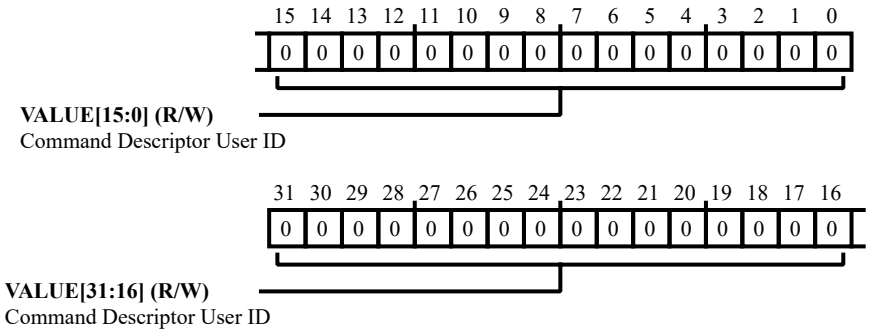


Figure 39-61: PKTE_USERID Register Diagram

Table 39-87: PKTE_USERID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Descriptor User ID. The PKTE_USERID.VALUE bit field gives identification to a command descriptor and the resultant result descriptor.

40 Public Key Accelerator (PKA)

The PKA helps offload computationally-intensive operations commonly found in public key cryptography algorithms.

PKA Features

The PKA engine provides the following basic operations:

- Large vector addition, subtraction, and combined addition/subtraction
- Large vector shift right or left
- Large vector multiplication, division (with and without quotient)
- Large vector compare and copy

The PKA engine provides the following complex operations:

- Large vector unsigned value modular exponentiation
- Large vector unsigned value modular exponentiation using the 'Chinese Remainders Theorem' (CRT) method with pre-calculated Q inverse vector
- Modular inversion: Given A and M, calculate B such that ((A × B) MOD M) = 1
- ECC point addition/doubling on elliptic curve $y^2 = x^3 + ax + b \pmod p$ with prime number *p* and input values *a* and *b* to the operation. Adding two identical points automatically performs point doubling.
- ECC point multiplication on elliptic curve $y^2 = x^3 + ax + b \pmod p$ with prime number *p* and input values *a* and *b* to the operation. A version of the 'Montgomery ladder' algorithm is used to provide side channel attack resistance.

The PKA also contains hardware logic to automatically zero out the PKA RAM buffer to clear out any information that is considered sensitive or secure.

PKA Functional Description

The following sections provide details on the function of the PKA module.

ADSP-2184x PKA Register List

The Public Key Accelerator module (PKA) provides security-related features. A set of registers governs PKA operations. For more information on PKA functionality, see the PKA register descriptions.

Table 40-1: ADSP-2184x PKA Register List

Name	Description
PKA_ALEN	PKA Vector_A Length
PKA_APTR	PKA Vector_A Address
PKA_BLEN	PKA Vector_B Length
PKA_BPTR	PKA Vector_B Address
PKA_COMPARE	PKA Compare Result
PKA_CPTR	PKA Vector_C Address
PKA_DIVMSW	PKA Most-Significant-Word of Divide Remainder
PKA_DPTR	PKA Vector_D Address
PKA_FUNC	PKA Function
PKA_RAM	Start of PKA RAM space
PKA_RESULTMSW	PKA Most-Significant-Word of Result Vector
PKA_SHIFT	PKA Bit Shift Value

PKA Definitions

The following definitions are helpful when using the PKA module.

Elliptic Curve Cryptography (ECC)

A form of public key cryptography based on elliptic curves over finite fields.

RSA

An acronym for Ron Rivest, Adi Shamir, and Leonard Adleman. It is another form of a public key cryptosystem.

Chinese Remainder Theorem (CRT)

A mathematical theorem used for simplifying time-consuming arithmetic used in public key algorithm computations.

Addition Chaining Table (ACT)

A method of speeding up exponentiation by repeatedly squaring the input, storing the result, and reusing the result as input. ACT2 uses a table with 2 address bits (4 entries) and ACT4 uses a table with 4 address bits (16 entries).

PKA Architectural Concepts

The following sections describe the PKA architecture.

Public Key Co-Processor (PKCP)

The Public Key Co-Processor (PKCP) handles the basic large vector processing such as addition, subtraction, multiplication, etc.

Sequencer

The sequencer is small processor that is part of the PKA which handles the more complicated vector processing for public key algorithms. Algorithms include modular exponentiation and the ECC addition and ECC multiply used in Elliptic Curve Cipher algorithms. It executes instructions stored from an internal pre-programmed ROM that handles these operations.

RAM

Input and output vectors are stored in a 4 kB RAM buffer that is part of the MMR space. The address of PKA_RAM is the beginning of the RAM space. This memory is also used as a scratchpad or workspace for the sequencer and PKCP. Programs must place the vectors appropriately following the constraints described in the ***Functional Description*** section of this chapter.

PKA Block Diagram

The ***PKA Block Diagram*** shows the top-level block diagram of the PKA engine. The PKA engine is comprised of five parts:

1. Registers for input, output, status, and control
2. Public Key Co-processor (PKCP) module which performs the basic suite of big number (vector) operations typically found in public key cryptography applications
3. Sequencer which controls modular exponentiation, elliptic curve cryptography, and modular inversion operations.
4. Program ROM associated with the PKA engine exclusively for the sequencer
5. PKA RAM holds the large input and output values as well as the workspace/scratchpad required from the sequencer and PKCP for operations.

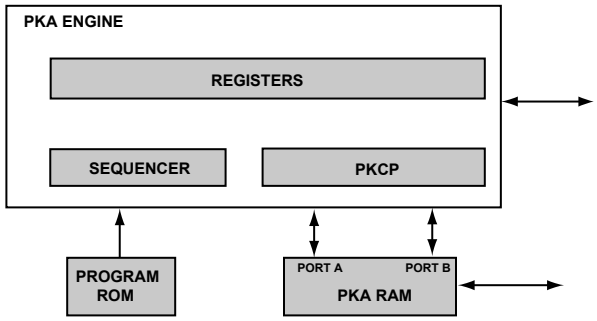


Figure 40-1: PKA Block Diagram

PKCP Vector Operations

The ***Summary of PKCP Vector Operations*** table lists the arguments and results for each PKCP vector operation.

Table 40-2: Summary of PKCP Vector Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
Multiply	A x B → C	Multiplicand	Multiplier	Product	N/A
Add	A + B → C	Addend	Addend	Sum	N/A
Subtract	A - B → C	Minuend	Subtracthend	Difference	N/A
AddSub	A + C - B → D	Addend	Subtracthend	Addend	Result
Right Shift	A >> Shift → C	Input	N/A	Result	N/A
Left Shift	A << Shift → C	Input	N/A	Result	N/A
Divide	A mod B → C, A div B → D	Dividend	Divisor	Remainder	Quotient
Modulo	A mod B → C	Dividend	Divisor	Remainder	N/A
Compare	A = B, A < B, A > B	Input 1	Input 2	N/A	N/A
Copy	A → C	Input	N/A	Result	N/A

To obtain correct result, the input vectors must meet the requirements presented in the ***Operational Restrictions on Input Vectors for PKCP Operations*** table.

Note the following:

- The PKCP does not check input restrictions
- A_Len and B_Len indicate the size of vectors A and B in 32-bit words
- Max_Len equals 128 (32-bit) words, for example, the standard maximum vector size is 4096 bits

Table 40-3: Operational Restrictions on Input Vectors for PKCP Operations

Function	Requirement
Multiply	0 < A_Len, B_Len <= Max_Len
Add	0 < A_Len, B_Len <= Max_Len
Subtract	0 < A_Len, B_Len <= Max_Len Result must be positive (A>=B)
AddSub	0 < A_Len <= Max_Len (B and C operands have A_Len as length, B_Len ignored) Result must be positive ((A + C) >= B)
Right Shift	0 < A_Len <= Max_Len
Left Shift	0 < A_Len <= Max_Len
Divide, Modulo	1 < B_Len <= A_Len <= Max_Len Most significant 32-bit word of B operand cannot be zero
Compare	0 < A_Len <= Max_Len (B operand has A_Len as length, B_Len ignored)
Copy	0 < A_Len <= Max_Len

The host processor is responsible for allocating a block of contiguous memory in PKA RAM for the result vectors. The ***PKCP Result Vector Memory Allocation*** table indicates how much memory is allocated for the result vectors.

Table 40-4: PKCP Result Vector Memory Allocation

Function	Result Vector	Result Vector Length (in 32-bit words)
Multiply	C	A_Len + B_Len + 6 (the 6 'scratchpad' words should be discarded)
Add	C	Max(A_Len, B_Len) + 1
Subtract	C	Max(A_Len, B_Len)
AddSub	D	A_Len + 1
Right Shift	C	A_Len
Left Shift	C	A_Len + 1 (when Shift Value is non-zero) A_Len (when Shift Value is zero)
Divide	C	Remainder → B_Len + 1 (one 'scratchpad' word should be discarded)
	D	Quotient → A_Len – B_Len + 1
Modulo	C	Remainder → B_Len + 1 (one 'scratchpad' word should be discarded)
Compare	None	Compare updates the PKA_COMPARE register
Copy	C	A_Len

Input vectors for an operation are always allowed to overlap in memory (partially or completely). The ***PKCP Result Vector/Input Overlap Restrictions*** table gives restrictions for the overlap of output and input vectors of the operations.

Table 40-5: PKCP Result Vector/Input Overlap Restrictions

Function	Result Vector	Restrictions
Multiply	C	No overlap with A or B vectors allowed
Add, Subtract	C	May overlap with A and/or B vector, provided the start address of the C vector does not lie above the start address of the vectors with which it overlaps
AddSub	D	May overlap with A, B and/or C vector, provided the start address of the D vector does not lie above the start address of the vectors with which it overlaps
Right Shift, Left Shift	C	May overlap with A vector, provided the start address of the C vector does not lie above the start address of the A vector
Divide	C	No overlap with A, B, or D vectors allowed
	D	No overlap with A, B, or C vectors allowed
Modulo	C	No overlap with A or B vectors allowed
Compare	None	Compare does not write a result vector
Copy	None	Same restrictions as for right or left shift, copy of a vector to a lower address is always allowed even if source and destination overlap†

†The copy operation can be used to fill memory by breaking the overlap restrictions, but it requires setting up TWO initial (32-bit) words. To zero a block of memory, set the A vector pointer to the block start, set the C vector pointer two words higher and the A vector length to the block length minus two (words). Fill the first two words of the block with constant zero and perform a PKCP copy operation to zero the remainder of the block.

Modular Exponentiation Operations

The ***Summary of ExpMod Operations*** table summarizes the modular exponentiation operations that the PKA supports.

Table 40-6: Summary of ExpMod Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	$C^A \bmod B \rightarrow D$	Exponent, length = A_Len	Modulus, length = B_Len	Base, length = B_Len	Result and Workspace
ExpMod-CRT	See below	Exp P followed by Exp Q at next higher even word address†, both A_Len long	Mod P + buffer word followed by Mod Q at next higher even word address‡, both B_Len long	Q inverse, length = B_Len	Input, Result (both 2xB-Len long) and Workspace

† If A_Len is even, Exp Q follows Exp P immediately – if A_Len is odd, there is one empty word between Exp Q and Exp P.

‡ If B_Len is even, there are two empty words between Mod P and Mod Q – if B_Len is odd, there is one empty (buffer) word between Mod Q and Mod P. Note that the engine may zero the words following Mod P and Mod Q.

The ExpMod-CRT operation performs the following computation steps. (These steps implement Garner's recombination algorithm after the basic exponentiations.)

- X <- (Input mod Mod P)Exp P mod Mod P
- Y <- (Input mod Mod Q)Exp Q mod Mod Q
- Z <- ((((X – Y) mod Mod P) · Q inverse) mod Mod P) · Mod Q
- Result <- Y + Z

The ExpMod-ACT2, -ACT4, and -variable functions implement the same mathematical operation but with a differently sized table with pre-calculated *odd powers*. The ExpMod-ACT2 function uses a table with two entries whereas ExpMod-ACT4 uses a table with eight entries. The ACT4 version gives better performance but needs more memory. ExpMod-variable and ExpMod-CRT operations allow the selection of a variable number (from 1 up to and including 16) of odd powers through the register normally used to specify the number of bits to shift for shift operations.

The exponentiation functions are extensions of the set of PKA functions. Input and result vectors are passed the same way as basic PKCP operations. The ***Restrictions on Input Vectors for ExpMod Operations*** table shows the restrictions on the input and result vectors for the exponentiation operations.

Table 40-7: Restrictions on Input Vectors for ExpMod Operations

Function	Requirements
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	0 < A_Len <= Max_Len 1 < B_Len <= Max_Len Modulus B must be odd (for example, the least significant bit must be ONE) Modulus B > 2^{32} Base C < Modulus B Vectors B and C must be followed by an empty 32-bit *buffer* word
ExpMod-CRT	0 < A_Len <= Max_Len 1 < B_Len <= Max_Len Mod P and Mod Q must be odd (for example, the least significant bits must be ONE) Mod P > Mod Q > 2^{32}(note that Mod P must be larger than Mod Q) Mod P and Mod Q must be co-prime (their GCD must be 1) 0 < Exp P < (Mod P – 1) 0 < Exp Q < (Mod Q – 1) (Q inverse · Mod Q) 1 (modulo Mod P) Input < (Mod P · Mod Q) Mod P and Mod Q must be followed by an empty 32-bit *buffer* word

The ***ExpMod Result Vector/Scratchpad Area Memory Allocation Starting at PKA_DPTR*** table shows the required scratchpad sizes for the exponentiation operations. These sizes depend on the PKA type. The 'M_Len' used in the table is the 'real' Modulus length in 32-bit words, for example, without trailing zero words at the end. (This description also applies to Mod P in an ExpMod-CRT operation and Modulus B in the other operations.) If the last word of the modulus vector as given is non-zero, 'M_Len' equals B_Len.

Table 40-8: ExpMod Result Vector/Scratchpad Area Memory Allocation Starting at PKA_DPTR

Function	Scratchpad Area Size (in 32-bit words), Result Vector is either M_Len or 2xM_Len 32-bit words long
ExpMod-ACT2	5 x (M_Len + 2)
ExpMod-ACT4	11 x (M_Len + 2)
ExpMod-variable	(# odd powers + 3) x (M_Len + 2)
ExpMod-CRT	(# odd powers + 3) x (M_Len + 2) + (M_Len + 2 – (M_Len MOD 2))

NOTE: During execution of an ExpMod-ACT2, -ACT4 or -variable operation, the last 34 bytes of the PKA RAM are used as the general scratchpad for the sequencer program execution. The ExpMod-CRT operation requires the last 72 bytes of the PKA RAM as the scratchpad. These (fixed location) areas may not overlap with any of the input vectors and/or the D vector scratchpad area. They can be used freely when executing basic PKCP operations.

Table 40-9: ExpMod Scratchpad Area / Input Vector Overlap Restrictions

Function	Result Vector	Restrictions
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	D	Scratchpad area starting at D may not overlap with any of the other vectors, except that base C may be co-located with result vector D to save space (for example, `PKA_CPTR`= `PKA_DPTR` is allowed).
ExpMod-CRT	D	Scratchpad area starting at D may not overlap with any of the other vectors. This area is also the location of the main input vector (with length 2 x B_Len)

The ***Maximum Number of Odd Powers*** table indicates the maximum number of odd powers that can be used for different standard PKA RAM sizes and PKA types (non-CRT operations using `PKA_CPTR` = `PKA_DPTR`). As a rule of thumb, for optimal performance, use one odd power for *Verify* operations and 4 (or as many as the implemented PKA RAM size allows) for *Sign* operations. Note the following points about odd powers:

- Using more than eight odd powers is not advisable as the speed advantage for each extra odd power decreases rapidly (and can even become negative for short exponent vector lengths due to the extra pre-processing required).
- The maximum number of odd powers is 16 (limited by the firmware). All '16 odd powers' entries in the table above hit this limit – they are not limited by the PKA RAM size.

Table 40-10: Maximum Number of Odd Powers

Operation	Modulus and Exponent Sizes	Maximum Number of Odd Powers
Non-CRT	1024 bits	16
	2048 bits	10
	4096 bits	2
CRT	2 × 512 bits	16
	2 ×1024 bits	16
	2 × 2048 bits	6

The ***2K-bit Modular Exponentiation PKA RAM Allocation Examples*** table shows example PKA RAM vector allocations for modular exponentiation operations with and without using CRT. The *free space* start address is the first free byte following the vector workspace. The sequencer execution scratchpad of 34 bytes (non-CRT) or 72 bytes (using CRT) must fit between this address and the end of the PKA RAM. Note that the non-CRT operations use `PKA_CPTR` = `PKA_DPTR` to save space.

Table 40-11: 2K-bit Modular Exponentiation PKA RAM Allocation Examples

Operation	(sub-)vector	Start address (Byte Offset)	Size (words)	Buffer (words)
non-CRT (`PKA_ALEN` = 0x040, `PKA_BLEN` = 0x040, 4 odd-powers)	Exponent	0x000 (`PKA_APTR` = 0x000)	64	0
	Modulus	0x100 (`PKA_BPTR` = 0x040)	64	2
	Base	0x208 (`PKA_CPTR` = 0x082)	64	2
	Result	0x208 (`PKA_DPTR` = 0x082)	64	2
	Vector Workspace	0x208 (= Result)	7 × (64+2)=462	0
	Free space	0x940 (2368 bytes used)	-	-
using CRT (`PKA_ALEN` = 0x020, `PKA_BLEN` = 0x020, 4 odd-powers)	Exp P	0x000 (`PKA_APTR` = 0x000)	32	0
	Exp Q	0x080	32	0
	Mod P	0x100 (`PKA_BPTR` = 0x040)	32	2
	Mod Q	0x188	32	2
	Q inverse	0x210 (`PKA_CPTR` = 0x084)	32	0
	Input, Result	0x290 (`PKA_DPTR` = 0x0A4)	64	0
	Vector workspace	0x290 (= Result)	7 × (32+2)+32+2-0 = 272	0
	Free space	0x6D0 (1744 bytes used)	-	-

The following example in pseudo-code describes the execution of a non-CRT modular exponentiation operation using a 512-bit modulus and a 160-bit exponent, using actual test vectors:

```
// Perform a 512/160 bit
modular exponentiation without CRT (using 4 'odd-powers')
// Exponent equals value 0x8FD84098_8A0930CC_9CDC1E8A_B246EB46_2D39F064
// write as vector A to PKA
```

```
RAM Byte offset 0x000:
Write PKA_RAM_BASE+0x000+0x00
0x2D39F064
Write PKA_RAM_BASE+0x000+0x04
0xB246EB46
Write PKA_RAM_BASE+0x000+0x08
0x9CDC1E8A
Write PKA_RAM_BASE+0x000+0x0C
0x8A0930CC
Write PKA_RAM_BASE+0x000+0x10
0x8FD84098
// Modulus equals value 0xF42F559D
1877CA5F_449492B9_42DC7C01_...
// A3C9085B_7236A085_2102B000_A093C6B4_...
// 9D0EDA0C_292DE841_29C23723_4048BDA3_...
// 373C4C9F_45CF15A7_5F049ABF_D8A01B9B
// write as vector B to PKA
RAM Byte offset 0x018 (following exp at next aligned 64-bit word):
Write PKA_RAM_BASE+0x018+0x00
0xD8A01B9B
Write PKA_RAM_BASE+0x018+0x04
0x5F049ABF
Write PKA_RAM_BASE+0x018+0x08
0x45CF15A7
Write PKA_RAM_BASE+0x018+0x0C
0x373C4C9F
Write PKA_RAM_BASE+0x018+0x10
0x4048BDA3
Write PKA_RAM_BASE+0x018+0x14
0x29C23723
Write PKA_RAM_BASE+0x018+0x18
0x292DE841
Write PKA_RAM_BASE+0x018+0x1C
0x9D0EDA0C
Write PKA_RAM_BASE+0x018+0x20
0xA093C6B4
Write PKA_RAM_BASE+0x018+0x24
0x2102B000
Write PKA_RAM_BASE+0x018+0x28
0x7236A085
Write PKA_RAM_BASE+0x018+0x2C
0xA3C9085B
Write PKA_RAM_BASE+0x018+0x30
0x42DC7C01
Write PKA_RAM_BASE+0x018+0x34
0x449492B9
Write PKA_RAM_BASE+0x018+0x38
0x1877CA5F
Write PKA_RAM_BASE+0x018+0x3C
0xF42F559D
```

```
// Base equals value 0x3D291F48_49064887_1149594B_67935110_...
// 14EB8FF0_AB291F3A_54A1B4D1_5E611E44_...
// C989251B_44904B45_0B060482_317F8352_...
// 18CE440E_9BF509F1_6EAF26F2_95F19F12
// write as vector C to PKA
RAM Byte offset 0x060 (following mod after buffer + align words):
Write PKA_RAM_BASE+0x060+0x00
0x95F19F12
Write PKA_RAM_BASE+0x060+0x04
0x6EAF26F2
Write PKA_RAM_BASE+0x060+0x08
0x9BF509F1
Write PKA_RAM_BASE+0x060+0x0C
0x18CE440E
Write PKA_RAM_BASE+0x060+0x10
0x317F8352
Write PKA_RAM_BASE+0x060+0x14
0x0B060482
Write PKA_RAM_BASE+0x060+0x18
0x44904B45
Write PKA_RAM_BASE+0x060+0x1C
0xC989251B
Write PKA_RAM_BASE+0x060+0x20
0x5E611E44
Write PKA_RAM_BASE+0x060+0x24
0x54A1B4D1
Write PKA_RAM_BASE+0x060+0x28
0xAB291F3A
Write PKA_RAM_BASE+0x060+0x2C
0x14EB8FF0
Write PKA_RAM_BASE+0x060+0x30
0x67935110
Write PKA_RAM_BASE+0x060+0x34
0x1149594B
Write PKA_RAM_BASE+0x060+0x38
0x49064887
Write PKA_RAM_BASE+0x060+0x3C
0x3D291F48
// The result value and scratchpad
(vector D) may be co-located with the base vector C for
// a normal modular exponentiation,
so these are located at PKA RAM Byte offset 0x060 too.
// Load pointer and length
registers:
Write PKA_APTR 0x000>>2 //
Exponent pointer
Write PKA_BPTR 0x018>>2 //
Modulus pointer
Write PKA_CPTR 0x060>>2 //
Base pointer
```

```
Write PKA_DPTR 0x060>>2 //
Result/scratchpad pointer
Write PKA_ALENGTH 0x00000005
// Exponent length in 32-bit words
Write PKA_BLENGTH 0x00000010
// Mod/base/result length in 32-bit words
// Start modular exponentiation
and wait until it's done:
Write PKA_SHIFT 0x00000004
// Number of 'odd powers'
Write PKA_FUNCTION 0x0000E000
// 'Run' bit set, 'Sequencer Operations' = 0b110
Wait PKA_FUNCTION[15] == '0'
// 'Run' bit clears itself - Host can also use interrupt!
// Result value equals 0xA497BF8B_DB729088_954005B0_B5CA6691_...
// A3EC491B_091A3D62_03C24214_0863A389_...
// 0C7C03CD_2333E231_35EC10ED_8F91281C_...
// 30F4253B_FE38FAFB_BB4A39DB_C14F2661
// written as vector D at
PKA RAM Byte offset 0x060:
Check PKA_RAM_BASE+0x060+0x00
== 0xC14F2661
Check PKA_RAM_BASE+0x060+0x04
== 0xBB4A39DB
Check PKA_RAM_BASE+0x060+0x08
== 0xFE38FAFB
Check PKA_RAM_BASE+0x060+0x0C
== 0x30F4253B
Check PKA_RAM_BASE+0x060+0x10
== 0x8F91281C
Check PKA_RAM_BASE+0x060+0x14
== 0x35EC10ED
Check PKA_RAM_BASE+0x060+0x18
== 0x2333E231
Check PKA_RAM_BASE+0x060+0x1C
== 0x0C7C03CD
Check PKA_RAM_BASE+0x060+0x20
== 0x0863A389
Check PKA_RAM_BASE+0x060+0x24
== 0x03C24214
Check PKA_RAM_BASE+0x060+0x28
== 0x091A3D62
Check PKA_RAM_BASE+0x060+0x2C
== 0xA3EC491B
Check PKA_RAM_BASE+0x060+0x30
== 0xB5CA6691
Check PKA_RAM_BASE+0x060+0x34
== 0x954005B0
Check PKA_RAM_BASE+0x060+0x38
== 0xDB729088
```

```
Check PKA_RAM_BASE+0x060+0x3C
== 0xA497BF8B
```

Modular Inversion

Besides modular exponentiation, the sequencer also controls modular inversion operations.

Table 40-12: Summary of ModInv Operation

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ModInv	A^{-1} mod B → D	NumToInvert, length = A_Len	Modulus, length = B_Len	Not Used	Result and Workspace

The above function is an extension of the set of basic PKCP functions with the following exceptions:

- Vector D not only addresses the result, but also a workspace.
- The `PKA_SHIFT` register field is used to return information on the operation's result.

Table 40-13: PKA_SHIFT Result Values for ModInv Operation

Function	PKA_SHIFT Register Field Value At Conclusion
ModInv	0 → success; VectorD holds result 7 → no inverse exists (GCD(A, B) != 1, for example, A and B have common factors); result undefined 31 → error, modulus even; result undefined other values are reserved

The following tables list the restrictions on the input and result vectors for the ModInv operation.

Table 40-14: Operational Restrictions on Input Vectors for the ModInv Operation

Function	Requirements
ModInv	0 < A_Len <= Max_Len 0 < B_Len <= Max_Len Modulus B must be odd (for example, the least significant bit must be ONE) Modulus B may not have value 1 (result is undefined, no error indicated) The highest word of the modulus vector, as indicated by B_Len, may not be zero.

Table 40-15: ModInv Scratchpad Area/Input Vector Overlap Restrictions

Function	Result Vector	Restrictions
ModInv	D	Scratchpad area starting at D may not overlap with any of the other vectors

The following table shows the required scratchpad sizes for the ModInv Operation.

Table 40-16: ModInv Result Vector/Scratchpad Area Memory Allocation (Both Starting at PKA_DPTR)

Function	Scratchpad area size (in 32-bit words), Result Vector is B_Length 32-bit words long
ModInv	5 x (M + ε(M)), with M = Max(A_Length, B_Length) ε(n) = 2 + (n MOD 2), for example, 2 (for n even) or 3 (for n odd)

NOTE: During execution of a ModInv operation, the last 34 bytes of the PKA RAM are used as general scratchpad for the sequencer program execution. This (fixed location) area may not overlap with any of the input vectors and/or the D vector scratchpad area during execution.

Modular Inversion with an Even Modulus

The ModInv operation requires the modulus to be odd. At first, this requirement appears to make the operation useless in the case of RSA key generation where the private key exponent *d* is derived from a chosen public exponent *e* as follows:

d = ModInv(e, φ); where φ = (p-1) × (q-1) and *p* and *q* both prime

Note that φ is even. However, since *e* must be odd (otherwise no inverse exists), *d* can be calculated as:

d = (1 + (φ × (e – ModInv(φ, e))) / e

With four more basic PKCP operations, ModInv can also be used to find inverse values in case the modulus is even.

Modular Inversion with a Prime Modulus

Modular inversion can be performed with a modular exponentiation using the modulus value minus two as exponent, provided that the modulus value is a prime. This is due to the following:

(A^M) mod M = A =>

(A^{M-1}) mod M = 1 =>

(A^{M-2}) mod M = A–1 (mod M)

Under the constraint that M is a prime value.

Especially with the large PKA engines containing an LNME, it is worthwhile to check whether this method is faster than using the ModInv operation directly. The modulus values for the ECC curves supported by this PKA engine must be prime, so this method can be used in ECDSA operations.

ECC Operations

Besides modular exponentiation and modular inversion, the sequencer also controls ECC operations.

Table 40-17: Summary of ECC Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ECC-ADD	Point addition/doubling† on elliptic curve: $y^2 = x^3 + ax + b \pmod{p}$ pntA + pntC → pntD	pntA.x followed‡ by pntA.y both B_Len long (A_Len *not* used)	Curve parameter *p* followed‡ by *a* (*b* is not needed) all B_Len long	pntC.x followed by pntC.y both B_Len long	Result, for example, pntD.x followed‡ by pntD.y and workspace
ECC-MUL	Point multiplication on elliptic curve: $y^2 = x^3 + ax + b \pmod{p}$ *k* x pntC → pntD	Scalar *k* A_Len long	Curve parameter *p* followed‡ by *a*and *b* all B_Len long.	pntC.x followed‡ by pntC.y both B_Len long	Result, for example pntD.x followed‡ by pntD.y and workspace

† If pntA = pntC, a point doubling operation is performed automatically.

‡ All input components must be located on a 64-bit boundary and must have extra 'buffer' words (of 32 bits each) after their most significant word. ε must be 3 (B_Len odd) or 2 (B_Len even). Each result component (for example, pntD.x, pntD.y) is followed by ε buffer (zero) words.

The above functions are extensions of the set of PKCP basic functions with the following exceptions:

- Input and result vectors can now be composite (for example, consist of two or three equal-sized subvectors).
- Vector D not only addresses the result, but also a workspace.
- The `PKA_SHIFT` register is used to return information on the result of the operation.

Table 40-18: PKA_SHIFT Result Values for ECC Operations

Function	PKA_SHIFT Register Field Value at Conclusion
ECC-ADD ECC-MUL	0 → success; VectorD holds result point 7 → result is point-at-infinity; VectorD result point undefined 31 → error, (p not odd, p too short, etc); VectorD result point undefined Other values are reserved

The following tables below list the restrictions on the input and result vectors for the ECC operations.

Table 40-19: Operational Restrictions on Input Vectors for ECC Operations

Function	Requirements
ECC-ADD	1 < B_Len <= 24 (maximum vector length is 768 bits) Modulus p must be a prime > 2^{63} Effective modulus size (in bits) must be a multiple of 32† The highest word of the modulus vector, as indicated by B_Len, may not be zero. *a*< p and *b*< p pntA and pntC must be on the curve (this condition is not checked) Neither pntA nor pntC can be the point-at-infinity, although ECC-ADD can return this point as a result
ECC-MUL	0 < A_Len <= 24 (maximum vector length is 768 bits) 1 < B_Len <= 24 (maximum vector length is 768 bits) Modulus p must be a prime > 2^{63} Effective modulus size (in bits) must be a multiple of 32† The highest word of the modulus vector, as indicated by B_Len, may not be zero. *a*< p and *b*< p pntC must be on the curve (this condition is not checked) pntC cannot be the point-at-infinity, although ECC-MUL can return this point as a result

† Modulus lengths of 112 and 521 bits are exceptions to this rule.

Table 40-20: ECC Scratchpad Area/Input Vector Overlap Restrictions

Function	Result Vector	Restrictions
ECC-ADD ECC-MUL	D	Scratchpad area starting at D may not overlap with any of the other vectors

The ***>ECC Result Vector/Scratchpad Area Memory Allocation*** table shows the required scratchpad sizes for the ECC operations:

Table 40-21: ECC Result Vector/Scratchpad Area Memory Allocation (Both Starting at PKA_DPTR)

Function	Scratchpad area size (in 32-bit words), Result Vector is 2x(B_Length+ ε †(B_Length)) 32-bit words long
ECC-ADD	2 × L + 5 x M, where L = B_Length + ε(B_Length) M = B_Length + 1 + ε(B_Length + 1)
ECC-MUL	18 x L + Max(8, L), where L = (B_Length + ε(B_Length))

† ε(*n*) = 2 + (*n* MOD 2), for example, 2 (for *n* even) or 3 (for *n* odd)

NOTE: During execution of an ECC-ADD or ECC-MUL operation, the last 72 bytes of the PKA RAM are used as a general scratchpad for the sequencer program execution. These (fixed location) areas must not overlap with any of the input vectors and the D vector scratchpad area during execution.

The ***ECC Point Multiplication PKA RAM Allocation Examples*** table shows example PKA RAM vector allocations for ECC point multiplication operations. The *free space* start address is the first free byte following the vector scratchpad. The sequencer execution scratchpad of 72 bytes must fit between this address and the end of the PKA RAM. Because of this requirement, a 521-bit ECC point multiplication cannot be performed with 2K byte PKA RAM.

Table 40-22: ECC Point Multiplication PKA RAM Allocation Examples

Modulus Length	(sub-)vector	Start Address (byte offset)	Size (words)	Buffer (words)
192 bits (=6 words, PKA_ALEN=0x006, PKA_BLEN=0x006)	Scalar *k*	0x000 (PKA_APTR = 0x000)	6	0
	p	0x018 (PKA_BPTR = 0x006)	6	2
	a	0x038	6	2
	b	0x058	6	2
	PntC.x (base)	0x078 (PKA_CPTR = 0x01E)	6	2
	PntC.y (base)	0x098	6	2
	PntD.x (result)	0x0B8 (PKA_DPTR = 0x02E)	6	2
	PntD.y (result)	0x0D8	6	0
	Vector scratchpad	0x0B8 (= PntD.x)	(18 × 8) + 8 = 152	0
	Free space	0x318 (792 bytes used)	-	-
384 bits (=12 words, ALENGTH=0x00C, BLENGTH=0x00C)	Scalar *k*	0x000 (PKA_APTR = 0x000)	12	0
	p	0x030 (PKA_BPTR = 0x00C)	12	2
	a	0x068	12	2
	b	0x0A0	12	2
	PntC.x (base)	0x0D8 (PKA_CPTR = 0x036)	12	2
	PntC.y (base)	0x110	12	2
	PntD.x (result)	0x148 (PKA_DPTR = 0x052)	12	2
	PntD.y (result)	0x180	12	0
	Vector scratchpad	0x148 (= PntD.x)	(18 × 14) + 14 = 266	0
	Free space	0x570 (1392 bytes used)	-	-
521 bits (=17 words, ALENGTH=0x011, BLENGTH=0x011)	Scalar *k*	0x000 (PKA_APTR = 0x000)	17	1 (to align *p*)
	p	0x048 (PKA_BPTR = 0x012)	17	3
	a	0x098	17	3
	b	0x0E8	17	3
	PntC.x (base)	0x138 (PKA_CPTR = 0x04E)	17	3
	PntC.y (base)	0x188	17	3
	PntD.x (result)	0x1D8 (PKA_DPTR = 0x076)	17	3

Table 40-22: ECC Point Multiplication PKA RAM Allocation Examples (Continued)

Modulus Length	(sub-)vector	Start Address (byte offset)	Size (words)	Buffer (words)
	PntD.y (result)	0x228	17	0
	Vector scratchpad	0x1D8 (= PntD.x)	(18 × 20) + 20 = 380	0
	Free space	0x7C8 (1992 bytes used)	-	-

The following example in pseudo-code describes the execution of a 192 bits ECC point multiplication, using actual test vectors (the curve parameters and generator point are from standard curve 'secp192r1').

```
// Perform a 192 bits ECC point multiplication using PKA RAM layout from table
above.
// Scalar 'k' equals value 0x8D98D058_9EFD018A_C9BCF3CF_2C33AEC0_24867D7F_6ADACBFF
// write as vector A to PKA RAM Byte offset 0x000:

    Write PKA_RAM_BASE+0x000+0x00 0x6ADACBFF
    Write PKA_RAM_BASE+0x000+0x04 0x24867D7F
    Write PKA_RAM_BASE+0x000+0x08 0x2C33AEC0
    Write PKA_RAM_BASE+0x000+0x0C 0xC9BCF3CF
    Write PKA_RAM_BASE+0x000+0x10 0x9EFD018A
    Write PKA_RAM_BASE+0x000+0x14 0x8D98D058
// Curve parameter 'p' equals value
0xFFFFFFFF_FFFFFFFF_FFFFFFFF_FFFFFFFE_FFFFFFFF_FFFFFFFF
// write as 1st part of vector B immediately following vector A at PKA RAM Byte
offset 0x018
// (no buffer word needed after 'k' vector, 64-bit alignment is OK):
    Write PKA_RAM_BASE+0x018+0x00 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x04 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x08 0xFFFFFFFE
    Write PKA_RAM_BASE+0x018+0x0C 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x10 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x14 0xFFFFFFFF
// Curve parameter 'a' equals value
0xFFFFFFFF_FFFFFFFF_FFFFFFFF_FFFFFFFE_FFFFFFFF_FFFFFFFC
// write as 2nd part of vector B after one buffer word and one re-alignment word
at 0x038:
    Write PKA_RAM_BASE+0x038+0x00 0xFFFFFFFC
    Write PKA_RAM_BASE+0x038+0x04 0xFFFFFFFF
    Write PKA_RAM_BASE+0x038+0x08 0xFFFFFFFE
    Write PKA_RAM_BASE+0x038+0x0C 0xFFFFFFFF
    Write PKA_RAM_BASE+0x038+0x10 0xFFFFFFFF
    Write PKA_RAM_BASE+0x038+0x14 0xFFFFFFFF
// Curve parameter 'b' equals value
0x64210519_E59C80E7_0FA7E9AB_72243049_FEB8DEEC_C146B9B1
// write as 3rd part of vector B after one buffer word and one re-alignment word
at 0x058:
    Write PKA_RAM_BASE+0x058+0x00
0xC146B9B1
```

```
    Write PKA_RAM_BASE+0x058+0x04 0xFEB8DEEC
    Write PKA_RAM_BASE+0x058+0x08 0x72243049
    Write PKA_RAM_BASE+0x058+0x0C 0x0FA7E9AB
    Write PKA_RAM_BASE+0x058+0x10 0xE59C80E7
    Write PKA_RAM_BASE+0x058+0x14 0x64210519
// X-coord of generator point is value
0x188DA80E_B03090F6_7CBF20EB_43A18800_F4FF0AFD_82FF1012
// write as 1st part of vector C following vector B after buffer + alignment
words at 0x078:
    Write PKA_RAM_BASE+0x078+0x00 0x82FF1012
    Write PKA_RAM_BASE+0x078+0x04 0xF4FF0AFD
    Write PKA_RAM_BASE+0x078+0x08 0x43A18800
    Write PKA_RAM_BASE+0x078+0x0C 0x7CBF20EB
    Write PKA_RAM_BASE+0x078+0x10 0xB03090F6
    Write PKA_RAM_BASE+0x078+0x14 0x188DA80E
// Y-coord of generator point is value
0x07192B95_FFC8DA78_631011ED_6B24CDD5_73F977A1_1E794811
// write as 2nd part of vector C after one buffer word and one re-alignment word
at 0x098:
    Write PKA_RAM_BASE+0x098+0x00 0x1E794811
    Write PKA_RAM_BASE+0x098+0x04 0x73F977A1
    Write PKA_RAM_BASE+0x098+0x08 0x6B24CDD5
    Write PKA_RAM_BASE+0x098+0x0C 0x631011ED
    Write PKA_RAM_BASE+0x098+0x10 0xFFC8DA78
    Write PKA_RAM_BASE+0x098+0x14 0x07192B95
// The result point and scratchpad (vector D) follow vector C after one buffer
word and one
// re-alignment word, so these are located at PKA RAM Byte offset 0x0B8.
// Load pointer and length registers:
    Write PKA_APTR 0x000>>2 // Scalar 'k' pointer
    Write PKA_BPTR 0x018>>2 // Curve parameters 'p', 'a' & 'b' pointer
    Write PKA_CPTR 0x078>>2 // Generator point X & Y coordinates pointer
    Write PKA_DPTR 0x0B8>>2 // Result point X & Y coordinates/scratchpad pointer
    Write PKA_ALENGTH 0x00000006 // Scalar 'k' length in 32-bit words
    Write PKA_BLENGTH 0x00000006 // Curve parameters and coordinate lengths in
32-bit words
// Start ECC point multiplication and wait until it's done:
    Write PKA_FUNCTION 0x0000D000
// 'Run' bit set, 'Sequencer Operations' = 0b101
    Wait PKA_FUNCTION[15] == '0'
// 'Run' bit clears itself - Host can also use interrupt!
    Check PKA_SHIFT == 0x00000000
// Shift field value 0 indicates success - check this
// X-coord of result point is value
0x759B9F39_0E81D268_18C82BB9_CB42BCF5_0E0AE958_85BA3097
// written as 1st part of vector D at PKA RAM Byte offset 0x0B8:
    Check PKA_RAM_BASE+0x0B8+0x00== 0x85BA3097
    Check PKA_RAM_BASE+0x0B8+0x04== 0x0E0AE958
    Check PKA_RAM_BASE+0x0B8+0x08== 0xCB42BCF5
    Check PKA_RAM_BASE+0x0B8+0x0C== 0x18C82BB9
```

```
    Check PKA_RAM_BASE+0x0B8+0x10== 0x0E81D268
    Check PKA_RAM_BASE+0x0B8+0x14== 0x759B9F39
// Y-coord of result point is value
0xECA14640_F92EFF07_CAF2BD55_3FBE28EF_D043F28E_1CC3D238
// written as 2nd part of vector D at PKA RAM Byte offset 0x0D8:
    Check PKA_RAM_BASE+0x0D8+0x00== 0x1CC3D238
    Check PKA_RAM_BASE+0x0D8+0x04== 0xD043F28E
    Check PKA_RAM_BASE+0x0D8+0x08== 0x3FBE28EF
    Check PKA_RAM_BASE+0x0D8+0x0C== 0xCAF2BD55
    Check PKA_RAM_BASE+0x0D8+0x10== 0xF92EFF07
    Check PKA_RAM_BASE+0x0D8+0x14== 0xECA14640
```

ADSP-2184x PKA Register Descriptions

Public Key Accelerator (PKA) contains the following registers.

Table 40-23: ADSP-2184x PKA Register List

Name	Description
PKA_ALEN	PKA Vector_A Length
PKA_APTR	PKA Vector_A Address
PKA_BLEN	PKA Vector_B Length
PKA_BPTR	PKA Vector_B Address
PKA_COMPARE	PKA Compare Result
PKA_CPTR	PKA Vector_C Address
PKA_DIVMSW	PKA Most-Significant-Word of Divide Remainder
PKA_DPTR	PKA Vector_D Address
PKA_FUNC	PKA Function
PKA_RAM	Start of PKA RAM space
PKA_RESULTMSW	PKA Most-Significant-Word of Result Vector
PKA_SHIFT	PKA Bit Shift Value

PKA Vector_A Length

During execution of basic PKCP operations, the PKA_ALEN register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_ALEN register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

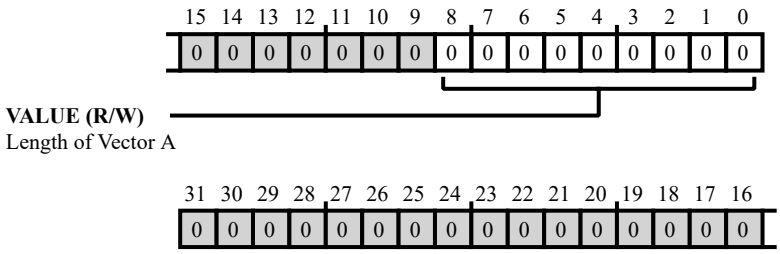


Figure 40-2: PKA_ALEN Register Diagram

Table 40-24: PKA_ALEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Length of Vector A. Length (in 32-bit words) of Vector A.

PKA Vector_A Address

During execution of basic PKCP operations, the PKA_APTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_APTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

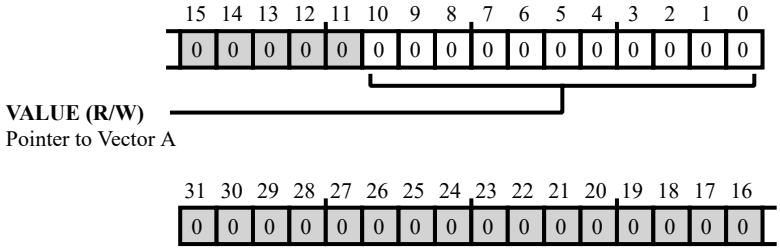


Figure 40-3: PKA_APTR Register Diagram

Table 40-25: PKA_APTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Pointer to Vector A. The PKA_APTR.VALUE bit field is the location of Vector A within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Vector_B Length

During execution of basic PKCP operations, the PKA_BLEN register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_BLEN register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

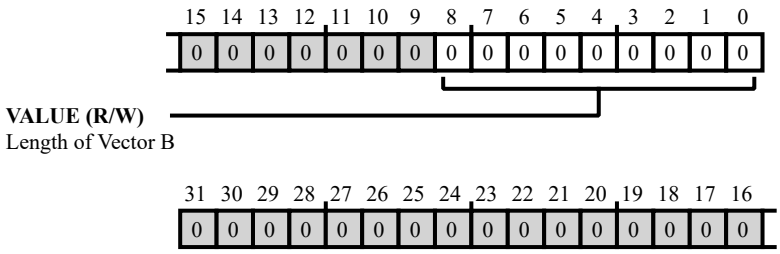


Figure 40-4: PKA_BLEN Register Diagram

Table 40-26: PKA_BLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Length of Vector B. Length (in 32-bit words) of Vector B.

PKA Vector_B Address

During execution of basic PKCP operations, the PKA_BPTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_BPTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

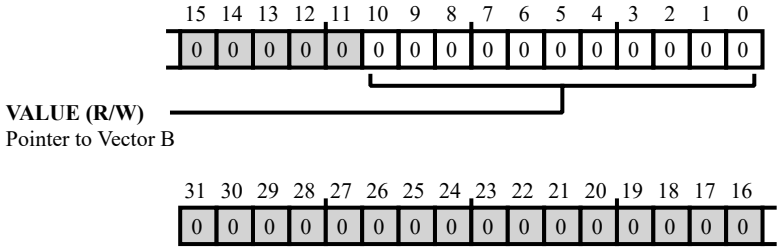


Figure 40-5: PKA_BPTR Register Diagram

Table 40-27: PKA_BPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Pointer to Vector B. The PKA_BPTR.VALUE bit field is the location of Vector B within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Compare Result

The PKA_COMPARE register provides the result of a basic PKCP Compare operation. It is updated when the PKA_FUNC.RUN bit is reset at the end of that operation. The status after a complex sequencer operation is unknown.

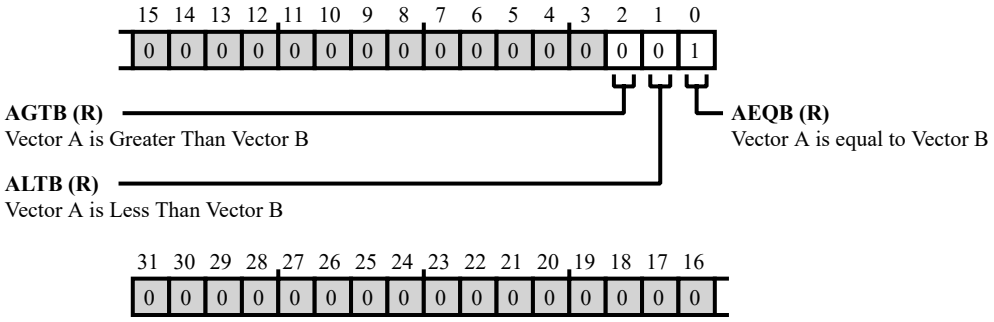


Figure 40-6: PKA_COMPARE Register Diagram

Table 40-28: PKA_COMPARE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/NW)	AGTB	Vector A is Greater Than Vector B. The PKA_COMPARE.AGTB bit shows the result of the basic compare operation is PKCP Vector_A is greater than Vector_B.
1 (R/NW)	ALTB	Vector A is Less Than Vector B. The PKA_COMPARE.ALTB bit shows the result of the basic compare operation is Vector_A is less than Vector_B.
0 (R/NW)	AEQB	Vector A is equal to Vector B. The PKA_COMPARE.AEQB bit shows the result of the basic compare operation is Vector_A is equal to Vector_B.

PKA Vector_C Address

During execution of basic PKCP operations, the PKA_CPTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_CPTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

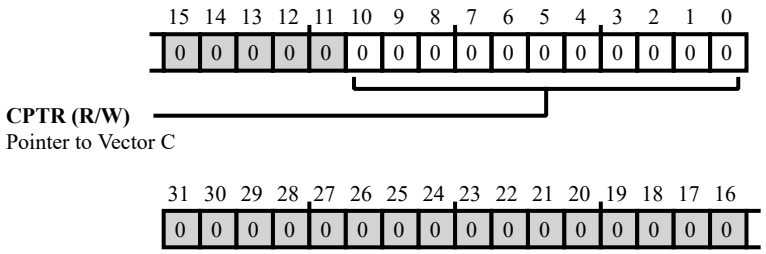


Figure 40-7: PKA_CPTR Register Diagram

Table 40-29: PKA_CPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	CPTR	Pointer to Vector C. The PKA_CPTR.CPTR bit field is the location of Vector C within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Most-Significant-Word of Divide Remainder

The PKA_DIVMSW register indicates the (32-bit word) address in the PKA RAM where the most significant non-zero 32-bit word of the Remainder result for the basic Divide and Modulo operations is stored. Bits [4:0] are loaded with the bit number of the most significant non-zero bit in the most significant non-zero word when MS one control bit is set. For Divide, Modulo and MS one reporting, this register is updated when the PKA_FUNC.RUN bit is reset at the end of the operation.

For the complex sequencer controlled operations, updating bits [4:0] of this register with the actual result's most significant bit location is done near the end of the operation. Note that the result is only meaningful if no errors were detected and that for ECC operations, the PKA_DIVMSW register provides information for the x-coordinate of the result point only.

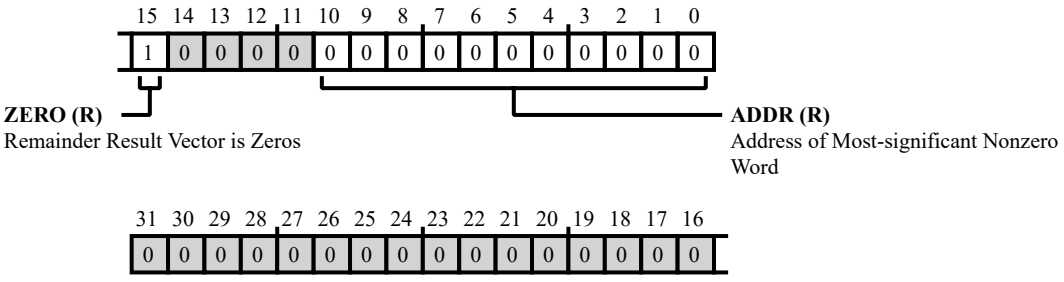


Figure 40-8: PKA_DIVMSW Register Diagram

Table 40-30: PKA_DIVMSW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/NW)	ZERO	Remainder Result Vector is Zeros. The PKA_DIVMSW.ZERO bit shows the remainder result vector is all zeros, ignore the address returned in bits [10:0].
10:0 (R/NW)	ADDR	Address of Most-significant Nonzero Word. The PKA_DIVMSW.ADDR bit shows the address of the most significant non-zero 32-bit word of the remainder result vector in PKA RAM.

PKA Vector_D Address

During execution of basic PKCP operations, the PKA_DPTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_DPTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

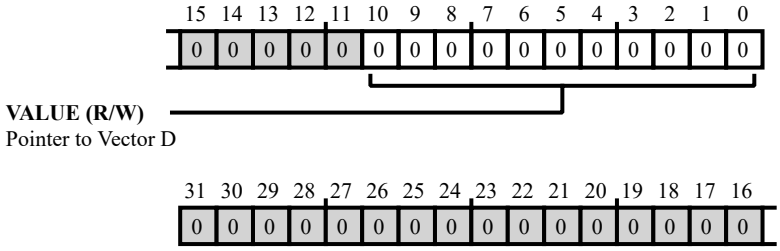


Figure 40-9: PKA_DPTR Register Diagram

Table 40-31: PKA_DPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Pointer to Vector D. The PKA_DPTR.VALUE bit field is the location of Vector D within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Function

The PKA_FUNC register contains the control bits to start basic PKCP as well as complex sequencer operations. The PKA_FUNC.RUN bit can be used to poll for the completion of the operation. Modifying bits [11:0] is made impossible during the execution of a basic PKCP operation.

During the execution of Sequencer controlled complex operations, this register is modified - the PKA_FUNC.RUN and PKA_FUNC.STALLRSLT bits are set to zero at the conclusion, but other bits are undefined.

Continuously reading this register to poll the PKA_FUNC.RUN bit is NOT allowed when executing complex sequencer operations (the sequencer cannot access the PKCP when this is done).

Leave at least one SCLK cycle between poll operations.

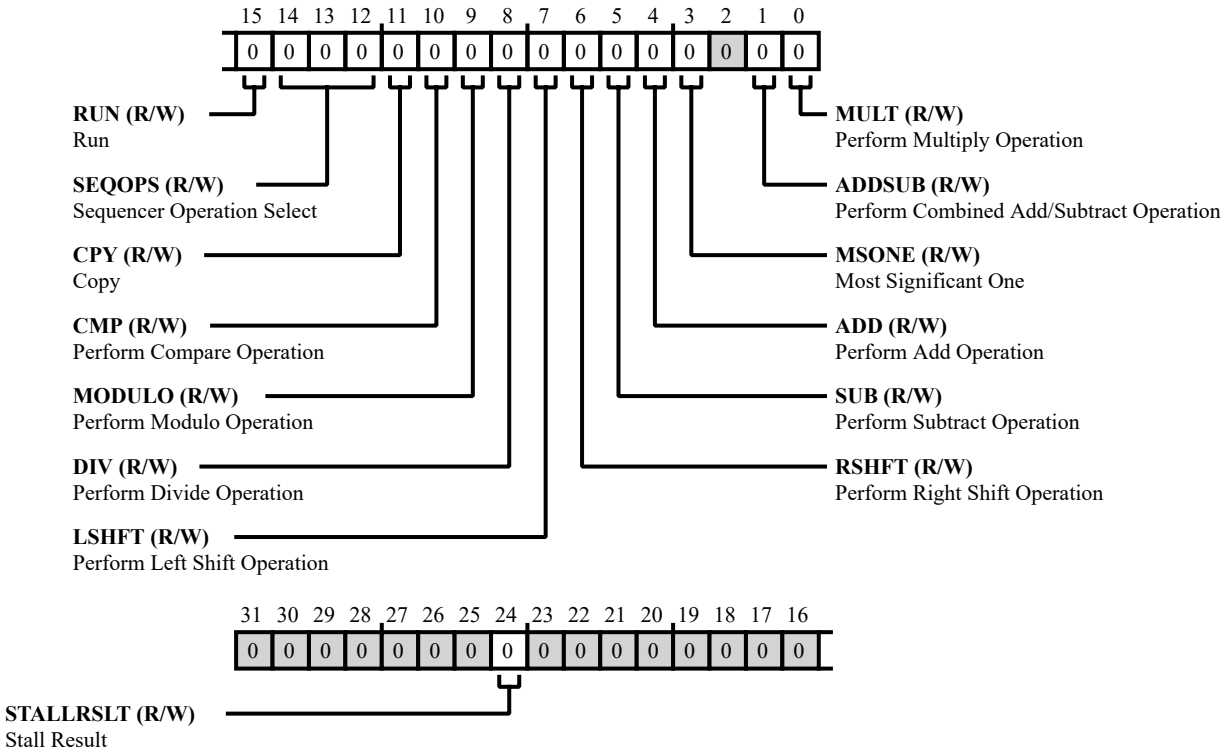


Figure 40-10: PKA_FUNC Register Diagram

Table 40-32: PKA_FUNC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	STALLRSLT	Stall Result. When the PKA_FUNC.STALLRSLT bit is set, updating the PKA_COMPARE, PKA_RESULTMSW and PKA_DIVMSW registers, as well as resetting the run bit, is stalled beyond the point that a running operation is actually finished. Use this to allow software enough time to read results from a previous operation when the newly started operation is known to take only a short amount of time. If a result is waiting, the result registers is updated and the run bit is reset in the clock cycle following writing the stall result bit back to 0. The stall result function may only be used for basic PKCP operations.	
15 (R/W)	RUN	Run. Set the PKA_FUNC.RUN bit to instruct the PKA module to begin processing the basic PKCP or complex sequencer operation. This bit is reset low automatically when the operation is complete. The complement of this bit is output as the pkaint1 interrupt. After a reset, the run bit is always set to 1b but the first sequencer firmware instruction sets this bit to 0 immediately after the hardware reset is released. A few clock cycles are needed before the first instruction is executed and the run bit state has been propagated.	
14:12 (R/W)	SEQOPS	Sequencer Operation Select. The PKA_FUNC.SEQOPS bit field select the complex sequencer operation to perform.	
		0	None
		1	ExpMod-CRT
		2	ExpMod-ACT4
		3	ECC-ADD
		4	ExpMod-ACT2
		5	ECC-MUL
		6	ExpMod-variable
		7	ModInv
11 (R/W)	CPY	Copy. Setting the PKA_FUNC.CPY bit performs the copy operation. For more information, see the "PKCP Vector Operations" section.	
10 (R/W)	CMP	Perform Compare Operation. Setting the PKA_FUNC.CMP bit performs the compare operation. For more information, see the "PKCP Vector Operations" section.	

Table 40-32: PKA_FUNC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	MODULO	Perform Modulo Operation. Setting the PKA_FUNC.MODULO bit performs the modulo operation. For more information, see the "PKCP Vector Operations" section.
8 (R/W)	DIV	Perform Divide Operation. Setting the PKA_FUNC.DIV bit performs the divide operation. For more information, see the "PKCP Vector Operations" section.
7 (R/W)	LSHFT	Perform Left Shift Operation. Setting the PKA_FUNC.LSHFT bit performs the left shift operation. For more information, see the "PKCP Vector Operations" section.
6 (R/W)	RSHFT	Perform Right Shift Operation. Setting the PKA_FUNC.RSHFT bit performs the right shift operation. For more information, see the "PKCP Vector Operations" section.
5 (R/W)	SUB	Perform Subtract Operation. Setting the PKA_FUNC.SUB bit performs the subtract operation. For more information, see the "PKCP Vector Operations" section.
4 (R/W)	ADD	Perform Add Operation. Setting the PKA_FUNC.ADD bit performs the add operation. For more information, see the "PKCP Vector Operations" section.
3 (R/W)	MSONE	Most Significant One. Setting the PKA_FUNC.MSONE bit loads the location of the most significant one bit within the result word indicated in the PKA_RESULTMSW register into bits [4:0] of the PKA_DIVMSW register; it can only be used with basic PKCP operations, except for divide, modulo and compare.
1 (R/W)	ADDSUB	Perform Combined Add/Subtract Operation. Setting the PKA_FUNC.ADDSUB bit performs the combined Add/Subtract operation. For more information, see the "PKCP Vector Operations" section.
0 (R/W)	MULT	Perform Multiply Operation. Setting the PKA_FUNC.MULT bit performs the multiply operation. For more information, see the "PKCP Vector Operations" section.

Start of PKA RAM space

The PKA_RAM register provides the starting location of the RAM space to hold the input, output and other vectors.

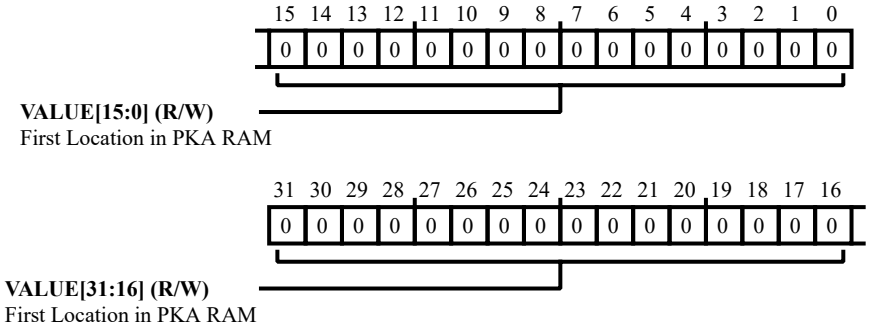


Figure 40-11: PKA_RAM Register Diagram

Table 40-33: PKA_RAM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	First Location in PKA RAM. The PKA_RAM.VALUE bit field provides the starting location of the RAM space to hold the input, output and other vectors.

PKA Most-Significant-Word of Result Vector

The PKA_RESULTMSW register indicates the (word) address in the PKA RAM where the most significant non-zero 32-bit word of the result is stored and should be ignored for modulo operations. For basic PKCP operations, the PKA_RESULTMSW register is updated when the PKA_FUNC.RUN bit is reset at the end of the operation.

For the complex sequencer controlled operations, updating the final value matching the actual result is done near the end of the operation. Note that the result is only meaningful if no errors are detected and that for ECC operations, the PKA_DIVMSW register provides information for the x-coordinate of the result point only.

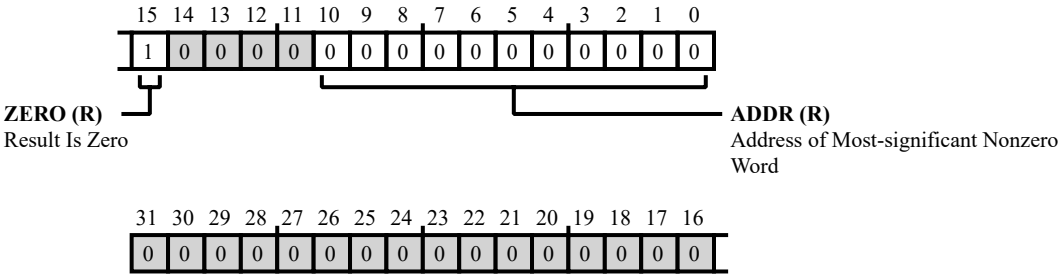


Figure 40-12: PKA_RESULTMSW Register Diagram

Table 40-34: PKA_RESULTMSW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/NW)	ZERO	Result Is Zero. The PKA_RESULTMSW.ZERO bit indicates the result vector is all zeros, ignore the address returned in bits [10:0].
10:0 (R/NW)	ADDR	Address of Most-significant Nonzero Word. The PKA_RESULTMSW.ADDR bit is the address of the most significant non-zero 32-bit word of the result vector in PKA RAM.

PKA Bit Shift Value

For basic PKCP operations, modifying the contents of the PKA_SHIFT register is made impossible while the operation is being performed. For the ExpMod-variable and ExpMod-CRT operations, the PKA_SHIFT register is used to indicate the number of odd powers to use (directly as a value in the range 1-16). For the ModInv and ECC operations, this register is used to hold a completion code.

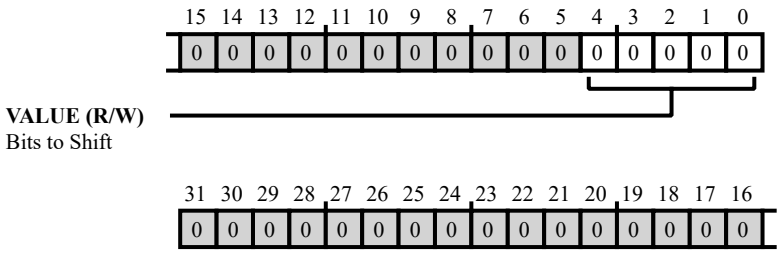


Figure 40-13: PKA_SHIFT Register Diagram

Table 40-35: PKA_SHIFT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	VALUE	Bits to Shift. The PKA_SHIFT.VALUE bit field is the number of bits to shift the input vector (in the range 0-31) during a Rshift or Lshift operation.

41 Public Key Interrupt Controller (PKIC)

The Public Key Accelerator (PKA) and the True Random Number Generator (TRNG) share a common interrupt controller, the Public Key Interrupt Controller (PKIC). The host processor configures the PKIC to generate interrupts when certain operations are complete, or interrupts are caused by errors.

PKIC Functional Description

The main purpose and function of the PKIC is to capture the interrupts from different sources, either from the PKA or the TRNG and combine them to one interrupt output. The interrupt controller is managed using the following register groups:

- Control for polarity, edge, and level detection and enabling of individual interrupts
- Acknowledgment (to clear edge detected interrupts)
- Status:
  - A raw source status register after edge detection, if edge selected.
  - A status register after masking with the interrupt enable control bits.

ADSP-2184x PKIC Register List

The Public Key Processor Interrupt Controller (PKIC) provides security-related features. A set of registers governs PKIC operations. For more information on PKIC functionality, see the PKIC register descriptions.

Table 41-1: ADSP-2184x PKIC Register List

Name	Description
PKIC_ACK	Acknowledge Register
PKIC_EN_CLR	Enable Clear Register
PKIC_EN_CTL	Enable Control Register
PKIC_EN_SET	Enable Set Register
PKIC_EN_STAT	Enabled Status Register
PKIC_POL_CTL	Polarity Control Register

Table 41-1: ADSP-2184x PKIC Register List (Continued)

Name	Description
PKIC_RAW_STAT	Raw Status Register
PKIC_TYPE_CTL	Type Control Register

ADSP-2184x PKIC Interrupt List

Table 41-2: ADSP-2184x PKIC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
225	PKIC0_IRQ	PKIC0 Interrupt	Level	

PKIC Programming Model

The following sections provide information on how to program the PKIC.

Enabling/Disabling and Status

The PKIC_EN_STAT register provides the mask to which interrupt source is enabled. There are two status registers, PKIC_RAW_STAT and PKIC_EN_STAT. They allow the host processor to read the status of the interrupt source before and after the mask is applied.

Level or Edge

All of the interrupt sources are level or edge events. The PKIC_TYPE_CTL register configures each interrupt to either level or edge. The PKIC_POL_CTL register controls the polarity of the signal.

These interrupts are latched at both status registers in case edge detection is selected. The edge detectors are reset by clearing the interrupts using the PKIC_ACK registers.

PKIC Programming Concepts

The following concepts help with proper programming for the PKIC module.

Interrupt Handling

When an interrupt is triggered, the handler must first examine this module, the PKIC to determine what triggered the interrupt. By reading the PKIC_EN_STAT, the bits that are set are the pending interrupts of the ones that were not masked. After determining the source of the interrupt, the appropriate action must be taken to service the interrupt from the corresponding module, the PKA, or the TRNG. After handling the interrupt in a particular module, the corresponding interrupt must be acknowledged and cleared in the PKIC to allow further interrupts.

While handling an interrupt, any events that would cause another interrupt would happen without triggering another interrupt.

Overlapping Registers

There are two sets of overlapping registers in the PKIC. The PKIC_EN_STAT and PKIC_ACK registers share one address. If read, the register tells which enabled interrupts are pending. If written to (W1C), the interrupt is acknowledged and cleared. The PKIC_RAW_STAT and PKIC_EN_SET registers are another pair that share the address. When read, the register tells which interrupts are pending and if written to will enabled certain interrupts. This register cannot be used to disable any interrupts.

ADSP-2184x PKIC Register Descriptions

Public Key Processor Interrupt Controller (PKIC) contains the following registers.

Table 41-3: ADSP-2184x PKIC Register List

Name	Description
PKIC_ACK	Acknowledge Register
PKIC_EN_CLR	Enable Clear Register
PKIC_EN_CTL	Enable Control Register
PKIC_EN_SET	Enable Set Register
PKIC_EN_STAT	Enabled Status Register
PKIC_POL_CTL	Polarity Control Register
PKIC_RAW_STAT	Raw Status Register
PKIC_TYPE_CTL	Type Control Register

Acknowledge Register

The PKIC_ACK register is used to acknowledge the interrupt and clear the corresponding interrupt bit in the status register.

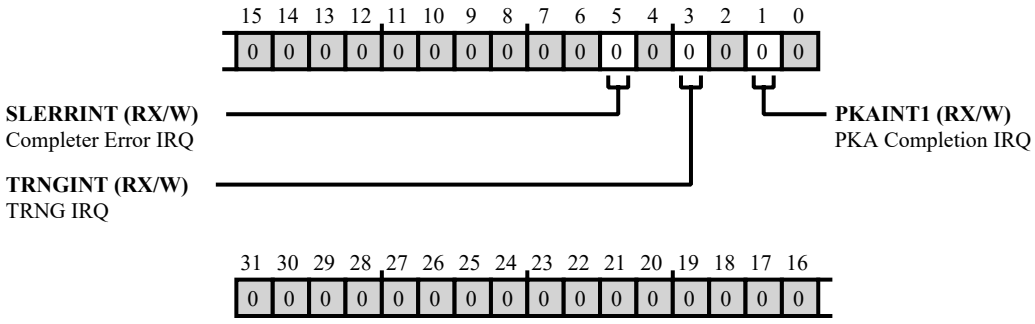


Figure 41-1: PKIC_ACK Register Diagram

Table 41-4: PKIC_ACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	SLERRINT	Completer Error IRQ. The PKIC_ACK.SLERRINT bit is the acknowledge bit for the completer error interrupt. When set =1 the PKIC_ACK.SLERRINT bit acknowledges the interrupt signal and clears the status bit (and is cleared automatically).	
		0	Do not acknowledge interrupt and clear status bit
		1	Acknowledge interrupt and clear status bit
3 (RX/W)	TRNGINT	TRNG IRQ. The PKIC_ACK.TRNGINT bit is the acknowledge bit for the TRNG interrupt. When set =1 the PKIC_ACK.TRNGINT bit acknowledges the interrupt signal and clears the status bit (and is cleared automatically).	
		0	Do not acknowledge interrupt and clear status bit
		1	Acknowledge interrupt and clear status bit
1 (RX/W)	PKAINT1	PKA Completion IRQ. The PKIC_ACK.PKAINT1 bit is the acknowledge bit for the PKA completion interrupt. When set =1 the PKIC_ACK.PKAINT1 bit acknowledges the interrupt signal and clears the status bit (and is cleared automatically).	

Enable Clear Register

The PKIC_EN_CLR register allows the user to disable certain interrupts without enabling others. The disabled interrupts are also reflected in PKIC_EN_CTL register.

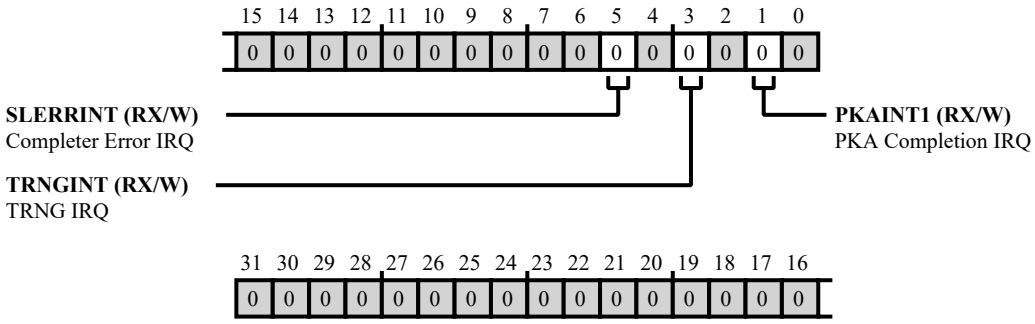


Figure 41-2: PKIC_EN_CLR Register Diagram

Table 41-5: PKIC_EN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	SLERRINT	Completer Error IRQ. The PKIC_EN_CLR.SLERRINT bit is the individual disable for the completer error interrupt. When set =1 this bit clears/resets the corresponding bit in the PKIC_EN_CTL register to 0 (disables the interrupt). This bit is cleared automatically.	
		0	No action
		1	Clear/reset corresponding CTL bit
3 (RX/W)	TRNGINT	TRNG IRQ. The PKIC_EN_CLR.TRNGINT bit is the individual disable for the TRNG interrupt. When set =1 this bit clears/resets the corresponding bit in the PKIC_EN_CTL register to 0 (disables the interrupt). This bit is cleared automatically. 0= no effect.	
		0	No action
		1	Clear/reset corresponding CTL bit
1 (RX/W)	PKAINT1	PKA Completion IRQ. The PKIC_EN_CLR.PKAINT1 bit is the individual disable for the PKA Completion interrupt. When set =1 this bit clears/resets the corresponding bit in the PKIC_EN_CTL register to 0 (disables the interrupt). This bit is cleared automatically. 0= no effect.	
		0	No action
		1	Clear/reset corresponding CTL bit

Enable Control Register

The PKIC_EN_CTL register provides individual enable bits for the interrupt sources.

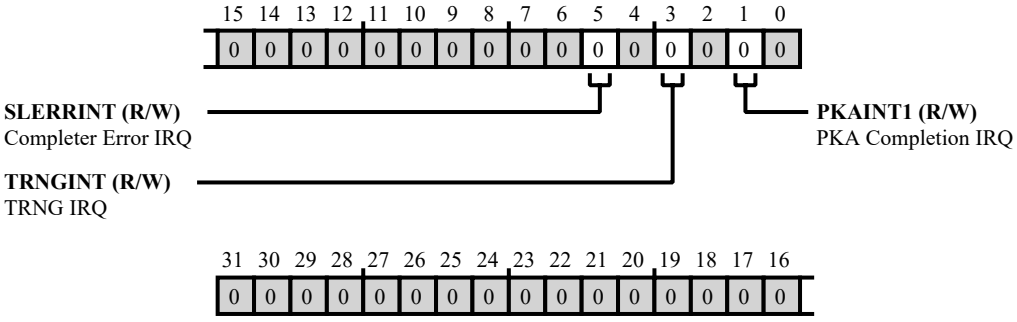


Figure 41-3: PKIC_EN_CTL Register Diagram

Table 41-6: PKIC_EN_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SLERRINT	Completer Error IRQ. The PKIC_EN_CTL.SLERRINT bit enables control for the completer error interrupt.	
		0	Disable interrupt
		1	Enable interrupt
3 (R/W)	TRNGINT	TRNG IRQ. The PKIC_EN_CTL.TRNGINT bit enables control for the TRNG interrupt.	
		0	Disable interrupt
		1	Enable interrupt
1 (R/W)	PKAINT1	PKA Completion IRQ. The PKIC_EN_CTL.PKAINT1 bit enables control for the PKA completion interrupt.	
		0	Disable interrupt
		1	Enable interrupt

Enable Set Register

The PKIC_EN_SET register allows the user to only set certain interrupt sources without disabling any others. The enabled interrupts are reflected in PKIC_EN_CTL register.

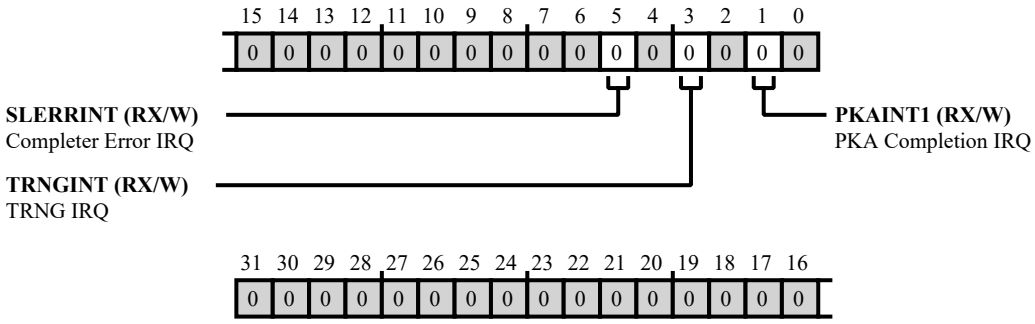


Figure 41-4: PKIC_EN_SET Register Diagram

Table 41-7: PKIC_EN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	SLERRINT	Completer Error IRQ. The PKIC_EN_SET.SLERRINT bit is the individual enable for the completer error interrupt. If =1, sets the corresponding bit in the PKIC_EN_CTL register to 1 (enables interrupt). This bit is cleared automatically. 0=no effect	
		0	No effect
		1	Enable interrupt
3 (RX/W)	TRNGINT	TRNG IRQ. The PKIC_EN_SET.TRNGINT bit is the individual enable for the TRNG interrupt. If =1, sets the corresponding bit in the PKIC_EN_CTL register to 1 (enables interrupt). This bit is cleared automatically.	
		0	No effect
		1	Enable interrupt
1 (RX/W)	PKAINT1	PKA Completion IRQ. The PKIC_EN_SET.PKAINT1 bit is the individual enable for the PKA Completion interrupt. If =1, sets the corresponding bit in the PKIC_EN_CTL register to 1 (enables interrupt). This bit is cleared automatically.	
		0	No effect
		1	Enable interrupt

Enabled Status Register

The PKIC_EN_STAT register is used to tell the status of the interrupts after the gating with the PKIC_EN_CTL register.

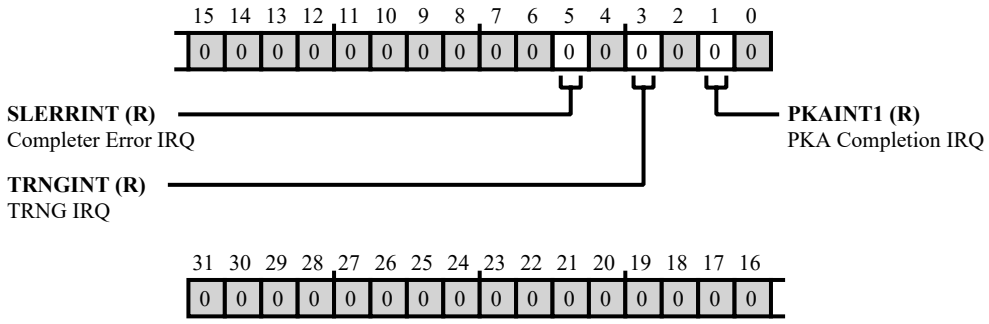


Figure 41-5: PKIC_EN_STAT Register Diagram

Table 41-8: PKIC_EN_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	SLERRINT	Completer Error IRQ. The PKIC_EN_STAT.SLERRINT bit provides the status of the completer error interrupt (after masking from the PKIC_EN_CTL register).	
		0	Interrupt is inactive
		1	Interrupt is pending
3 (R/NW)	TRNGINT	TRNG IRQ. The PKIC_EN_STAT.TRNGINT bit provides the status of the TRNG interrupt (after masking from the PKIC_EN_CTL register).	
		0	Interrupt is inactive
		1	Interrupt is pending
1 (R/NW)	PKAINT1	PKA Completion IRQ. The PKIC_EN_STAT.PKAINT1 bit provides the status of the PKA Completion interrupt (after masking from the PKIC_EN_CTL register).	
		0	Interrupt is inactive
		1	Interrupt is pending

Polarity Control Register

The PKIC_POL_CTL register is used to configure the signal polarity for each individual interrupt. During the initialization phase of the PKA the host processor must set each interrupt in this register to (high level/rising edge or low level/falling edge).

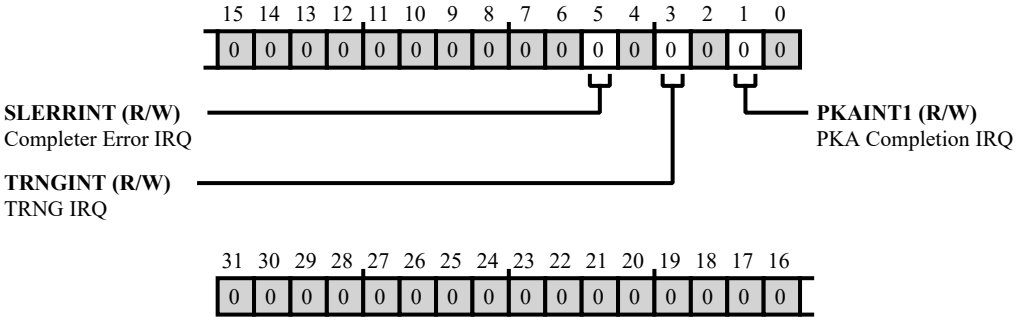


Figure 41-6: PKIC_POL_CTL Register Diagram

Table 41-9: PKIC_POL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SLERRINT	Completer Error IRQ. The PKIC_POL_CTL.SLERRINT bit provides polarity control for the completer error interrupt.	
		0	Low level/falling edge
		1	High level/rising edge
3 (R/W)	TRNGINT	TRNG IRQ. The PKIC_POL_CTL.TRNGINT bit provides polarity control for the TRNG interrupt.	
		0	Low level/falling edge
		1	High level/rising edge
1 (R/W)	PKAINT1	PKA Completion IRQ. The PKIC_POL_CTL.PKAINT1 bit provides polarity control for PKA completion interrupt.	
		0	Low level/falling edge
		1	High level/rising edge

Raw Status Register

The PKIC_RAW_STAT register reflects the status of the individual interrupts before masking with the PKIC_EN_CTL register.

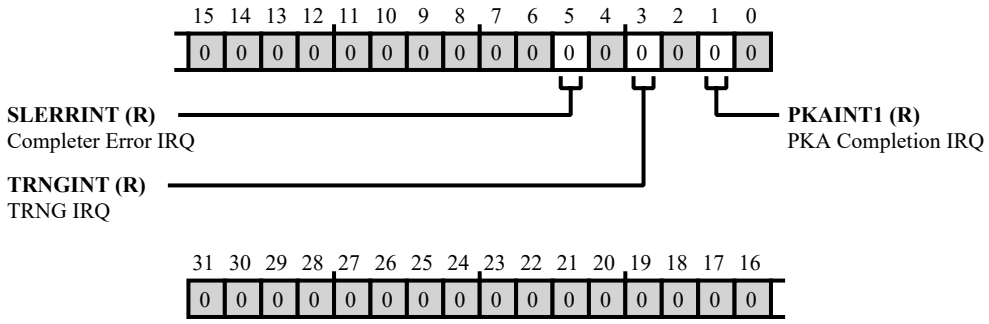


Figure 41-7: PKIC_RAW_STAT Register Diagram

Table 41-10: PKIC_RAW_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	SLERRINT	Completer Error IRQ. The PKIC_RAW_STAT.SLERRINT bit provides the raw status of the completer error interrupt.	
		0	Inactive interrupt
		1	Pending interrupt
3 (R/NW)	TRNGINT	TRNG IRQ. The PKIC_RAW_STAT.TRNGINT bit provides the raw status of the TRNG interrupt where 1=pending and 0=inactive.	
		0	Inactive interrupt
		1	Pending interrupt
1 (R/NW)	PKAINT1	PKA Completion IRQ. The PKIC_RAW_STAT.PKAINT1 bit provides raw status of the PKA completion interrupt where 1=pending and 0=inactive.	
		0	Inactive interrupt
		1	Pending interrupt

Type Control Register

The PKIC_TYPE_CTL register is used to configure the signal type for each individual interrupt. During the initialization phase of the PKA the host processor must set each interrupt in this register to level or edge.

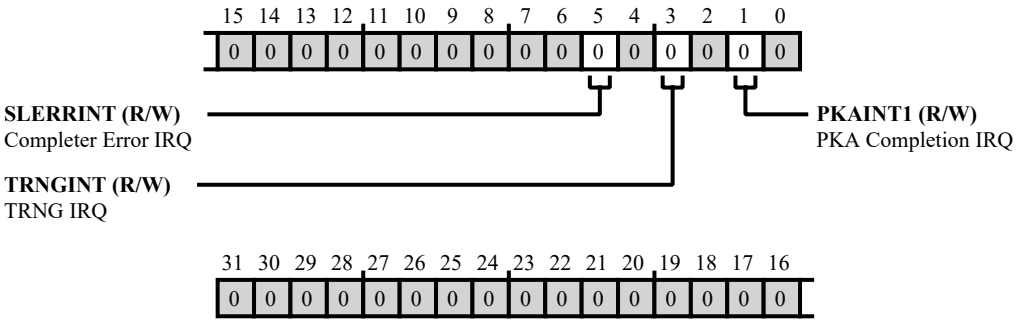


Figure 41-8: PKIC_TYPE_CTL Register Diagram

Table 41-11: PKIC_TYPE_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SLERRINT	Completer Error IRQ. The PKIC_TYPE_CTL.SLERRINT bit provides signal type control for the completer error interrupt.	
		0	Level
		1	Edge
3 (R/W)	TRNGINT	TRNG IRQ. The PKIC_TYPE_CTL.TRNGINT bit provides signal type control for the TRNG interrupt.	
		0	Level
		1	Edge
1 (R/W)	PKAINT1	PKA Completion IRQ. The PKIC_TYPE_CTL.PKAINT1 bit provides signal type control for the PKA completion interrupt.	
		0	Level
		1	Edge

42 True Random Number Generator (TRNG)

The TRNG engine provides a true, non-deterministic, noise source for generating keys, Initialization Vectors (IVs), and other random number requirements. Other non-cryptographic purposes include statistical sampling, retry timers for communications protocols and noise generation.

TRNG Features

The TRNG features include:

- Hardware-based non-deterministic random number generator
- ANSI X9.31 postprocessing (depending on processor)
- Redundant 'fail-safe' design with self-test circuits
- Reliable shot noise oscillator implementation with auto-tuning
- Debug output to allow monitoring of internal operation
- Alarm count overflow and auto-tuning error interrupts
- Buffer to allow generation of large blocks of random data in the background

TRNG Functional Description

The following sections provide details on the function of the TRNG module.

ADSP-2184x TRNG Register List

The True Random Number Generator (TRNG) provides random number generation, intended mainly for security-related applications. A set of registers governs TRNG operations. For more information on TRNG functionality, see the TRNG register descriptions.

Table 42-1: ADSP-2184x TRNG Register List

Name	Description
TRNG_ALMCNT	TRNG Alarm Counter Register

Table 42-1: ADSP-2184x TRNG Register List (Continued)

Name	Description
TRNG_ALMMSK	TRNG Alarm Mask Register
TRNG_ALMSTP	TRNG Alarm Stop Register
TRNG_BLKCNT	TRNG Block Count Register
TRNG_CFG	TRNG Configuration Register
TRNG_CNT	Counter Register
TRNG_CTL	TRNG Control Register
TRNG_FRODETUNE	TRNG FRO De-tune Register
TRNG_FROEN	TRNG FRO Enable Register
TRNG_INPUT[n]	TRNG Input Registers
TRNG_INTACK	TRNG Interrupt Acknowledge Register
TRNG_KEY[n]	Post-Process Key Registers
TRNG_LFSR_H	TRNG LFSR Access Register
TRNG_LFSR_L	TRNG LFSR Access Register
TRNG_LFSR_M	TRNG LFSR Access Register
TRNG_MONOBITCNT	TRNG Monobit Test Result Register
TRNG_OUTPUT[n]	TRNG Output Registers
TRNG_POKER[n]	TRNG Poker Test Result Registers
TRNG_RUNCNT	TRNG Run Count Registers
TRNG_RUN[n]	TRNG Run Test State and Result Registers
TRNG_STAT	TRNG Status Register
TRNG_TEST	TRNG Test Register
TRNG_V[n]	TRNG Post-Process "V" Value Registers

Random Number Generation

The random numbers that the TRNG generates are produced by sampling Free Running Oscillators (FRO). The (TRNG_CTL.STARTUPCYC) bit field along with TRNG_CFG.MINREFCYC (minimum refill cycles) and TRNG_CFG.MAXREFCYC (maximum refill cycles) bit fields determine the number of samples taken to generate the first random value and subsequent random values.

1. After enabling the TRNG (TRNG_CTL.TRNGEN bit =1), a number of FRO output samples defined by the TRNG_CTL.STARTUPCYC bit field are gathered in the main linear-feedback shift register (LFSR) before taking a snapshot of the LFSR and storing that snapshot in the random data buffer (after optional post-processing).

2. After taking a snapshot of the LFSR, a number of FRO output samples defined by the TRNG_CFG.MINREFCYC bit field, are gathered in the main LFSR. If the random data buffer is full, sample-taking continues until the number of samples (counting from the snapshot) matches the number of samples defined by the TRNG_CFG.MAXREFCYC bit field. At that point, the TRNG switches off the FROs and powers down.
3. If, after the TRNG_CFG.MINREFCYC sampling period, the random data buffer is not completely filled, a new snapshot of the LFSR is taken and stored in the random data buffer (after optional postprocessing). Control branches back to point step 2 above and a new TRNG_CFG.MINREFCYC sample period starts.

Locking Detection and Prevention

Lock detection in functional mode uses the sampled outputs of the individual FROs. A FRO alarm event is declared when a repeating pattern (of up to four sample lengths) is detected continuously for the number of samples defined in the alarm threshold TRNG_ALMCNT.ALMTHRESH bit field. The alarm event is logged by setting the bit that corresponds to the FRO that caused the alarm in the TRNG_ALMMSK register. If that bit was already set, the corresponding bit in the TRNG_ALMSTP register is set. The FRO is switched off to prevent further alarm events from that FRO. If the TRNG_ALMMSK register bit was not yet set, the FRO is restarted automatically in an attempt to break locking.

The shutdown count field in the alarm count TRNG_ALMCNT.SHDNCNT register monitors the number of FROs switched off. (It counts the number of 1 bits in the TRNG_ALMSTP register.) The shutdown threshold field (TRNG_ALMCNT.SHDNTHRESH) can be configured to generate the shutdown overflow interrupt (TRNG_STAT.SHDNOVR). When the shutdown count in the TRNG_ALMCNT.SHDNCNT bit field exceeds the shutdown threshold in the TRNG_ALMCNT.SHDNTHRESH bit field, the shutdown overflow bit (TRNG_STAT.SHDNOVR) is set to 1 (which can be used to generate an interrupt).

Software can use two strategies for the TRNG operation:

- ***Monitored Operation***. Software checks the TRNG_ALMMSK register at regular intervals (on the order of seconds). If a bit is set in that register, then the program must also check the TRNG_ALMSTP register to determine if a FRO was shut down due to multiple alarm events. If no FROs are shut down, the program clears the TRNG_ALMMSK register to remove the incidental alarm events. If one or more FROs are shut down, the host processor can modify the delay selection of those FROs using the TRNG_FRODETUNE register to prevent further locking. For this type of operation, the shutdown threshold is normally set to a low value (two, for example). The shutdown overflow interrupt can then be used to signal abnormal operation conditions or the breakdown of FROs.
- ***Unmonitored Operation.*** Software sets the shutdown threshold to the acceptable number of FROs to be shut down before taking corrective actions. It then uses the shutdown overflow interrupt to initiate corrective actions (clearing the TRNG_ALMMSK and TRNG_ALMSTP registers, toggling bits in the TRNG_FRODETUNE register). The software must monitor the time interval between these interrupts. If they occur too often (for example, within a minute after each other), this frequency indicates abnormal operating conditions or the breakdown of FROs.

Run Testing

Run Test

The TRNG block counts the number of consecutive zeros and ones (runs) in the data stream shifted into the main LFSR. The run length and bit value is then used to increment a specific bucket counter for these values. After 20,000 bits, the bucket counters must be within specified limits for this test to pass. If not, a run fail interrupt (RUNFAIL) is generated.

Table 42-2: Allowable Limits on Runs of 0's and 1's

Run Count	Bit Value	Min (inclusive)	Max (inclusive)
1	0	2267	2733
	1	2267	2733
2	0	1079	1421
	1	1079	1421
3	0	502	748
	1	502	748
4	0	233	402
	1	233	402
5	0	90	223
	1	90	223
6 and up	0	90	233
	1	90	233

Long Run Test

The long run test fails immediately when a run longer than 33 bits is found and a long run fail (`TRNG_STAT.LRUNFAIL`) interrupt is generated.

Noise Source Test

A noise source failure is declared when a run of 48 or more identical bits is found and a noise fail interrupt (`TRNG_STAT.NOISEFAIL`) is generated.

The status and counts are stored in the `TRNG_RUNCNT` and `TRNG_RUN[n]` registers. Unless otherwise indicated, all counters and state bits in these registers are reset when writing a 1 to either the monobit fail acknowledge (`TRNG_INTACK.MBITFAIL`), the run fail acknowledge (`TRNG_INTACK.RUNFAIL`), or the poker fail acknowledge (`TRNG_INTACK.PKRFAIL`) bits.

Monobit Testing

The TRNG block performs the monobit test on blocks of 20,000 bits (in parallel with the run test and poker test). It monitors the number of zeros and ones in the data stream shifted into the main LFSR. At the start of the block,

the counter is initialized to 10,000. Each 1 value increments the counter, each 0 value decrements the counter. After 20,000 bits, the counter value must be within 9310–10690 (inclusive) for this test to pass. If not, a monobit fail interrupt (TRNG_STAT.MBITFAIL) is generated. The AIS-31 standard (test T1, ref 0) specifies this run time testing of the TRNG and the parameters.

NOTE: The actual limits stated here are different than the limits stated in the AIS-31 standard due to the implementation. The circuitry in the TRNG uses an up-down counter while the standard just evaluates the sum of the 20,000 bits.

When the continue poker (TRNG_TEST.CONTPKR) bit is set to 1, the test is not stopped after 20,000 bits. The counter keeps incrementing and decrementing (protected against overflow and underflow). But, no actual limit checking happens. The offset from starting value 10,000 indicates the balance of 0 and 1 bits that were checked since the start of the continuous test. The offset is twice the number of missing or extra 1 bits. (An extra 1 bit adds an increment operation and removes one decrement operation. So, having 10,001 1 bits in the block gives a counter value of 10,002.)

Poker Testing

The poker test is run in parallel with the monobit test and run test. Counters in the TRNG_POKER[n] registers are used to count the occurrences of one specific 4-bit value in the data stream fed into the main LFSR. All of the counters are decremented by one every 64 data bits and reset to their start value every 20,000 bits. All counters start at a value of –1 and are decremented 312 times during the 20,000-bit test run.

Each 8-bit counter holds a two's complement value and does not overflow past the range of –128 to +127. At the end of the 20,000-bits block, the values of the counters that contain a single 1 bit appended at the least significant bit side are individually squared and then added. The poker test fails with a poker fail (PKRFAIL) interrupt when:

- the resulting sum (accumulated in the TRNG_MONOBITCNT register) is outside the range 1288 – 71750 (inclusive), or
- one of the counters tries to increment or decrement outside its limit range

NOTE: The poker test fails when the 4-bit values of the data stream are distributed too evenly (with eight counters having incremented 312 times and the others incremented 313 times). This failure is intentional. The minimum mean deviation from the expected value of 312.5 is 4.5. Failure at counter overflow is not an official part of the poker test as specified in the AIS-31 standard (ref 0). It can be shown that the maximum deviation for one counter's value from the mean value of 312.5 (without the poker test failing) is 129.5. Since this deviation is more than 40% of the mean value, it indicates that something is wrong. Here, the counter overflow failure is combined with the official poker test failure.

When the continue poker (TRNG_TEST.CONTPKR) bit is set to 1, the test does not stop after 20,000 bits. The counters keep incrementing and decrementing (the latter every 64 bits). A PKRFAIL interrupt is generated when one of the counters tries to increment or decrement outside its limit range.

Data for Tests

For the monobit test, run test and poker test circuits self-test, the test data written to the TRNG_INPUT[n]0 register is used for the monobit test and run test bit-by-bit. It executes from bit 31 down to bit 0. For the poker test, the written test data is used in eight blocks of 4 bits, starting from bits 31:28 down to bits 3:0.

When the TRNG_TEST.CONTPKR bit =0 during run test and poker test circuits self-tests, the state of the poker and runs test status registers is frozen after all calculations are made and after inputting 20,000 test bits. This state allows time to read the test results. These calculations take around 20 clock cycles to complete. The status registers are reset to their starting states when the first word for the next test block of test bits is written to the TRNG_INPUT[n]0 register. Then, the contents of this word are processed.

X9.31 Postprocessing

Postprocessing is available on some parts using the TRNG. If available, the postprocessing block is situated after the main LFSRs and before the output buffer that stores the random numbers for consumption. The online test logic for monobit, run, and poker testing is before the postprocessor block. The bits used for testing are from the main LFSR.

The postprocessor is based on the ANSI X9.31 specification using 3-Key 3DES cipher algorithm. It does not provide any more entropy than is what is already achieved from sampling the Free Running Oscillators (FROs). The use of the postprocessor only helps with applications requiring the use of an X9.31 compliant RNG.

The following section is an example of postprocessing.

The variables include:

- DT is a 64-bit date/time vector. This value is the input coming from random bits from the main LFSR.
- I is an intermediate value, a 64-bit value
- R is the final result, a 64-bit value
- V is a 64-bit seed value that is to be kept secret
- K is the 3-key for 3DES, each being 64-bits

The postprocessing uses the steps:

1. The intermediate value I is calculated: I = $3DES_{EDE}$(K, DT) with 3DES.
2. The result R is calculated: R = $3DES_{EDE}$(K, I XOR V).
3. Finally, V is updated: V = $3DES_{EDE}$(K, R XOR I).

TRNG Block Diagram

The ***System Block Diagram of the TRNG*** diagram shows the system block for the TRNG. The system includes:

- Free Running Oscillators (FROs) that are the source of sampled bits
- A post-processing unit (processor dependent) for standards compliance

- Test circuitry to detect non-randomness due to failures in the system

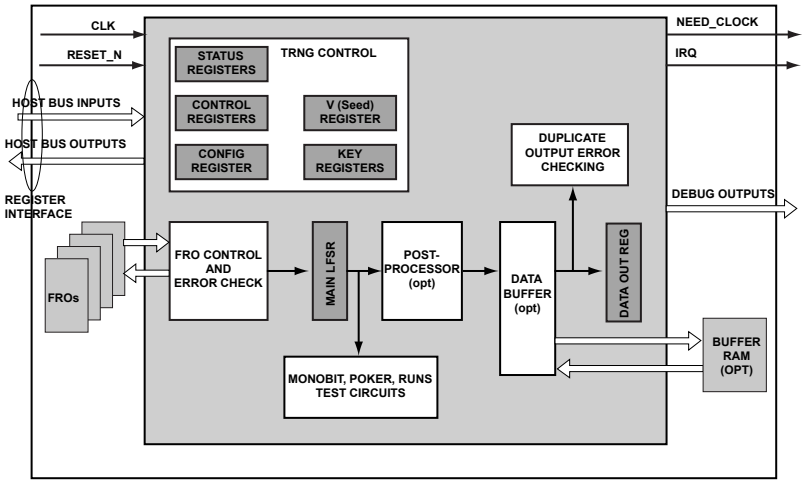


Figure 42-1: System Block Diagram of the TRNG

TRNG Architectural Concepts

The random numbers are accessible to the host processor in four 32-bit registers allowing a single burst read of a 128-bit random number. Acknowledging the data ready (interrupt) state causes the TRNG to move a new value (when available in the data buffer) to the TRNG output register. The TRNG always tries to keep the data buffer completely full. Pulling out data starts the regeneration of a new number by:

- Enabling the FROs
- Capturing their outputs in the LFSR, and
- Cryptographically postprocessing the values from the LFSR

The process produces new random values to replenish the buffer.

The major functional blocks of the TRNG module are:

- The actual TRNG core with control and test circuits, optional post-processor, and optional data buffer control logic
- Free Running Oscillators (FROs) instantiated outside the TRNG core
- A 128-byte buffer RAM instantiated outside the TRNG core

In the TRNG core, the true entropy source uses FROs as the basic building block. The accumulation of timing jitter, caused (for the largest part) by shot noise, creates uncertainty intervals for the output transitions of each FRO. Sampling within an uncertainty interval generates a single bit of entropy, which is ‘accumulated’ in a LFSR. As the uncertainty interval is very narrow compared to the cycle time of a FRO, the mean amount of entropy generated per sample is small (less than 1/100 bit per sample). To increase the entropy generation rate, multiple FROs are used in parallel.

The FROs are asynchronous to one another and asynchronous to the sampling clock to make their behavior truly non-deterministic.

The output signals of the FROs are sampled at regular intervals (in general, at the TRNG core module clock frequency). The samples feed into an error detection circuit that checks for repeating patterns coming out of a FRO. If a repeating pattern persists for a configurable number of samples, the FRO is suspect of having synchronized to (a harmonic of) the sampling interval. This activity drastically reduces the amount of entropy generated by that FRO. The error detection circuit signals this activity as a FRO error event.

Error events can occur during normal operation. The FRO control circuit attempts to restart a FRO that on a first error event. A second error event causes an automatic shutdown of the FRO. Because there are multiple FROs, shutting down a FRO reduces the amount of entropy generated, but it does not immediately jeopardize the TRNG operation. A limit can be configured below which the number of operational FROs does not drop. If this limit is crossed, an interrupt can be generated on the host processor. Software on the host processor can then attempt to prevent frequent locking of a FRO by *de-tuning* it to a slightly different frequency.

An XOR tree combines the sampled outputs and feeds them into an 81-bit LFSR to accumulate entropy and whiten the random bits stream.

NOTE: Here, *whitening* means balancing the number of one and zero bits in the stream.

TRNG Operating Modes

The TRNG has the following operating modes.

Normal Reading Mode

In normal reading mode, random data can only be read out of the output registers of the TRNG module when the ready bit (`TRNG_STAT.RDY`) =1. Acknowledging the data by writing a 1 to the ready acknowledge bit in the `TRNG_INTACK.RDY` register clears the ready bit from the status register and clears the output registers. The registers remain at zero until the next 128-bit data block is available.

Secure Reading Mode

An attacker can try to read the output registers (without acknowledging the data) to obtain a copy of data to be read later by an application. To block this attack, secure reading mode is used. In this mode, enable reading from the output registers (by writing 0x00 to the open read gate bits (`TRNG_INTACK.OPENRDGATE`) before it is possible to access the output registers. Enabling the reading starts a timeout (controlled by the `TRNG_CFG.RDTIMEOUT` bit field). When this timeout expires, reading is disabled, and the offered data is acknowledged so that it is not offered again. The host processor must set this timeout such that there is enough time to read the output registers and perform a normal data acknowledge (which stops the timeout).

Test Mode

In addition to the test circuitry that operates during normal operation, the TRNG has a test mode that allows further diagnosis when errors occur. In test mode, programs have access to the main LFSR through the

TRNG_LFSR_L, TRNG_LFSR_M, and TRNG_LFSR_H registers. Programs can also control the finite state machine sample counter using the TRNG_CNT register.

In test mode, the TRNG can be configured to test individual or a chosen set of FROs. It can also be configured to feed test patterns to the delay chain.

TRNG Data Transfer Modes

The host processor reads four 32-bit registers to access the random numbers. Once the registers are read and the data ready interrupt has been acknowledged, the TRNG moves more data from the internal buffer to the output registers (TRNG_OUTPUT[n]).

TRNG Event Control

There are eight events that the TRNG generates. The events are common error events from the run-time testing of the TRNG. While the TRNG is operating and generating random bits, test circuitry is also running statistical tests. The tests determine if the sources of the random bits have started to fail and are not truly producing random bits. There is also a single event to signal when data is ready in the output registers.

These events are captured in the TRNG_STAT register and can generate a single interrupt in the Public Key Interrupt Controller (PKIC). The individual events can be masked so the interrupt is not triggered. This configuration uses the TRNG_CTL register. The event and associated interrupt can be acknowledged in the TRNG_INTACK register.

The ***TRNG Interrupt Signals*** table lists all events or interrupts that the TRNG can generate.

TRNG Interrupt Signals

The TRNG provides a total of eight interrupts multiplexed into one output.

Table 42-3: TRNG Interrupt Signals

Interrupt	Name	Description
0	RDY	Ready. Random number is ready to be read from registers
1	SHDNOVR	Shutdown Overflow. The number of FRO's automatically shut down due to failures or errors have gone above the threshold specified in TRNG_ALMCNT.SHDNTHRESH.
2	STUCKOUT	Stuck Out. Logic circuitry around the output registers has detected the same output has been provided twice.
3	NOISEFAIL	Noise Fail. Logic circuitry monitoring the data shifted into the main LFSR detected 48 identical bits, which is considered a noise source failure.
4	RUNFAIL	Run Fail. Logic circuitry monitoring the data shifted into the main LFSR detected an out-of-bounds value for at least one of the TRNG_RUN[n] counters after checking 20,000 bits.

Table 42-3: TRNG Interrupt Signals (Continued)

Interrupt	Name	Description
5	LRUNFAIL	Long Run Fail. Logic circuitry monitoring the data shifted into the main LFSR detected 34 identical bits.
6	PKRFAIL	Poker Fail. Logic circuitry monitoring the data shifted into the main LFSR detected an out-of-bounds value in at least one of the 16 TRNG_POKER[n] counters or an out-of-bounds sum of squares values after checking 20,000 bits.
7	MBITFAIL	Monobit Fail. Logic circuitry monitoring the data shifted into the main LFSR detected an out-of-bounds number of 1's after checking 20,000 bits.

ADSP-2184x TRNG Register Descriptions

True Random Number Generator (TRNG) contains the following registers.

Table 42-4: ADSP-2184x TRNG Register List

Name	Description
TRNG_ALMCNT	TRNG Alarm Counter Register
TRNG_ALMMSK	TRNG Alarm Mask Register
TRNG_ALMSTP	TRNG Alarm Stop Register
TRNG_BLKCNT	TRNG Block Count Register
TRNG_CFG	TRNG Configuration Register
TRNG_CNT	Counter Register
TRNG_CTL	TRNG Control Register
TRNG_FRODETUNE	TRNG FRO De-tune Register
TRNG_FROEN	TRNG FRO Enable Register
TRNG_INPUT[n]	TRNG Input Registers
TRNG_INTACK	TRNG Interrupt Acknowledge Register
TRNG_KEY[n]	Post-Process Key Registers
TRNG_LFSR_H	TRNG LFSR Access Register
TRNG_LFSR_L	TRNG LFSR Access Register
TRNG_LFSR_M	TRNG LFSR Access Register
TRNG_MONOBITCNT	TRNG Monobit Test Result Register
TRNG_OUTPUT[n]	TRNG Output Registers
TRNG_POKER[n]	TRNG Poker Test Result Registers
TRNG_RUNCNT	TRNG Run Count Registers
TRNG_RUN[n]	TRNG Run Test State and Result Registers

Table 42-4: ADSP-2184x TRNG Register List (Continued)

Name	Description
TRNG_STAT	TRNG Status Register
TRNG_TEST	TRNG Test Register
TRNG_V[n]	TRNG Post-Process "V" Value Registers

TRNG Alarm Counter Register

The TRNG_ALMCNT register, together with the TRNG_ALMMSK and TRNG_ALMSTP registers, can be used by the host processor to determine if the FRO/sample cycle locking is a problem. Note that incidental alarm events are expected to occur during normal operation. This register also controls the way the monobit test and poker test circuits operate (using the standard 20,000 bit blocks or running continuously).

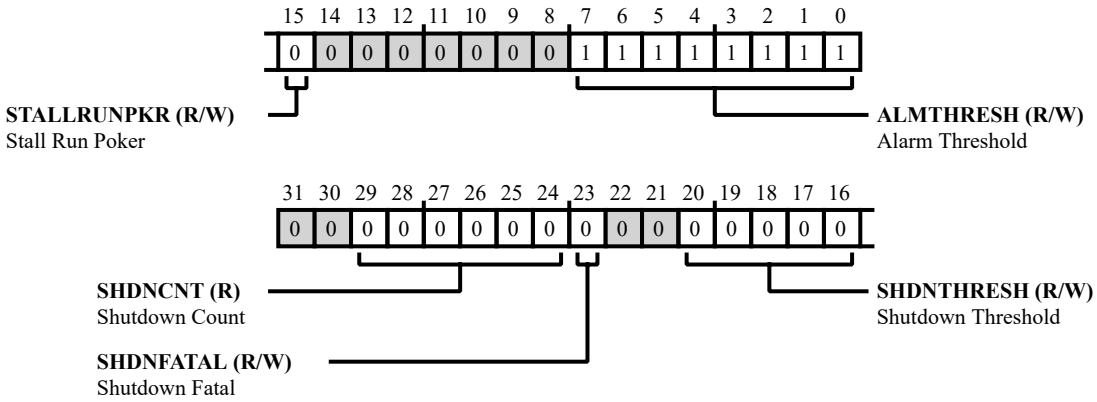


Figure 42-2: TRNG_ALMCNT Register Diagram

Table 42-5: TRNG_ALMCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/NW)	SHDNCNT	Shutdown Count. This read-only field indicates the number of 1 bits in the TRNG_ALMSTP register, the number of FRO's that's been turned off.
23 (R/W)	SHDNFATAL	Shutdown Fatal. When the TRNG_ALMCNT.SHDNFATAL bit field is set, the shutdown overflow (SHDNOVR) interrupt is considered a fatal error requiring taking the complete TRNG engine off-line.
20:16 (R/W)	SHDNTHRESH	Shutdown Threshold. The TRNG_ALMCNT.SHDNTHRESH bit field provides the threshold setting for generating the shutdown overflow (SHDNOVR) interrupt, which is activated when the shutdown count (TRNG_ALMCNT.SHDNCNT) value in this register exceeds the threshold value set here.
15 (R/W)	STALLRUNPKR	Stall Run Poker. When the TRNG_ALMCNT.STALLRUNPKR bit is set, stalls the Monobit Test, Run Test and Poker Test circuits when either the TRNG_STAT.MBITFAIL, TRNG_STAT.RUNFAIL or TRNG_STAT.PKRFAIL bits =1. This allows inspection of the state of the result counters (which would otherwise be reset immediately for the next 20,000 bits block to test).

Table 42-5: TRNG_ALMCNT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	ALMTHRESH	Alarm Threshold. The `TRNG_ALMCNT.ALMTHRESH` bit field sets the alarm detection threshold for the repeating pattern detectors on each FRO. A FRO alarm event is declared when a repeating pattern (of up to four samples length) is detected continuously for the number of samples defined by this fields value. Reset value 255 (decimal) should keep the number of alarm events to a manageable level.

TRNG Alarm Mask Register

A set bit (=1) in the TRNG_ALMMSK register signifies an alarm event and is used by the host processor to determine which of the individual FROs generated the alarm. If a bit in this register is set, the corresponding bit in the TRNG_ALMSTP register is set and the FRO is turned off by clearing the corresponding bit in the TRNG_FROEN register. If a bit is not set, the FRO restarts automatically to try to break sample cycle locking that could have caused the alarm event.

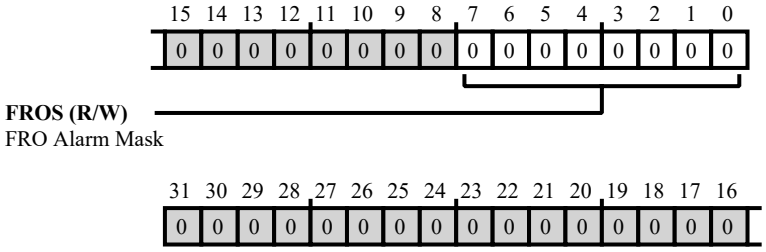


Figure 42-3: TRNG_ALMMSK Register Diagram

Table 42-6: TRNG_ALMMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	FRO Alarm Mask. The TRNG_ALMMSK.FROS bit field provides logging for the alarm events of individual FROs. A 1 in bit [n] indicates FRO n experienced an alarm event.

TRNG Alarm Stop Register

The TRNG_ALMSTP register is used by the host processor to determine which of the individual FROs generated more than one alarm event in quick succession. If a FRO generates an alarm event while a previous event is still logged in the TRNG_ALMMSK register, the corresponding bit in this register is set (=1) and the FRO is turned off by clearing (=0) the corresponding bit in the TRNG_FROEN register. The TRNG_ALMCNT.SHDNCNT bit field keeps track of the number of bits that are set in this register.

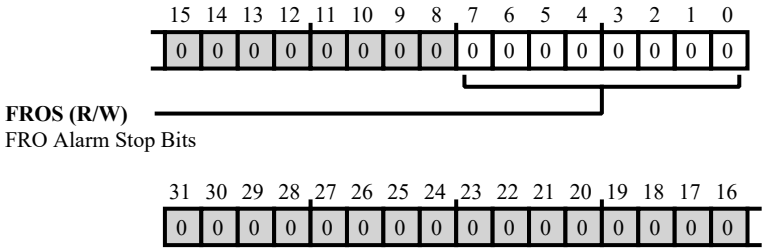


Figure 42-4: TRNG_ALMSTP Register Diagram

Table 42-7: TRNG_ALMSTP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	FRO Alarm Stop Bits. The TRNG_ALMSTP.FROS bit field provides logging for the alarm events of individual FROs. A 1 in bit [n] indicates FRO n experienced more than one alarm event in quick succession and has been turned off. A 1 in this field forces the corresponding bit in the TRNG_FROEN register to 0.

TRNG Block Count Register

The TRNG_BLKCNT register is the counter for the 128-bit blocks generated by the post-processor. These bits are forced to zero when the post-processor is disabled and are cleared to zero when an internal re-seed operation has finished. This register can be used by driver software to determine when to re-seed the post-processor.

The whole 32 bits of this register represent the amount of data (in bytes) generated since a re-seed. The TRNG_BLKCNT register is only present when a post-processor is available.

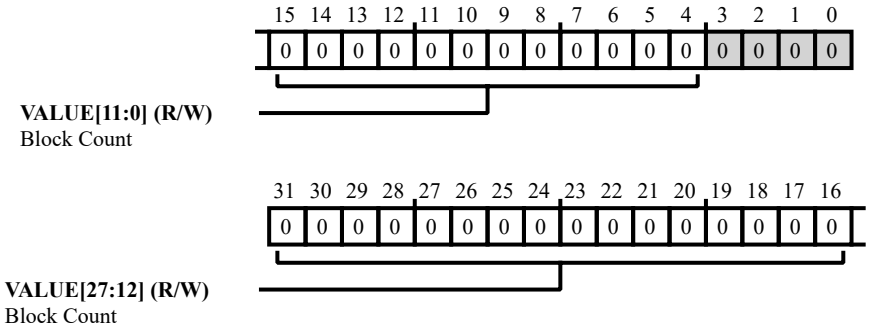


Figure 42-5: TRNG_BLKCNT Register Diagram

Table 42-8: TRNG_BLKCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/W)	VALUE	Block Count. The TRNG_BLKCNT.VALUE bit field is the counter for the 128-bit blocks generated by the post-processor. These bits are forced to zero when the post-processor is disabled and are cleared to zero when an internal re-seed operation has finished.

TRNG Configuration Register

The TRNG_CFG register holds the lower and upper limits of the samples taken from the FROs in order to refill the random data buffer. This register also holds the time out value used for secure reading mode.

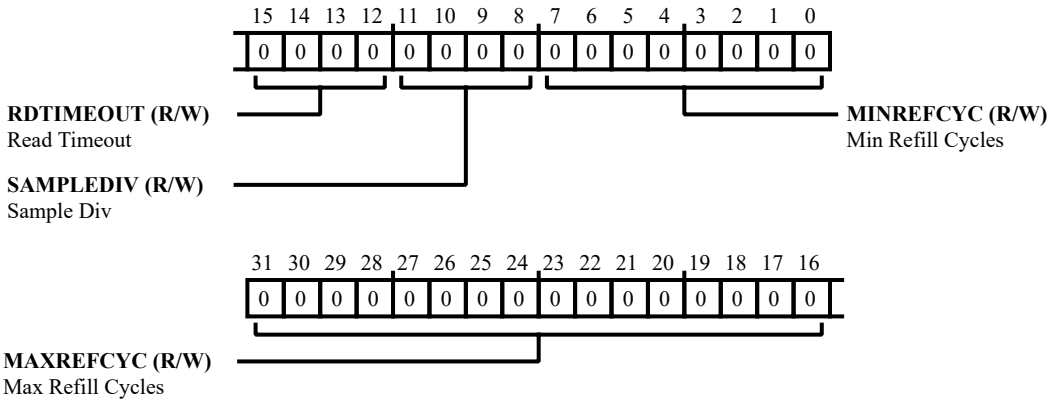


Figure 42-6: TRNG_CFG Register Diagram

Table 42-9: TRNG_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	MAXREFCYC	Max Refill Cycles. The TRNG_CFG.MAXREFCYC bit field determines the maximum number of samples (between 2^8 and 2^{24}) taken to re-generate entropy from the FROs after reading out a 64-bit random number. If the written value of this field is zero, the number of samples is 2^{24}, otherwise the number of samples equals the written value times 2^8. This field can only be modified while the TRNG_CTL.TRNGEN bit =0.
15:12 (R/W)	RDTIMEOUT	Read Timeout. The TRNG_CFG.RDTIMEOUT bit field controls the secure reading mode. When this field is 0, secure reading mode is disabled. Values in the range 1-15 enable secure reading and set a read gate closure timeout of approximately (read_timeout + 1) x 16 clock input cycles. This field can only be modified while the TRNG_CTL.TRNGEN bit =0.

Table 42-9: TRNG_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11:8 (R/W)	SAMPLEDIV	Sample Div. The TRNG_CFG.SAMPLEDIV bit field directly controls the number of input cycles between samples taken from the FROs. The default value 0 indicates that samples are taken every cycle, maximum value 15 (decimal) takes one sample every 16 cycles. This field must be set to a value such that the slowest FRO (even under worst-case conditions) has a cycle time less than twice the sample period. The default configuration of the FROs allows this field to remain 0. This field can only be modified while TRNG_CTL.TRNGEN=0. Note: In order to avoid oversampling the FRO, do not to use a TRNG_CFG.SAMPLEDIV value of 0.
7:0 (R/W)	MINREFCYC	Min Refill Cycles. The TRNG_CFG.MINREFCYC bit field determines the minimum number of samples (between 2^6 and 2^{24}) taken to re-generate entropy from the FROs after reading out a 64-bit random number. If the value of this field is zero, the number of samples is fixed to the value determined by the maximum refill cycles (TRNG_CFG.MAXREFCYC) field, otherwise the minimum number of samples equals the written value times 64 (which can be up to 2^{14}). The number of samples defined here cannot be higher than the number defined by the TRNG_CFG.MAXREFCYC field (i.e. that field takes precedence). This field can only be modified while the TRNG_CTL.TRNGEN bit =0.

Counter Register

The TRNG_CNT register is used to access the main control Finite State Machine's (FSM) sample counter while the TRNG_CTL.TSTMODE bit =1.

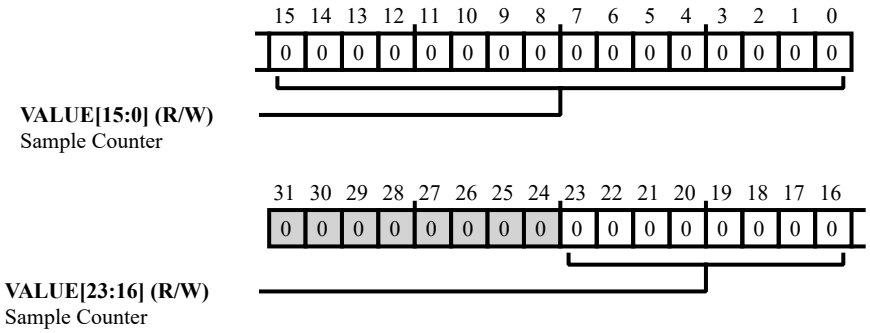


Figure 42-7: TRNG_CNT Register Diagram

Table 42-10: TRNG_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/W)	VALUE	Sample Counter. The TRNG_CNT.VALUE bit field is the sample counter used by control finite state machine. This counter can only be accessed when the TRNG_CTL.TSTMODE bit =1.

TRNG Control Register

The TRNG_CTL register must be written to start accumulating entropy before random numbers can be generated. In most cases, the TRNG_CFG register must also be written prior to writing the TRNG_CTL register. To enable the TRNG, set the TRNG_CTL.TRNGEN bit. This register also controls post-processing (if available). Note that when the TRNG_CTL.TRNGEN bit =1, the start up cycles field (TRNG_CTL.STARTUPCYC) and the post-processing enable bit (if available) are locked. Any writes to these fields are ignored.

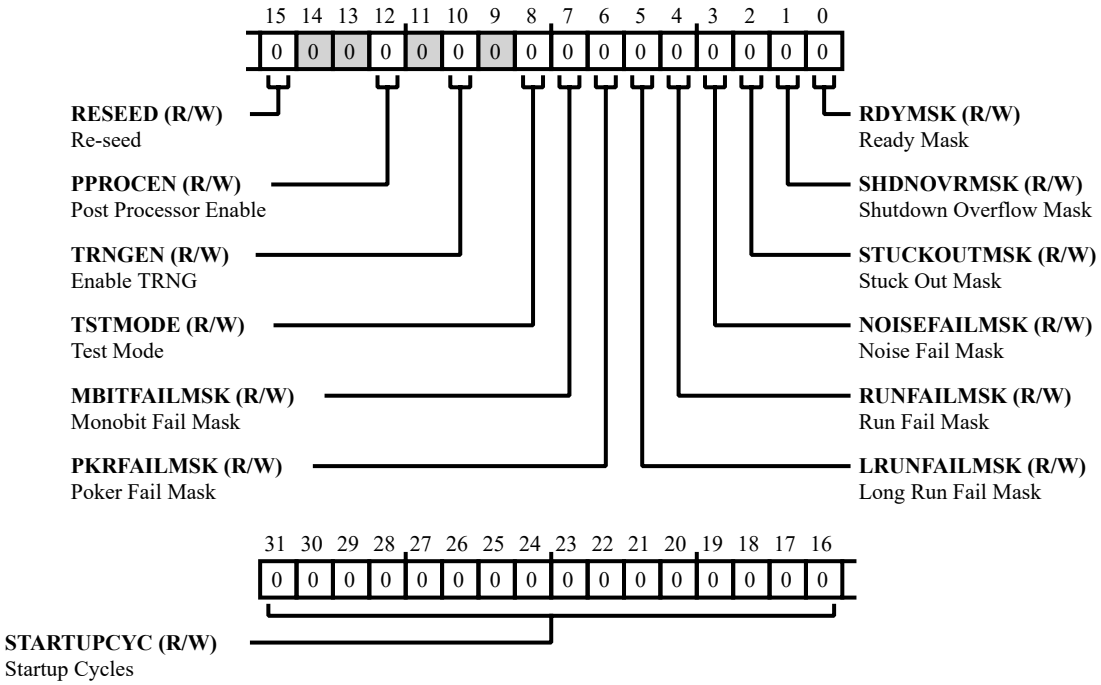


Figure 42-8: TRNG_CTL Register Diagram

Table 42-11: TRNG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	STARTUPCYC	Startup Cycles. The TRNG_CTL.STARTUPCYC bit field determines the number of samples (between 2^8 and 2^{24}) taken to gather entropy from the FROs during startup. If the written value of this field is zero, the number of samples is 2^{24}, otherwise the number of samples equals the written value times 2^8 . This field can only be written when TRNG_CTL.TRNGEN=0 before the write.

Table 42-11: TRNG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	RESEED	Re-seed. The TRNG_CTL.RESEED bit is set-only, writing a 1 starts a re-seed cycle and writing a 0 has no effect. A re-seed cycle entails loading the TRNG_KEY[n] and TRNG_V[n] registers with random values generated internally; these values are not visible outside the TRNG core. This bit falls back to 0 automatically after the re-seed operation is complete; at that time the TRNG_BLKCNT register is reset to zero and the random data buffer is zeroized so that any new data read from the TRNG will use the new seed values. This bit is only present when post-processing is available and can only be set to 1 when TRNG_CTL.TRNGEN=1 before the write. Note that re-seeding can be done with the post-processor disabled (normally used to seed the post-processor before enabling it). When writing a 1 to this bit, all other bits in this register remain unchanged.
12 (R/W)	PPROCEN	Post Processor Enable. Setting the TRNG_CTL.PPROCEN bit enables the FIPS post-processor. If this bit is reset to 0, the post-processor is forced back into the idle state immediately. This bit is only present when post-processing is available and can only be changed when the TRNG_CTL.TRNGEN bit (enable TRNG) was 0 before the write. To change the TRNG_CTL.PPROCEN bit during operation, first put the TRNG into reset (TRNG_CTL.TRNGEN =0). If the post-processor is enabled, it can be disabled by a subsequent write of 0 to this bit (writing this bit when the TRNG_CTL.TRNGEN bit is still 1 has no effect). The post-processor then stops immediately. To enable it, this bit bit must be written with 1. Changing the enabled/disabled state does not affect the contents of the TRNG_KEY[n] and TRNG_V[n] registers. After changing the state, the TRNG must be started again by setting the TRNG_CTL.TRNGEN bit to 1. Note that it is required to re-gather entropy, so the same number of start-up cycles must be used as when starting the TRNG out of a system reset state.
10 (R/W)	TRNGEN	Enable TRNG. Setting the TRNG_CTL.TRNGEN bit to 1 starts the TRNG, gathering entropy from the FROs for the number of samples determined by the value in the TRNG_CTL.STARTUPCYC (Startup Cycles) field. Resetting this bit to 0 forces all TRNG logic back into the idle state immediately. Resetting this bit to 0 also performs the Un-instantiate operation, clearing all internal post-processor registers.
8 (R/W)	TSTMODE	Test Mode. When the TRNG_CTL.TSTMODE bit is set, access is enabled to the TRNG_CNT and TRNG_LFSR_L, TRNG_LFSR_M and TRNG_LFSR_H registers (the latter are cleared before enabling access) and sets the TRNG_STAT.NEEDCLK bit for testing purposes. This bit must be set to 1 before various test modes in the TRNG_TEST register can be enabled.

Table 42-11: TRNG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	MBITFAILMSK	Monobit Fail Mask. When the TRNG_CTL.MBITFAILMSK bit is set, this mask allows the TRNG_STAT.MBITFAIL bit to activate the (active HIGH) interrupt output.
6 (R/W)	PKRFAILMSK	Poker Fail Mask. When the TRNG_CTL.PKRFAILMSK bit is set, this mask allows the TRNG_STAT.PKRFAIL bit to activate the (active HIGH) interrupt output.
5 (R/W)	LRUNFAILMSK	Long Run Fail Mask. When the TRNG_CTL.LRUNFAILMSK bit is set, this mask allows the TRNG_STAT.LRUNFAIL bit to activate the (active HIGH) interrupt output.
4 (R/W)	RUNFAILMSK	Run Fail Mask. When the TRNG_CTL.RUNFAILMSK bit is set, this mask allows the TRNG_STAT.RUNFAIL bit to activate the (active HIGH) interrupt output.
3 (R/W)	NOISEFAILMSK	Noise Fail Mask. When the TRNG_CTL.NOISEFAILMSK bit is set, this mask allows the TRNG_STAT.NOISEFAIL bit to activate the (active HIGH) interrupt output.
2 (R/W)	STUCKOUTMSK	Stuck Out Mask. When the TRNG_CTL.STUCKOUTMSK bit is set, this mask allows the TRNG_STAT.STUCKOUT bit to activate the (active HIGH) interrupt output.
1 (R/W)	SHDNOVRMSK	Shutdown Overflow Mask. When the TRNG_CTL.SHDNOVRMSK bit is set, this mask allows the TRNG_STAT.SHDNOVR bit to activate the (active HIGH) interrupt output.
0 (R/W)	RDYMSK	Ready Mask. When the TRNG_CTL.RDYMSK bit is set, this mask allows the TRNG_STAT.RDY bit to activate the (active HIGH) interrupt output.

TRNG FRO De-tune Register

The TRNG_FRODETUNE register is used by the host processor to change the frequencies of individual FROs. This can reduce the number of alarm events generated by a specific FRO.

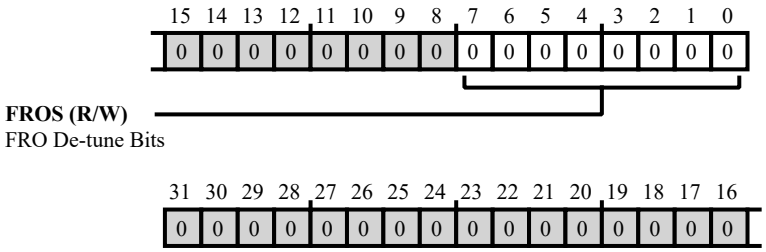


Figure 42-9: TRNG_FRODETUNE Register Diagram

Table 42-12: TRNG_FRODETUNE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	FRO De-tune Bits. The TRNG_FRODETUNE.FROS bits De-tune the FROs. A 1 in bit [n] lets FRO n run approximately 5% faster. The value of one of these bits may only be changed while the corresponding FRO is turned off (by temporarily writing a 0 in the corresponding bit of the TRNG_FROEN register).

TRNG FRO Enable Register

The TRNG_FROEN register can be used by the host processor to enable and disable FROs individually. Only enabled FROs contribute to entropy generation, but require power to do so. Disabled FROs cannot generate alarm events.

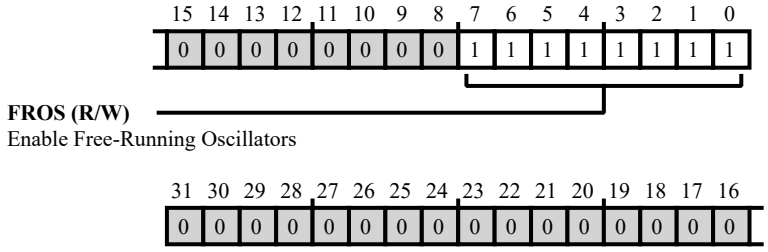


Figure 42-10: TRNG_FROEN Register Diagram

Table 42-13: TRNG_FROEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	Enable Free-Running Oscillators. The TRNG_FROEN.FROS bits are the enables for the individual FROs. A 1 in bit [n] enables FRO n. The default state is all ones to enable all FROs after power-up. Note that the FROs are not actually started up before the TRNG_CTL.TRNGEN bit is set to 1. These bits are automatically forced to 0 (and cannot be written to 1) when the corresponding bit in the TRNG_ALMSTP register has value 1.

TRNG Input Registers

The TRNG_INPUT[n] registers are used as input for post-processor testing (if post processing is available) and as input for Monobit Test, Run Test and Poker Test functionality tests (TRNG_INPUT0 only). They share their addresses with the corresponding TRNG_OUTPUT[n] registers. The least significant word is contained in the TRNG_INPUT0 register.

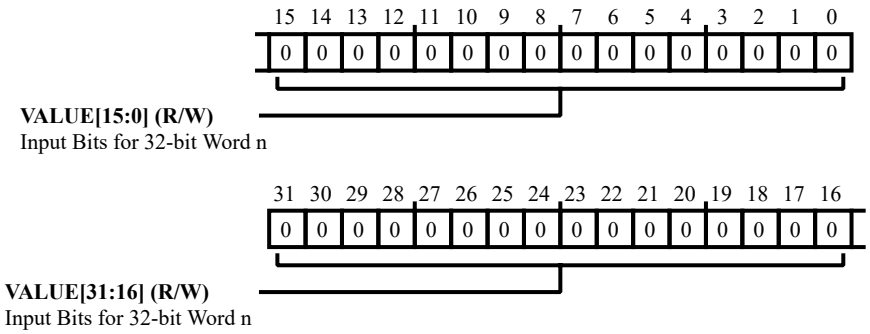


Figure 42-11: TRNG_INPUT[n] Register Diagram

Table 42-14: TRNG_INPUT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Input Bits for 32-bit Word n. The TRNG_INPUT[n].VALUE bit field is used to hold 32-bits of the 64-bit word of test data for 3-DES post-processor (if available). Or, the TRNG_INPUT[n].VALUE bit field is used to hold 32-bit data word for Run Test and Poker Test circuits self test. Can only be written to when the TRNG_STAT.TSTRDY bit =1.

TRNG Interrupt Acknowledge Register

The TRNG_INTACK register is written to acknowledge interrupts indicated in bits [7:0] of the TRNG_STAT register. Writing a 1 to any of the bits [7:2] has side effects in resetting various parts of the TRNG core logic which can also be used even if no interrupts are actually active.

When acknowledging the interrupts, these bits are write '1' to clear the associated bit in TRNG_STAT register. The bit in this register will also automatically be reset to zero.

When secure reading mode is enabled, write bits [7:0] of this register with zeros to enable TRNG data reads from the TRNG_OUTPUT[n] registers. Writing bits [7:0] also starts the (configurable) timeout counter that automatically acknowledges the TRNG data (and disables reads) if the TRNG_INTACK.RDY bit is not written with a 1 within that timeout period.

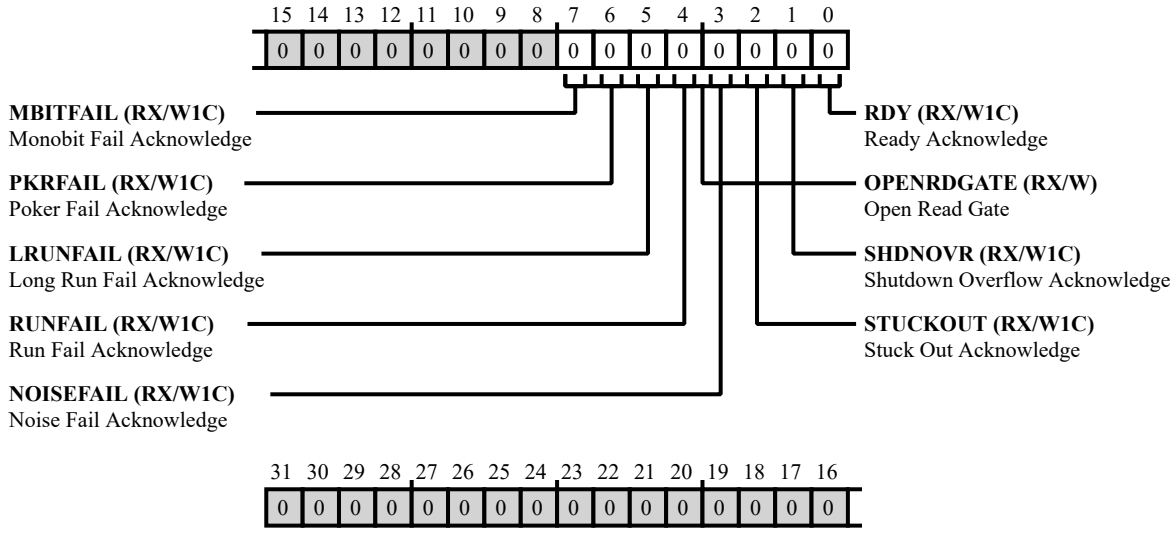


Figure 42-12: TRNG_INTACK Register Diagram

Table 42-15: TRNG_INTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (RX/W1C)	MBITFAIL	Monobit Fail Acknowledge. Set the TRNG_INTACK.MBITFAIL bit to acknowledge the Monobit Fail Interrupt. This also resets all counter and state bits in the TRNG_RUN[n], TRNG_MONOBITCNT and the TRNG_POKER[n] registers (except for the TRNG_RUNCNT.LENMAX field).
6 (RX/W1C)	PKRFAIL	Poker Fail Acknowledge. Set the TRNG_INTACK.PKRFAIL bit to acknowledge the Poker Fail Interrupt. This also resets all counter and state bits in the TRNG_RUN[n], TRNG_MONOBITCNT and the TRNG_POKER[n] registers (except for the TRNG_RUNCNT.LENMAX field).

Table 42-15: TRNG_INTACK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (RX/W1C)	LRUNFAIL	Long Run Fail Acknowledge. Set the TRNG_INTACK.LRUNFAIL bit to acknowledge the Long Run Fail Interrupt. Also clears the TRNG_RUNCNT.LENMAX field.
4 (RX/W1C)	RUNFAIL	Run Fail Acknowledge. Set the TRNG_INTACK.RUNFAIL bit to acknowledge the Run Fail Interrupt. Also resets all counter and state bits in the TRNG_RUN[n], TRNG_MONOBITCNT and the TRNG_POKER[n] registers (except for the TRNG_RUNCNT.LENMAX field).
3 (RX/W1C)	NOISEFAIL	Noise Fail Acknowledge. Set the TRNG_INTACK.NOISEFAIL bit to acknowledge the Noise Fail Interrupt. Setting this bit also clears the TRNG_RUNCNT.LENMAX (Run Length Max) field, the random data buffer, the TRNG_OUTPUT[n] registers and the TRNG_STAT.RDY bit.
2 (RX/W1C)	STUCKOUT	Stuck Out Acknowledge. Set the TRNG_INTACK.STUCKOUT bit to acknowledge the Stuck Out Interrupt. Setting this bit also clears the random data buffer, the TRNG_OUTPUT[n] registers and the TRNG_STAT.RDY bit.
1 (RX/W1C)	SHDNOVR	Shutdown Overflow Acknowledge. Set the TRNG_INTACK.SHDNOVR bit to acknowledge the Shutdown Overflow Interrupt.
0 (RX/W1C)	RDY	Ready Acknowledge. The TRNG_INTACK.RDY bit allows a new number (if it is ready in the random data buffer), to directly move into the result register. Once done, the TRNG_STAT.RDY bit is reset, after at most size clock cycles.
7:0 (RX/W)	OPENRDGATE	Open Read Gate. In Secure Reading Mode, the TRNG_INTACK.OPENRDGATE bit writes an all zeros value to bits [7:0] to enable reading of TRNG data from the TRNG_OUTPUT[n] registers. This starts the timeout counter that automatically acknowledges the TRNG data (and disables reading) if the TRNG_INTACK.RDY bit is not written with a 1 within that timeout period.

Post-Process Key Registers

The TRNG_KEY[n] registers are used to load the key used for post-processing (if available). These registers are write-only. Reads return the values of the other registers mapped at the same addresses.

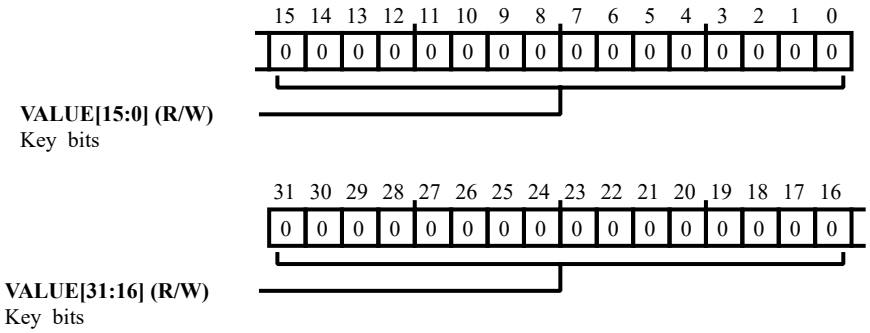


Figure 42-13: TRNG_KEY[n] Register Diagram

Table 42-16: TRNG_KEY[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Key bits. Bits for cipher key used in the post-processor.

TRNG LFSR Access Register

The TRNG_LFSR_H register is used to access bits [80:64] of the main entropy accumulation LFSR while in test mode (TRNG_CTL.TSTMODE =1).

For security reasons, the LFSR contents are zeroed before enabling access.

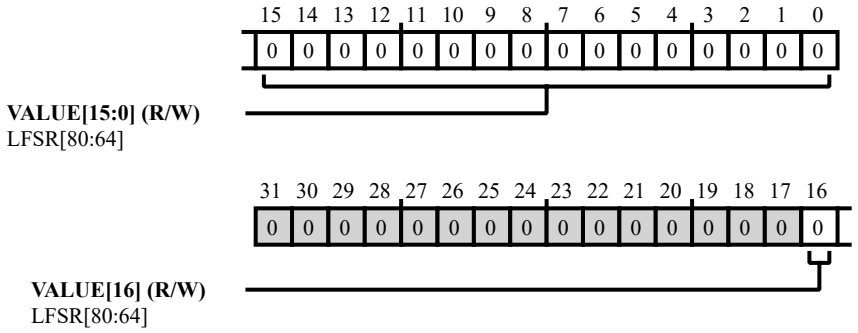


Figure 42-14: TRNG_LFSR_H Register Diagram

Table 42-17: TRNG_LFSR_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
16:0 (R/W)	VALUE	LFSR[80:64]. The TRNG_LFSR_H.VALUE bit field contains bits [80:64] of the main entropy accumulation LFSR. This field can only be accessed when the TRNG_CTL.TSTMODE bit =1. Contents are cleared (=0) before access is enabled.

TRNG LFSR Access Register

The TRNG_LFSR_L register is used to access bits [31:0] of the main entropy accumulation LFSR while in test mode (TRNG_CTL.TSTMODE =1).

For security reasons, the LFSR contents are zeroed before enabling access.

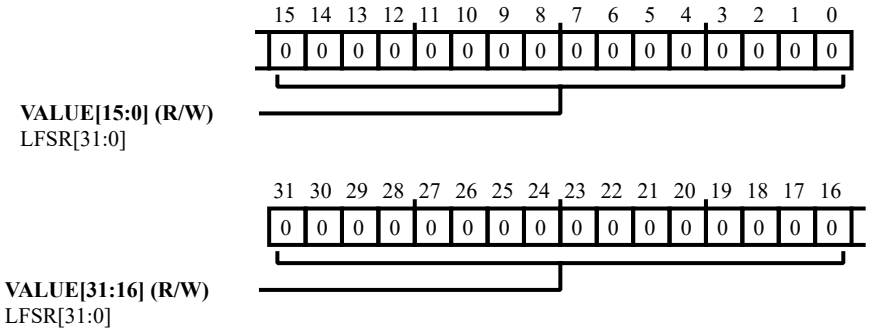


Figure 42-15: TRNG_LFSR_L Register Diagram

Table 42-18: TRNG_LFSR_L Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	LFSR[31:0]. The TRNG_LFSR_L.VALUE bit field contains bits [31:0] of the main entropy accumulation LFSR. This field can only be accessed when the TRNG_CTL.TSTMODE bit =1. Contents are cleared (=0) before access is enabled.

TRNG LFSR Access Register

The TRNG_LFSR_M register is used to access bits [63:32] of the main entropy accumulation LFSR while in test mode (TRNG_CTL.TSTMODE =1).

For security reasons, the LFSR contents are zeroed before enabling access.

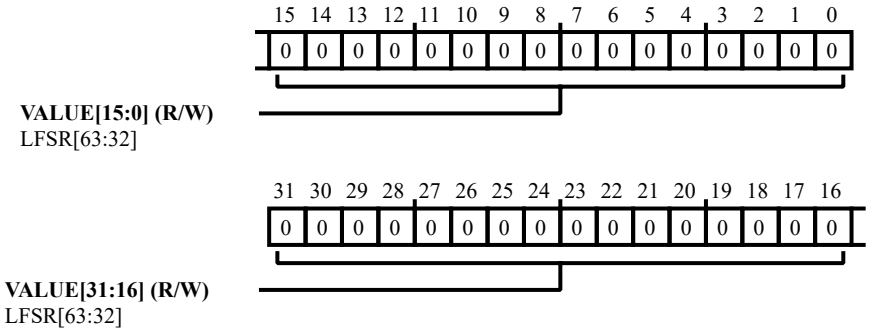


Figure 42-16: TRNG_LFSR_M Register Diagram

Table 42-19: TRNG_LFSR_M Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	LFSR[63:32]. The TRNG_LFSR_M.VALUE bit field contains bits [63:32] of the main entropy accumulation LFSR. This field can only be accessed when the TRNG_CTL.TSTMODE bit =1. Contents are cleared (=0) before access is enabled.

TRNG Monobit Test Result Register

The TRNG_MONOBITCNT register accesses the counter used perform a Monobit Test as specified by the AIS-31 standard (test T1, ref 4). This test is performed on blocks of 20,000 bits (in parallel to the run test and Poker Test).

Note: Immediately after performing the actual Monobit Test at the end of the 20,000 bits block, the counter is used to accumulate the Poker Test results. As a result, the actual Monobit Test count result value can only be read in the TRNG_MONOBITCNT register if the test fails and the stall run Poker (TRNG_ALMCNT.STALLRUNPKR) bit =1.

The monobit test result register is read-only; writing it accesses the registers mapped at the same address. The counter in this register is reset when writing a 1 to either the monobit fail acknowledge (TRNG_INTACK.MBITFAIL), run fail acknowledge (TRNG_INTACK.RUNFAIL) or the poker fail acknowledge (TRNG_INTACK.PKRFAIL) bits.

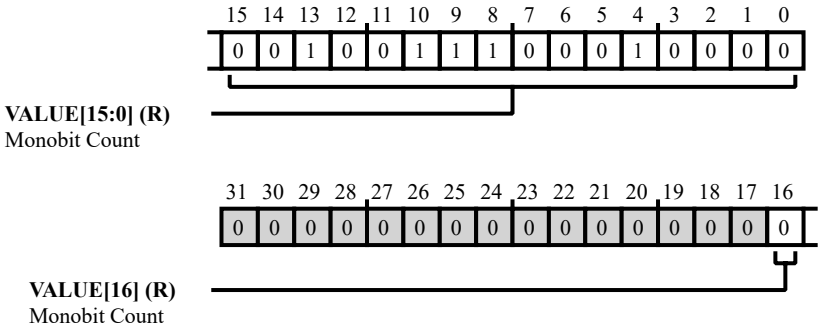


Figure 42-17: TRNG_MONOBITCNT Register Diagram

Table 42-20: TRNG_MONOBITCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
16:0 (R/NW)	VALUE	Monobit Count. The TRNG_MONOBITCNT.VALUE bit field is the up/down counter which monitors 1 and 0 bits. After 20,000 bits, this counter should have a value in the range 9310 through 10690 (inclusive) to pass the Monobit Test. This counter is protected against overflow and underflow.

TRNG Output Registers

The TRNG_OUTPUT[n] registers provide read access to the 128-bit random number output. A subset of these registers are also used as output for post-processor testing (if available). They share their addresses with the TRNG_INPUT0 through TRNG_INPUT3 registers. The least significant word is contained in the TRNG_OUTPUT0 register.

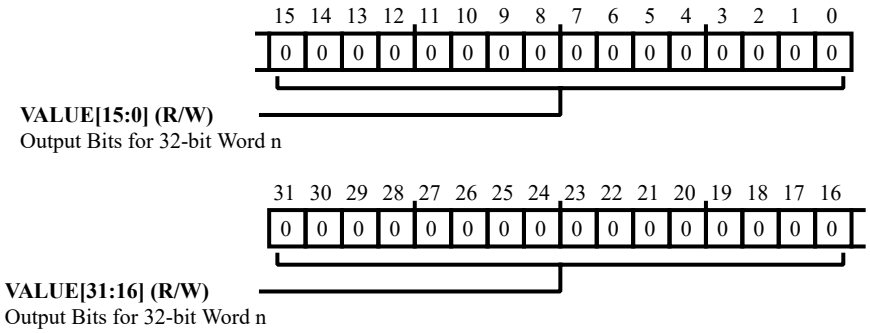


Figure 42-18: TRNG_OUTPUT[n] Register Diagram

Table 42-21: TRNG_OUTPUT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Output Bits for 32-bit Word n. The TRNG_OUTPUT[n].VALUE bit field is used to holds 32 bits of the 128-bit word of random data. Only valid when the TRNG_STAT.RDY bit =1. Alternatively, this register holds the 32-bits of the 42-bit word of result data for 3-DES post-processing testing. Only valid when the TRNG_STAT.TSTRDY bit =1.

TRNG Poker Test Result Registers

The TRNG_POKER[n] registers are used to access the 16 counters used perform a poker test on blocks of 20,000 bits (in parallel to the monobit and run tests).

Poker test result registers are read-only; writing them accesses the registers mapped at these same addresses. All counters in these registers are reset when writing a 1 to either the monobit fail acknowledge (TRNG_INTACK.MBITFAIL), run fail acknowledge (TRNG_INTACK.RUNFAIL) or the poker fail acknowledge (TRNG_INTACK.PKRFAIL) bits.

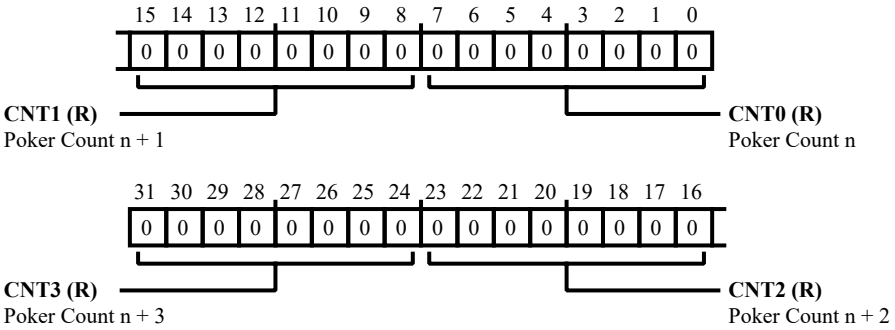


Figure 42-19: TRNG_POKER[n] Register Diagram

Table 42-22: TRNG_POKER[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	CNT3	Poker Count n + 3. The TRNG_POKER[n].CNT3 bit field provides the counter for 4-bit value 0x3, 0x7, 0xB, 0xF in TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.
23:16 (R/NW)	CNT2	Poker Count n + 2. The TRNG_POKER[n].CNT2 bit field provides the counter for 4-bit value 0x2, 0x6, 0xA, 0xE in TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.
15:8 (R/NW)	CNT1	Poker Count n + 1. The TRNG_POKER[n].CNT1 bit field provides the counter for 4-bit value 0x1, 0x5, 0x9, 0xD in TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.
7:0 (R/NW)	CNT0	Poker Count n. The TRNG_POKER[n].CNT0 bit field provides the counter for 4-bit value 0x0, 0x4, 0x8, 0xC in the TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.

TRNG Run Count Registers

The TRNG_RUNCNT registers are used to access the 10 counters that perform a run test and long run test as specified by the AIS-31 standard (tests T3 and T4, ref 4). They are also used to perform the noise source failure test proposed in section E.5 of that same standard.

The TRNG_RUNCNT registers are read-only; writing them accesses the other registers which are mapped at the same addresses. Unless otherwise indicated, all counters and state bits in these registers are reset when writing a 1 to either the Monobit Fail acknowledge (TRNG_INTACK.MBITFAIL), Run Fail acknowledge (TRNG_INTACK.RUNFAIL) or the Poker Fail acknowledge (TRNG_INTACK.PKRFAIL) bits.

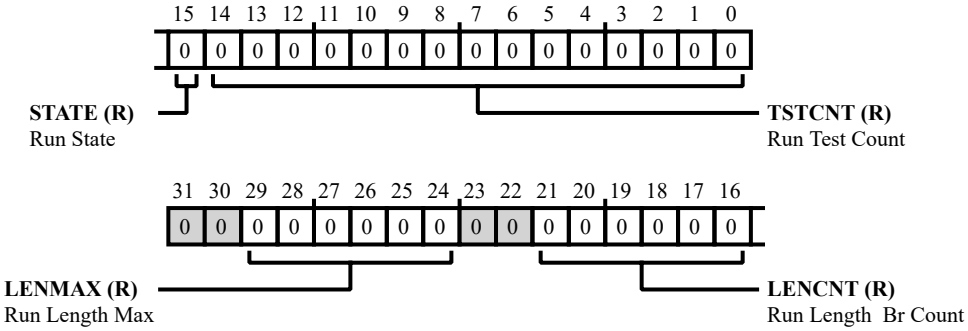


Figure 42-20: TRNG_RUNCNT Register Diagram

Table 42-23: TRNG_RUNCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/NW)	LENMAX	Run Length Max. The TRNG_RUNCNT.LENMAX bit field configures the maximum run length count value encountered since start of test. This value is reset back to zero when writing a 1 to either the Noise Fail acknowledge (TRNG_INTACK.NOISEFAIL) or the Long Run Fail acknowledge (TRNG_INTACK.LRUNFAIL) bits.
21:16 (R/NW)	LENCNT	Run Length Br Count. The TRNG_RUNCNT.LENCNT bit field configures the counter for the current run of consecutive 0/1 bits; cannot increment past its maximum value of 63.
15 (R/NW)	STATE	Run State. The TRNG_RUNCNT.STATE bit field provides th state of bits in the current run.
14:0 (R/NW)	TSTCNT	Run Test Count. The TRNG_RUNCNT.TSTCNT bit field configures the block length counter for the run and poker tests - counts up for 20,000 tested bits and then controls testing of the run_X_count_... and poker_count_X counters to contain expected values, after which they - and this counter - are reset for the next block.

TRNG Run Test State and Result Registers

The TRNG_RUN[n] registers holds the counts for the associated run bucket for 1's and 0's.

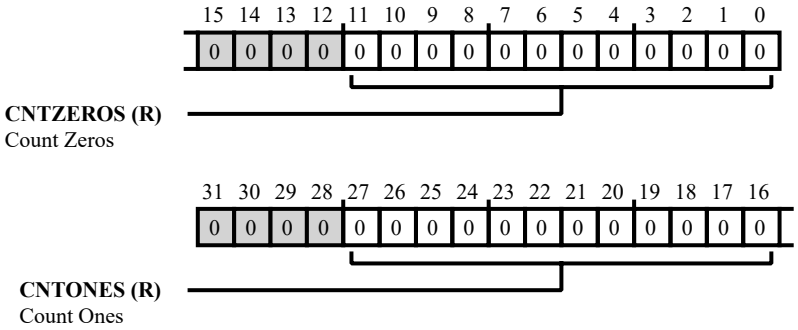


Figure 42-21: TRNG_RUN[n] Register Diagram

Table 42-24: TRNG_RUN[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/NW)	CNTONES	Count Ones. In TRNG_RUN1, this counter is for single bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 2267 to 2733 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 4095. In TRNG_RUN2, this counter is for two bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 1079 to 1421 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 2047. In TRNG_RUN3, this counter is for three bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 502 to 748 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 1023. In TRNG_RUN4, this counter for four bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 233 to 402 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 511. In TRNG_RUN5, this counter is for five bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 90 to 223 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255. In TRNG_RUN6, this counter for six and higher bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 90 to 233 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255.

Table 42-24: TRNG_RUN[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/NW)	CNTZEROS	Count Zeros. In TRNG_RUN1, this counter is for single bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 2267 to 2733 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 4095. In TRNG_RUN2, this counter is for two bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 1079 to 1421 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 2047. In TRNG_RUN3, this counter is for three bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 502 to 748 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 1023. In TRNG_RUN4, this counter is for four bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 233 to 402 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 511. In TRNG_RUN5, this counter is for five bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 90 to 223 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255. In TRNG_RUN6, this counter is for six and higher bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 90 to 233 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255.

TRNG Status Register

The TRNG_STAT register provides status results. This register shares the same address as the Interrupt Acknowledge (TRNG_INTACK) register.

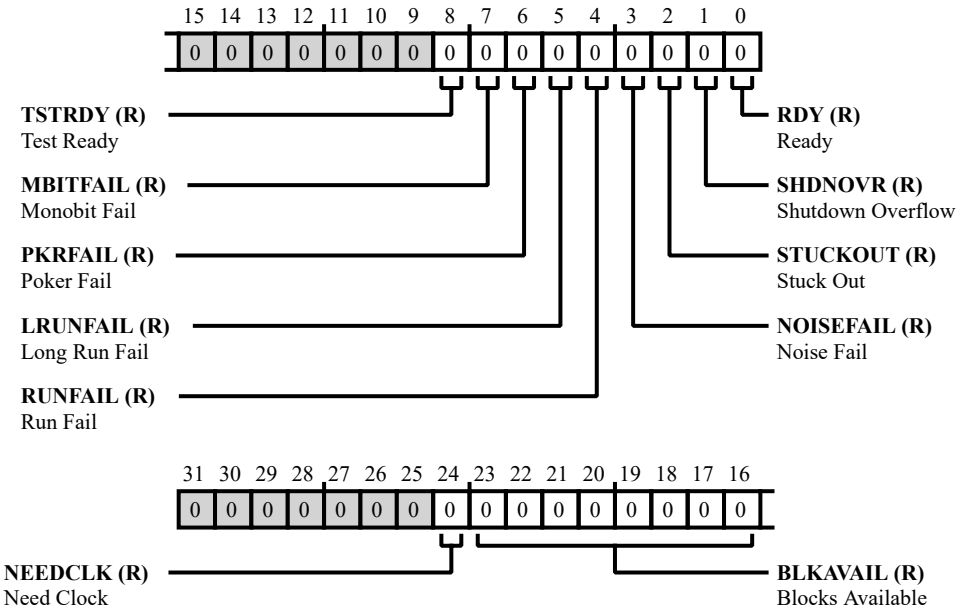


Figure 42-22: TRNG_STAT Register Diagram

Table 42-25: TRNG_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/NW)	NEEDCLK	Need Clock. When the TRNG_STAT.NEEDCLK bit is set, it indicates that the TRNG is busy generating entropy or is in one of its test modes; the module clock may not be turned off.
23:16 (R/NW)	BLKAVAIL	Blocks Available. This field indicates the number of 128-bit blocks of random data that are available in the random data buffer. If this value is non-zero, the output registers will be refilled from the random data buffer immediately after acknowledging the TRNG_STAT.RDY by writing a 1 to TRNG_INTACK.RDY.
8 (R/NW)	TSTRDY	Test Ready. When the TRNG_STAT.TSTRDY bit is set, it indicates that data for known-answer tests on the monobit test, run test, poker test and post-processor functions can be written to the TRNG_INPUT[n] registers. When testing the post-processor, result data can be read from those same registers when this bit has become 1 again (after dropping to 0).

Table 42-25: TRNG_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	MBITFAIL	Monobit Fail. When the TRNG_STAT.MBITFAIL bit is set, the monobit test logic monitoring data, shifted into the main LFSR, detected an out-of-bounds number of 1s after checking 20,000 bits (test T1 as specified in the AIS-31 standard).
6 (R/NW)	PKRFAIL	Poker Fail. When the TRNG_STAT.PKRFAIL bit is set, the poker test logic monitoring data shifted into the main LFSR detected an out-of-bounds value in at least one of the 16 poker counters or an out of bounds "sum of squares" value after checking 20,000 bits (test T2 as specified in the AIS-31 standard).
5 (R/NW)	LRUNFAIL	Long Run Fail. When the TRNG_STAT.LRUNFAIL bit is set, the run test logic monitoring data shifted into the main LFSR detected a sequence of 34 identical bits (test T4 as specified in the AIS-31 standard).
4 (R/NW)	RUNFAIL	Run Fail. When the TRNG_STAT.RUNFAIL bit is set, the run test logic monitoring data shifted into the main LFSR detected an out-of-bounds value for at least one of the TRNG_RUN[n].CNTZEROS or TRNG_RUN[n].CNTONES counters after checking 20,000 bits (test T3 as specified in the AIS-31 standard).
3 (R/NW)	NOISEFAIL	Noise Fail. When the TRNG_STAT.NOISEFAIL bit is set, the Run Test logic monitoring data shifted into the main LFSR detected a sequence of 48 identical bits, which is considered a noise source failure as proposed in section E.5 of the AIS-31 standard.
2 (R/NW)	STUCKOUT	Stuck Out. When the TRNG_STAT.STUCKOUT bit is set, the logic around the output data registers detected that the TRNG generates the same value twice in a row.
1 (R/NW)	SHDNOVR	Shutdown Overflow. When the TRNG_STAT.SHDNOVR bit is set, the number of FROs shut down after a second error event (the number of 1 bits in the TRNG_ALMSTP register) has exceeded the threshold set by the TRNG_ALMCNT.SHDNTHRESH bit field.
0 (R/NW)	RDY	Ready. When the TRNG_STAT.RDY bit is set, data is available in the TRNG_OUTPUT0 to TRNG_OUTPUT3 registers. If a new number is already available in the random data buffer, that number is directly moved into the result register. In this case the ready status bit is asserted again, after at most six module clock cycles.

TRNG Test Register

The TRNG_TEST register can be used by the host processor to perform a number of tests on the TRNG logic including:

- Register controlled characterization by connecting the tst_fro_clk_out output to a selected FRO clock output
- FRO logic connectivity and error event detection checking by feeding known patterns through the FRO delay line and error event detection circuits
- Direct XOR-ed FRO outputs capture by disabling the main LFSR feedback logic
- Extend the Monobit Test and Poker Test by not resetting the Monobit count and Poker Test X counters after each 20,000 bits block
- Perform known answer tests on the Run Test, Poker Test and post-processor functions.

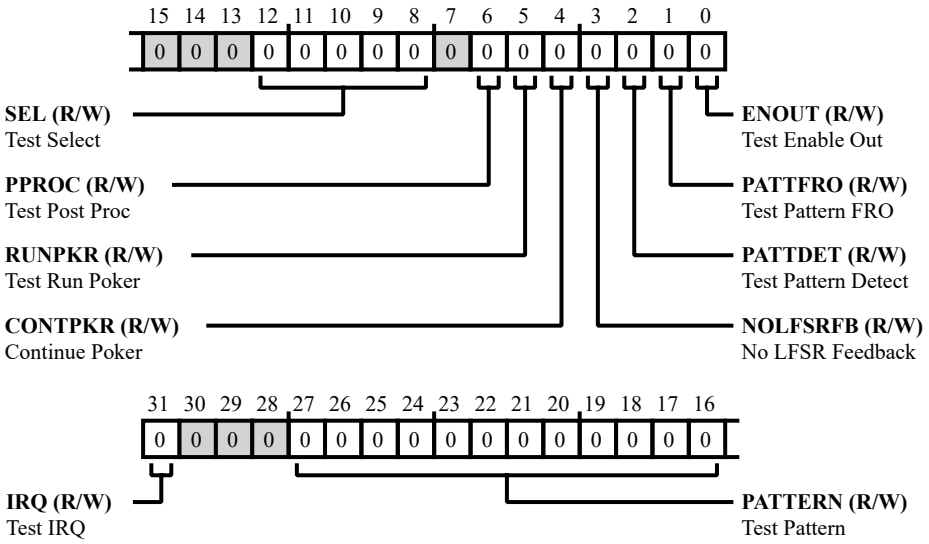


Figure 42-23: TRNG_TEST Register Diagram

Table 42-26: TRNG_TEST Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	IRQ	Test IRQ. When the TRNG_TEST.IRQ bit is set force irq output HIGH for interrupt signal connectivity testing.	
		0	Do not force IRQ high
		1	Force IRQ output high

Table 42-26: TRNG_TEST Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27:16 (R/W)	PATTERN	Test Pattern. The TRNG_TEST.PATTERN bit field sets up a repeating sequence of bits to be fed into the selected FRO delay chain TRNG_TEST.PATTFRO =1 and/or the selected FRO error detection circuit TRNG_TEST.PATTDET =1. This field is rotated right over one bit, once every sample period, when either of these control bits is 1. Therefore, bit [16] is the actual pattern bit fed into the test target.	
12:8 (R/W)	SEL	Test Select. The TRNG_TEST.SEL bit field configures the number of the FRO to be tested, the value should be in the range of 0 to 7.	
6 (R/W)	PPROC	Test Post Proc. When the TRNG_TEST.PPROC bit is set, it provides direct access to the post-processor for known-answer tests (writing input data to the TRNG_INPUT[n] registers). While this bit is set, the TRNG can continue to generate entropy in the main LFSR and any buffered random data is preserved, to be loaded into the output registers as soon as this bit is reset to 0 again. The need clock output is forced active while this bit is set. For X9.31 post-processors, it is advisable to re-seed the post-processor after running known-answer tests as the original key and V values are modified to known values. This bit is only present when post-processing is available and can only be set to 1 when the TRNG_CTL.PPROCEN bit =1 and the test run poker (TRNG_TEST.RUNPKR) bit in this register is 0.	
5 (R/W)	RUNPKR	Test Run Poker. When the TRNG_TEST.RUNPKR bit is set, it provides direct access to the inputs of the Monobit, Run and Poker Test circuits (writing input data in chunks of 32 bits to the TRNG_INPUT0 register). While this bit is 1, the TRNG is not allowed to generate entropy but any buffered random data is preserved, to be loaded into the output registers as soon as this bit is reset to 0 again. The TRNG_STAT.NEEDCLK bit is forced active while this bit is 1. The Monobit, Run and Poker Test circuits are reset to their initial states on any change of this bit.	
4 (R/W)	CONTPKR	Continue Poker. When the TRNG_TEST.CONTPKR bit is set, Monobit Test and Poker Test keep running continuously by not resetting the Monobit count (TRNG_MONOBITCNT) and poker counters (TRNG_POKER[n] register) at the end of each 20,000 bits test block. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Do not continue poker test
		1	Continue poker test

Table 42-26: TRNG_TEST Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	NOLFSRFB	No LFSR Feedback. When the TRNG_TEST.NOLFSRFB bit is set, it removes XNOR feedback from the main LFSR, converting it into a normal shift register for the XOR-ed outputs of the FROs (shifting data in on the LSB side). A 1 also forces the LFSR to sample continuously. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Keep XNOR feedback
		1	Remove XNOR feedback
2 (R/W)	PATTDET	Test Pattern Detect. When the TRNG_TEST.PATTDET bit is set, it repeatedly feeds test pattern (PATTERN) into the error detection circuit of the FRO selected by the test select (TRNG_TEST.SEL) field. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Do not repeat feed test pattern
		1	Repeat feed test pattern
1 (R/W)	PATTFRO	Test Pattern FRO. When the TRNG_TEST.PATTFRO bit is set, it repeatedly feeds test pattern (PATTERN) into the delay chain of the FRO selected by the test select (TRNG_TEST.SEL) field by forcing the corresponding FRO enable (FROEN) output LOW. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Do not repeat feed test pattern
		1	Repeat feed test pattern
0 (R/W)	ENOUT	Test Enable Out. When the TRNG_TEST.ENOUT bit is set, it enables the tst_fro_clk_out output, connecting to the FRO selected by the test select (TRNG_TEST.SEL) field. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Disable tst_fro_clk_out
		1	Enable tst_fro_clk_out

TRNG Post-Process "V" Value Registers

The TRNG_V[n] registers are used to load the V value used for post-processing (if available). These registers are write-only. Reads return the values of the other registers mapped at the same addresses.

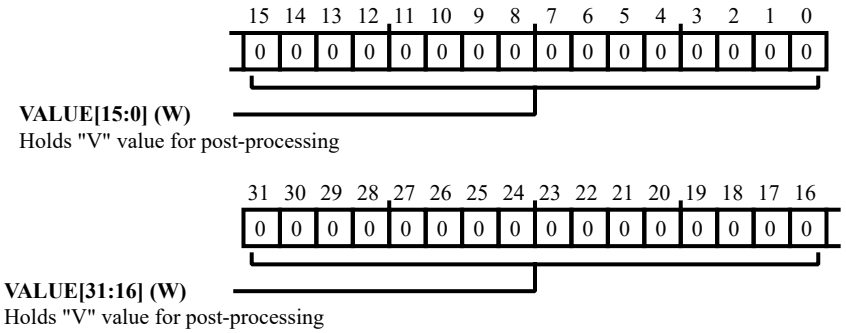


Figure 42-24: TRNG_V[n] Register Diagram

Table 42-27: TRNG_V[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Holds "V" value for post-processing. Bits of the post-processing 'V' value.

43 FIR Accelerator (FIR)

Finite Impulse Response (FIR) filters are frequently used in digital signal processing applications. The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce the instruction processing load on the core. FIR filters are used in a wide array of applications including multi-rate processing with an interpolator or decimator.

Features

This hardware module can perform FIR filters without core intervention. This allows programs to use the core to implement complex algorithms, effectively adding more bandwidth to the processor.

The FIR supports the following features:

- Fixed-point and 32-bit IEEE floating-point format
- Four SHARC+ core compatible MAC units that operate in parallel
- Rounding modes compatible with SHARC+ core MACs
- Single rate or multi-rate window processing
- Programmable rates with decimation or interpolation mode
- Up to 32 filter channels available in TDM in legacy mode
- Burst transfers on data or coefficient loads

NOTE: The FIR accelerator module has a local memory that the core cannot access during regular operation. Unlike previous SHARC processors, the FIR accelerator modules each have access to the system memory (on-chip or off-chip).

Also, unlike the previous SHARC processors, where only one of the FIR or IIR accelerators can be used at a time, the SHARC+ processor can use both accelerators simultaneously.

Clocking

The FIR accelerator runs at the speed of the core clock frequency (CCLK).

Functional Description

The ***FIR Block Diagram*** shows the 1024-TAP FIR hardware accelerator. The accelerator consists of a 1024 word coefficient memory, a 1024 deep delay line for data, and four MAC units. The accelerator runs at /CCLK frequency.

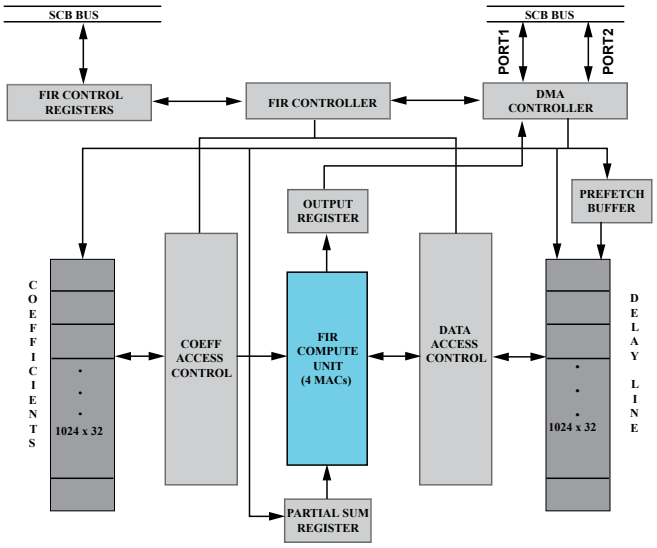


Figure 43-1: FIR Block Diagram

The FIR accelerator has the following logical sub blocks:

- A datapath unit that consists of:
  - A 1024 deep coefficient memory
  - A 1024 deep delay line for the data
  - Four 32-bit floating-point and fixed-point multiplier and adder units
  - One 32-bit prefetch buffer that operates in a pipelined fashion
  - One 32-bit buffer to hold the previous partial sum
  - One 32-bit buffer to hold the output
- Configuration registers for the number of TAPs, number of channels, filter enable, interrupt control, DMA enable, up sample or down sample control, and ratios.
- Core access interface for writing to the DMA and filter configuration registers and reading the status register.
- DMA bus interface for transferring data and coefficients to and from the accelerator.
- DMA configuration registers including chain pointer, input, output, and coefficient registers.

ADSP-2184x FIR Register List

The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce the instruction processing load on the core.

Table 43-1: ADSP-2184x FIR Register List

Name	Description
FIR_CHNPTR	FIR Chain Pointer Register
FIR_COEFCNT	FIR Coefficient Count Register
FIR_COEFIDX	FIR Coefficient Index Register
FIR_COEFMOD	FIR Coefficient Modifier Register
FIR_CTL1	FIR Global Control Register
FIR_CTL2	FIR Channel Control Register
FIR_DBG_ADDR	Debug Address Register
FIR_DBG_CTL	FIR Debug Control Register
FIR_DBG_RDDAT	FIR Debug Data Read Register
FIR_DBG_WRDAT	FIR Debug Data Write Register
FIR_DMASTAT	FIR DMA Status Register
FIR_INBASE	FIR Input Data Base Register
FIR_INCNT	FIR Input Data Count Register
FIR_INIDX	FIR Input Data Index Register
FIR_INMOD	FIR Input Data Modifier Register
FIR_MACSTAT	FIR MAC Status Register
FIR_OUTBASE	FIR Output Data Base Register
FIR_OUTCNT	FIR Output Data Count Register
FIR_OUTIDX	FIR Output Data Index Register
FIR_OUTMOD	FIR Output Data Modifier Register
FIR_SCTL1	Software Control Register 1
FIR_SCTL2	Software Control Register 2
FIR_SGCTL	Secondary Global Control Register

ADSP-2184x FIR Interrupt List

Table 43-2: ADSP-2184x FIR Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
108	FIR0_DMA	FIR0 Core 1 DMA	Edge	
109	FIR0_STAT	FIR0 Core 1 Status	Edge	
110	FIR1_DMA	FIR1 Core 1 DMA	Edge	

Table 43-2: ADSP-2184x FIR Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
111	FIR1_STAT	FIR1 Core 1 Status	Edge	
112	FIR0_BUS_ERR	FIR0 Core 1 FIR Bus Error	Level	
113	FIR1_BUS_ERR	FIR1 Core 1 FIR Bus Error	Level	

ADSP-2184x FIR Trigger List

Table 43-3: ADSP-2184x FIR Trigger List Generators

Trigger ID	Name	Description	Sensitivity
53	FIR0_DMA	FIR0 Core 1 DMA	Edge
54	FIR1_DMA	FIR1 Core 1 DMA	Edge

Table 43-4: ADSP-2184x FIR Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
98	FIR0_TRGI	FIR0 Core 1 FIR Wait on Trigger Input	Pulse
99	FIR1_TRGI	FIR1 Core 1 FIR Wait on Trigger Input	Pulse

Compute Block

The MAC unit, shown in the ***FIR MAC Unit*** figure, has four multiply accumulators. The accumulators operate simultaneously on a single filter as described below.

- The MAC unit operates on the data and coefficient fetched from the data and coefficient RAMs
- Each MAC can perform 32-bit floating-point or 32-bit fixed-point MAC operations
- Floating-point format is IEEE-compliant
- Multiply and accumulation operation (addition) are pipelined
- A 32-bit floating-point MAC operation generates 32-bit multiply results
- A 32-bit fixed-point operation generates 80-bit results (64-bit result + 16 guard bits)

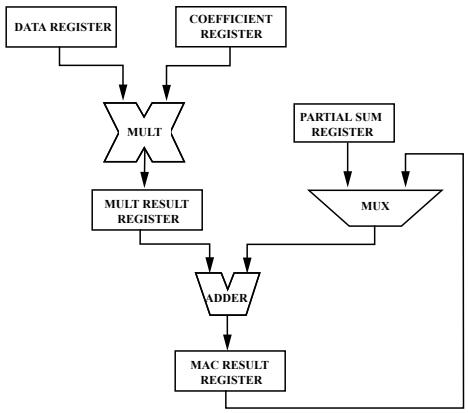


Figure 43-2: FIR MAC Unit

Partial Sum Register

The partial sum register is useful for Floating-point Multi-Iteration mode. For a particular channel, the intermediate MAC result is written to the output buffer of the system memory (on-chip or off-chip). If the same channel is requested again, the partial result register is updated with the intermediate MAC result through DMA from the output buffer of the system memory. The result is added to the current MAC result after each iteration. This process repeats until all iterations complete (the entire soft filter length is processed).

Delay Line Memory

The accelerator has a 1024 TAP delay line to hold the data locally. The DMA controller fetches the data from system memory and loads it into the delay line. Four read accesses can be made to the delay line simultaneously.

Coefficient Memory

The accelerator has a 1024 deep coefficient memory to store the coefficients. The DMA controller loads the coefficients from system memory into coefficient memory. Four coefficients can be fetched from the coefficient memory simultaneously. If the soft filter length is more than 1024, processing happens in multi-iteration mode.

Prefetch Data Buffer

The prefetch data buffer enables pipeline operation. One data sample is prefetched when the compute unit is operating on the delay line corresponding to the current sample. The data prefetched in this buffer is later used to update the delay line for the next sample. This operation happens in parallel again when the compute unit is not accessing the delay line. In other words, it happens when the compute unit is adding the output from the four MACs and the partial sum register.

Table 43-5: Pipeline Operation for Window Size = 1

Cycles	1	2	3	4	5	6
Output DMA			N	N1	N2	N3
Compute		N	N1	N2	N3	

Table 43-5: Pipeline Operation for Window Size = 1 (Continued)

Cycles	1	2	3	4	5	6
Input DMA	N	prefetch N1	prefetch N2	prefetch N3		

Processing Output

The accelerator uses all four MACs simultaneously to calculate one output sample as shown in the ***Multi-Iteration Filtering Flow*** figure and the following procedure.

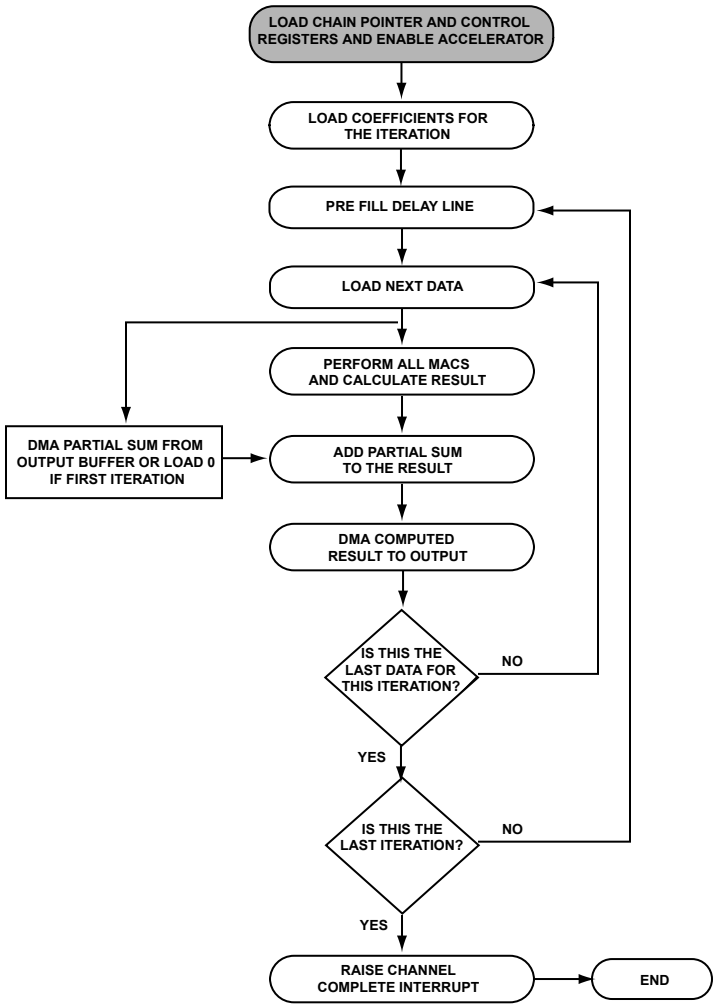


Figure 43-3: Multi-Iteration Filtering Flow

1. The accelerator fetches four input data from the delay line and four corresponding coefficients from the coefficient memory and feeds them to the MAC units for multiply and accumulation operations.
2. The accelerator repeats the procedure with the next four input data and coefficients until all the TAPs complete. For example, this procedure happens N/4 times for an N TAP filter.

3. When all the TAPs are complete, the accelerator adds the four MAC outputs together to the previous partial sum (if any) to calculate the final result.
4. Finally, that output sample is stored back in system memory.

System Memory Storage

The following sections describe the storage format for the accelerator.

CAUTION: Store all data in memory aligned to the word address boundaries. Any other programmed addresses do not flag an error.

Coefficients and Input Buffer Storage

For any N TAP filter with coefficients:

```
C[i] i = 0,1,
...
N - 1
```

store the coefficients in system memory buffer in the order:

```
C[N - 1], C[N - 2]
...
C[1],C[0]
```

and CI should point to `C(N - 1)`.

Single Rate Input Filtering

The total size of the input buffer must at least be equal to `N - 1 + W`. If the input buffer that needs to be processed is:

```
 x[n],x[n+1],x[n+2]
 ...
 x[n+W-1]
```

store it in the memory as

```
x[n-(N-1)], x[n-(N-2)]
...
x[n-1], x[n], x[n+1]
...
x[n+W-1]
```

and the `FIR_INIDX` register value should point to `x[n - (N - 1)]`

Decimation

Assuming M = decimation ratio, the total size of the input buffer should at least be equal to `N - 1 + W x M`. If the input buffer that needs to be processed is:

```
x[n],x[n+1],x[n+2]....x[n+WxM-1],
```

store it in the memory as:

```
x[n-(N-1)], x[n-(N-2)]....x[n-1],
x[n], x[n+1]....x[n+WxM-1]
```

and the FIR_INIDX register value should point to x[n-(N-1)].

Interpolation

Assuming L= interpolation ratio, the total size of the input buffer should be at least equal to:

Ceil ((N-1)/L)+W/L.

If the input buffer that needs to be processed is:

```
x[n], x[n+1], x[n+2]....x[n+W/L-1],
and K = Ceil((N-1)/L)
```

store it in the memory as:

```
x[n-k], x[n-(K-1)], x[n-(K-2)]....x[n-1],
x[n], x[n+1].... x[n+W/L-1]
```

and the FIR_INIDX register value should point to x[n-K].

Operating Modes

The FIR core performs a sum-of-products operation to compute the convolution sum. It supports single-rate, decimation, and interpolation functions.

Single Rate Processing

In a single-rate filter, the output result rate is equal to the input sample rate. The filter output Y(n) is computed according to following equation where N is the number of filter coefficients: c(k) k = 0 to N – 1 are the filter coefficients and x(n) represents the input time-series.

$$Y(n)= \sum_{k=0}^{N-1} c(k) \cdot x(n-k)$$

Figure 43-4: Filter Output Calculation

Single Iteration

Results are computed in a single iteration when the soft filter length is less than or equal to 1024.

Floating-point Multi-Iteration

Results are computed in multiple iterations when the soft filter length is greater than 1024 (for example, 2048 TAPs on a 1024 hard filter length). In this mode, the accelerator implements two iterations of 1024 TAPs.

NOTE: If the soft filter length is not a multiple of the hard filter length, the accelerator iterates until the soft filter length is satisfied.

Example: 550 taps on a 256 tap filter. In this example, the FIR accelerator implements two iterations of 256 taps and one iteration of 38 taps.

NOTE: Multi-iteration mode is not supported in fixed-point format.

Window Processing

In window-based mode, multiple output samples (up to 1024) equal to the window size of that channel are calculated. After these calculations are complete, the accelerator begins processing the next channel. A configurable window size parameter is provided to specify the length of the window.

To select sample-based processing mode, configure the window size to 1. In this mode, one sample from a particular channel is processed through all the taps of that channel and the final output sample is calculated.

Multi-Rate Processing

Multi-rate filters change the sampling rate of a signal—they convert the input samples of a signal to a different set of data that represents the same signal sampled at a different rate.

Decimation

A decimation filter provides a single output result for every M input samples, where M is the decimation ratio. The output rate is 1/M th of the input rate. The filter implementation exploits the low output sample rate by not starting a computation until a new set of M input samples is available.

In this mode, after low-pass filtering (for anti-aliasing), FIR logic discards the ratio – 1 samples of output data. For performance optimization, FIR logic skips the computation of output samples, which are discarded.

The input buffer size for decimation filters is N – 1 + (W × M) where:

- N is the number of taps
- W is the window size
- M is the decimation ratio

The window size (`FIR_CTL2.WINDOW` bits) must be programmed with the number of output samples.

To start this mode, programs set the `FIR_CTL2.RATIO` and `FIR_CTL2.UPSAMP` bits (along with normal filter setting). Also, the `FIR_CTL2.TAPLEN` bit field value should be greater than or equal to the `FIR_CTL2.RATIO` bit field value for the decimation filter.

Interpolation

An interpolation filter provides L output results for each new input sample, where L is the interpolation ratio. The output rate is L times the input rate.

In this mode, according to the ratio specified in configuration register, FIR logic inserts L – 1 zeros between any two input samples (upsampling). It then performs the interpolation (through the FIR filter).

Both upsampling and downsampling do not support multi-iteration mode. Therefore, the filtering operation only happens on up to 1024 TAPs and the ratio of up and downsampling can only be an integer value.

In an interpolation filter, FIR logic inserts L – 1 zeros between each sample. The program must ensure that these zeros fully shift out of the delay line before moving on to the next channel. This operation puts a restriction on window size in terms of L – *the sample ratio* as showing in the expression:

WINDOWSIZE = *n* × *SAMPLERATIO*, where *n* is the number of input samples.

The input buffer size is smallest integer greater than or equal to (N – 1 + W) L for interpolation filters where:

- N is the number of taps
- W is the window size
- L is the interpolation ratio

To start the mode, programs configure the `FIR_CTL2.RATIO` and `FIR_CTL2.UPSAMP` bits (along with filter settings).

Floating-Point Data Format

The FIR accelerator treats data and coefficients in 32-bit floating-point format as the default functional mode.

Fixed-Point Data Format

In fixed-point mode, the 32-bit input data or coefficient is treated as fixed-point. A 32-bit fixed-point MAC operation generates an 80-bit result. Fixed-point data or coefficients can be unsigned integer, unsigned fractional and signed integer.

NOTE: In fixed-point mode, the entire 80-bit result register is always written back in bursts of 3 × 32 bits. The first word is the LSW, the second word is the MSW, and the third word is a 16-bit overflow. The remaining 16 bits are padded with zeros. Therefore, for fixed-point mode: WINDOWSIZE = WINDOWSIZE × 3.

If the signed fractional format is used, the output must be scaled by 2. The MAC does not right shift to remove the redundant sign bit. A final routine must decimate the output buffer to the desired samples.

Multi-iteration mode is not supported in this format. Therefore, the maximum TAP length is 1024.

Auto Configuration Mode (ACM)

The accelerator can be operated in legacy mode or Auto Configuration Mode (ACM), controlled by the `FIR_CTL1.ACM` bit. The default functional mode is legacy mode. The accelerator mode can only be changed when the accelerator is disabled.

The ACM provides the following additional features:

- Halt and queuing

The core may pause the current Transfer Control Block (TCB) chain being processed by setting the FIR_CTL1.HALT bit. The accelerator acknowledges the core by setting the FIR_DMASTAT.HALT_STAT bit. The core can appropriately take action to submit/insert new TCB's. After the core acts, the accelerator processing can be resumed by clearing the FIR_CTL1.HALT bit. Before halting the accelerator, if the initial TCB chain is processed, the accelerator comes to the idle state. In this case, the accelerator has to be disabled and enabled by toggling the FIR_CTL1.EN bit and the FIR_CTL1.HALT bit must be cleared to resume processing.

- No channel number limitation

  Unlike legacy mode, there is no fixed channel number limitation and, therefore, the accelerator ignores the value programmed in the FIR_CTL1.CH field. The application can queue an unlimited number of channels/TCBs dynamically and the accelerator keeps processing the TCBs until the chain pointer becomes null.

- Selective interrupt

  The core can enable/mask interrupt generation for each channel using the FIR_CTL2.IMASK bit. If the bit is cleared, an interrupt is generated after completion of the channel.

- Selective controller / target trigger

  The core can enable/mask trigger generation by the accelerator after the end of processing of each channel using the FIR_CTL2.TMASK bit. The accelerator can also wait for a trigger after loading the TCB and coefficients and before processing a channel for which the FIR_CTL2.TWAIT bit is set.

In addition to the above features, there are three additional fields as part of the TCB FIR_SCTL1, FIR_SCTL2, and FIR_SGCTL registers. The FIR_SGCTL register can be used to change the FIR_CTL1 parameters such as rounding mode and fixed-point mode for each channel. The FIR_SCTL1 and FIR_SCTL2 registers can be used as general-purpose registers.

Data Transfer

The FIR filter works exclusively through DMA.

Chain Assignment

The structure of a TCB is conceptually the same as a traditional linked list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB can point to itself to continuously re-run the same DMA. The FIR accelerator reads each word of the TCB and loads it into the corresponding register. The end of the chain (no further TCBs are loaded) is indicated by a TCB with a chain pointer register value of zero.

The FIR accelerator DMA supports circular buffer chained DMA. The FIR accelerator does not support circular buffering for the coefficient buffer.

Table 43-6: TCBs for Chained DMA in Legacy Mode

Address	Register
TCB	FIR_CHNPTR
TCB + 0x1	FIR_COEFCNT
TCB + 0x2	FIR_COEFMOD
TCB + 0x3	FIR_COEFIDX
TCB + 0x4	FIR_OUTBASE
TCB + 0x5	FIR_OUTCNT
TCB + 0x6	FIR_OUTMOD
TCB + 0x7	FIR_OUTIDX
TCB + 0x8	FIR_INBASE
TCB + 0x9	FIR_INCNT
TCB + 0xA	FIR_INMOD
TCB + 0xB	FIR_INIDX
TCB + 0xC	FIR_CTL2

Table 43-7: TCBs for Chained DMA in Auto Configuration Mode

Address	Register
TCB	FIR_CHNPTR
TCB + 0x1	FIR_SCTL1
TCB + 0x2	FIR_SCTL2
TCB + 0x3	FIR_SGCTL
TCB + 0x4	FIR_COEFCNT
TCB + 0x5	FIR_COEFMOD
TCB + 0x6	FIR_COEFIDX
TCB + 0x7	FIR_OUTBASE
TCB + 0x8	FIR_OUTCNT
TCB + 0x9	FIR_OUTMOD
TCB + 0xA	FIR_OUTIDX
TCB + 0xB	FIR_INBASE
TCB + 0xC	FIR_INCNT
TCB + 0xD	FIR_INMOD
TCB + 0xE	FIR_INIDX
TCB + 0xF	FIR_CTL2

The FIR_COEFCNT register is loaded with the values in the FIR_COEFCNT TCB field and is decremented from that value onwards. However, coefficient loading continues until the number of coefficients, equal to the tap length, are read. This condition is true even if the FIR_COEFCNT register reaches zero as in the case of a tap length = 10, and the FIR_COEFCNT bit field in the TCB is initialized to 0. The value in the FIR_COEFCNT register is –10 after all coefficients are loaded.

NOTE: Initialize the FIR_CHNPTR register to TCB + 12 in legacy mode and TCB + 15 in ACM mode.

DMA Access

The FIR accelerator has two DMA channels (accelerator input and output) to connect to the system memory. The DMA controller fetches the data and coefficients from memory and stores the result.

Burst Access Support

Burst access enhances the throughput of the DMA channel and reduces the overall load on the system fabric. Burst support is provided for TCB, data, and coefficient loads on the DMA channel. The FIR module supports burst transfers of size SINGLE, INCR4 INCR8 and INCR16. The DMA access are automatically split into these four types of burst by the FIRA accelerator for optimum performance. For a TCB load, a burst access of length 16 is issued which reduces the TCB load time.

An additional bit (FIR_CTL1.BURSTEN) enables the burst feature. Burst transfers are always of size SINGLE when the modifier is other than 1. In cases of burst transfers around the circular buffer boundary, the design ensures that the burst does not cross the buffer boundary.

Data and Coefficient Load in Parallel

To reduce the overall loading time of data and coefficients, the operation occurs over separate channels. After the TCB load, both data and coefficients are loaded in parallel over two DMA channels (CH0 and CH1). If there is no conflict in the system when accessing the data and coefficients, the loading time is significantly improved over when loaded sequentially using a single channel. This mode is enabled by default and can be disabled by setting the FIR_CTL1.DCP_DIS bit. When disabled, coefficient load and data load occur sequentially over a single channel (CH0).

Prefetch Buffer for DMA Channels

An access latency exists in every burst transfer for the address issued for the first data to arrive. To reduce the latency, both channels of the FIR contain a prefetch buffer. When one burst access is in-progress, the FIR prefetch buffer receives the next burst, which reduces the latency for consecutive accesses. The FIR prefetch buffer is disabled by default. Setting the FIR_CTL1.PFB_EN bit enables the prefetch buffer.

Accelerator TCB

The location of the DMA parameters for the next sequence comes from the chain pointer register that points to the next set of DMA parameters stored in the internal memory of the processor. In chained DMA operations, the

accelerator automatically initializes and starts another DMA transfer when the current DMA transfer is complete. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Pointer DMA

The DMA controller supports circular buffer chain pointer DMA. One TCB must be configured for each channel. The DMA controller contains the following:

- A control register value to configure the filter parameters (such as filter tap length, window size, sample rate conversion settings) for each channel. In ACM mode, additional parameters such as interrupt mask, trigger mask, and trigger wait are also available.
- Software control register values in ACM mode for each channel.
- Secondary control register value to configure rounding mode, fixed-point mode, and two's complement for each channel.
- DMA parameter register values for the input data (delay line).
- DMA parameter register values for coefficient load.
- DMA parameter register values for output data.

Intermediate results in multi-iteration mode are saved in the output buffer.

As shown in the ***Circular Buffer Addressing*** figure, the accelerator loads the TCB into its internal registers and uses these values to fetch coefficients and data and to store results. After processing a window of data for any channel, the accelerator writes back the appropriate values to the `FIR_INIDX` and `FIR_OUTIDX` bit fields of the TCB in memory. Then, data processing can begin from where it left off during the next time slot of that channel.

The write-back value for input buffer is:

- `FIR_INIDX` + W for single rate filtering
- `FIR_INIDX` + W × M for decimation (M = decimation ratio)
- `FIR_INIDX` + W/L for interpolation (L = interpolation ratio)
- The write-back value for output buffer in floating point mode is: `FIR_OUTIDX` + W
- The write-back value for output buffer in fixed-point mode is: `FIR_OUTIDX` + 3 × W

NOTE: The `FIR_CTL2` register is part of the FIR TCB. This configuration allows programming individual FIR channels with different control attributes.

The above index updates are valid only for legacy mode. In ACM mode, the `FIR_INIDX` and `FIR_OUTIDX` bit fields of the TCB in memory are updated to 0x00000000 and 0xFFFFFFFF by the accelerator after processing a window of data.

In ACM mode, when the `FIR_CTL1.SMQ_LIUPS_EN` bit is set, the accelerator updates the `FIR_INIDX` and `FIR_OUTIDX` bit fields of the TCB in memory after processing a window of data

and according to the circular buffer scheme. When the FIR_CTL1.SMQ_LIUPS_EN bit is cleared, the accelerator updates the FIR_INIDX and FIR_OUTIDX bit fields of the TCB in memory to 0x00000000 and 0xFFFFFFFF after processing a window of data. The FIR_CTL1.SMQ_LIUPS_EN bit is only valid in ACM mode.

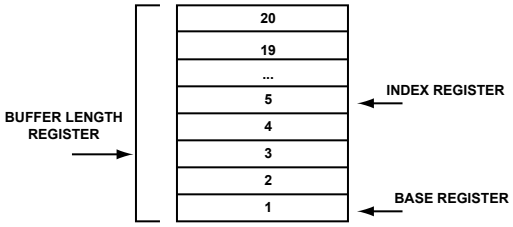


Figure 43-5: Circular Buffer Addressing

Programming Model

The following sections provide general programming information for the FIR accelerator.

Legacy Mode

The ***Wait for Core Intervention ≥ Idle (if auto channel iterate bit = 0)*** figure shows the diagram for multichannel filtering. Multiple channels are processed in a time division multiplexed (TDM) format. After completing all the channels, the accelerator can either repeat the slots or wait for core intervention.

For multichannel filtering, use the following steps.

1. Program the number of channels using the FIR_CTL1.CH bits.
2. Configure the TCBs in system memory with one channel's TCB pointing to the next channel's TCB.
3. Write the first TCB value into the FIR_CHNPTR register and enable the accelerator.
    a. The accelerator fetches the first channel's TCB and, using it as pointer, prefills the delay line and coefficient memory and loads the FIR_CTL2 register to configure the filter parameters corresponding to that channel.
    b. The accelerator then calculates output samples corresponding to one window and stores the data back in internal memory.
    c. At the end of the window the accelerator updates the FIR_INIDX and FIR_OUTIDX registers in the TCB of system memory and moves to the next channel.
    d. When all the channels are finished and the auto channel iterate bit (FIR_CTL1.CAI) =1, the accelerator processes the first channel again and iterates through the channels. If the FIR_CTL1.CAI bit =0, the accelerator waits for core intervention.

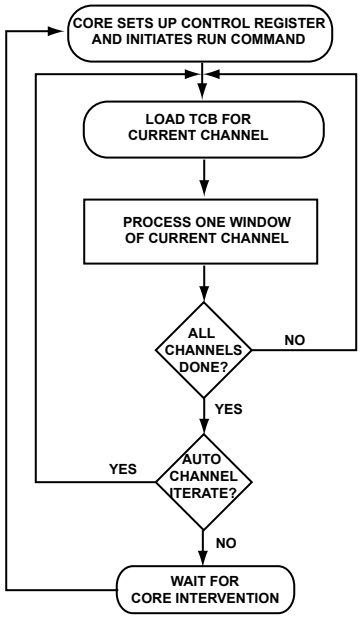


Figure 43-6: Wait for Core Intervention ≥ Idle (if auto channel iterate bit = 0)

NOTE: All the addresses programmed in the TCB correspond to 32-bit address boundaries and should not contain the lower 2 bits (assumed as zeros).

Auto Configuration Mode (ACM)

The figure shows multi-channel filtering in ACM. Multiple channels are processed in a time multiplexed format (TDM).

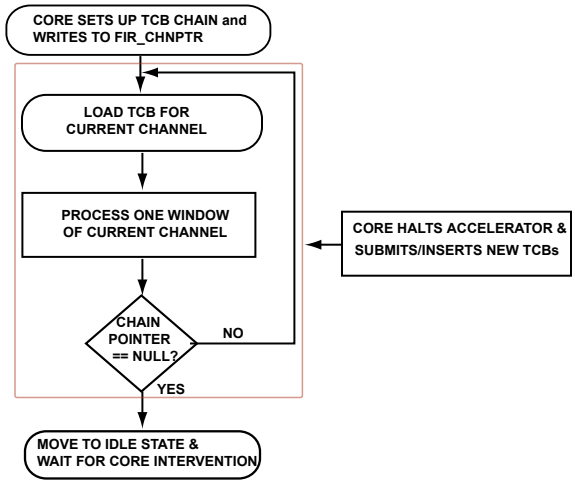


Figure 43-7: Multi-Channel Filtering in Auto Configuration Mode

For multi-channel filtering:

1. Configure the Transfer Control Blocks (TCBs) in system memory with one channel's TCB pointing to the next channel's TCB. There is no limit on the number of channels to be configured.

2. Write the first TCB value to the FIR_CHNPTR register and enable the accelerator.

   The accelerator fetches the first channel's TCB. Using it as pointer, it prefills the delay line and coefficient memory and loads the FIR_CTL2 and FIR_SGCTL registers to configure the filter parameters corresponding to that channel.

   When the FIR_CTL2.TWAIT bit is set, the accelerator waits for a input trigger to start the window processing for the channel.

   The accelerator then calculates output samples corresponding to one window and stores the data back in internal memory. When the FIR_CTL2.TMASK bit is cleared the accelerator sends an output trigger after completion of processing of the channel.

   When the FIR_CTL2.IMASK bit is cleared, the accelerator interrupts the core after completion of processing of a particular channel.

   At the end of the window, the accelerator updates the FIR_INIDX and FIR_OUTIDX registers to 0x00000000 and 0xFFFFFFFF in the TCB of system memory and moves to the next channel.

3. At any instant, as required, the core halts the accelerator. It sets the FIR_CTL1.HALT bit and appropriately takes action to submit or insert new TCBs and clears the FIR_CTL1.HALT bit to resume channel processing.

   If the FIR_CHNPTR register is zero (last channel is being processed or channel processing is complete) after halting the accelerator, the FIR_CHNPTR register is appropriately written before resuming the accelerator channel processing.

   When the accelerator is idle after halting, FIR_CTL1.EN bit is toggled to disable and re-enable the accelerator and the FIR_CTL1.HALT bit is cleared, and the accelerator resumes channel processing.

4. The accelerator continues processing until all the channels are complete. Repeat *Step 3* (if required) to submit/insert new channels.

NOTE: Channel auto iterate (FIR_CTL1.CAI) is not supported in ACM.

Debug Mode

The next sections show the steps required for reading and writing local memory in debug mode.

Write to Local Memory

1. Clear the FIR_CTL1.DMAEN bit.
2. Set the FIR_DBG_CTL.EN, FIR_DBG_CTL.MEM, and FIR_DBG_CTL.HLD bits.
3. Set the FIR_DBG_CTL.ADRINC bit for address auto increment.
4. Write the start address to the FIR_DBG_ADDR register.

   NOTE: If bit 11 in the FIR_DBG_ADDR register is set, coefficient memory is selected.

5. Write data to the FIR_DBG_WRDAT register.

Read from Local Memory

1. Clear the FIR_CTL1.DMAEN bit.
2. Set the FIR_DBG_CTL.EN, FIR_DBG_CTL.MEM, and FIR_DBG_CTL.HLD bits.
3. Set the FIR_DBG_CTL.ADRINC bit for address auto increment.
4. Write the start address to the FIR_DBG_ADDR register.

   NOTE: If bit 11 in the FIR_DBG_ADDR register is set, coefficient memory is selected.

5. Read data from the FIR_DBG_RDDAT register.

Single-Step Mode

Single-step mode can be used for debug purposes. An extra debug register is used in this mode.

1. Enable stop DMA during breakpoint hit in the emulator settings.
2. Clear the FIR_DBG_CTL.HLD bit and enable the FIR_DBG_CTL.EN and FIR_DBG_CTL.RUN bits.
3. Program the FIR module according to the application.
4. In single-step mode, each iteration is updated in the emulator session.

Computing FIR Output, Tap Length Greater than 4096

With little core intervention, the FIR accelerator can also be used to calculate output for a tap length greater than 4096 taps. The section demonstrates the calculation with an example of 8192 taps.

1. Divide the transfer function of an 8192 FIR filter into two 4096 FIR filters:

   $H(Z) = b_0 + b_1Z^{-1} + b_2Z^{-2} + ...b_{4095}Z^{-4095} + b_{4096}Z^{-4096}b_{4097}Z^{-4097} + ...b_{8191}Z^{-8191}$

   $= b_0 + b_1Z^{-1} + b_2Z^{-2} + ...b_{4095}Z^{-4095} + Z^{-4096}(b_{4096} + b_{4097} + ...b_{8191}Z^{-4096})$

2. Divide the filter coefficients of an 8192 tap filter among two 4096 tap FIR filters:

   Filter 1

   Coefficients = b0, b1, b2,……, b4095

   Input data = x[n], x [n – 1],…….x[n – 4095]

   Filter 2

   Coefficients = b4096, b4097,……..,b8191

   Input data = x[n – 4096], x[n – 4097],……x[n – 8191]

The accelerator can be used in two-channel mode where:

- channel 1 operates on x[n]…x[n – 4095] input data with the filter coefficients of filter 1 and

- channel 2 operates on x[n – 4096]…x[n – 8191] with the filter coefficients of filter 2.

Once both the channels are processed, add the partial sum output of both the channels to get the final output. Implement this approach (tap length = TAPS = 8192, window size = WINDOW) using the following programming steps.

1. Create a circular input data buffer in system memory (IBUF). The buffer must be large enough to avoid overwriting data before the accelerator processes it. Ideally, the input buffer size for a channel is TAPS + WINDOW – 1.
2. Create a coefficient buffer of size TAPS (8192) (CBUF).
3. Create one output buffer of size WINDOW (OBUF) and another temporary output buffer (OBUF1) to store the partial sum.
4. Create two TCBs in system memory with first TCB chained to the second and second chained to the first in a circular manner.
    a. Point the `FIR_COEFIDX` bit field of the first TCB to the start address of the coefficient buffer (CBUF) and that of the second TCB to 4096 offset from the start of the coefficient buffer (CBUF + 4096).
    b. Point the `FIR_OUTBASE` and `FIR_OUTIDX` bit fields of the first TCB to the start address of OBUF and that of the second TCB to the start address of OBUF1.
    c. Point the `FIR_INIDX` bit field of the first TCB to the start address of IBUF and that of the second TCB to 4096 offset from the start address of IBUF.
    d. Configure the `FIR_CTL2` bit field of both the TCB for tap length = TAP/2 = 4096 and window size = WINDOW.
5. Initialize the `FIR_CHNPTR` register to point to the first TCB.
6. Program the `FIR_CTL1` register to initiate the accelerator processing by setting the `FIR_CTL1.EN` and `FIR_CTL1.DMAEN` bits and the number of channels configured as 2.
7. Wait for the FIR all channels done interrupt (`FIR_DMASTAT.ACDONE`) to occur. Inside the ISR, add the partial sum results using the core from both the output buffers (OBUF and OBUF1) to get the final output. To save memory, replace the contents of the buffer OBUF with the final output result.

Debug Features

The following sections provide information for debugging the FIR accelerator.

Local Memory Access

The contents of FIR delay line and coefficient memories are made observable for debug by setting the `FIR_DBG_CTL.EN`/`FIR_DBG_CTL.MEM` and `FIR_DBG_CTL.HLD` bits. The debug address register (`FIR_DBG_ADDR`) and two data registers are provided for debug operations. Bit 11 of the `FIR_DBG_ADDR` register selects coefficient memory when set (=1) and selects delay line memory when cleared (=0).

In the debug mode, the read data register (FIR_DBG_RDDAT) returns the contents of the memory location pointed to by the address register. Data can be written into any memory location using FIR_DBG_WRDAT register writes. If the address auto-increment bit (FIR_DBG_CTL.ADRINC) is set, the address register auto-increments on FIR_DBG_WRDAT writes and FIR_DBG_RDDAT reads. During auto-increment the FIR_DBG_ADDR register cannot cross the data memory or coefficient memory boundary.

Single-Step Mode

Programs can single step through the MAC operations and observe the memory contents after each step. The FIR_DBG_CTL.EN, FIR_DBG_CTL.HLD, and FIR_DBG_CTL.MEM bits control the FIR MAC units.

Emulation Considerations

In FIR debug mode, the DMA operations are not observable.

Interrupts

The ***FIR Interrupt Overview*** table provides the source of interrupt and service instructions for the FIR interrupts.

Table 43-8: FIR Interrupt Overview

Accelerator Mode	Default Programmable Interrupt	Sources	Masking	Service
Legacy Mode	FIR_DMA	Window Complete	N/A	ROC from FIR_DMASTAT + RTI instruction
		All channels complete		
	FIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from FIR_MACSTAT + RTI instruction
		MAC fixed point overflow		
Auto Configuration Mode	FIR_DMA	Window Complete	FIR_CTL2.IMASK	ROC from FIR_DMASTAT + RTI instruction
	FIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from FIR_MACSTAT + RTI instruction
		MAC fixed point overflow	N/A	

Sources

The FIR module drives two interrupt signals: FIR_DMA for the DMA status and FIR_STAT for the MAC status. The FIR module generates interrupts as described in the following sections.

Window Complete

This interrupt is generated at the end of each channel when all the output samples are calculated corresponding to a window and updated index values are written back.

In legacy mode, if the FIR_CTL1.CCINTR bit is set, an interrupt is generated after completion of window processing of each channel. In ACM, the interrupt generation can be selectively masked using the FIR_CTL2.IMASK bit for each channel.

All Channels Complete

This interrupt is generated when all the channels are complete or when one iteration of time slots completes. Note that the interrupt follows the access completion rule, where the interrupt is generated when all data are written back to system memory.

The all channels complete interrupt source is only applicable in legacy mode. If the FIR_CTL1.CCINTR bit is not set in legacy mode, the all channel complete interrupt is generated. In ACM mode, interrupt generation for each channel is controlled using the FIR_CTL2.IMASK bit.

NOTE: The FIR_CTL1.CCINTR bit is valid only in legacy mode. In ACM mode, interrupt generation for each channel is controlled using the FIR_CTL2.IMASK bit.

MAC Status

A MAC status interrupt is generated under the following conditions and is reflected in the FIR_MACSTAT register.

- Multiplier result zero – Set if multiplier result is zero
- Multiplier result infinity – Set if multiplier result is infinity
- Multiply invalid – Set if multiply operation is invalid
- Adder result zero – Set if adder result is zero
- Adder result infinity – Set if adder result is infinity
- Adder invalid – Set if addition is invalid
- Adder overflow – for fixed-point operation

Service

The DMA interrupt status bits are sticky and are cleared when the DMA status register is read. When a MAC status interrupt occurs, programs can find this state by reading the MAC status register (FIR_MACSTAT). The read of the register clears the (sticky) MAC interrupt status bits.

The status interrupt sources are derived from the FIR_MACSTAT register. A status interrupt can occur due to the last set of MAC operations of a processing iteration that correspond to a particular channel. The interrupt is generated continuously and cannot be stopped, even after disabling the accelerator. The interrupt can only be stopped when another processing iteration results in a non-zero or valid multiply or add result.

ADSP-2184x FIR Register Descriptions

The FIR accelerator (FIR) contains the following registers.

Table 43-9: ADSP-2184x FIR Register List

Name	Description
FIR_CHNPTR	FIR Chain Pointer Register
FIR_COEFCNT	FIR Coefficient Count Register
FIR_COEFIDX	FIR Coefficient Index Register
FIR_COEFMOD	FIR Coefficient Modifier Register
FIR_CTL1	FIR Global Control Register
FIR_CTL2	FIR Channel Control Register
FIR_DBG_ADDR	Debug Address Register
FIR_DBG_CTL	FIR Debug Control Register
FIR_DBG_RDDAT	FIR Debug Data Read Register
FIR_DBG_WRDAT	FIR Debug Data Write Register
FIR_DMASTAT	FIR DMA Status Register
FIR_INBASE	FIR Input Data Base Register
FIR_INCNT	FIR Input Data Count Register
FIR_INIDX	FIR Input Data Index Register
FIR_INMOD	FIR Input Data Modifier Register
FIR_MACSTAT	FIR MAC Status Register
FIR_OUTBASE	FIR Output Data Base Register
FIR_OUTCNT	FIR Output Data Count Register
FIR_OUTIDX	FIR Output Data Index Register
FIR_OUTMOD	FIR Output Data Modifier Register
FIR_SCTL1	Software Control Register 1
FIR_SCTL2	Software Control Register 2
FIR_SGCTL	Secondary Global Control Register

FIR Chain Pointer Register

The FIR_CHNPTR register contains the chain pointer address.

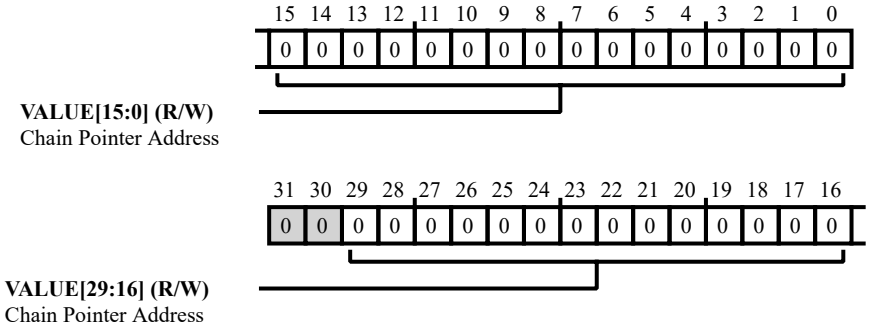


Figure 43-8: FIR_CHNPTR Register Diagram

Table 43-10: FIR_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Chain Pointer Address. The FIR_CHNPTR.VALUE bit field contains the chain pointer address.

FIR Coefficient Count Register

The FIR_COEFCNT register contains the 16-bit coefficient buffer count.

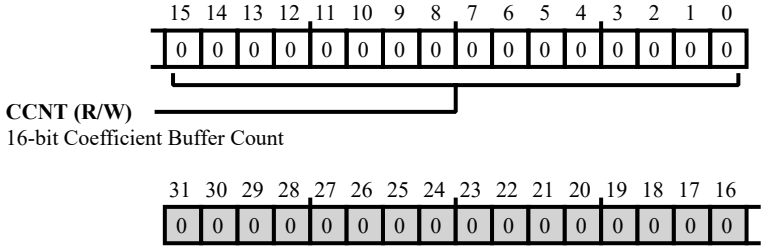


Figure 43-9: FIR_COEFCNT Register Diagram

Table 43-11: FIR_COEFCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	CCNT	16-bit Coefficient Buffer Count. The FIR_COEFCNT.CCNT bit field contains the 16-bit coefficient buffer count.

FIR Coefficient Index Register

The FIR_COEFIDX register contains the coefficient index word address with the lower two bits removed.

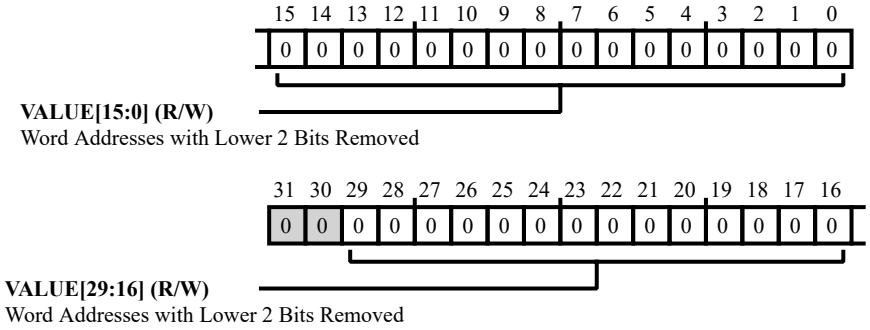


Figure 43-10: FIR_COEFIDX Register Diagram

Table 43-12: FIR_COEFIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Addresses with Lower 2 Bits Removed. The FIR_COEFIDX.VALUE bit field contains the word addresses with the lower 2 bits removed.

FIR Coefficient Modifier Register

The FIR_COEFMOD register contains the 16-bit coefficient index modifier.

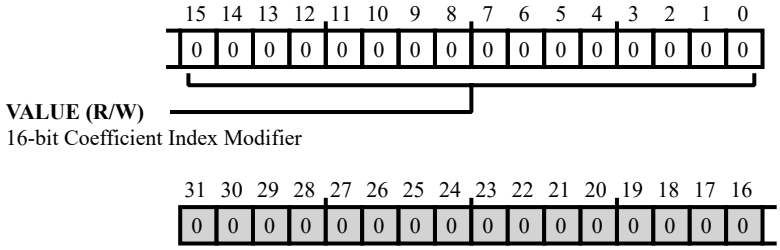


Figure 43-11: FIR_COEFMOD Register Diagram

Table 43-13: FIR_COEFMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Coefficient Index Modifier. The FIR_COEFMOD.VALUE bit field contains the 16-bit coefficient index modifier.

FIR Global Control Register

The FIR_CTL1 register is used to configure the global parameters for the accelerator. These parameters include the number of channels, channel auto iterate, DMA enable, and accelerator enable.

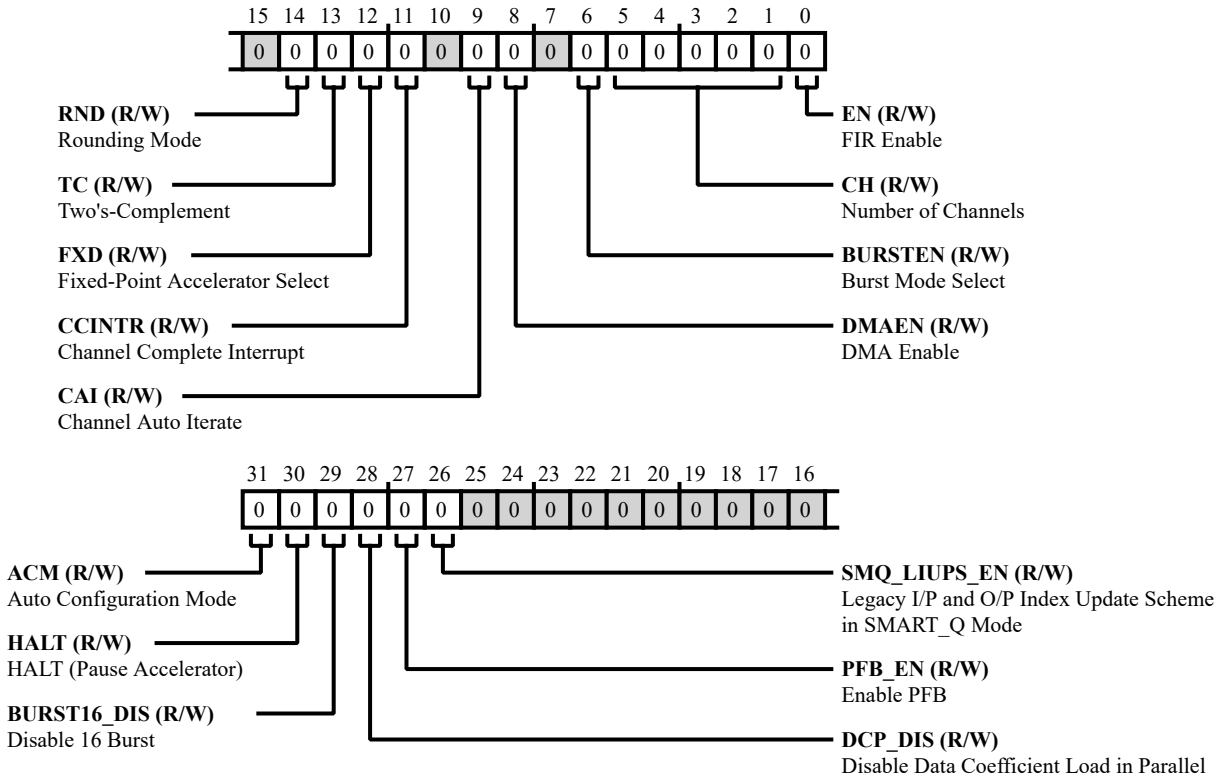


Figure 43-12: FIR_CTL1 Register Diagram

Table 43-14: FIR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ACM	Auto Configuration Mode. The FIR_CTL1.ACM bit configures the mode for loading the TCB.	
		0	Legacy Mode. Load TCB without the FIR_CTL1 register. Core intervention may be required.
		1	Auto Configuration Mode. Load TCB with the FIR_CTL1 register. Core intervention is not required.

Table 43-14: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	HALT	HALT (Pause Accelerator). This bit is only valid in Auto Configuration mode (ACM)	
		0	Release. Release accelerator for further processing of data.
		1	Pause. Pause accelerator for core to check status and modify or submit jobs.
29 (R/W)	BURST16_DIS	Disable 16 Burst. The `FIR_CTL1.BURST16_DIS` bit configures the accelerator to disable 16 burst when in burst mode.	
		0	Maximum burst of INCR16 in burst mode for DMA and TCB
		1	Maximum burst of INCR8 in burst mode for DMA and non-burst for TCB
28 (R/W)	DCP_DIS	Disable Data Coefficient Load in Parallel. The `FIR_CTL1.DCP_DIS` bit enables data and coefficient loading to occur in parallel.	
		0	Enable
		1	Disable
27 (R/W)	PFB_EN	Enable PFB. When set (=1), the `FIR_CTL1.PFB_EN` bit enables the prefetch buffer on both FIR channels.	
		0	Disable
		1	Enable
26 (R/W)	SMQ_LIUPS_EN	Legacy I/P and O/P Index Update Scheme in SMART_Q Mode. The `FIR_CTL1.SMQ_LIUPS_EN` bit configures the scheme the accelerator uses to update the input index (II) and output index (OI). This bit is only valid in ACM.	
		0	Update the II with all '0' and the OI with all 'F', respectively
		1	Update II and OI after circular buffer scheme
14 (R/W)	RND	Rounding Mode. The `FIR_CTL1.RND` bit configures the accelerator to use a rounding mode.	
		0	Round to Nearest
		1	Truncate (round towards zero)

Table 43-14: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TC	Two's-Complement. The FIR_CTL1.TC bit configures the accelerator to use either unsigned integer or signed integer	
		0	Unsigned Integer
		1	Signed Integer
12 (R/W)	FXD	Fixed-Point Accelerator Select. The FIR_CTL1.FXD bit configures the accelerator to use either 32-bit IEEE floating-point or 32-bit fixed-point.	
		0	32-bit IEEE Floating-Point
		1	32-bit Fixed Point
11 (R/W)	CCINTR	Channel Complete Interrupt. The FIR_CTL1.CCINTR bit configures the accelerator to generate an interrupt when each or all channels are done. This bit is only valid in Legacy Mode. In Auto Configuration Mode (ACM), interrupt generation for each channel is controlled using the FIR_CTL2.IMASK bit.	
		0	Interrupt is Generated Only When All Channels are Done
		1	Interrupt is Generated After Each Channel is Done
9 (R/W)	CAI	Channel Auto Iterate. The FIR_CTL1.CAI bit, if cleared, causes the TDM processing to stop (idle) once all channels are done. If set, processing moves to the first channel and continues TDM processing in a loop when all channels are done. Channel Auto Iterate is not available in Auto Configuration Mode (ACM).	
		0	TDM Processing Stops (IDLE) Once All Channels are Over
		1	Moves to First Channel and Continues TDM Processing in a Loop When All Channels are Over
8 (R/W)	DMAEN	DMA Enable. The FIR_CTL1.DMAEN bit enables and disables DMA on the FIR accelerator.	
		0	DMA Disabled
		1	DMA Enabled
6 (R/W)	BURSTEN	Burst Mode Select. When the FIR_CTL1.BURSTEN bit is set, burst mode is enabled for coefficient loads and data loads. When cleared, burst mode is disabled. By default, burst mode is disabled.	

Table 43-14: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:1 (R/W)	CH	Number of Channels. The `FIR_CTL1.CH` bit field configures the number of channels and is programmable from 0 to 31. This bit field is only valid in Legacy Mode. There is no channel number limitation in Auto Configuration Mode (ACM) as the accelerator keeps processing the TCBs until the chain pointer becomes null.	
		0	Channel 1
		1-30	Channel 2-31
		31	Channel 32
0 (R/W)	EN	FIR Enable. The `FIR_CTL1.EN` bit enables and disables the FIR accelerator.	
		0	Disable FIR
		1	Enable FIR

FIR Channel Control Register

The FIR_CTL2 register is used to configure the channel specific parameters such as filter TAP length, window size, sample rate conversion, up/down sampling and ratio. In Auto Configuration Mode(ACM), this register is also used to configure additional channel specific parameters like interrupts and triggers.

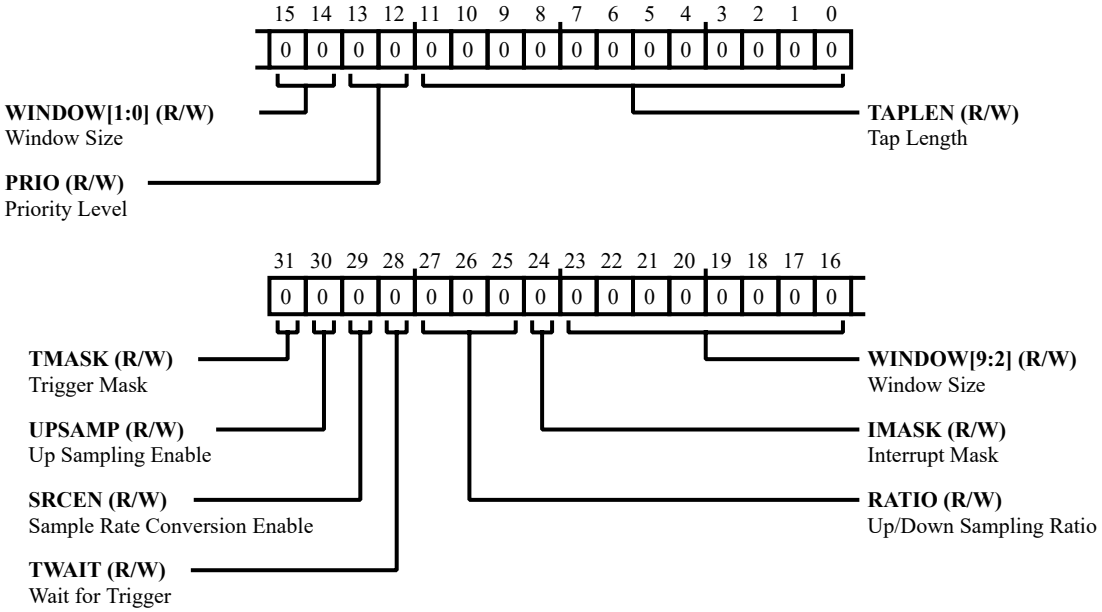


Figure 43-13: FIR_CTL2 Register Diagram

Table 43-15: FIR_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TMASK	Trigger Mask. The FIR_CTL2.TMASK bit enables trigger generation for the channel. It is only valid in Auto Configuration Mode (ACM).	
		0	Enable
		1	Mask
30 (R/W)	UPSAMP	Up Sampling Enable. The FIR_CTL2.UPSAMP bit enables up sampling.	
		0	Down Sampling
		1	Up Sampling

Table 43-15: FIR_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SRCEN	Sample Rate Conversion Enable. The `FIR_CTL2.SRCEN` bit field enables sample rate conversion.	
		0	Disabled
		1	Enabled
28 (R/W)	TWAIT	Wait for Trigger. The `FIR_CTL2.TWAIT` bit disables the wait for the trigger. It is only valid in Auto Configuration Mode (ACM).	
		0	Disable wait for trigger for the channel
		1	Enable wait for the external trigger assertion
27:25 (R/W)	RATIO	Up/Down Sampling Ratio. The `FIR_CTL2.RATIO` bit field sets the sampling ratio (`FIR_CTL2.RATIO` + 1).	
24 (R/W)	IMASK	Interrupt Mask. The `FIR_CTL2.IMASK` bit enables interrupt generation for the channel. This bit is only valid in Auto Configuration Mode (ACM).	
		0	Enable
		1	Mask
23:14 (R/W)	WINDOW	Window Size. The `FIR_CTL2.WINDOW` bit field sets the window size which specifies the number of sample/block to process (sample based processing = window size of 0).	
13:12 (R/W)	PRIO	Priority Level. The `FIR_CTL2.PRIO` bit field indicates the priority.	
		0	Level 0 (lowest)
		1	Level 1
		2	Level 2
		3	Level 3 (highest)
11:0 (R/W)	TAPLEN	Tap Length. The `FIR_CTL2.TAPLEN` bit field sets the tap length which is programmable between 0-4095 (Tap Length = `FIR_CTL2.TAPLEN` + 1).	

Debug Address Register

The FIR_DBG_ADDR register holds the debug address.

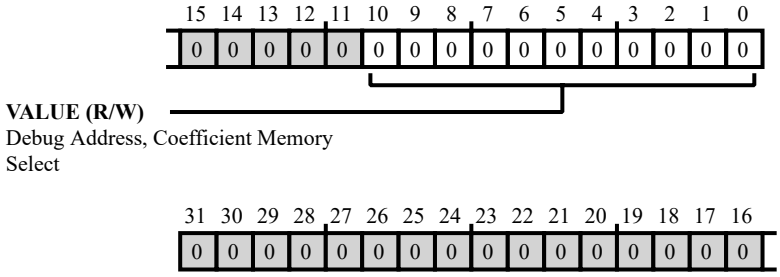


Figure 43-14: FIR_DBG_ADDR Register Diagram

Table 43-16: FIR_DBG_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Debug Address, Coefficient Memory Select. The FIR_DBG_ADDR.VALUE bit field holds the debug address (bits 0-10). Bit 11 configures whether the memory access is to coefficient memory (=0) or to delay line memory (=1).

FIR Debug Control Register

The FIR_DBG_CTL register controls debugging operations such as enabling debug mode running, hold or single stepping.

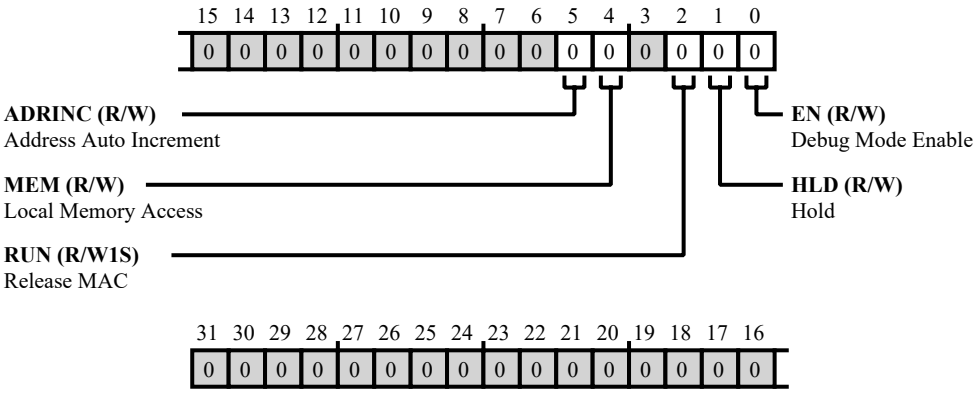


Figure 43-15: FIR_DBG_CTL Register Diagram

Table 43-17: FIR_DBG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	ADRINC	Address Auto Increment. The FIR_DBG_CTL.ADRINC bit bit allows the address register to auto-increment on FIR_DBG_WRDAT writes and FIR_DBG_RDDAT reads.	
4 (R/W)	MEM	Local Memory Access. When the FIR_DBG_CTL.MEM bit is set, the data and coefficients memory can be indirectly accessed.	
2 (R/W1S)	RUN	Release MAC. The FIR_DBG_CTL.RUN bit releases the MAC. This bit is self-clearing after one FIR clock cycle.	
1 (R/W)	HLD	Hold. The FIR_DBG_CTL.HLD bit holds or single-steps through the FIR.	
		0	Hold
		1	Single-step
0 (R/W)	EN	Debug Mode Enable. The FIR_DBG_CTL.EN bit enables debug mode. For local memory access, the FIR_CTL1 register can be cleared.	
		0	Disable Debug Mode
		1	Enable Debug Mode

FIR Debug Data Read Register

The FIR_DBG_RDDAT register hold the debug read data.

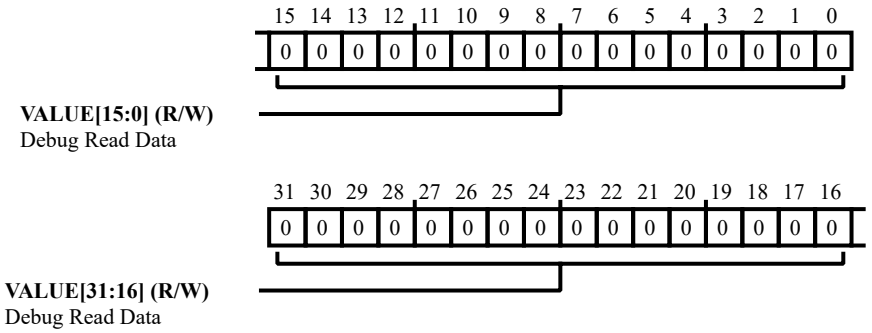


Figure 43-16: FIR_DBG_RDDAT Register Diagram

Table 43-18: FIR_DBG_RDDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Read Data. The FIR_DBG_RDDAT.VALUE bit field holds the debug read data.

FIR Debug Data Write Register

The FIR_DBG_WRDAT register holds the debug write data.

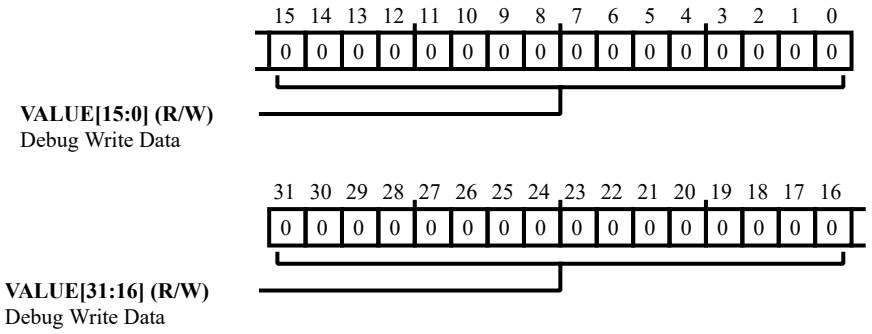


Figure 43-17: FIR_DBG_WRDAT Register Diagram

Table 43-19: FIR_DBG_WRDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Write Data. The FIR_DBG_WRDAT.VALUE bit field holds the debug write data.

FIR DMA Status Register

The FIR_DMASTAT register provides information about chain pointer loading, coefficient DMA, data preload DMA, processing in progress, window complete, all channels complete.

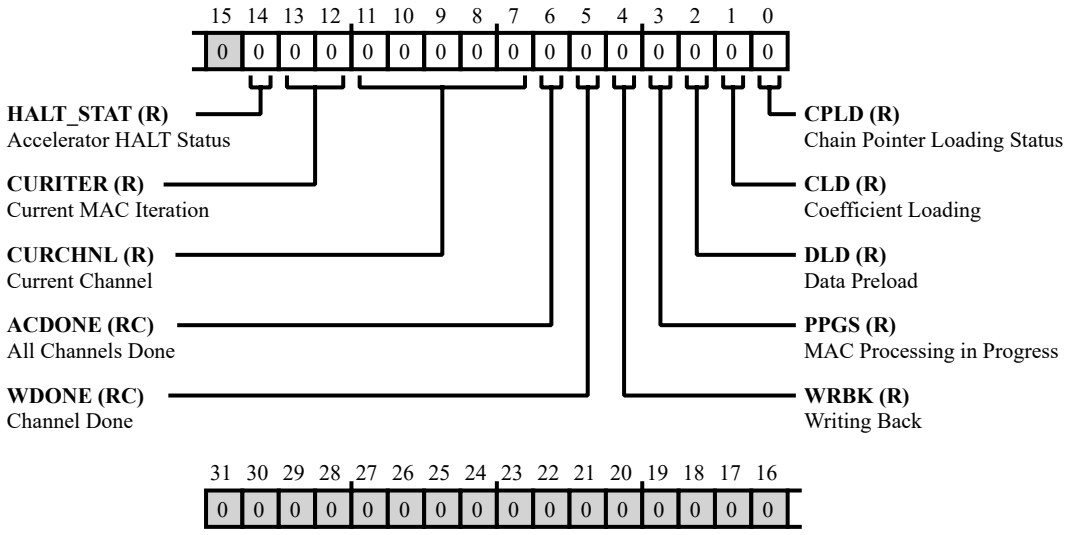


Figure 43-18: FIR_DMASTAT Register Diagram

Table 43-20: FIR_DMASTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	HALT_STAT	Accelerator HALT Status. Accelerator HALT status. 1 - Acknowledge to core that acceleration is in halt state. 0 - Not in halt state
13:12 (R/NW)	CURITER	Current MAC Iteration. The FIR_DMASTAT.CURITER bit indicates the current MAC iteration in multi-iteration mode. Zero indicates the final iteration.
11:7 (R/NW)	CURCHNL	Current Channel. The FIR_DMASTAT.CURCHNL bit indicates the channel that is being processed in the TDM slot. Zero indicates the last slot.
6 (RC/NW)	ACDONE	All Channels Done. The FIR_DMASTAT.ACDONE bit indicates the accelerator that processing all channels is complete. This is a sticky bit and is cleared on a register read. The FIR_CTL1.CCINTR bit does not affect the FIR_DMASTAT.ACDONE bit.
5 (RC/NW)	WDONE	Channel Done. The FIR_DMASTAT.WDONE bit indicates the accelerator that processing the current channel is complete. This is a sticky bit and is cleared on a register read. The FIR_CTL1.CCINTR bit does not affect the FIR_DMASTAT.WDONE bit.

Table 43-20: FIR_DMASTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/NW)	WRBK	Writing Back. The `FIR_DMASTAT.WRBK` bit indicates the accelerator is writing back the updated index registers.
3 (R/NW)	PPGS	MAC Processing in Progress. The `FIR_DMASTAT.PPGS` bit indicates MAC processing in progress.
2 (R/NW)	DLD	Data Preload. The `FIR_DMASTAT.DLD` bit indicates data preloading.
1 (R/NW)	CLD	Coefficient Loading. The `FIR_DMASTAT.CLD` bit indicates coefficient loading.
0 (R/NW)	CPLD	Chain Pointer Loading Status. The `FIR_DMASTAT.CPLD` bit indicates the state machine is in chain pointer load state.

FIR Input Data Base Register

The FIR_INBASE register contains the input word base address with the lower two bits removed.

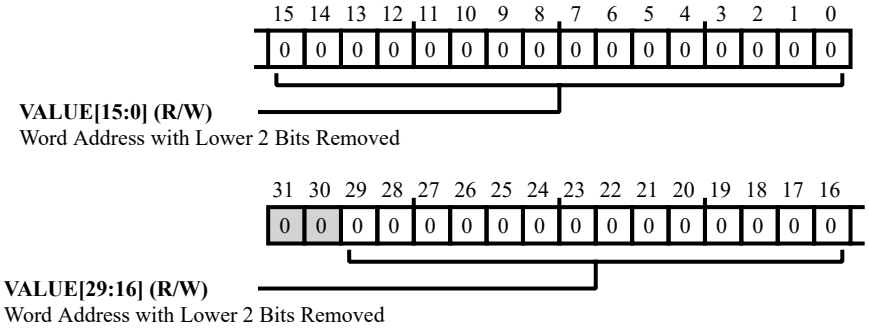


Figure 43-19: FIR_INBASE Register Diagram

Table 43-21: FIR_INBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_INBASE.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Input Data Count Register

The FIR_INCNT register contains the 16-bit input data count.

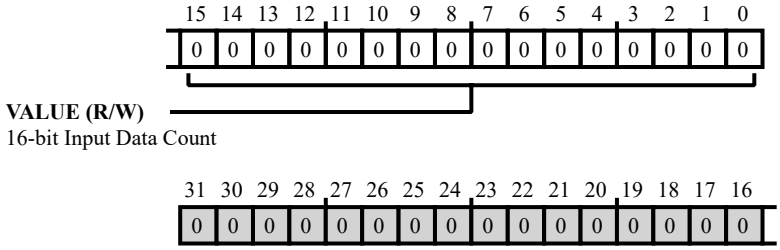


Figure 43-20: FIR_INCNT Register Diagram

Table 43-22: FIR_INCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Count. The FIR_INCNT.VALUE bit field contains the 16-bit input data count.

FIR Input Data Index Register

The FIR_INIDX register contains the input word address with the lower two bits removed.

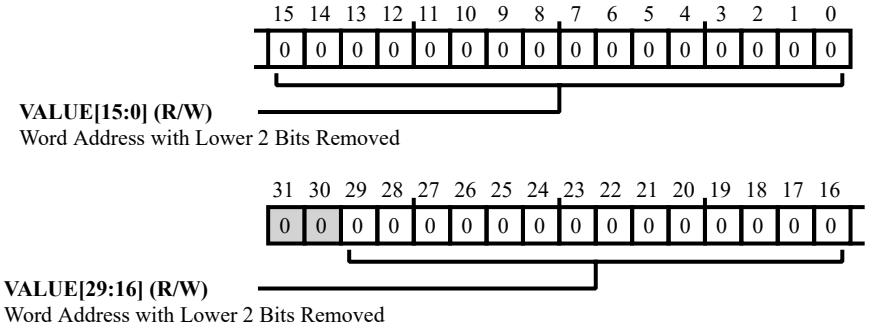


Figure 43-21: FIR_INIDX Register Diagram

Table 43-23: FIR_INIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_INIDX.VALUE bit field contains the input word address with the lower two bits removed.

FIR Input Data Modifier Register

The FIR_INMOD register contains the 16-bit input data buffer modifier.

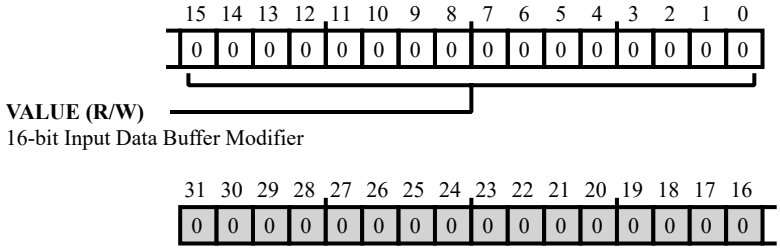


Figure 43-22: FIR_INMOD Register Diagram

Table 43-24: FIR_INMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Modifier. The FIR_INMOD.VALUE bit field contains the 16-bit input data buffer modifier.

FIR MAC Status Register

The FIR_MACSTAT register provides the status of MAC operations. The status of all four multipliers/adders are available separately for programs to poll. In fixed-point mode only, the ARIx bits are used (all other bits are reserved).

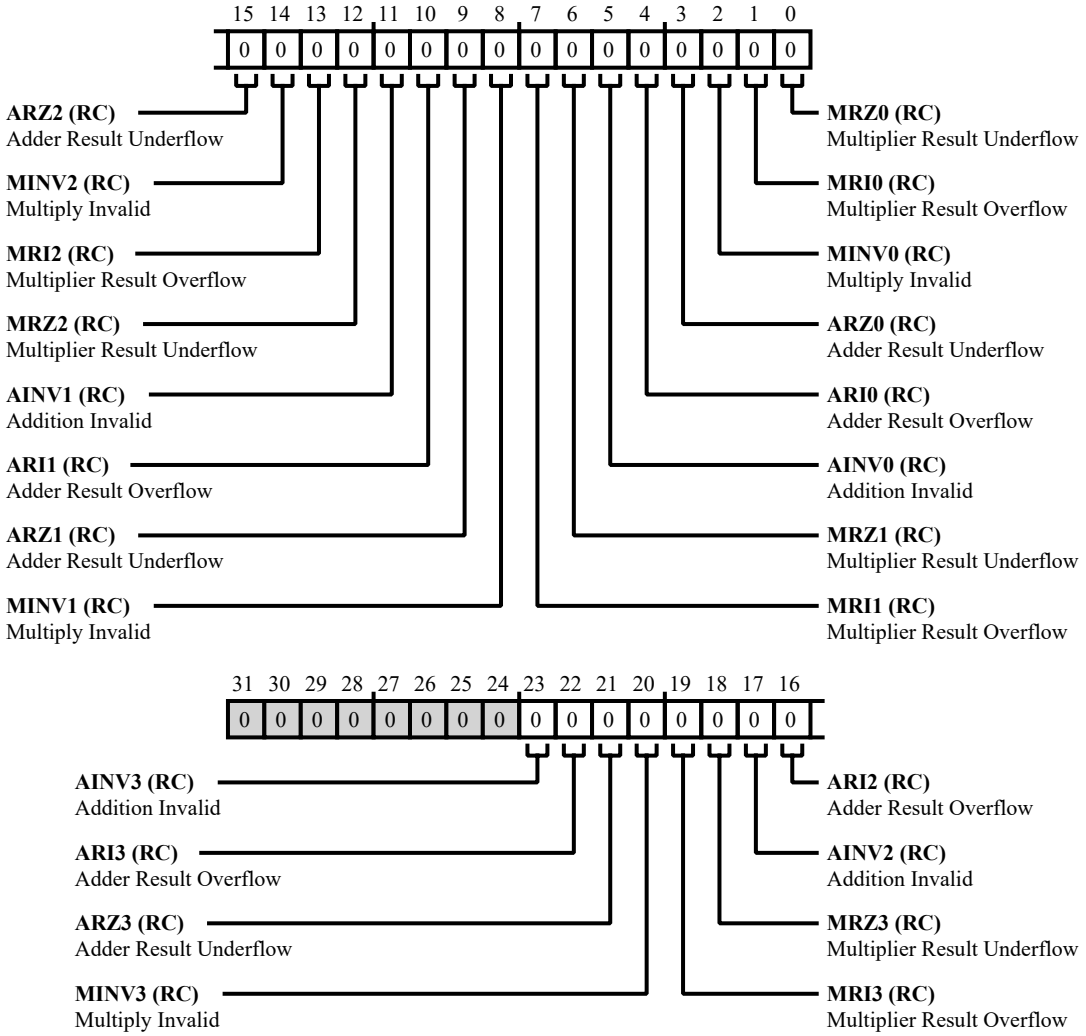


Figure 43-23: FIR_MACSTAT Register Diagram

Table 43-25: FIR_MACSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23 (RC/NW)	AINV3	Addition Invalid. The FIR_MACSTAT.AINV3 bit is set if the adder 3 addition is invalid.

Table 43-25: FIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
22 (RC/NW)	ARI3	Adder Result Overflow. The `FIR_MACSTAT.ARI3` bit is set if the adder 3 result is infinity. Indicates overflow in fixed-point mode.
21 (RC/NW)	ARZ3	Adder Result Underflow. The `FIR_MACSTAT.ARZ3` bit is set if the adder 3 result is zero.
20 (RC/NW)	MINV3	Multiply Invalid. The `FIR_MACSTAT.MINV3` bit is set if the multiplier 3 multiply operation is invalid.
19 (RC/NW)	MRI3	Multiplier Result Overflow. The `FIR_MACSTAT.MRI3` bit is set if the multiplier 3 result is infinity.
18 (RC/NW)	MRZ3	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ3` bit is set if the multiplier 3 result is zero.
17 (RC/NW)	AINV2	Addition Invalid. The `FIR_MACSTAT.AINV2` bit is set if the adder 2 addition is invalid.
16 (RC/NW)	ARI2	Adder Result Overflow. The `FIR_MACSTAT.ARI2` bit is set if the adder 2 result is infinity. Indicates overflow in fixed-point mode.
15 (RC/NW)	ARZ2	Adder Result Underflow. The `FIR_MACSTAT.ARZ2` bit is set if the adder 2 result is zero.
14 (RC/NW)	MINV2	Multiply Invalid. The `FIR_MACSTAT.MINV2` bit is set if the multiplier 2 multiply operation is invalid.
13 (RC/NW)	MRI2	Multiplier Result Overflow. The `FIR_MACSTAT.MRI2` bit is set if the multiplier 2 result is infinity.
12 (RC/NW)	MRZ2	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ2` bit is set if the multiplier 2 result is zero.
11 (RC/NW)	AINV1	Addition Invalid. The `FIR_MACSTAT.AINV1` bit is set if the adder 1 addition is invalid.
10 (RC/NW)	ARI1	Adder Result Overflow. The `FIR_MACSTAT.ARI1` bit is set if the adder 1 result is infinity. Indicates overflow in fixed-point mode.
9 (RC/NW)	ARZ1	Adder Result Underflow. The `FIR_MACSTAT.ARZ1` bit is set if the adder 1 result is zero.

Table 43-25: FIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8 (RC/NW)	MINV1	Multiply Invalid. The `FIR_MACSTAT.MINV1` bit is set if the multiplier 1 multiply operation is invalid.
7 (RC/NW)	MRI1	Multiplier Result Overflow. The `FIR_MACSTAT.MRI1` bit is set if the multiplier 1 result is infinity.
6 (RC/NW)	MRZ1	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ1` bit is set if the multiplier 1 result is zero.
5 (RC/NW)	AINV0	Addition Invalid. The `FIR_MACSTAT.AINV0` bit is set if the adder 0 addition is invalid.
4 (RC/NW)	ARI0	Adder Result Overflow. The `FIR_MACSTAT.ARI0` bit is set if the adder 0 result is infinity. Indicates overflow in fixed-point mode.
3 (RC/NW)	ARZ0	Adder Result Underflow. The `FIR_MACSTAT.ARZ0` bit is set if the adder 0 result is zero.
2 (RC/NW)	MINV0	Multiply Invalid. The `FIR_MACSTAT.MINV0` bit is set if the multiplier 0 multiply operation is invalid.
1 (RC/NW)	MRI0	Multiplier Result Overflow. The `FIR_MACSTAT.MRI0` bit is set if the multiplier 0 result is infinity.
0 (RC/NW)	MRZ0	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ0` bit is set if multiplier 0 result is zero.

FIR Output Data Base Register

The FIR_OUTBASE register contains the output word base address with the lower two bits removed

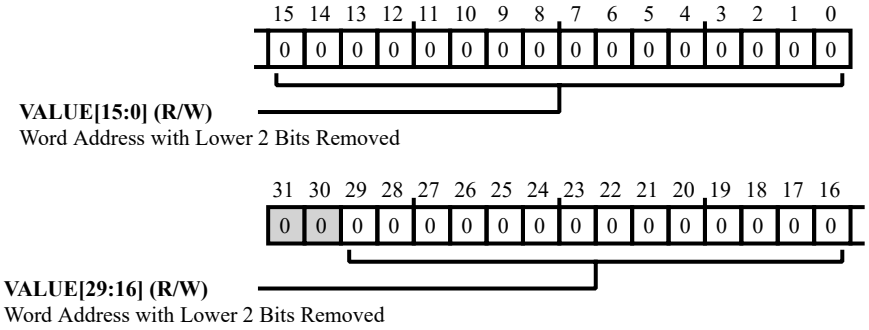


Figure 43-24: FIR_OUTBASE Register Diagram

Table 43-26: FIR_OUTBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_OUTBASE.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Output Data Count Register

The FIR_OUTCNT register contains the 16-bit output buffer count.

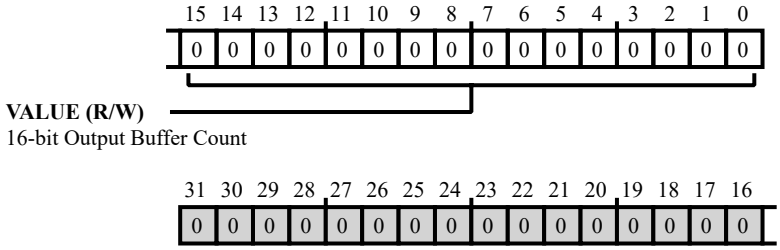


Figure 43-25: FIR_OUTCNT Register Diagram

Table 43-27: FIR_OUTCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Buffer Count. The FIR_OUTCNT.VALUE bit field contains the 16-bit output buffer count.

FIR Output Data Index Register

The FIR_OUTIDX register contains the output word address with the lower two bits removed.

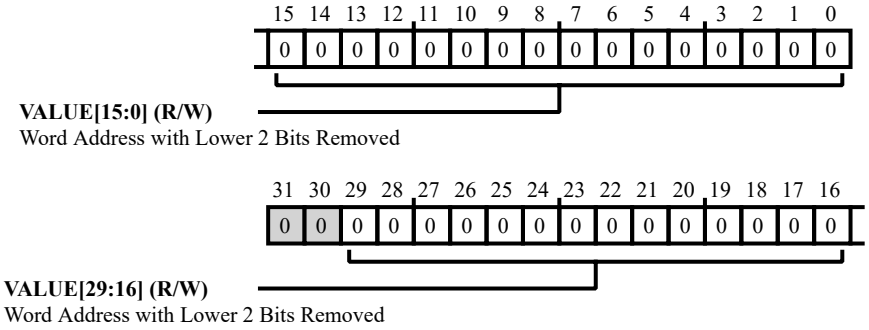


Figure 43-26: FIR_OUTIDX Register Diagram

Table 43-28: FIR_OUTIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_OUTIDX.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Output Data Modifier Register

The FIR_OUTMOD register contains the 16-bit output data buffer modifier.

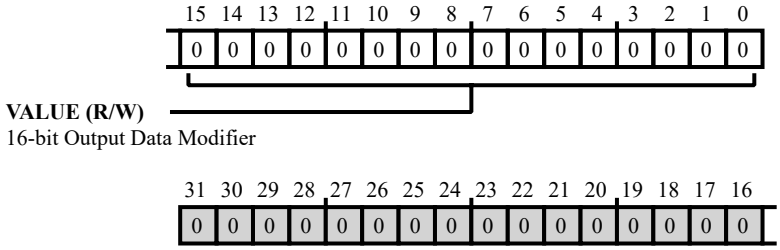


Figure 43-27: FIR_OUTMOD Register Diagram

Table 43-29: FIR_OUTMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Data Modifier. The FIR_OUTMOD.VALUE bit field contains the 16-bit output data modifier.

Software Control Register 1

The FIR_SCTL1 register is used to control software.

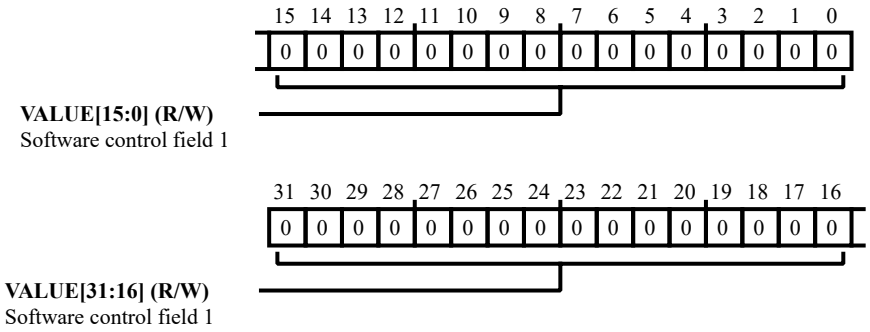


Figure 43-28: FIR_SCTL1 Register Diagram

Table 43-30: FIR_SCTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software control field 1.

Software Control Register 2

The FIR_SCTL2 register is used to control software.

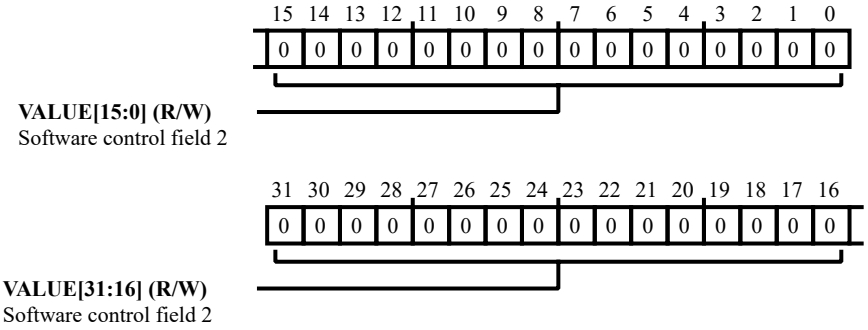


Figure 43-29: FIR_SCTL2 Register Diagram

Table 43-31: FIR_SCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software control field 2.

Secondary Global Control Register

The FIR_SGCTL register is used to configure the global parameters for the accelerator in ACM mode for loading CTL1 register bits as part of TCB.

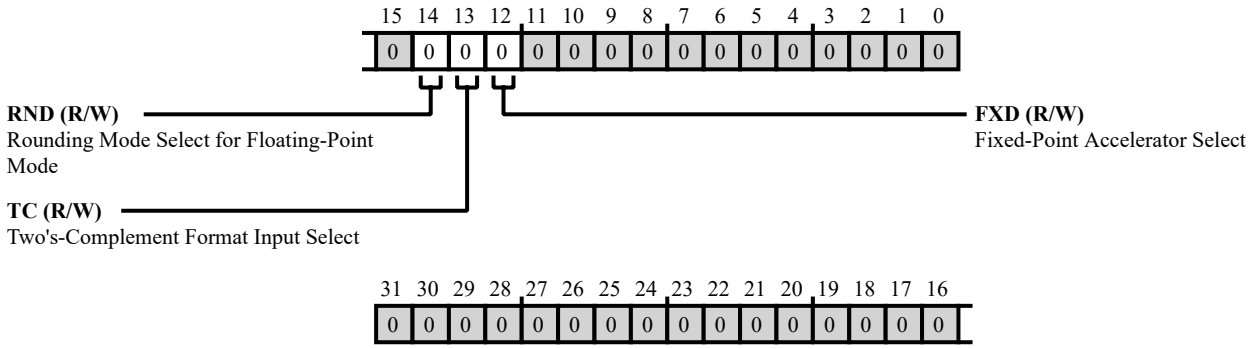


Figure 43-30: FIR_SGCTL Register Diagram

Table 43-32: FIR_SGCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	RND	Rounding Mode Select for Floating-Point Mode. The FIR_SGCTL.RND bit indicates whether the rounding mode is to the nearest or truncated (round towards zero).	
		0	Round to nearest
		1	Truncate
13 (R/W)	TC	Two's-Complement Format Input Select. The FIR_SGCTL.TC bit indicates the two's complement format input for fixed-point mode.	
		0	Unsigned integer
		1	Signed integer
12 (R/W)	FXD	Fixed-Point Accelerator Select. When set (=1), the FIR_SGCTL.FXD bit indicates a 32-bit fixed point accelerator. When cleared (=0), the bit indicates a 32-bit floating-point compatible with SHARC-XI.	
		0	32-bit floating-point
		1	32-bit fixed point

44 IIR Accelerator (IIR)

The processor includes an Infinite Impulse Response (IIR) filter accelerator implemented in hardware that reduces the processing load on the core, freeing it up for other tasks.

Features

The accelerator supports a maximum of 32 channels in legacy mode. In Auto Configuration Mode (ACM), there is no limit on number of channels. The accelerator locally stores all the biquad coefficients of 32 channels in legacy mode. Window size can be configured from 1 (sample-based) to 1024.

The IIR has the following features:

- Supports IEEE floating point format 32/40-bit
- Rounding modes compatible with SHARC+ core MACs
- Sample-based or window-based processing
- Up to 64 cascaded biquads per channel
- Up to 32 filter channels available in TDM
- Allows biquad save state storage
- Supports real-time coefficient slewing using a dedicated hardware slewing engine

NOTE: The IIR accelerator module has local memory that is not accessible by the core during regular operation mode. Unlike in the previous SHARC processors, the IIR accelerator modules each have access to the system memory (on-chip/off-chip).

Unlike in the previous SHARC processors, where only one of the IIR or FIR accelerators can be enabled at a time, the processor can simultaneously use both the IIR and the FIR accelerators.

IIR Definitions

The following definitions are helpful when using the IIR filter and calculating slewing coefficients.

Infinite Impulse Response (IIR) filter

A recursive filter calculates its output by using current and previous inputs, along with previous outputs. IIR filters are digital filters that have an infinite impulse response.

Automatic Configuration Mode (ACM)

The IIR accelerator loads biquad coefficients for only the current channel before processing the current channel.

Legacy Mode

The IIR accelerator loads biquad coefficients for all channels before processing the first channel.

Task Control Block (TCB)

A method to store task-specific parameters and calculation states in a user-initialized memory buffer and permitting user tasks to directly access the values. The reduction of the amount of time required for context switching can lessen the execution time for user tasks.

Biquad (biquadratic)

A second-order recursive digital linear filter, containing two poles and two zeros. In the Z domain, its transfer function consists of the ratio of two quadratic functions. The biquad form II implements the equation in two parts with difference equations and two less storage delay components.

Multiply and Accumulate (MAC) Unit

The IIR subblock with a pipelined multiplier and accumulator that performs floating-point calculations on data and fetches coefficients. Can perform 32-bit or 40-bit operations.

Window-based Processing Mode

Includes using a window size of one or greater (maximum of 1024) for a channel.

Save Biquad State Mode

The mode where the IIR filter stores the current biquad states in local memory when it must switch to another high-priority accelerator task. After resuming, these states can be reloaded, and IIR processing continued.

Slewing

Computation of IIR filter coefficients in real time and in parallel with IIR filtering. The slewing engine calculates coefficients using a slew rate, slew frequency (number of samples), and slewing time (number of iterations).

Slew Rate (α)

A 32-bit floating-point number ($0 \leq \alpha \leq 1$) that indicates the slewing process rate. Alpha (α) values approximating zero indicates a rapidly changing slew process. Values approximating one indicate a slow changing slew.

Slew Frequency (n)

Determines the number of samples (n+1) that pass before the IIR coefficients are slewed.

Slew Time (N)

Determines the number of iterations of coefficient slewing performed by the slewing engine.

Slew Mode

To achieve multiple target values for each coefficient, change the slew rate (either constant or linear) or set the slewing parameters for each channel.

Clocking

The IIR accelerator runs at the speed of core clock.

Functional Description

The ***IIR Accelerator Block Diagram*** shows the blocks of the IIR hardware accelerator.

The accelerator has:

- a coefficient memory size of 1440 × 40 bits (288 biquads x 5 coefficients).
- a data memory size of 576 × 40 bits (288 biquads x 2 states).
- one SHARC+ core compatible MAC unit with an input data buffer to supply data to the MAC.

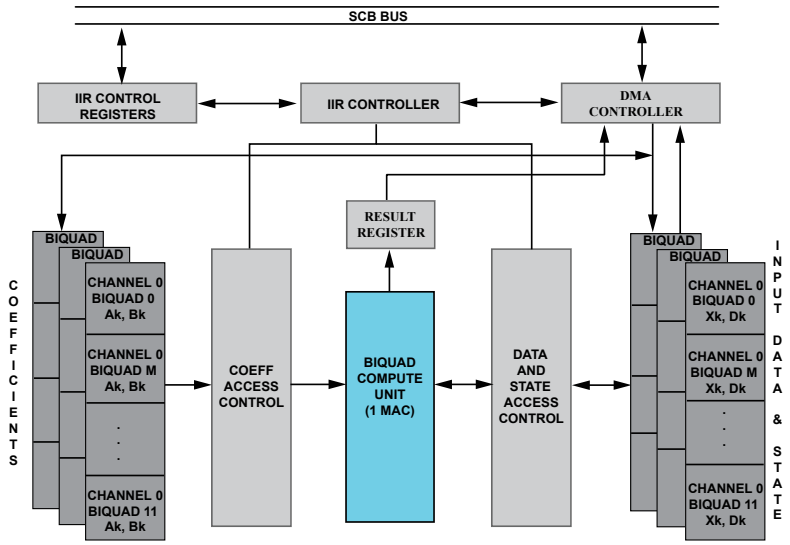


Figure 44-1: IIR Accelerator Block Diagram

A transposed direct form II biquadratic is used to implement the IIR accelerator, which reduces coefficient sensitivity. The ***Transposed Direct Form II Biquad*** figure shows the signal flow graph for the biquad structure.

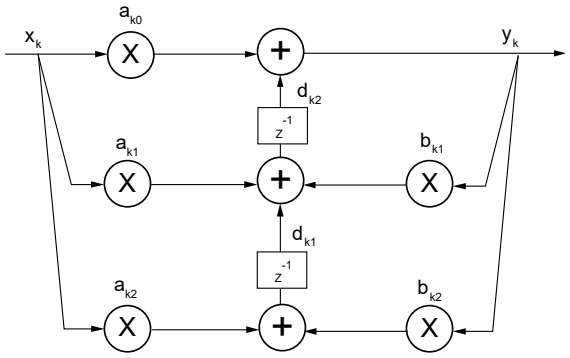


Figure 44-2: Transposed Direct Form II Biquad

The accelerator has the following logical subblocks.

- A data path unit with the following elements:
  - 32/40-bit coefficient memory (Ak, Bk) for storing biquad coefficients
  - 32/40-bit input data (Xk) and state (Dk)
  - One 40/32-bit floating-point multiplier and adder (MAC) unit
  - An input data buffer to efficiently supply data to MAC
  - One 40-bit result register to hold result of the biquad
- Configuration registers for controlling various parameters such as the number of biquads, the number of channels, interrupt control, and DMA control.

- A core access interface for writing the DMA/filter configuration registers and for reading the status registers.
- A DMA bus interface for transferring data to and from the accelerator. This interface also preloads the coefficients (Ak, Bk) and state (Dk) at startup.
- DMA configuration registers for the transfer of input data, output data, and coefficients.

ADSP-2184x IIR Register List

The IIR module reduces the processing load on the core. For more information on IIR functionality, see the IIR register descriptions.

Table 44-1: ADSP-2184x IIR Register List

Name	Description
IIR_ALPHA[n]	Channel n Alpha
IIR_CHNPTR	Chain Pointer Register
IIR_COEFIDX	Coefficient Buffer Index Register
IIR_COEFLEN	Coefficient Buffer Length Register
IIR_COEFMOD	Coefficient Index Modifier Register
IIR_CTL1	Global Control Register
IIR_CTL2	Channel Control Register
IIR_DBG_ADDR	IIR Debug Address Register
IIR_DBG_CTL	IIR Debug Control Register
IIR_DBG_RDDAT_HI	IIR Debug Read Data High Register
IIR_DBG_RDDAT_LO	IIR Debug Read Data Low Register
IIR_DBG_WRDAT_HI	IIR Debug Write Data High Register
IIR_DBG_WRDAT_LO	IIR Debug Write Data Low Register
IIR_DMASTAT	DMA Status Register
IIR_INBASE	Input Buffer Base Register
IIR_INIDX	Input Data Index Register
IIR_INLEN	Input Data Buffer Length Register
IIR_INMOD	Input Data Index Modifier Register
IIR_MACSTAT	MAC Status Register
IIR_OUTBASE	Output Buffer Base Register
IIR_OUTIDX	Output Data Buffer Index Register
IIR_OUTLEN	IIR Output Data Buffer Length Register
IIR_OUTMOD	IIR Output Data Index Modifier Register

Table 44-1: ADSP-2184x IIR Register List (Continued)

Name	Description
IIR_SCTL1	Software Control Register1
IIR_SCTL2	Software Control Register2
IIR_SGCTL	Secondary Global Control Register
IIR_SLEW_CNFG[n]	Channel 0 Slewing Duration and Frequency

ADSP-2184x IIR Interrupt List

Table 44-2: ADSP-2184x IIR Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
131	IIR0_DMA	IIR0 Core 1 DMA	Edge	
132	IIR0_STAT	IIR0 Core 1 Status	Edge	
133	IIR1_DMA	IIR1 Core 1 DMA	Edge	
134	IIR1_STAT	IIR1 Core 1 Status	Edge	
135	IIR2_DMA	IIR2 Core 1 DMA	Edge	
136	IIR2_STAT	IIR2 Core 1 Status	Edge	
137	IIR3_DMA	IIR3 Core 1 DMA	Edge	
138	IIR3_STAT	IIR3 Core 1 Status	Edge	
139	IIR0_BUS_ERR	IIR0 Core 1 IIR Bus Error	Edge	
140	IIR1_BUS_ERR	IIR1 Core 1 IIR Bus Error	Edge	
141	IIR2_BUS_ERR	IIR2 Core 1 IIR Bus Error	Edge	
142	IIR3_BUS_ERR	IIR3 Core 1 IIR Bus Error	Edge	

ADSP-2184x IIR Trigger List

Table 44-3: ADSP-2184x IIR Trigger List Generators

Trigger ID	Name	Description	Sensitivity
60	IIR0_DMA	IIR0 Core 1 DMA	Edge
61	IIR1_DMA	IIR1 Core 1 DMA	Edge
62	IIR2_DMA	IIR2 Core 1 DMA	Edge
63	IIR3_DMA	IIR3 Core 1 DMA	Edge

Table 44-4: ADSP-2184x IIR Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
102	IIR0_TRGI	IIR0 Core 1 IIR Wait on Trigger Input	Pulse
103	IIR1_TRGI	IIR1 Core 1 IIR Wait on Trigger Input	Pulse
104	IIR2_TRGI	IIR2 Core 1 IIR Wait on Trigger Input	Pulse
105	IIR3_TRGI	IIR3 Core 1 IIR Wait on Trigger Input	Pulse

Multiply and Accumulate (MAC) Unit

The ***IIR MAC Unit*** figure shows a pipelined multiplier and accumulator unit that operates on the data and coefficients fetched from the data and coefficient memory. The MAC can perform 32-bit floating-point or 40-bit floating-point MAC operations. The 32-bit floating-point operations generate 32-bit results and the 40-bit floating-point operations generate 40-bit results.

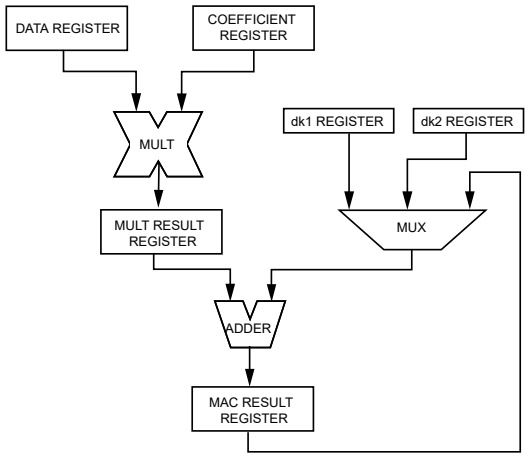


Figure 44-3: IIR MAC Unit

Input Data and Biquad State

The size of the data memory is 576 × 40 bits and locally holds the dk1 and dk2 state of all biquads. The DMA controller fetches the sample data from internal memory and calculates the output plus the dk1 and dk2 values for each biquad and stores them in local data memory.

Coefficient Memory

The size of coefficient memory is 1440 × 40 bits and stores all the coefficients for all biquads. At startup, the DMA loads the coefficients from system memory into local coefficient memory.

Internal Memory Storage

This section describes the required storage model for the IIR accelerator.

Coefficient Memory Storage

Store the coefficients and Dk values for a specific biquad BQD[k] in internal memory in the following order: Ak0, Ak1, Bk1, Ak2, Bk2, Dk2, and Dk1.

NOTE: The naming convention for the filter coefficients used here is different from the one used in MATLAB. The following conversion should be used when using MATLAB generated coefficients:

(Akx = bx and Bkx = -ax).

Store the coefficients for each biquad in the order:

```
b0, b1, -a1, b2, -a2, dk2, dk1
```

For N biquad stages, store the coefficients in the order:

```
b01, b11, -a11, b21, -a21, dk21, dk11,
b02, b12, -a12, b22, -a22, dk22, dk12,
,......
b0N, b1N, -a1N, b2N, -a2N, dk2N, dk1N.
```

where bxN and axN are the coefficients ([b, a]) for the Nth biquad stage.

Target Coefficient Memory Storage

For coefficient slewing, the structure of the target coefficient array is like the coefficient memory storage. The first five coefficient of each biquad are stored in internal memory with the order: Ak0, Ak1, Bk1, Ak2, and Bk2. Following the five coefficients, two zeroes replace the initial state values Dk1 and Dk2.

IMPORTANT: The naming convention used in this hardware reference and that used in MATLAB follows the same conversion rule described for the Coefficient Memory Storage (Akx = bx and Bkx = -ax).

Operating Modes

The accelerator operates:

- In Window Processing Mode, that also includes sample-based processing mode
- In 40-Bit Floating-Point Mode
- In Save Biquad State Mode
- In Auto Configuration Mode (ACM)
- Calculates Coefficient Slewing

Window Processing Mode

Select the sample-based processing mode by configuring window size to one. All biquads for a particular channel process one sample, and then calculate a final output sample in this mode.

In window-based mode, multiple output samples (maximum of 1024, equal to the window size) of that channel are calculated. After these calculations are complete, the accelerator begins processing the next channel. A register field provides a window size parameter to specify the length of the window.

40-Bit Floating-Point Mode

In 40-bit floating-point mode, the accelerator treats input data and coefficients as a 40-bit floating-point numbers. The 40-bit floating-point MAC operations generate 40-bit results. This mode can be selected by setting the `IIR_CTL1.FORTYBIT` bit.

In ACM mode, 40-bit floating-point mode can be selected by setting the `IIR_SGCTL.FORTYBIT` bit field for each channel.

As the DMA bus width is 32 bits, in 40-bit mode the IIR accelerator performs two packed 32-bit accesses to the memory to:

- fetch one 40-bit input or coefficient data
- store one 40-bit output word

The first 32-bit word provides the lower 32 bits and the 8 LSBs of the second 32-bit word provides rest of the upper 8 bits of the complete 40-bit word. The ***32-Bit to 40-Bit Packing*** figure shows the 32– to 40–bit packing used by the accelerator.

NOTE: Overhead time may be required to pack the input 40-bit data into the format acceptable by the IIR accelerator and for unpacking the output of accelerator to a format acceptable to the rest of the application.

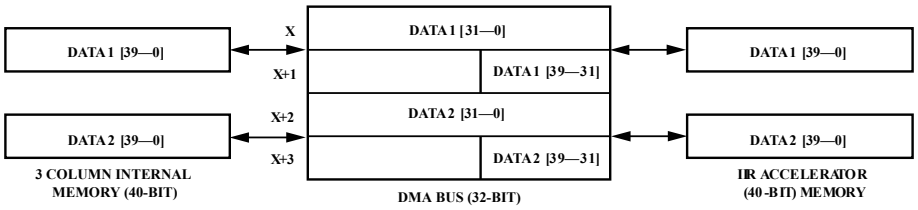


Figure 44-4: 32-Bit to 40-Bit Packing

Save Biquad State Mode

The `IIR_CTL1.SS` bit stores the current biquad states in local memory (writes all the Dk1 and Dk2 states back into the system memory states). This is useful for applications that require fast switching to another high-priority accelerator task—a required IIR to FIR processing transition for example. After resuming, these states can be reloaded and IIR processing can be continued. DMA status is automatically stored after each iteration.

NOTE: In legacy mode, the accelerator loads the biquad coefficients for all the channels before processing the first channel. After processing the last channel, when the `IIR_CTL1.SS` bit is set, the accelerator stores the biquad states of all the channels in local memory (writes all the Dk1 and Dk2 states back into the system memory states). In ACM mode, the accelerator loads the biquad coefficients for only the current channel

before starting to process the current channel. When the IIR_SGCTL.SS bit is set, the accelerator stores the biquad states of the current channel in local memory before starting calculations for the next channel.

NOTE: During the save state operation, it is not permitted to have write access to any of the IIR accelerator registers that are loaded by chaining. Attempted writes to these registers can result in the blocking of IOP core reads until the save state operation completes.

Auto Configuration Mode (ACM)

The accelerator can be operated in legacy mode or Auto Configuration Mode (ACM). ACM is configured by setting the IIR_CTL1.ACM bit. The default functional mode is legacy mode. The accelerator must be disabled before changing the accelerator mode.

The ACM provides the following features:

- Halt and Queuing

  The core can pause processing the current Transfer Control Block (TCB) chain by setting the IIR_CTL1.HALT bit. The accelerator acknowledges the core by setting the IIR_DMASTAT.HALT_STAT bit. The core can then submit or insert new TCBs. After the core acts, the accelerator resumes processing by clearing the IIR_CTL1.HALT bit. Before halting the accelerator, when the initial TCB chain is processed, the accelerator goes to the idle state. To do this, disable the accelerator and then enable it by toggling the IIR_CTL1.EN. Clear the IIR_CTL1.HALT bit to resume processing.

- No Channel Number Limitation

  Unlike in legacy mode, there is no fixed channel number limitation, and the accelerator ignores the value programmed in the IIR_CTL1.CH field. The application can queue an unlimited number of channels and TCBs dynamically, plus the accelerator keeps processing the TCBs until the chain pointer is null.

- Selective Interrupt

  The core can enable and mask interrupt generation for each channel by setting the IIR_CTL2.IMASK bit. When the bit is cleared, an interrupt generates after the channel completes.

- Selective Controller/Target Trigger

  The core can enable/mask trigger generation by the accelerator after the end of processing of each channel using the IIR_CTL2.TMASK bit. The accelerator can also wait for a trigger after loading the TCB and coefficients and before processing a channel for which the IIR_CTL2.TWAIT bit is set.

  This feature can also be enabled in legacy mode by setting the IIR_CTL1.L_TIMASK_EN and IIR_CTL1.L_TWAIT_EN bits. In legacy mode, when the IIR_CTL1.L_TIMASK_EN and IIR_CTL1.L_TWAIT_EN bits are set, then the IIR_CTL2.TMASK and IIR_CTL2.TWAIT bits can be used to enable/mask trigger generation and wait for a trigger for each channel respectively.

Additionally, use the IIR_CTL1 register to change parameters such as rounding mode, 40-bit mode, and save state for each channel. The IIR_SCTL1 and IIR_SCTL2 registers can be used as general-purpose registers.

Coefficient Slewing

The following sections describe the features, parameters, and real-time processing for coefficient slewing.

- Coefficient Slewing Features
- Coefficient Slewing Parameters
- Coefficient Real-time Slewing

Coefficient Slewing Features

Enable the slewing feature by setting `IIR_CTL1.SLEW_EN` bit. When this bit is cleared, the IIR operates in legacy mode. Once set, this bit should not be cleared until the `IIR_CTL1.EN` bit is cleared.

The slewing feature does not support the ACM operating mode.

The slewing feature can be enabled in burst/non-burstmode, 40-bit or 32-bit mode, and in the channel auto interate (CAI) mode.

The IIR instances IIR0 and IIR1 each have one slewing engine.

Slewing calculations happen in parallel to its biquad execution within the IIR block. When slewed coefficients are needed for the next sample calculation, slewing of coefficients happens during the current sample.

Coefficients can be slewed for each sample (sample -based) or after a configured interval of samples.

Slew parameters can be configured on a per channel basis using the:

- Slewing Rate (α)
- Slewing Frequency (n)
- Slewing Time (N)

These parameters are part of the TCB for each channel. The TCB length is increased by three for legacy mode when the coefficient slewing is enabled.

With appropriate values of α, n and N, the user can ensure that the coefficients approximate their target values at the end of the slewing process for each channel. Without proper values, the user may observe audio glitches after slewing completion because of the differences in target and final coefficients.

A separate memory for storing the Target Coefficients (C_T) resides inside the IIR accelerator.

The slewing rate (α) can be kept constant or can be linearly changing from one to zero. The `IIR_CTL2.SLEW_LALPHA_MODE` control bit defines the alpha configuration (constant or varying) operating for each channel.

Whenever the slewing saturates or slewing time expires (that is, after N iterations), C[n] either remains at the last iterated value or attains C_T, based on the `IIR_CTL2.SLEW_SAT_EN` bit set for each channel.

Slewing parameters (α, N, n) and target coefficients (C_T) can be updated during the slewing process by enabling the `IIR_CTL1.SLEW_PARAMS_UPDATE` bit and by disabling and reenabling the DMA.

During channel switching, the slewing engine remembers the state of slewing in the current channel and restores the state once the channel comes back.

Coefficient Slewing Parameters

Slewing Rate (α)

The slewing rate (α) is 32-bit floating-point number with a value zero through one. When α approximates one, it indicates a slow slewing process. When α approximates zero, it indicates a fast slewing process. The slewing rate provides a 32-bit floating-point number as the sixth element in the task control block (that is, `TCB+0x5` or the `IIR_ALPHA[n]` register for the nth channel).

Slewing Frequency (n)

The slewing frequency (n) determines after how many samples the slew calculation for the IIR coefficients happens. When n = 0, beginning from the second sample every sample output is calculated using the updated slewed coefficients (see the *IIR Coefficient Slewing (constant alpha mode) n=0 and n=3 Examples* figure). For n > 0, the accelerator coefficients are slewed after every n+1 samples and output of n+1 samples are calculated using last slewed coefficients. An internal counter keeps track of the number of samples completed to take a decision on performing the slewing for the next sample.

For instance, when n = 3, starting from the 5th sample, every fourth sample (N+1) output is calculated using new coefficients (that is, fifth through eighth samples). Coefficients are slewed again at the nineth sample and every fourth sample thereafter.

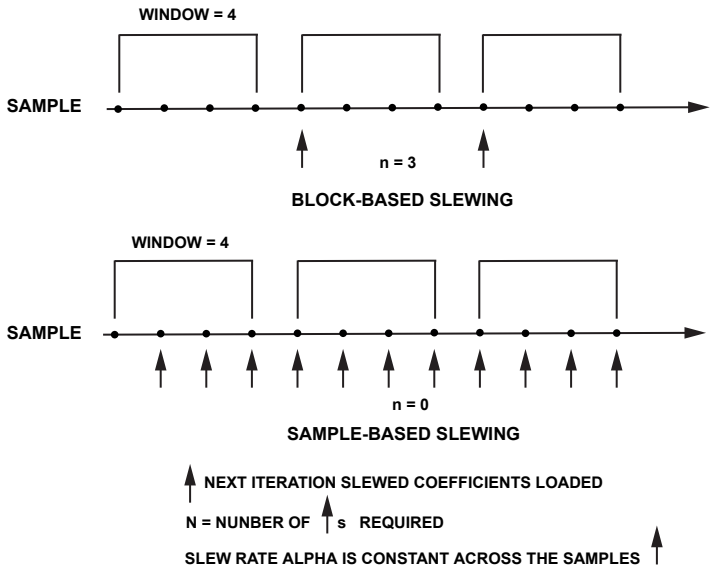


Figure 44-5: IIR Coefficient Slewing (constant alpha mode) n=0 and n=3 Examples

Slewing Time (N)

The slewing time (N) indicates the number of iterations of coefficient slewing performed by the slewing engine. This parameter is defined for each channel and decremented inside the slewing engine after each slewing iteration.

Once it reaches zero, the slewing stops. The total number of samples processed during an entire slewing interval is (n*N) samples.

Coefficient Real-time Slewing

This processor has a dedicated hardware to slew the IIR coefficients in real time and in parallel to the IIR filtering. Typically, slewing done through software takes up significant core MIPS and DMA overhead for loading slewed coefficients. The dedicated hardware engine placed inside the IIR filter can slew IIR biquad coefficients in real time and avoids using the core. Slewing ensures a smooth transition of IIR coefficients during the change of IIR parameters, that can otherwise result in audio click noises.

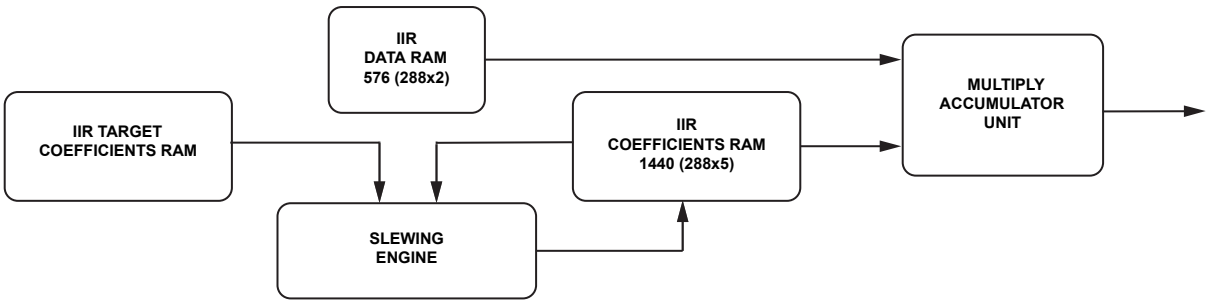


Figure 44-6: IIR Slewing Block Diagram

One slewing engine connects to each of the IIR filters. The coefficient slewing is done based on the following recursive equation:

$C[n] = (1-\alpha) * C_T + \alpha * C[n-1]$, where $C = \{Ak0, Ak1, Bk1, Ak2, Bk2\}$ and $0 \leq \alpha \leq 1$
where C_T is the target coefficient, α is the slewing rate, and n is sample number.

Data Transfers

The IIR filter works exclusively through DMA.

IIR Accelerator TCB

The location of the DMA parameters for the next sequence comes from the chain pointer register. This register points to the next set of DMA parameters stored in the system memory of the processor known as the Task Control Block (TCB). In chained DMA operations, the processor automatically initializes and then begins another DMA transfer once the current DMA transfer completes. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Assignment

The structure of a TCB is conceptually the same as the structure of a linked list. Each TCB has several data values and a pointer to the next TCB. The chain pointer of a TCB can point to itself to re-run the same DMA continuously. The IIR accelerator reads each word of the TCB and loads it into the corresponding register. A TCB with a chain pointer register value of zero indicates the end of the chain (no further TCBs to load). The IIR

accelerator supports circular buffer chained DMA. The following tables show the required TCBs for chained DMA in Legacy Mode and ACM. TCB refers to the start address of the TCB array.

NOTE: The IIR accelerator DMA has two available TCB loading sequences:

- Loads five parameters for the coefficients (IIR_CTL2, IIR_COEFIDX, IIR_COEFMOD, IIR_COEFLEN, and IIR_CHNPTR).
- Loads 10 parameters for the data (IIR_CTL2, IIR_INIDX, IIR_INMOD, IIR_INLEN, IIR_INBASE, IIR_OUTIDX, IIR_OUTMOD, IIR_OUTLEN, IIR_OUTBASE, and IIR_CHNPTR).

Initialize IIR_CHNPTR to TCB+12 in legacy mode and TCB+15 in ACM.

Table 44-5: IIR TCBs for Chained DMA in Legacy Mode

Address	Register
TCB	IIR_CHNPTR
TCB + 0x1	IIR_COEFLEN
TCB + 0x2	IIR_COEFMOD
TCB + 0x3	IIR_COEFIDX
TCB + 0x4	IIR_OUTBASE
TCB + 0x5	IIR_OUTLEN
TCB + 0x6	IIR_OUTMOD
TCB + 0x7	IIR_OUTIDX
TCB + 0x8	IIR_INBASE
TCB + 0x9	IIR_INLEN
TCB + 0xA	IIR_INMOD
TCB + 0xB	IIR_INIDX
TCB + 0xC	IIR_CTL2

Table 44-6: IIR TCBs for Chained DMA in ACM

Address	Register
TCB	IIR_CHNPTR
TCB + 0x1	IIR_SCTL1
TCB + 0x2	IIR_SCTL2
TCB + 0x3	IIR_SGCTL
TCB + 0x4	IIR_COEFLEN
TCB + 0x5	IIR_COEFMOD

Table 44-6: IIR TCBs for Chained DMA in ACM (Continued)

Address	Register
TCB + 0x6	IIR_COEFIDX
TCB + 0x7	IIR_OUTBASE
TCB + 0x8	IIR_OUTLEN
TCB + 0x9	IIR_OUTMOD
TCB + 0xA	IIR_OUTIDX
TCB + 0xB	IIR_INBASE
TCB + 0xC	IIR_INLEN
TCB + 0xD	IIR_INMOD
TCB + 0xE	IIR_INIDX
TCB + 0xF	IIR_CTL2

Table 44-7: Slewing TCB Structure

Address	Register
TCB	IIR_CHNPTR
TCB+0x1	IIR_COEFLEN
TCB+0x2	IIR_COEFMOD
TCB+0x3	IIR_COEFIDX
TCB+0x4	IIR_CTIDX
TCB+0x5	IIR_ALPHA[n]
TCB+0x6	IIR_Nn
TCB+0x7	IIR_OUTBASE
TCB+0x8	IIR_OUTLEN
TCB+0x9	IIR_OUTMOD
TCB+0xA	IIR_OUTIDX
TCB+0xB	IIR_INBASE
TCB+0xC	IIR_INLEN
TCB+0xD	IIR_INMOD
TCB+0xE	IIR_INIDX
TCB+0xF	IIR_CTL2.SLEW_LALPHA_MODE

DMA Access

The IIR accelerator has two DMA channels (accelerator input and output) to connect to the system memory. The DMA controller fetches the data and coefficients from memory and stores the result.

Burst Mode Access

Burst mode enhances the throughput of the DMA channel and reduces the overall load on the system. When the `IIR_CTL1.BURSTEN` bit is set, it loads all the biquads for all the channels with burst enabled. The IIR module supports only INCR8 burst.

Chain Pointer DMA

The DMA controller supports a circular buffer chain pointer DMA. One transfer control block (TCB) must be configured for each channel. The TCB contains:

- A control register value to configure the filter parameters (such as number of biquads, window size) for each channel. For ACM, additional parameters such as an interrupt mask, trigger mask, and trigger wait exist.
- Software control register values in ACM for each channel.
- Secondary control register value in ACM to configure rounding mode, 40-bit mode, and save state for each channel.
- DMA parameter register values for the input data.
- DMA parameter register values for coefficient load.
- DMA parameter register values for output data.

In legacy mode, The chain pointer (`IIR_CHNPTR`) field of the last channel TCB should point to the first channel TCB. This configuration exists so that when the IIR accelerator is enabled, the module:

1. Loads the coefficients (Ak, Bk) and state variables (Dk) for all the channels into the local coefficient memory of the IIR.
2. Loops back to the first channel again to start fetching the input data for processing.

In ACM, the chain pointer field of the last channel TCB need not point to the first channel TCB.

When the accelerator is enabled, the module:

1. Only loads the coefficients (Ak, Bk) and state variables (Dk) for the current channel being processed.
2. Starts fetching input data for processing.
3. After processing the current channel, when the `IIR_SGCTL.SS` bit is set, it saves the biquad states of the current channel in local memory and moves to the next channel.

In ACM, the accelerator keeps processing the TCBs until the chain pointer becomes null.

The accelerator loads the TCB into its internal registers and uses these values to fetch coefficients, fetch data, and store results. After processing a window of data for any channel, the accelerator writes back the `IIR_INIDX` (input

index register) and IIR_OUTIDX (output index register) values to the TCB in memory. Data processing can then begin from where it left off during the next time slot of that channel.

For 32-bit mode, the write-back values for the index registers are equal to IIRII + W and IIROI + W.

For 40-bit mode, the write-back values are: IIR_INIDX + 2 × W and IIR_OUTIDX + 2 × W.

Accelerator input and output channels connect to system memory.

NOTE: The IIR_CTL2 register is part of the IIR TCB. This configuration allows software to program individual IIR channels having different control attributes.

In ACM, when the IIR_CTL1.SMQ_LIUPS_EN bit is set, the accelerator updates the IIR_INIDX and IIR_OUTIDX bit fields of the TCB in memory after processing a window of data and according to the circular buffer scheme. When the IIR_CTL1.SMQ_LIUPS_EN bit is cleared, the accelerator updates the IIR_INIDX and IIR_OUTIDX bit fields of the TCB in memory to 0x00000000 and 0xFFFFFFFF after processing a window of data. The IIR_CTL1.SMQ_LIUPS_EN bit is only valid in ACM.

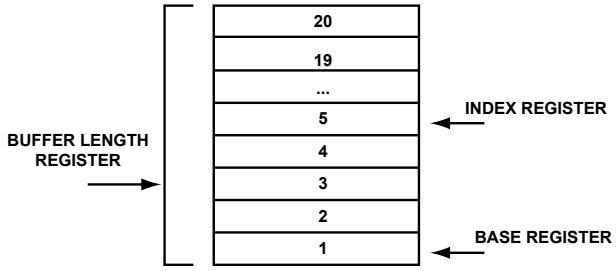


Figure 44-7: Circular Buffer Addressing

Interrupts

The ***IIR Interrupt Overview*** table provides the source of interrupt and service instructions for the IIR interrupts.

Table 44-8: IIR Interrupt Overview

Accelerator Mode	Default Programmable Interrupt	Sources	Masking	Service
Legacy Mode	IIR_DMA	Window Complete	N/A	ROC from IIR_DMASTAT + RTI instruction
		All channels complete		
	IIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from IIR_MACSTAT + RTI instruction
		MAC fixed point overflow		

Table 44-8: IIR Interrupt Overview (Continued)

Accelerator Mode	Default Programmable Interrupt	Sources	Masking	Service
Auto Configuration Mode	IIR_DMA	Window Complete	IIR_CTL2.IMASK	ROC from IIR_DMASTAT + RTI instruction
	IIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from IIR_MACSTAT + RTI instruction
		MAC fixed point overflow	N/A	

Sources

The IIR module drives two interrupt signals, IIR_DMA for the DMA status and IIR_STAT for the MAC status. The IIR module generates interrupts as described in the following sections.

Window Complete

This interrupt generates at the end of each channel when all the output samples are calculated for a window and the channel writes back updated index values.

In legacy mode, when the IIR_CTL1.CCINTR bit is set, an interrupt generates after completion of window processing of each channel. In ACM, the interrupt generation can be selectively masked using the IIR_CTL2.IMASK bit for each channel.

Interrupt masking for each channel can be enabled in legacy mode by setting the IIR_CTL1.L_TIMASK_EN bit. In legacy mode, when the IIR_CTL1.L_TIMASK_EN bit is set, interrupt generation for each channel is controlled using the IIR_CTL2.IMASK bit.

All Channels Complete

This interrupt generates when all the channels are complete or when one iteration of time slots completes. The interrupt follows the access completion rule, where the interrupt generates when all data are written back to system memory.

The All Channels Complete interrupt source only applies in legacy mode. When the IIR_CTL1.CCINTR and IIR_CTL1.L_TIMASK_EN bits are not set in Legacy Mode, the all channel complete interrupt generates. In ACM, interrupt generation for each channel is controlled using the IIR_CTL2.IMASK bit.

NOTE: In ACM, the Output DMA complete interrupt can be generated after the Save State completes for the channel by clearing the IIR_SGCTL.SSESEL bit . When this bit is set, the Output DMA interrupt generates before the Save State completes for the channel.

In Legacy Mode, when the IIR_CTL1.CCINTR and IIR_CTL1.SSESEL bits are cleared, the All Channels Done Interrupt generates after the Save State operation completes for all the channels.

MAC Status

A MAC status interrupt generates under the following conditions:

- Multiplier result zero–set when multiplier result is zero
- Multiplier result infinity–set when multiplier result is infinity
- Multiply invalid–set when multiply operation is invalid
- Adder result zero–set when adder result is zero
- Adder result infinity–set when adder result is infinity
- Adder invalid–set when addition is invalid

The slewing engine inside IIR accelerator has a floating-point subtractor, multiplier, and adder for calculating the next iteration of coefficients. The slewing engine interrupts are ORed with the previous IIR MAC Status interrupt and only sampled when slewing is enabled (set the `IIR_CTL1.SLEW_MAC_INT_EN` bit). A MAC status interrupt generates for any of the following conditions:

- Adder output is infinity, invalid, or zero
- Multiplier output is infinity, invalid, or zero
- Subtractor output is infinity or invalid

Service

The DMA interrupt status bits are sticky and are cleared when the DMA status register is read. When a MAC status interrupt occurs, programs can find this state (and clear) by reading the MAC status register (`IIR_MACSTAT`). The MAC interrupt status bits are sticky.

The status interrupt sources are derived from the `IIR_MACSTAT` register. A status interrupt can occur due to the last set of MAC operations of a processing iteration that correspond to a particular channel. The interrupt is generated continuously and cannot be stopped, even after disabling the accelerator. The interrupt can only be stopped when another processing iteration results in a non-zero or valid multiply or add result.

Programming Model

The following sections provide programming examples for legacy and auto configuration modes.

The Coefficient Slewing Programming section provides detailed programming steps for the different modes to calculate slewing coefficients.

Programming Legacy Mode and ACM

- Legacy Mode Programming
- Auto Configuration Mode Programming

Legacy Mode Programming

The IIR accelerator supports up to 32 channels which are time division multiplexed (TDM). Each channel can have a maximum of 64 biquads. The window size for each channel is configurable using the control registers. The maximum configured window size is 1024 samples.

For programming multi-channel filtering in Legacy Mode:

1. Program the number of channels using the `IIR_CTL1.CH` bits.
2. Configure the TCBs in system memory with one channel TCB pointing to the next channel TCB. The TCB of the last channel should point to the TCB of the first channel.
3. Write the first TCB value into the `IIR_CHNPTR` register and enable the accelerator.
4. When the `IIR_CTL1.BURSTEN` bit is set, ensure that the coefficient buffer end address plus seven words does not fall outside the valid address range and the buffer is initialized. When the burst feature is enabled, the IIR module issues an INCR8 burst. When the data count % 8== 7, the extra seven words are always read, but ignored by the IIR module.

NOTE: Data count = ∑(Bn*7) or ∑(Bn*14); where Bn is number of biquads for channel N. Each biquads contains 7 words for 32-bit mode or 14 words in 40-bit mode.

The ***Biquad Processing Program Flow*** figure shows the biquad processing program flow.

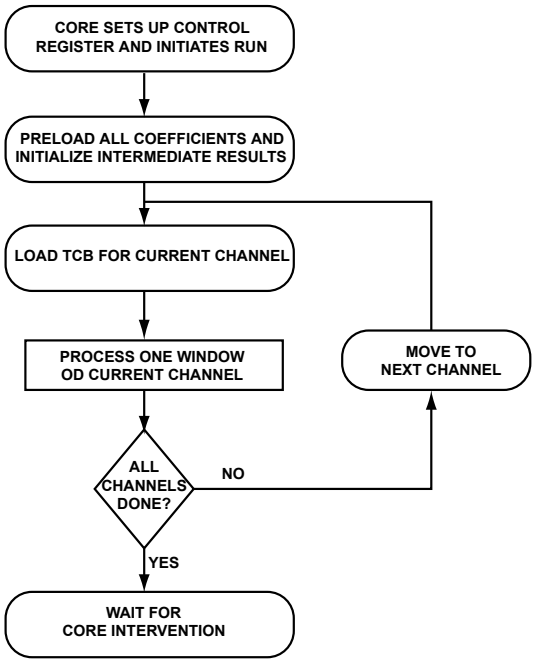


Figure 44-8: Biquad Processing Program Flow

The accelerator uses the following procedure to process biquads.

1. The controller loads all coefficients of all the channels into local storage.

2. Once all the coefficients are loaded, the controller goes to the first biquad of the first channel and calculates the output of the first biquad and updates the intermediate results for that biquad. When the `IIR_CTL1.L_TWAIT_EN` bit is set and the `IIR_CTL2.TWAIT` bit is set for the channel, the accelerator waits for the input trigger to start the computation.
3. The accelerator then moves to the next biquad of that channel and repeats the process until all the biquads for that channel are completed and the results are stored to memory.
4. The procedure is repeated for the next sample until processing completes for one window of the corresponding channel.
5. After one channel window is processed, the accelerator moves to the next channel and computes the results.

NOTE: All the addresses must be aligned on 32-bit address boundaries and must not contain the lower two bits (assumed to be zeros).

Auto Configuration Mode Programming

For ACM, there is no limit on the number of channels the accelerator can process. The accelerator keeps processing the TCBs until the chain pointer becomes null. Each channel can have a maximum of 64 biquads. The maximum configured window size is 1024 samples.

The Multi-channel Filtering in Auto Configuration Mode figure shows multi-channel filtering in ACM. Multiple channels are processed in a time division multiplexed (TDM) format.

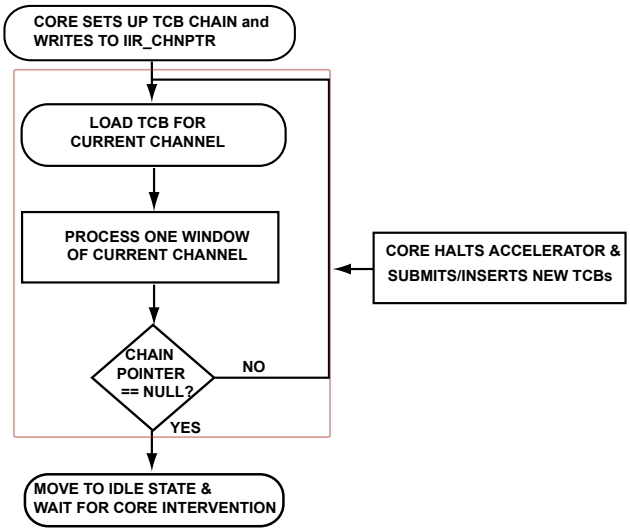


Figure 44-9: Multi-channel Filtering in Auto Configuration Mode

To program and understand the multi-channel filtering in ACM:

1. Configure the TCBs in system memory with the one channel TCB pointing to the next channel TCB. There is no limit on the number of channels to be configured. The TCB of the last channel does not need to point to the TCB of the first channel.

2. Write the first TCB value to the IIR_CHNPTR register and enable the accelerator.

   The accelerator fetches the first channel TCB. Using it as pointer, loads the biquad coefficients and biquad states for that channel into the IIR local memory and loads the IIR_CTL2 and IIR_SGCTL registers to configure the filter parameters corresponding to that channel.

   When the IIR_CTL2.TWAIT bit is set, the accelerator waits for a input trigger to start the window processing for the channel.

   The accelerator computes the output of the first biquad and updates the intermediate results for that biquad. Then, the accelerator moves to the next biquad of that channel and repeats the process until all the biquads for that channel are completed and the results are stored to memory. The process is repeated with the next sample until one window of the corresponding channel is processed. When the IIR_CTL1.SS bit is set, the accelerator saves the states of the biquads of the channel into system memory. If the IIR_CTL2.TMASK bit is cleared the accelerator sends an output trigger after completion of processing the channel.

   When the IIR_CTL2.IMASK bit is cleared, the accelerator interrupts the core after completion of processing a particular channel.

   At the end of the window, the accelerator updates the IIR_INIDX and IIR_OUTIDX registers to 0x00000000 and 0xFFFFFFFF in the TCB of system memory and moves to the next channel.

3. At any instant, the core can halt the accelerator as needed. It sets the IIR_CTL1.HALT bit and appropriately takes action to submit/insert new TCBs and clears the IIR_CTL1.HALT bit to resume channel processing.

   When the IIR_CHNPTR register is zero (last channel is being processed or channel processing is complete) after halting the accelerator, the IIR_CHNPTR register is written before resuming the accelerator channel processing.

   When the accelerator is idle after halt, IIR_CTL1.EN bit is toggled to disable and re-enable the accelerator and the IIR_CTL1.HALT bit is cleared and the accelerator resumes channel processing.

4. The accelerator continues processing until all the channels are complete. Repeat *Step 3* (as required) to submit or insert new channels.

Coefficient Slewing Programming

- Coefficient Slewing Program Flow
- Programming Constant Alpha Slewing Mode
- Programming Linear Changing Alpha Slewing Mode
- Programming Intermediate Target Coefficient Update Mode

NOTE: The save state bit should be disabled (IIR_CTL1.SS =0) when the coefficient slewing bit is enabled (IIR_CTL1.SLEW_EN =1).

Coefficient Slewing Program Flow

The *Slewing Operation Program Flow* figure captures the coefficient slewing program flow that is common for all slewing modes described:

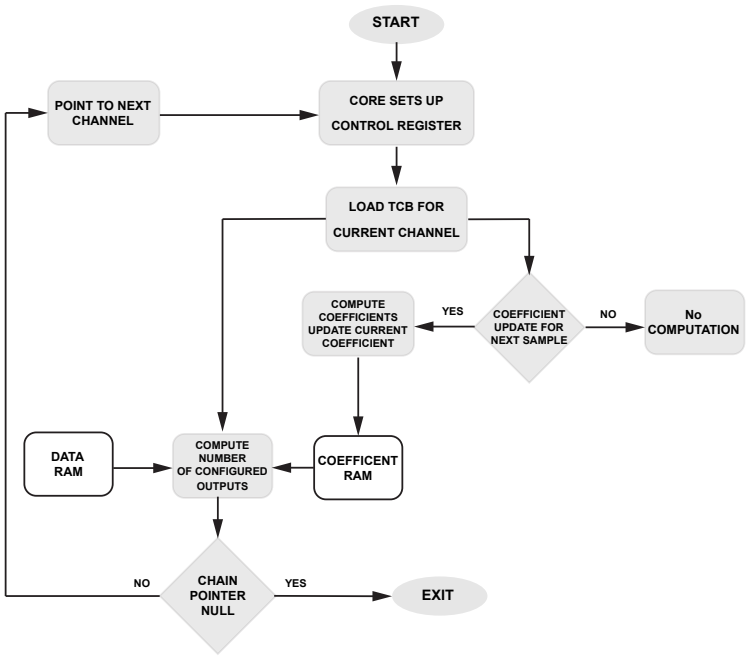


Figure 44-10: Slewing Operation Program Flow

Programming Constant Alpha Slewing Mode

To program slewing engine in constant alpha (α) mode, follow the below mentioned steps:

1. Enable coefficient slewing by setting the `IIR_CTL1.SLEW_EN` bit.
2. Provide the pointer to the target coefficient memory (`TCB+0x4`, IIR_CTIDX) along with current coefficient memory pointer (`TCB+0x3`, `IIR_COEFIDX`).
3. Set slewing rate (α) for the current channel using the 32-bit float format (`TCB+0x5`, `IIR_ALPHA[n]`).
4. Program the slewing frequency (n) and slewing time (N) values at (`TCB+0x6`, IIR_Nn) using the format in the `IIR_SLEW_CNFG[n]` register. Bits[0:7] should hold the value of slewing frequency (n) and bits[23:8] should hold the value of slewing time (N). The remaining bits must be zero.
5. Clear the `IIR_CTL2.SLEW_LALPHA_MODE` bit (`TCB+0xF`) to configure the slewing engine for constant alpha slewing mode.
6. When exact target coefficients are required after N iterations of slewing, set the `IIR_CTL2.SLEW_SAT_EN` bit to one.

NOTE: In constant alpha slewing mode, the window size should be a multiple of the slewing frequency (n+1).

Programming Linear Changing Alpha Slewing Mode

In this mode, the slewing rate (α) changes along with the coefficients after every iteration of slewing. During N slewing iterations, the value of alpha changes as (N-1)/N, (N-2)/N, … ,1/N, 0/N. Because α=0 for the last iteration of slewing, exact target coefficients are reached for each biquad.

To program the slewing engine in linearly changing alpha (α) mode, follow these steps:

1. Enable coefficient slewing by setting the `IIR_CTL1.SLEW_EN` bit.
2. Provide the pointer to the target coefficient memory (`TCB+0x4`, IIR_CTIDX) along with current coefficient memory pointer (`TCB+0x3`, `IIR_COEFIDX`).
3. Set slewing rate (α) for the current channel using the 32-bit float format (`TCB+0x5`, `IIR_ALPHA[n]`).
4. Program the slewing frequency (n) and slewing time (N) values at (`TCB+0x6`, IIR_Nn) using the format in the `IIR_SLEW_CNFG[n]` register. Bits[0:7] should hold the value of slewing frequency (n) and bits[23:8] should hold the value of slewing time (N). The remaining bits must be zero.
5. Set the `IIR_CTL2.SLEW_LALPHA_MODE` bit to enable linear varying alpha mode for this channel.
6. Program the slewing rate (α) in (`TCB+0x5`, `IIR_ALPHA[n]`) as 1/N in 32-bit float format. The slewing engine internally calculates the actual alpha to be used for each iteration.
7. The value of alpha is only updated at the start of the window. This restricts the slewing frequency (n) to a multiple of the window sample size. That is, `n = p*(window size-1)`, where p=1, 2, 3…

Programming Intermediate Target Coefficient Update Mode

Use Intermediate Target Coefficient Update Mode for achieving multiple target values for each coefficient after each slewing interval.

To program the slewing engine in Coefficient Update Mode, follow these steps:

1. Enable coefficient slewing by setting the `IIR_CTL1.SLEW_EN` bit.
2. Provide the pointer to the target coefficient memory (`TCB+0x4`, IIR_CTIDX) along with current coefficient memory pointer (`TCB+0x3`, `IIR_COEFIDX`).
3. Set slewing rate (α) for the current channel using the 32-bit float format (`TCB+0x5`, `IIR_ALPHA[n]`).
4. Program the slewing frequency (n) and slewing time (N) values at (`TCB+0x6`, IIR_Nn) using the format in the `IIR_SLEW_CNFG[n]` register. Bits[0:7] should hold the value of slewing frequency (n) and bits[23:8] should hold the value of slewing time (N). The remaining bits must be zero.
5. The slewing parameters slew rate(α), C_T, slewing time (N), and slewing frequency (n) can be updated during the slewing process.
6. When servicing a DMA interrupt, toggle the `IIR_CTL1.DMAEN` and `IIR_CTL1.SLEW_MAC_INT_EN` bits from zero to one to enable this mode.
7. When this mode is enabled, the IIR loads updated values of alpha, N, n, and the target coefficients.

ADSP-2184x IIR Register Descriptions

The IIR filter accelerator (IIR) contains the following registers.

Table 44-9: ADSP-2184x IIR Register List

Name	Description
IIR_ALPHA[n]	Channel n Alpha
IIR_CHNPTR	Chain Pointer Register
IIR_COEFIDX	Coefficient Buffer Index Register
IIR_COEFLEN	Coefficient Buffer Length Register
IIR_COEFMOD	Coefficient Index Modifier Register
IIR_CTL1	Global Control Register
IIR_CTL2	Channel Control Register
IIR_DBG_ADDR	IIR Debug Address Register
IIR_DBG_CTL	IIR Debug Control Register
IIR_DBG_RDDAT_HI	IIR Debug Read Data High Register
IIR_DBG_RDDAT_LO	IIR Debug Read Data Low Register
IIR_DBG_WRDAT_HI	IIR Debug Write Data High Register
IIR_DBG_WRDAT_LO	IIR Debug Write Data Low Register
IIR_DMASTAT	DMA Status Register
IIR_INBASE	Input Buffer Base Register
IIR_INIDX	Input Data Index Register
IIR_INLEN	Input Data Buffer Length Register
IIR_INMOD	Input Data Index Modifier Register
IIR_MACSTAT	MAC Status Register
IIR_OUTBASE	Output Buffer Base Register
IIR_OUTIDX	Output Data Buffer Index Register
IIR_OUTLEN	IIR Output Data Buffer Length Register
IIR_OUTMOD	IIR Output Data Index Modifier Register
IIR_SCTL1	Software Control Register1
IIR_SCTL2	Software Control Register2
IIR_SGCTL	Secondary Global Control Register
IIR_SLEW_CNFG[n]	Channel 0 Slewing Duration and Frequency

Channel n Alpha

The `IIR_ALPHA[n]` register gives access to Alpha, which is in 32-bit floating-point format. The register has both read and write access. However, a write to this register cannot be performed during non-IDLE state of DMA.

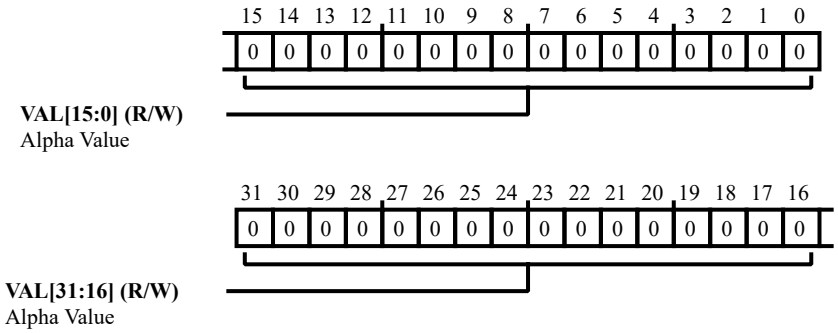


Figure 44-11: IIR_ALPHA[n] Register Diagram

Table 44-10: IIR_ALPHA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VAL	Alpha Value. The `IIR_ALPHA[n].VAL` bit field indicates the alpha value for channel n in slewing mode.

Chain Pointer Register

The `IIR_CHNPTR` register should be written with word address without the lower 2 bits.

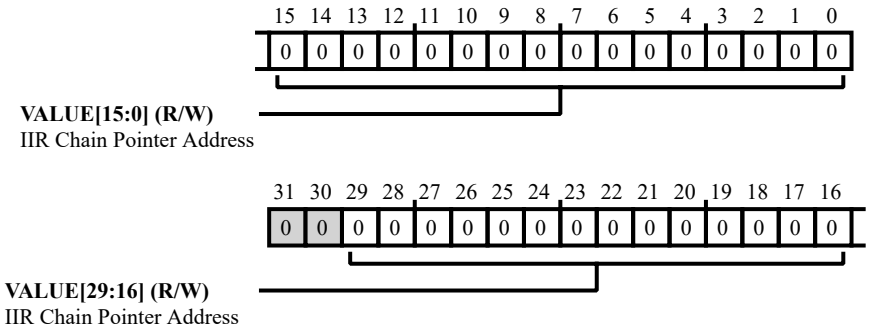


Figure 44-12: IIR_CHNPTR Register Diagram

Table 44-11: IIR_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	IIR Chain Pointer Address. The `IIR_CHNPTR.VALUE` bit field contains the chain pointer address.

Coefficient Buffer Index Register

The IIR_COEFIDX register contains the word address with the lower 2 bits removed.

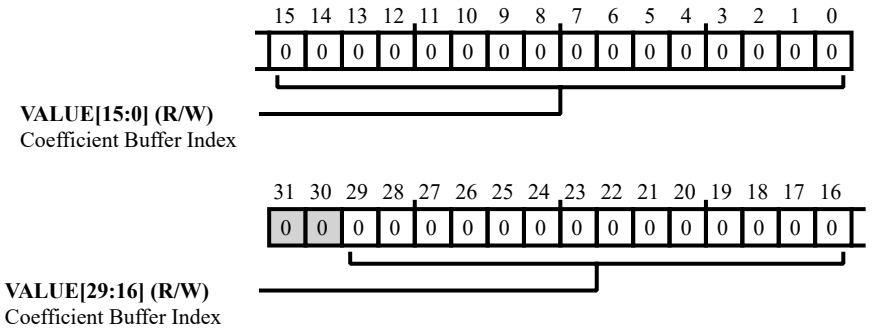


Figure 44-13: IIR_COEFIDX Register Diagram

Table 44-12: IIR_COEFIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Coefficient Buffer Index. The IIR_COEFIDX.VALUE bit field provides the coefficient buffer index.

Coefficient Buffer Length Register

The `IIR_COEFLEN` register provides the coefficient buffer length.

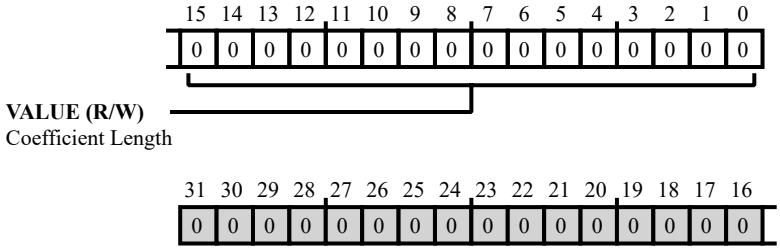


Figure 44-14: IIR_COEFLEN Register Diagram

Table 44-13: IIR_COEFLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Coefficient Length. The `IIR_COEFLEN.VALUE` bit field provides the coefficient buffer length.

Coefficient Index Modifier Register

The `IIR_COEFMOD` register provides the coefficient index modifier.

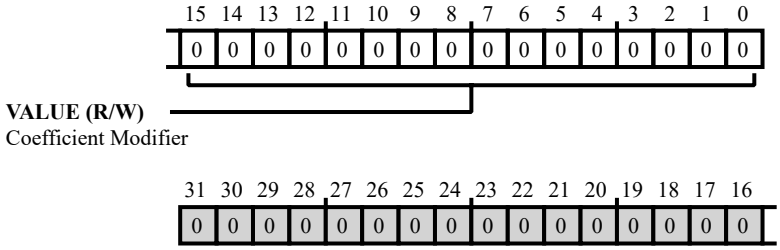


Figure 44-15: IIR_COEFMOD Register Diagram

Table 44-14: IIR_COEFMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Coefficient Modifier. The `IIR_COEFMOD.VALUE` bit field provides the coefficient modifier.

Global Control Register

The `IIR_CTL1` register is used to configure the global parameters for the accelerator. These parameters include the number of channels, channel auto iterate, DMA enable, and accelerator enable.

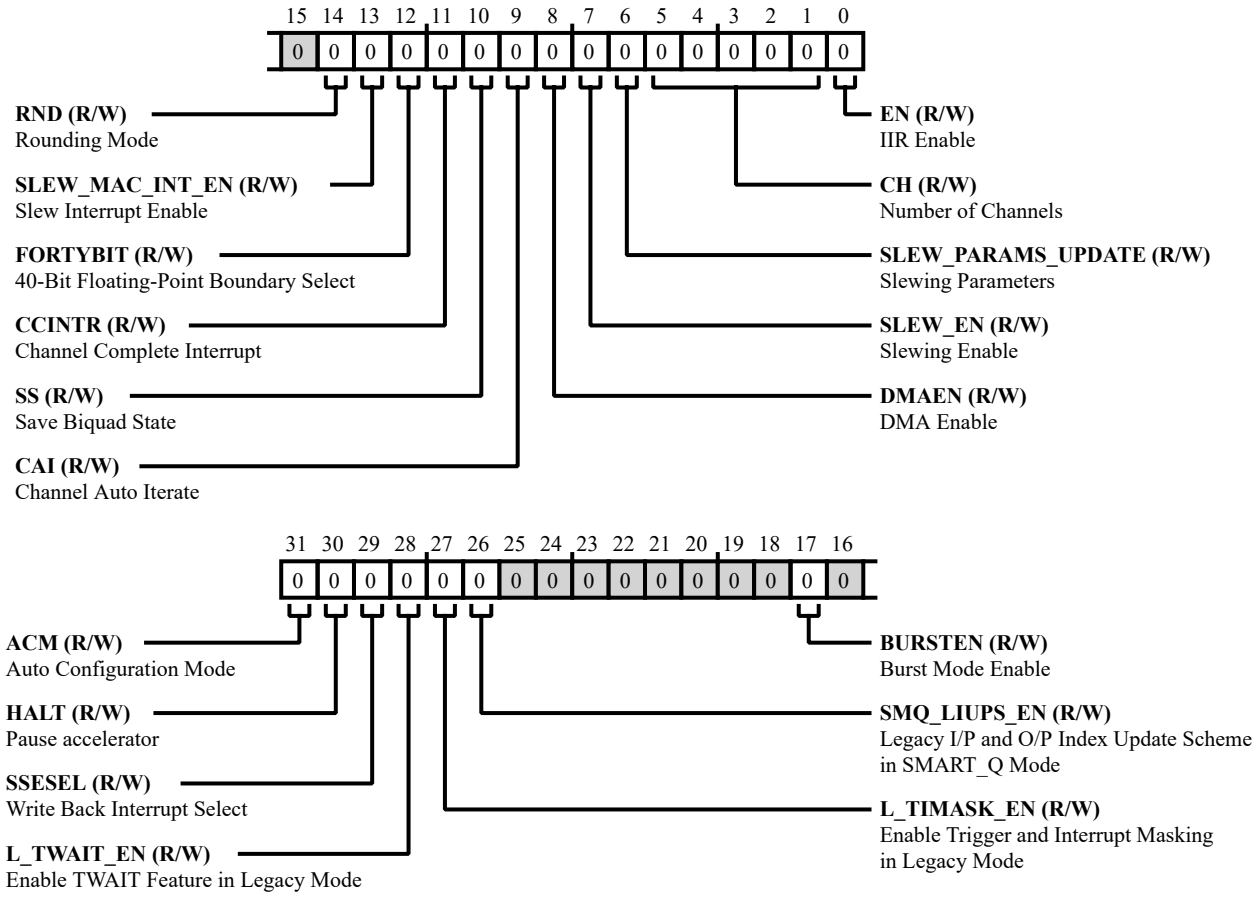


Figure 44-16: IIR_CTL1 Register Diagram

Table 44-15: IIR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ACM	Auto Configuration Mode. The `IIR_CTL1.ACM` bit configures the mode for loading the TCB.	
		0	Legacy Mode
		1	Auto Configuration Mode

Table 44-15: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	HALT	Pause accelerator. The `IIR_CTL1.HALT` bit determines whether the accelerator pauses so that the core can check the status and modify or submit a job or the accelerator is released for further processing of data. This bit is only valid in Auto Configuration Mode (ACM).	
		0	Release accelerator
		1	Pause accelerator
29 (R/W)	SSESEL	Write Back Interrupt Select. The `IIR_CTL1.SSESEL` bit selects whether to generate an interrupt after the write back or after save state completion.	
		0	Generate Interrupt after save state completion
		1	Generate interrupt after write back completion
28 (R/W)	L_TWAIT_EN	Enable TWAIT Feature in Legacy Mode. The `IIR_CTL1.L_TWAIT_EN` bit indicates whether the TWAIT feature is available in legacy mode. This bit is only valid in legacy mode.	
		0	Not Available
		1	Available
27 (R/W)	L_TIMASK_EN	Enable Trigger and Interrupt Masking in Legacy Mode. The `IIR_CTL1.L_TIMASK_EN` bit enables trigger and interrupt masking in legacy mode. This bit is only valid in legacy mode.	
		0	Disable
		1	Enable
26 (R/W)	SMQ_LIUPS_EN	Legacy I/P and O/P Index Update Scheme in SMART_Q Mode. The `IIR_CTL1.SMQ_LIUPS_EN` bit configures the scheme the accelerator uses to update the input index (II) and output index (OI). This bit is only valid in ACM.	
		0	Update the II with all '0' and the OI with all 'F', respectively
		1	Update II and OI after circular buffer scheme
17 (R/W)	BURSTEN	Burst Mode Enable. The `IIR_CTL1.BURSTEN` bit field is set, burst access is enabled.	
		0	Disable burst mode
		1	Enable burst mode

Table 44-15: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	RND	Rounding Mode. The `IIR_CTL1.RND` bit field selects the rounding mode for floating-point format.	
		0	Round to nearest (even)
		1	Truncate (Round towards zero)
13 (R/W)	SLEW_MAC_INT_EN	Slew Interrupt Enable. The `IIR_CTL1.SLEW_MAC_INT_EN` bit enables interrupts from slewing floating point units.	
12 (R/W)	FORTYBIT	40-Bit Floating-Point Boundary Select. The `IIR_CTL1.FORTYBIT` bit selects between 32-bit IEEE floating-point format or 40-bit IEEE floating-point format.	
		0	32-bit IEEE floating-point
		1	40-bit IEEE floating-point
11 (R/W)	CCINTR	Channel Complete Interrupt. The `IIR_CTL1.CCINTR` bit configures the channel complete interrupt to generate when all channels are done or after each channel is done. This bit is only valid in legacy mode when the IIR_CTL1.L_TIMASK_EN bit is cleared (=0). In Auto Configuration Mode (ACM) or when the IIR_CTL1.L_TIMASK_EN is set (=1), interrupt generation for each channel is controlled using IIR_CTL2.IMASK bit.	
		0	Interrupt is generated only when all channels are done (default)
		1	Interrupt is generated after each channel is done (default)
10 (R/W)	SS	Save Biquad State. The `IIR_CTL1.SS` bit configures the accelerator to store the Dk register settings into the internal memory. This can be used to save the biquad states before switching to another high priority accelerator task. This bit is only valid in legacy mode. In Auto Configuration Mode (ACM), the save state for each channel is controlled using the IIR_SGCTL.SS bit	

Table 44-15: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	CAI	Channel Auto Iterate. The `IIR_CTL1.CAI` bit sets whether TDM processing stops (idle) once all channels complete processing or moves to first channel and continues TDM processing in a loop when all channels complete processing. Channel Auto Iterate is not available in Auto Configuration Mode (ACM). The accelerator keeps processing the TCBs until the chain pointer becomes null in ACM.	
		0	TDM processing stops (idle) once all channels complete processing
		1	Moves to first channel and continues TDM processing in a loop when all channels complete processing
8 (R/W)	DMAEN	DMA Enable. The `IIR_CTL1.DMAEN` bit enables DMA on the accelerator.	
		0	Disable
		1	Enable
7 (R/W)	SLEW_EN	Slewing Enable. The `IIR_CTL1.SLEW_EN` bit enables control of IIR coefficients slewing engine.	
		0	Disable Slewing Engine (default)
		1	Enable Slewing Engine
6 (R/W)	SLEW_PARAMS_UP-DATE	Slewing Parameters. Slewing parameters (alpha, slewing duration, slewing frequency) and target coefficients can be updated during the process of slewing by enabling this `IIR_CTL1.SLEW_PARAMS_UPDATE` bit and by disabling and re-enabling the DMA.	
5:1 (R/W)	CH	Number of Channels. The `IIR_CTL1.CH` bit field configures the number of channels and is programmable between 0-31 (channels = NCH + 1). This bit field is only valid in legacy mode. In Auto Configuration Mode (ACM), there is no limit on the number of channels. the accelerator keeps processing the TCBs until the chain pointer becomes null.	
0 (R/W)	EN	IIR Enable. The `IIR_CTL1.EN` bit enables or disables the IIR accelerator.	
		0	IIR disabled
		1	IIR enabled

Channel Control Register

The `IIR_CTL2` register is used to configure the channel specific parameters. These parameters include the number of biquads and window size. In Auto Configuration Mode (ACM), this register is also used to configure additional channel specific parameters like interrupts and triggers.

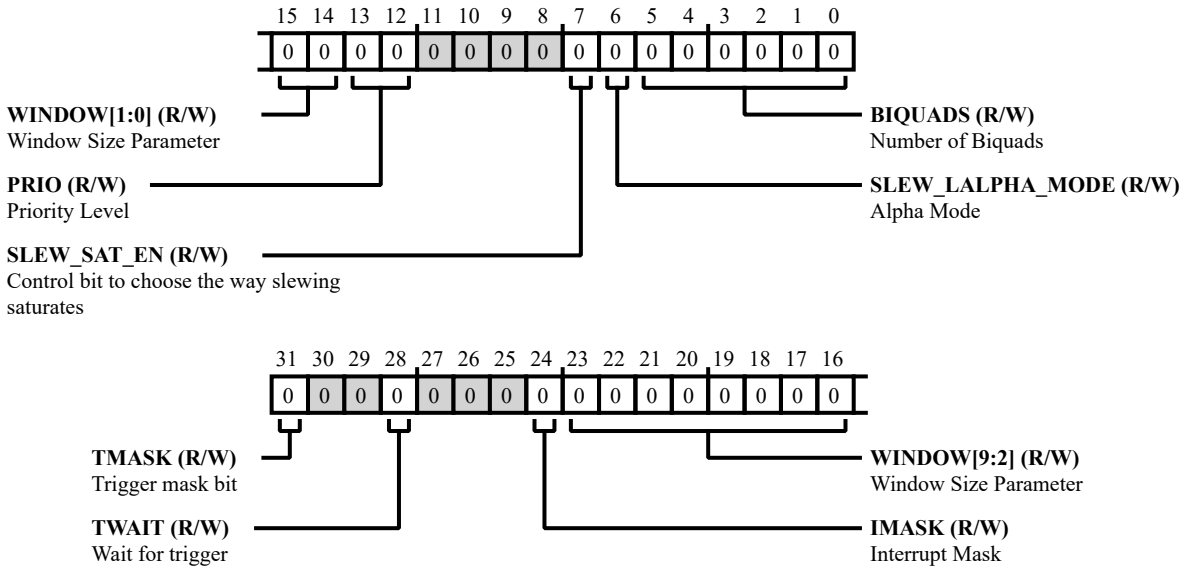


Figure 44-17: IIR_CTL2 Register Diagram

Table 44-16: IIR_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TMASK	Trigger mask bit. The `IIR_CTL2.TMASK` bit enables trigger generation for the channel. It is valid in Auto Configuration Mode (ACM) or if CTL1.L_TIMASK_EN is set.	
		0	Enable
		1	Mask
28 (R/W)	TWAIT	Wait for trigger. The `IIR_CTL2.TWAIT` bit disables the wait for the trigger. It is valid in Auto Configuration Mode (ACM) or if CTL1.L_TIMASK_EN is set.	
		0	Disable wait for the trigger for the channel
		1	Enable wait for external trigger input assertion for the channel

Table 44-16: IIR_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	IMASK	Interrupt Mask. The `IIR_CTL2.IMASK` bit enables interrupt generation for the channel. This bit is valid in Auto Configuration Mode (ACM) or if CTL1.L_TIMASK_EN is set.	
		0	Enable
		1	Mask
23:14 (R/W)	WINDOW	Window Size Parameter. The `IIR_CTL2.WINDOW` bit field configures the window size which specifies the number of samples/blocks to process (sample based processing = window size of 1). This bit field should be programmed to "actual window size required -1". For example, for sample based processing this bit field should be programmed to 0.	
13:12 (R/W)	PRIO	Priority Level. The `IIR_CTL2.PRIO` bit field indicates the priority.	
		0	Level 0 (lowest)
		1	Level 1
		2	Level 2
		3	Level 3 (highest)
7 (R/W)	SLEW_SAT_EN	Control bit to choose the way slewing saturates. `IIR_CTL2.SLEW_SAT_EN` bit field is set to HIGH to assume target coefficients after slewing duration. If this bit is not set, coefficients will remain at the last iteration values	
6 (R/W)	SLEW_LALPHA_MODE	Alpha Mode. The `IIR_CTL2.SLEW_LALPHA_MODE` bit controls whether to slew in constant alpha mode or linearly changing alpha mode. Program `IIR_CTL2.SLEW_LALPHA_MODE` bit field as, 0: Default setting; Constant alpha mode 1: Linearly changing alpha mode	
		0	Constant Alpha Mode (default)
		1	Linearly Changing Alpha Mode
5:0 (R/W)	BIQUADS	Number of Biquads. The `IIR_CTL2.BIQUADS` bit field configures the number of biquads and is programmable between 0-63 (number of Biquads = BIQUADS + 1).	

IIR Debug Address Register

The `IIR_DBG_ADDR` register holds the debug address. If bit 11 is set, coefficient memory is selected.

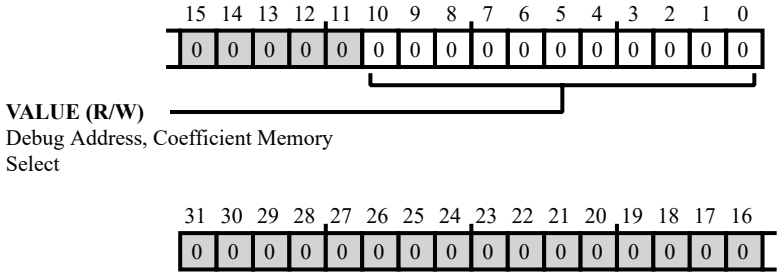


Figure 44-18: IIR_DBG_ADDR Register Diagram

Table 44-17: IIR_DBG_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Debug Address, Coefficient Memory Select. The `IIR_DBG_ADDR.VALUE` bit field holds the debug address (bits 0-10). Bit 11 configures whether the memory access is to coefficient memory (=0) or to delay line memory (=1).

IIR Debug Control Register

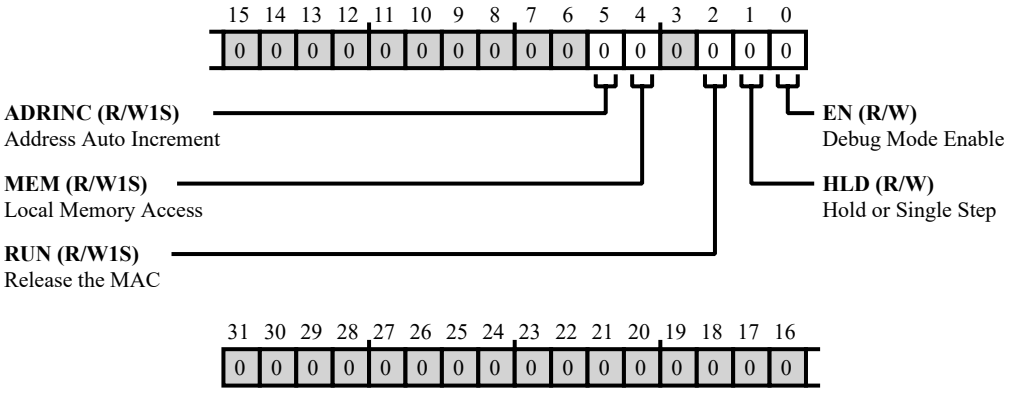


Figure 44-19: IIR_DBG_CTL Register Diagram

Table 44-18: IIR_DBG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1S)	ADRINC	Address Auto Increment. The `IIR_DBG_CTL.ADRINC` bit allows the address register to auto increment on `IIR_DBG_WRDAT_HI`/`IIR_DBG_WRDAT_LO` writes and `IIR_DBG_RDDAT_HI`/`IIR_DBG_RDDAT_LO` reads.	
4 (R/W1S)	MEM	Local Memory Access. The `IIR_DBG_CTL.MEM` bit allows the data and coefficients memory to be indirectly accessed.	
2 (R/W1S)	RUN	Release the MAC. The `IIR_DBG_CTL.RUN` bit releases the MAC; it is is self clearing after one IIR clock cycle.	
1 (R/W)	HLD	Hold or Single Step. The `IIR_DBG_CTL.HLD` bit function is based on the `IIR_DBG_CTL.MEM` bit setting. For `IIR_DBG_CTL.MEM` = 0 this bit sets single step. For `IIR_DBG_CTL.MEM` = 1 this bit sets hold data.	
		0	No effect
		1	Single step (IIR_DBGMEM=0) or Hold data (IIR_DBGMEM=1)
0 (R/W)	EN	Debug Mode Enable. The `IIR_DBG_CTL.EN` bit enables debug mode. For local memory access, the `IIR_CTL1` register can be cleared.	
		0	Disable
		1	Enable

IIR Debug Read Data High Register

The `IIR_DBG_RDDAT_HI` register is part of the 40-bit wide debug mode read data register and holds the upper 8 bits.

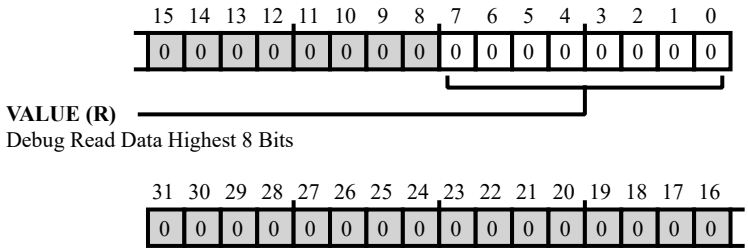


Figure 44-20: IIR_DBG_RDDAT_HI Register Diagram

Table 44-19: IIR_DBG_RDDAT_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VALUE	Debug Read Data Highest 8 Bits. The `IIR_DBG_RDDAT_HI.VALUE` bit field holds the upper 8-bit read data.

IIR Debug Read Data Low Register

The IIR_DBG_RDDAT_LO register is part of the 40-bit wide debug mode read data register and holds the lower 32 bits.

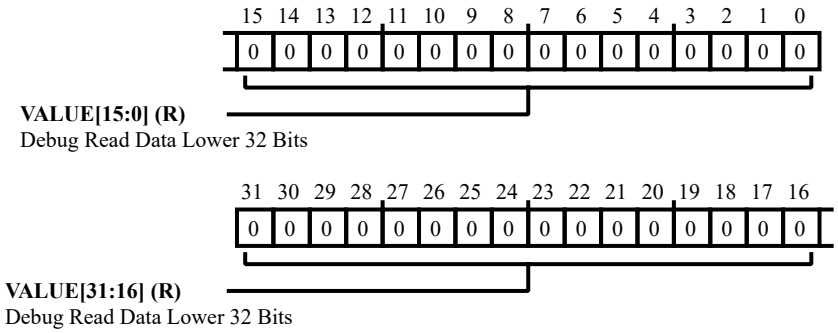


Figure 44-21: IIR_DBG_RDDAT_LO Register Diagram

Table 44-20: IIR_DBG_RDDAT_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Debug Read Data Lower 32 Bits. The IIR_DBG_RDDAT_LO.VALUE bit field holds the lower 32-bit read data.

IIR Debug Write Data High Register

The `IIR_DBG_WRDAT_HI` register is part of the 40-bit wide debug mode write data register and holds the upper 8 bits.

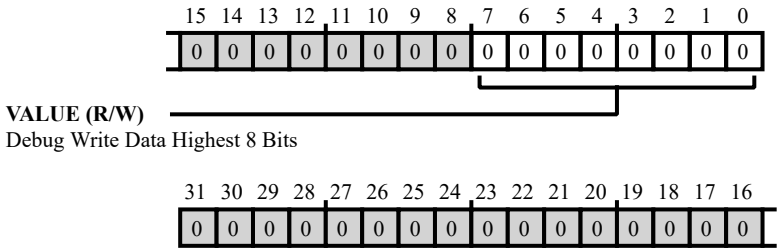


Figure 44-22: IIR_DBG_WRDAT_HI Register Diagram

Table 44-21: IIR_DBG_WRDAT_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Debug Write Data Highest 8 Bits. The `IIR_DBG_WRDAT_HI.VALUE` bit field holds the upper 8-bit write data.

IIR Debug Write Data Low Register

The IIR_DBG_WRDAT_LO register is part of the 40-bit wide debug mode write data register and holds the lower 32 bits.

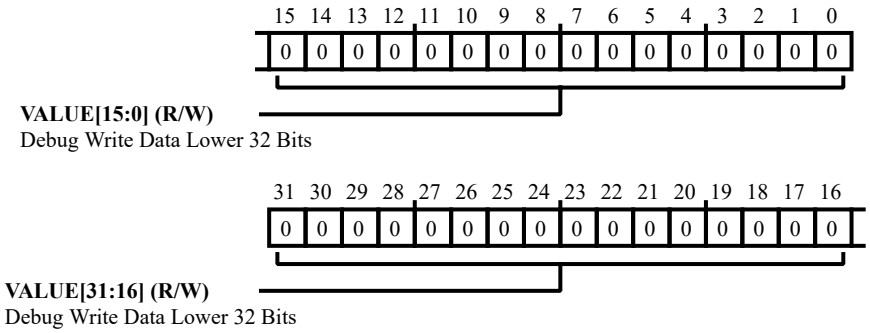


Figure 44-23: IIR_DBG_WRDAT_LO Register Diagram

Table 44-22: IIR_DBG_WRDAT_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Write Data Lower 32 Bits. The IIR_DBG_WRDAT_LO.VALUE bit field holds the lower 32-bit write data.

DMA Status Register

The IIR_DMASTAT registers indicate the status of DMA operations.

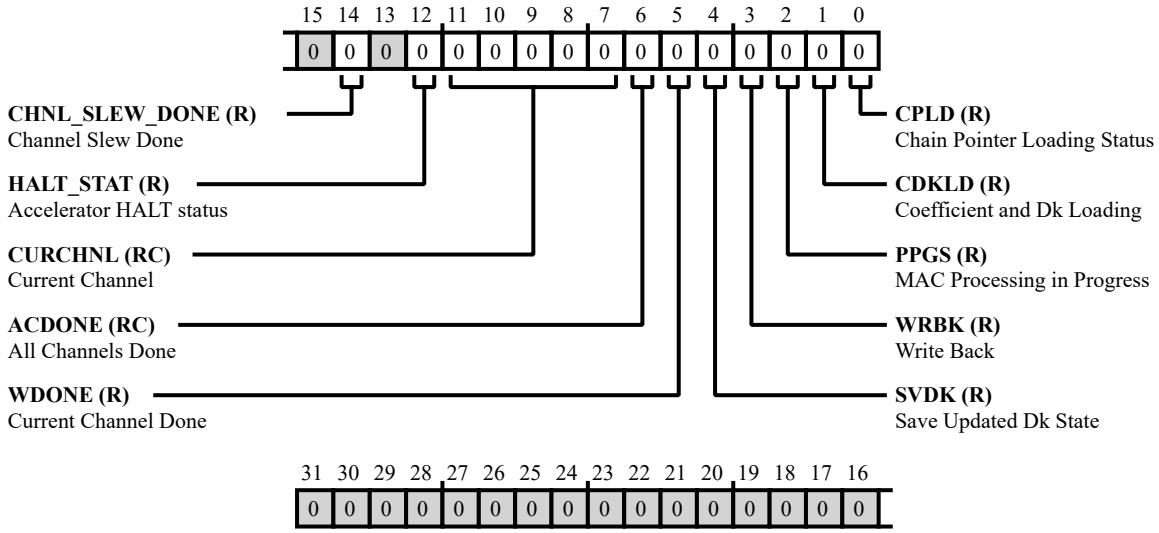


Figure 44-24: IIR_DMASTAT Register Diagram

Table 44-23: IIR_DMASTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	CHNL_SLEW_DONE	Channel Slew Done. The IIR_DMASTAT.CHNL_SLEW_DONE bit field indicates the completion of slewing (N iterations) in the current channel.
12 (R/NW)	HALT_STAT	Accelerator HALT status. Acknowledge to core that accelerator is in halt state, if set.
11:7 (RC/NW)	CURCHNL	Current Channel. The IIR_DMASTAT.CURCHNL bit field indicates the channel that is being processed in the TDM slot. Zero indicates the last slot.
6 (RC/NW)	ACDONE	All Channels Done. The IIR_DMASTAT.ACDONE bit indicates all channels are done processing. Note that the IIR_CTL1.CCINTR bit does not affect this status bit. This bit is sticky and is cleared on register read.
5 (R/NW)	WDONE	Current Channel Done. The IIR_DMASTAT.WDONE bit indicates the processing of the current channel is complete. Note that the IIR_CTL1.CCINTR bit does not affect this status bit. This bit is sticky and is cleared on register read.

Table 44-23: IIR_DMASTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	SVDK	Save Updated Dk State. If there is more than one channel (`IIR_CTL1.CH`>0), the `IIR_DMASTAT.SVDK` bit toggles between 0 and 1 as it starts and completes the save state operation on one channel at a time.	
3 (R/NW)	WRBK	Write Back. The `IIR_DMASTAT.WRBK` bit indicates the accelerator is writing back updated index registers.	
2 (R/NW)	PPGS	MAC Processing in Progress. The `IIR_DMASTAT.PPGS` bit indicates MAC processing is in progress.	
1 (R/NW)	CDKLD	Coefficient and Dk Loading. The `IIR_DMASTAT.CDKLD` bit indicates the coefficient and Dk are loading.	
0 (R/NW)	CPLD	Chain Pointer Loading Status. The `IIR_DMASTAT.CPLD` bit indicates the IIR is in the chain pointer load state.	
		0	State machine not in chain pointer load state
		1	State machine in chain pointer load state

Input Buffer Base Register

The IIR_INBASE register contains the word address with the lower 2 bits removed.

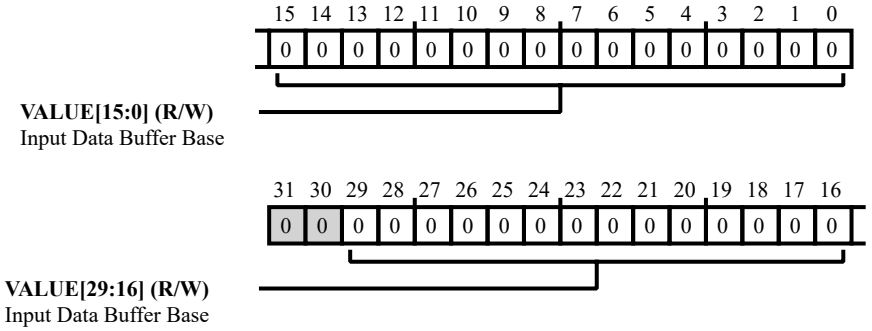


Figure 44-25: IIR_INBASE Register Diagram

Table 44-24: IIR_INBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Input Data Buffer Base. The IIR_INBASE.VALUE bit field value is the input data buffer base address.

Input Data Index Register

The IIR_INIDX register contains a word address with the lower 2 bits removed.

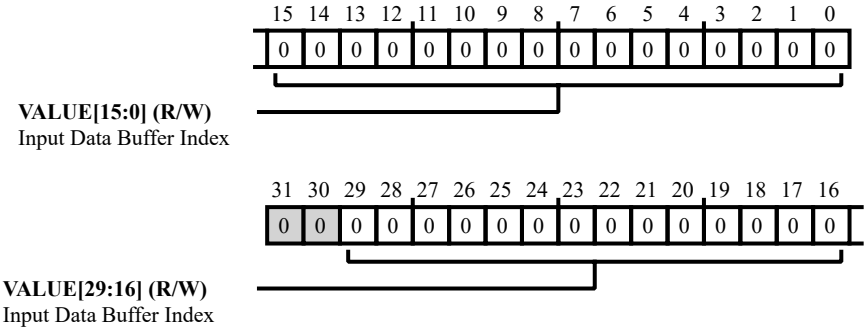


Figure 44-26: IIR_INIDX Register Diagram

Table 44-25: IIR_INIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Input Data Buffer Index. The IIR_INIDX.VALUE bit field value is the input data buffer index.

Input Data Buffer Length Register

The `IIR_INLEN` register provides the input data buffer length.

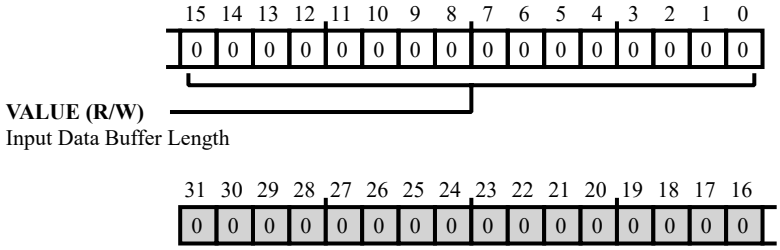


Figure 44-27: IIR_INLEN Register Diagram

Table 44-26: IIR_INLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Input Data Buffer Length. The `IIR_INLEN.VALUE` bit field value is the input data buffer length.

Input Data Index Modifier Register

The IIR_INMOD register provides the 16-bit input data buffer index modifier.

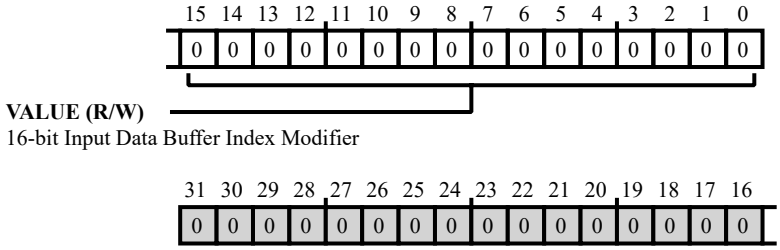


Figure 44-28: IIR_INMOD Register Diagram

Table 44-27: IIR_INMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Index Modifier. The IIR_INMOD.VALUE bit field value is the 16-bit input data buffer modifier.

MAC Status Register

The IIR_MACSTAT register indicates the status of MAC operations.

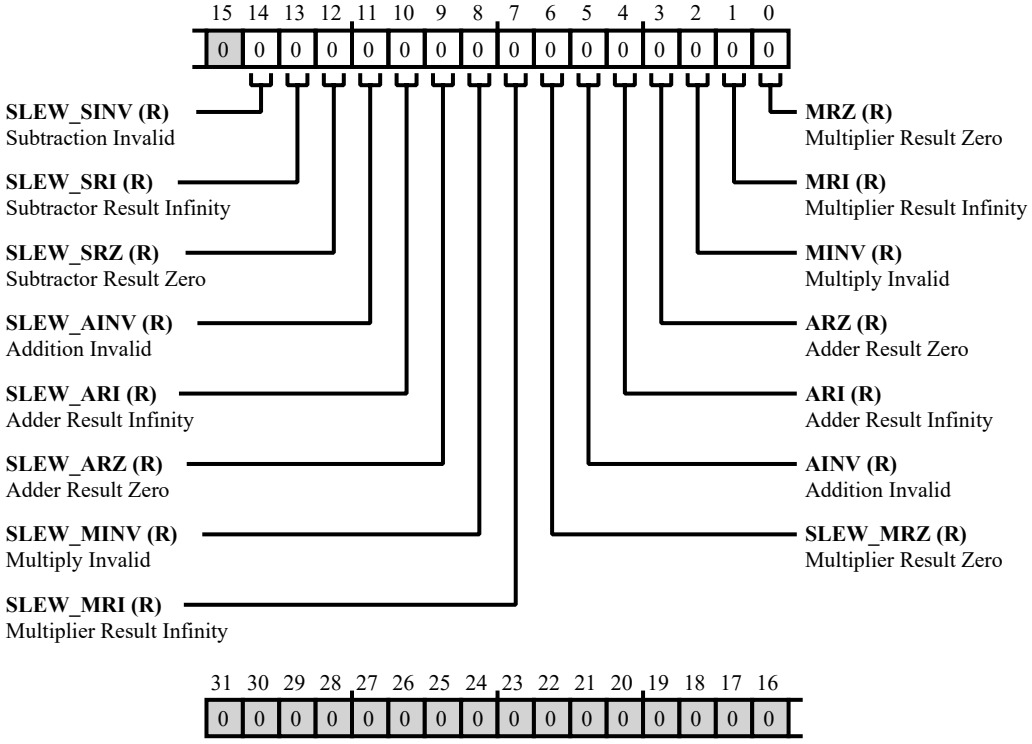


Figure 44-29: IIR_MACSTAT Register Diagram

Table 44-28: IIR_MACSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	SLEW_SINV	Subtraction Invalid. The IIR_MACSTAT.SLEW_SINV bit indicates the subtraction is invalid.
13 (R/NW)	SLEW_SRI	Subtractor Result Infinity. The IIR_MACSTAT.SLEW_SRI bit indicates the subtractor result is infinity.
12 (R/NW)	SLEW_SRZ	Subtractor Result Zero. The IIR_MACSTAT.SLEW_SRZ bit indicates the subtractor result is zero.
11 (R/NW)	SLEW_AINV	Addition Invalid. The IIR_MACSTAT.SLEW_AINV bit indicates the addition is invalid.
10 (R/NW)	SLEW_ARI	Adder Result Infinity. The IIR_MACSTAT.SLEW_ARI bit indicates the adder result is infinity.

Table 44-28: IIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/NW)	SLEW_ARZ	Adder Result Zero. The `IIR_MACSTAT.SLEW_ARZ` bit indicates the adder result is zero.
8 (R/NW)	SLEW_MINV	Multiply Invalid. The `IIR_MACSTAT.SLEW_MINV` bit indicates the multiply operation is invalid.
7 (R/NW)	SLEW_MRI	Multiplier Result Infinity. The `IIR_MACSTAT.SLEW_MRI` bit indicates the multiplier result is infinity.
6 (R/NW)	SLEW_MRZ	Multiplier Result Zero. The `IIR_MACSTAT.SLEW_MRZ` bit indicates the multiplier result is zero.
5 (R/NW)	AINV	Addition Invalid. The `IIR_MACSTAT.AINV` bit indicates the addition is invalid.
4 (R/NW)	ARI	Adder Result Infinity. The `IIR_MACSTAT.ARI` bit indicates the adder result is infinity.
3 (R/NW)	ARZ	Adder Result Zero. The `IIR_MACSTAT.ARZ` bit indicates the adder result is zero.
2 (R/NW)	MINV	Multiply Invalid. The `IIR_MACSTAT.MINV` bit indicates the multiply operation is invalid.
1 (R/NW)	MRI	Multiplier Result Infinity. The `IIR_MACSTAT.MRI` bit indicates the multiplier result is infinity.
0 (R/NW)	MRZ	Multiplier Result Zero. The `IIR_MACSTAT.MRZ` bit indicates the multiplier result is zero.

Output Buffer Base Register

The `IIR_OUTBASE` register contains the word address with the lower 2 bits removed.

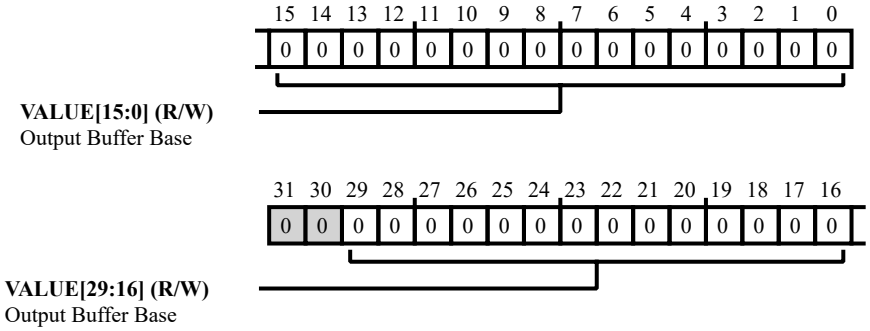


Figure 44-30: IIR_OUTBASE Register Diagram

Table 44-29: IIR_OUTBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Output Buffer Base. The `IIR_OUTBASE.VALUE` bit field provides the output buffer base address.

Output Data Buffer Index Register

The IIR_OUTIDX register is written with word address without the lower 2 bits.

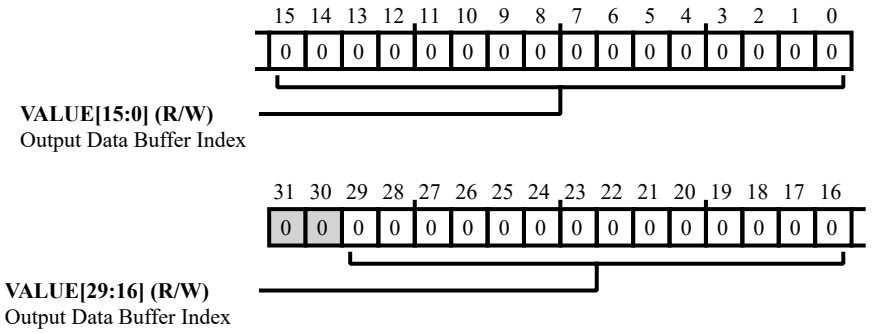


Figure 44-31: IIR_OUTIDX Register Diagram

Table 44-30: IIR_OUTIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Output Data Buffer Index. The IIR_OUTIDX.VALUE bit field provides the output data buffer index.

IIR Output Data Buffer Length Register

The `IIR_OUTLEN` register provides the output data buffer length.

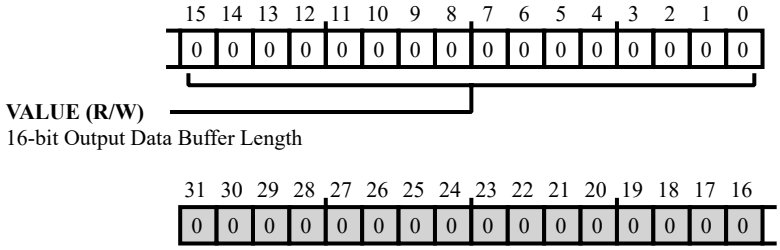


Figure 44-32: IIR_OUTLEN Register Diagram

Table 44-31: IIR_OUTLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Data Buffer Length. The `IIR_OUTLEN.VALUE` bit field provides the output data buffer length.

IIR Output Data Index Modifier Register

The `IIR_OUTMOD` register provides the output data index modifier.

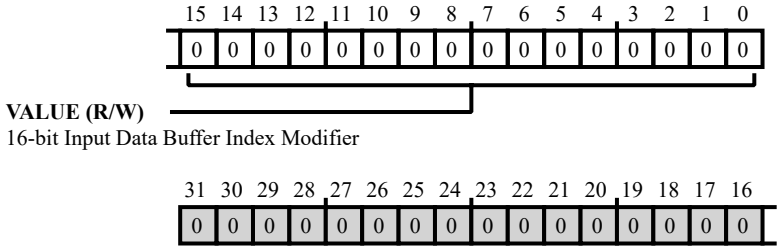


Figure 44-33: IIR_OUTMOD Register Diagram

Table 44-32: IIR_OUTMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Index Modifier. The `IIR_OUTMOD.VALUE` bit field provides the output data buffer index modifier.

Software Control Register1

The `IIR_SCTL1` register is used by Smart Queuing APIs in ACM mode to exchange relevant job information amongst APIs.

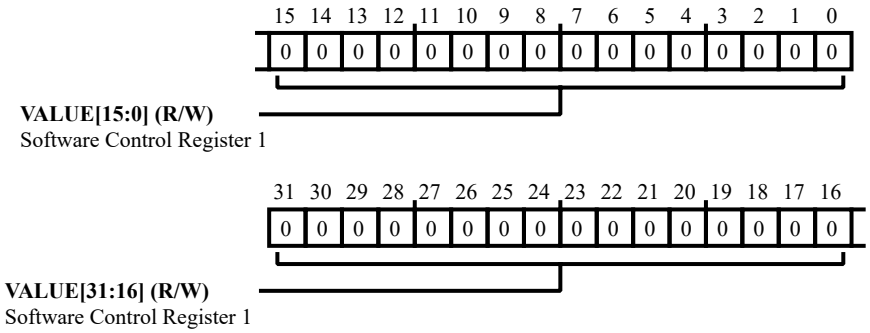


Figure 44-34: IIR_SCTL1 Register Diagram

Table 44-33: IIR_SCTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software Control Register 1.

Software Control Register2

The `IIR_SCTL2` register is used by Smart Queuing APIs in ACM mode to exchange relevant job information amongst APIs.

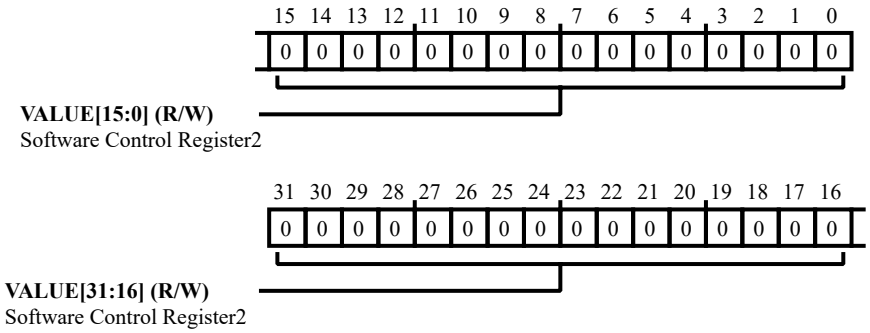


Figure 44-35: IIR_SCTL2 Register Diagram

Table 44-34: IIR_SCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software Control Register2.

Secondary Global Control Register

The `IIR_SGCTL` register configures the global parameters for the accelerator in ACM mode for loading CTL1 register as part of TCB.

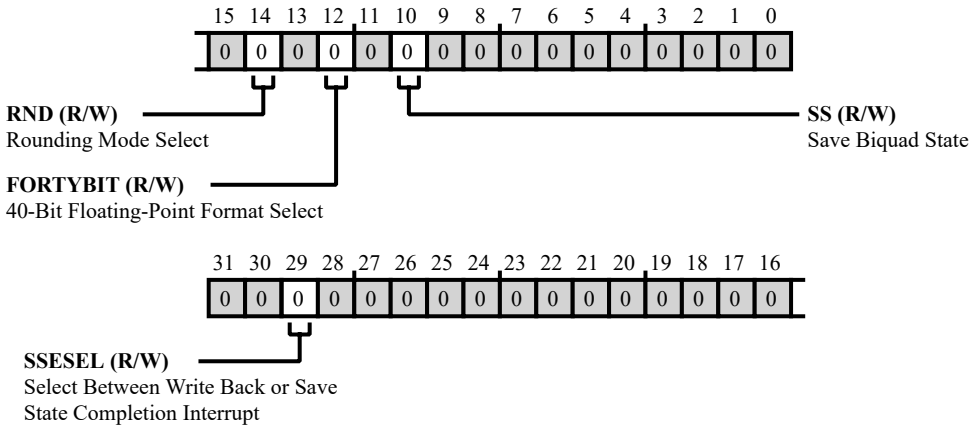


Figure 44-36: IIR_SGCTL Register Diagram

Table 44-35: IIR_SGCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SSESEL	Select Between Write Back or Save State Completion Interrupt.	
14 (R/W)	RND	Rounding Mode Select. The `IIR_SGCTL.RND` bit selects the rounding mode for a floating-point mode compatible with SHARC+.	
		0	Round to nearest (even)
		1	Truncate (Round towards zero)
12 (R/W)	FORTYBIT	40-Bit Floating-Point Format Select. The `IIR_SGCTL.FORTYBIT` bit selects the floating point format.	
		0	32-bit floating point
		1	40-bit floating point
10 (R/W)	SS	Save Biquad State. The `IIR_SGCTL.SS` bit configures the accelerator to store the Dk register settings into the internal memory. This can be used to save the biquad states before switching to another high priority accelerator task.	

Channel 0 Slewing Duration and Frequency

The `IIR_SLEW_CNFG[n]` registers give access to the slewing duration, slewing frequency, and counter values of the particular channel. The register has both read and write access. However, a write to this register cannot be performed during non-IDLE state of DMA.

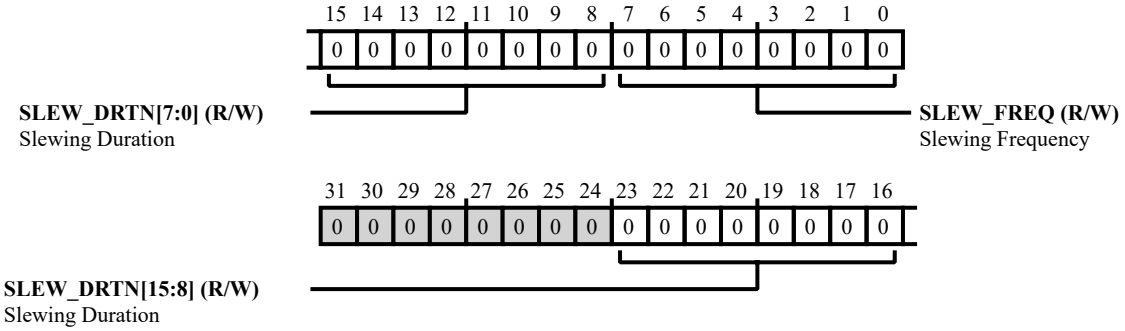


Figure 44-37: IIR_SLEW_CNFG[n] Register Diagram

Table 44-36: IIR_SLEW_CNFG[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:8 (R/W)	SLEW_DRTN	Slewing Duration. The `IIR_SLEW_CNFG[n].SLEW_DRTN` bit field programs number of samples processing during which slewing is done.
7:0 (R/W)	SLEW_FREQ	Slewing Frequency. The `IIR_SLEW_CNFG[n].SLEW_FREQ` bit field indicates the number of samples after which coefficients slewing is performed.

45 System Crossbars (SCB)

A modern system on a chip (SoCs) contains multi-cores, memory controllers, security blocks, and other high speed peripherals. As system integration increases, SoCs need to provide bus connectivity that allows better throughput to reduce performance bottlenecks. While traditional point-to-point connection buses have performed well in smaller systems, there is a need to use advanced switching based bus architectures for efficient handling of data transfer between multiplicity of data sources and sinks in the system. Additionally, mixing various traffic types in the same SoC (for example control, communication over peripherals and computing) while sharing the same bus resources, create different requirements from the Quality of Service (QoS) perspective.

The system crossbars (SCB) are the fundamental building blocks of a switch-fabric style for (on-chip) system bus interconnection. The SCBs connect system bus requesters to system bus completers, providing concurrent data transfer between multiple bus requesters and multiple bus completers. The SCB architecture addresses the challenges described above. The SCB provides sustainable throughput for simultaneous transactions in the system with configurable quality of service for each type of transaction (traffic) as required. A hierarchical model, built from multiple SCBs, provides a power and area efficient system interconnect, which satisfies the performance and flexibility requirements of a specific system.

SCB Features

The SCBs provide the following features:

- Efficient, pipelined bus transfer protocol for sustained throughput
- Full-duplex bus operation for flexibility and reduced latency
- Concurrent bus transfer support to allow multiple bus requesters to access bus completers simultaneously
- Simple priority-based QoS based arbitration model
- Programmable quality of service

SCB Functional Description

The following sections provide a functional description of the SCB.

- SCB Architectural Concepts

SCB Architectural Concepts

This section describes the components of the SCB and the modules connected to it. The basic elements in the SCB are SCB requesters, completers, requester interfaces, and completer interfaces.

Requesters

The system bus requesters include the peripheral Direct Memory Access (DMA) channels. These include the Serial Port (SPORT) DMA, SPI DMAs, Memory-to-Memory DMA channels (MDMA), the L1 code fill block, and the processor cores.

Completers

Completers are SCB connections that are responding to transfer requests. Completers include MMR registers, memory units, and various peripherals depending upon individual configurations. Each system completer has its own latencies and response times.

SCB Block Diagram

The SCB architectural model is shown in the ***SCB Overview*** figure. This figure shows a high level representation of a basic SCB connecting n completers to x requesters. A variable number of requesters connect to a variable number of completers in each SCB. In this example, all SIs connect to all MIs as indicated by the lines connecting them.

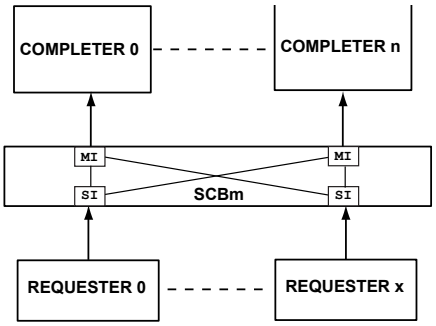


Figure 45-1: SCB Overview

Hierarchy Block Diagram

A system interconnect built from multiple SCBs in a hierarchical model is illustrated in the ***SCB Hierarchy Overview*** figure. The system requester node level SCBs requester connects multiple SIs to a single MI, which in turn connects to an SI of the system completer level node SCB.

As discussed above, all the requesters in the system are distributed across different SCBs. A given SCB at system requester node level connects directly to the system requesters. These SCBs connect to SCB0 through its SIs forming a hierarchical structure. While a requester must access any completer, its first access goes through the SCB it is connected to, and then through SCB0, to reach its intended completer. This simplifies the connecting hardware in the basic SCB block by limiting the requesters. Care must be taken when sharing requesters to allow adequate throughput for their individual data transfer requirements.

In this example, all SIs are connected to all MIs.

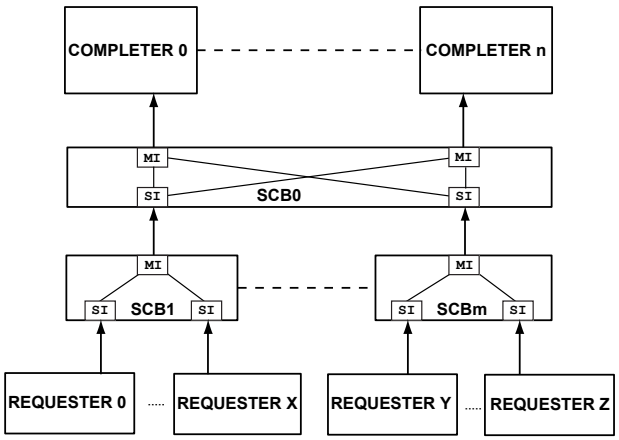


Figure 45-2: SCB Hierarchy Overview

NOTE: For an overall diagram of all SCB interconnections, see the SCB Block Diagram.

SCB Block Diagram

The ***SCB Block Diagram*** shows the functional blocks of the SCB module.

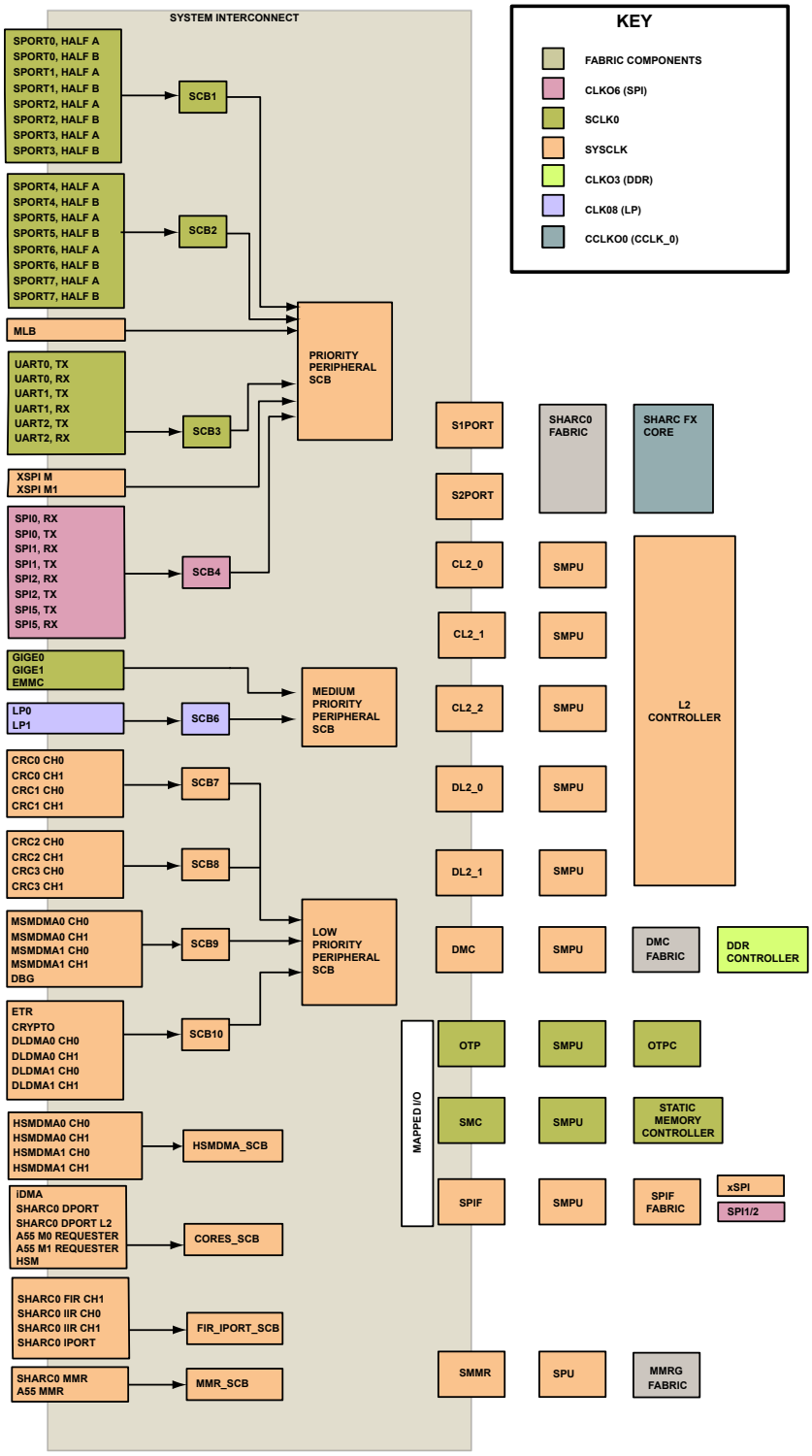


Figure 45-3: SCB Block Diagram

The following are important points related to the system fabric on the ADSP-SC84x processors.

- The hierarchy of SCBs manages the system bus interconnections, multiplexing, and arbitration among the peripherals on the processor.
- The SCBs connections support DMA channels for some peripherals and support dedicated connections for others. The connections also support memory-mapped register access for internal memory (L1 and L2) and for external memory.

  The completer interface of the crossbar where requesters such as DMA connect to performs two functions. The first function is arbitration. SCBx handles arbitration. The second function is clock conversion. The programmable QoS registers can be viewed as being associated with the SCBx.
- Most of the peripherals and the SCBs are in the *SCLK0* domain. MDMA0-7, EMDMA0/1, Crypto, FIR/IIR accelerators, MLB, DBG, ETR, SHARCFX and Arm cores and their SCBs are in the *SYSCLK* domain. The link port is in the CLK08 domain. SPI is in the CLK06 domain.
- MSMDMA0, MSMDMA1, HSMDMA0 and HSMDMA1 channels are unidirectional. For example, MSMDMA0 CH0 is read only and MSMDMA0 CH1 is write only.
- Access to the GPV (Global Programmers View) space is allowed only in secure mode. This space has registers for programming the QoS and CDC relationships and the remapping of various requesters.
- The S2 completer port of SHARC0 is intended primarily for MDMA. Hence, only the HSMDMA0 (MDMA3) and HSMDMA1 (MDMA7) is given access to S2 ports while all other requesters are given access to S1 port. This arrangement also means that S1 and S2 port share the same address space.
- LP to SPI2/xSPI access is allowed.

IMPORTANT: The following points are important changes from ADSP-2159x processors:

- SHARC-FX DPORT accesses L2 RAM and ROM over CL2_0 and CL2_1 are through a dedicated requester interface (SHARC0 DPORT L2); accesses to the rest of the system are over SHARC0 DPORT L2.
- Other new requesters are iDMA (from FX core), A55 data requester ports (a55_m0s_axi, a55_m1_axi), A55 MMR requester port (a55_mmr), SPI5(TX), SPI5(RX), xSPI requester port (xspi_m, xspi_m1), HSM, and eMSI.
- All requesters that can access DMC have been given access to SPI/xSPI.
- The SPIF completer in the system fabric has remapping options in the SPIF fabric for SPI[n] and xSPI in the same address space. The address map configuration is programmable using the remap registers.
- SH0_FIR_CH0 is moved to the HSMDMA SCB, and access to L2 is through dl2_0.

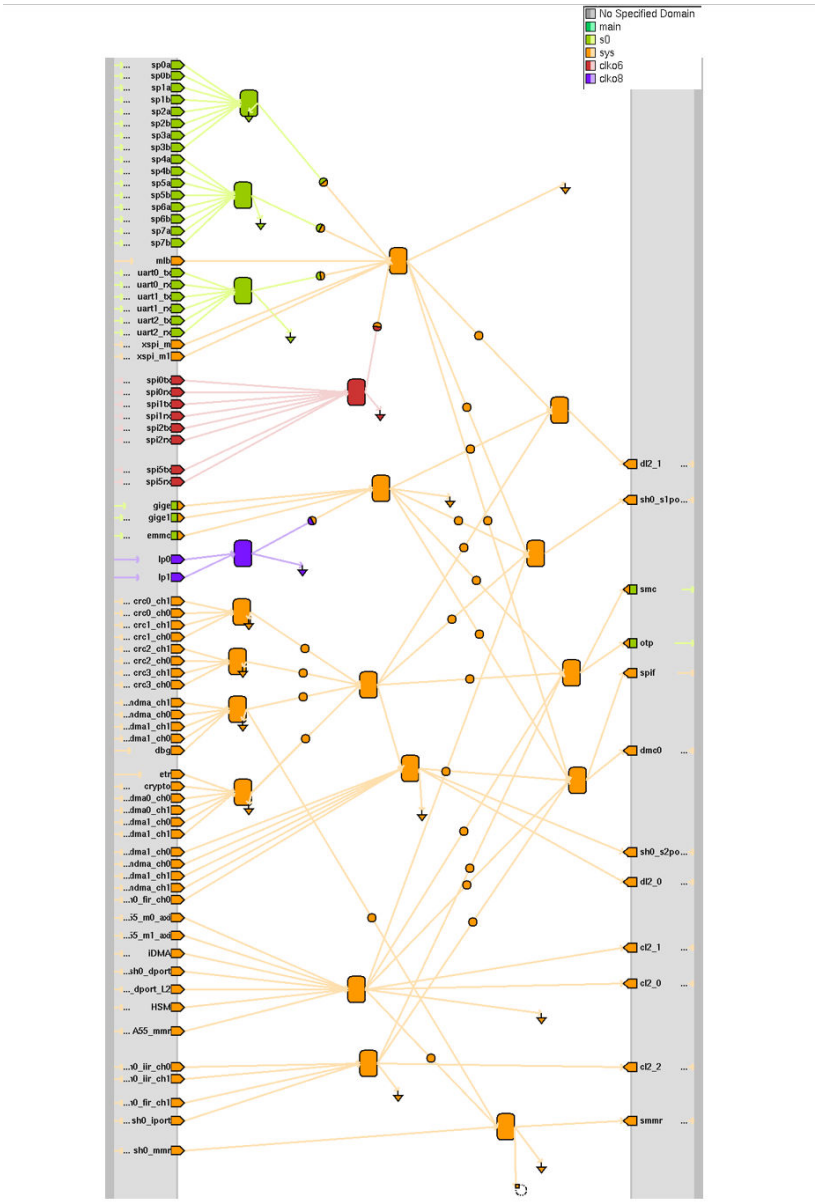


Figure 45-4: SCB Interconnections

The following acronyms are used in the ***SCB Interconnections*** figure.

SCB0-10

Indicate SCB interfaces, connecting the system bus requesters and completers

SCLK0, SYSCLK, CLK02, CLK06, CLK08

Indicate clock domains in which the specific SCBs operate. For more information on clock domains, see the Clock Generation Unit (CGU) chapter and the product data sheet.

iDMA

Indicates the integrated DMA in the SHARC-FX core

xSPI

Indicates the SPI hyperbus

A55

Indicates the ARM Coretex A55 microprocessor

Hardware Security Module CMRT-641

Indicates a 128-bit L2 completer

DL2_0, DL2_1

Indicates a 128-bit L2 completer

L1C0_S1

Indicates the S1 completer of the SHARC0 processor

L1C0_S2

Indicates the S2 completer of the SHARC0 processor

SHARC0_DPORT

Indicates the SHARC0 processor instruction requester port

SHARC0_IPORT

Indicates the SHARC0 processor instruction requester port

SHARC0_MMR

Indicates the SHARC0 processor MMR interface

MSMDMA

Indicates medium speed memory to memory DMA

HSMDMA

Indicates high speed memory to memory DMA

SHARC0, SHARC, SHARCFX

Indicates the SHARC-FX core (used interchangeably)

QoS

Indicates quality of service

SHARC Fabric with FIR/IIR Accelerator

The accelerator block on the SHARC core of the processor has two FIR and four IIR accelerators. All instances of the FIR and IIR accelerators operate at the core clock (CCLK) frequency. Accelerator requester ports can directly access the SHARC L1 memory with reduced latency. The access does not go through the system fabric. The SHARC core can also directly access the MMR registers of its accelerator.

The ***SHARC Fabric Connectivity*** figure shows a block diagram of the SHARC fabric with FIR/IIR accelerators integrated closely with the SHARC core.

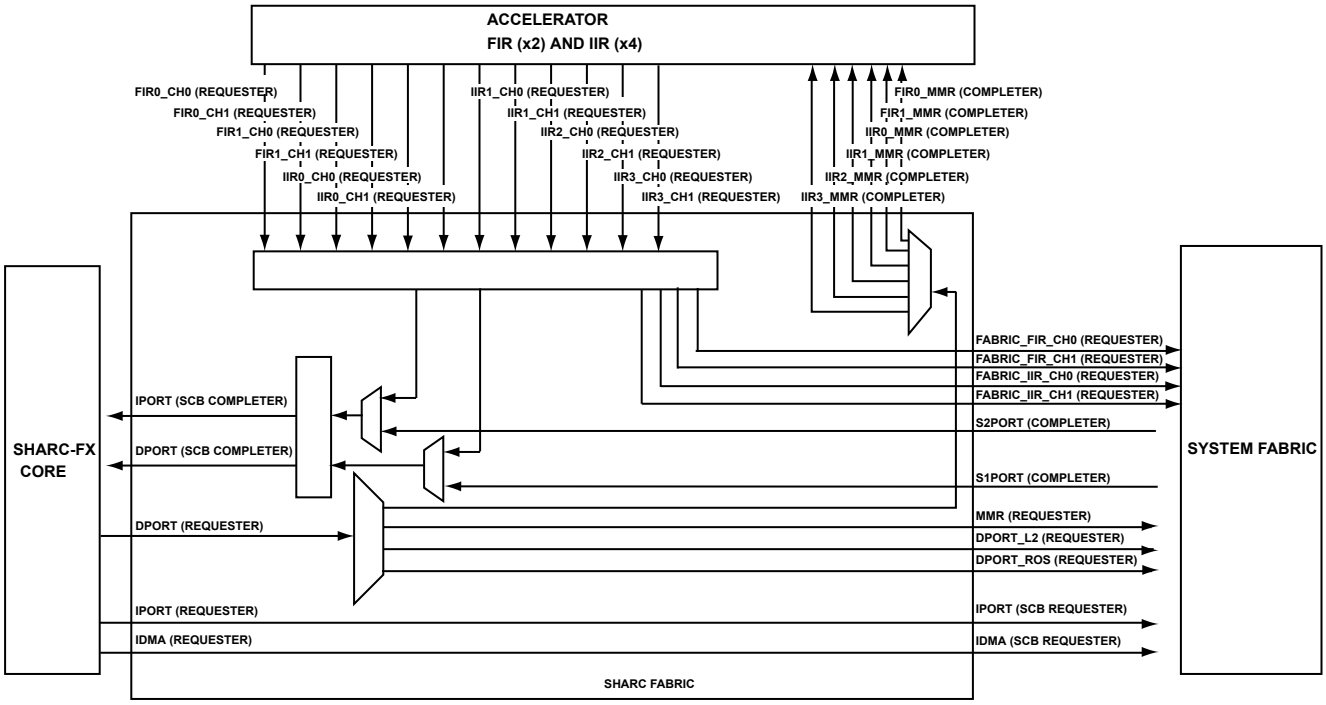


Figure 45-5: SHARC Fabric Connectivity

Port list and Parameters

Requester Ports (AXI-M)

- Data-Requester

  Inst-Requester

- iDMA

Completer Ports (AXI-S)

- DRAM-Completer
- IRAM-Completer

To ensure complete bandwidth utilization and optimal performance of the L2 ports, the controllers are allocated as shown in ***L2 Port Requester Allocation*** table.

Table 45-1: L2 Port Requester Allocation

Requesters	CL2_0	CL2_1	CL2_2	DL2_0	DL2_1
SHARC0 DPORT	✓	✓			
SHARC0 iDMA	✓	✓			
SHARC0 IPORT			✓		
A55 M0 AXI		✓			
A55 M1 AXI	✓				
HSM	✓	✓			
SHARC0_FIR_CH0				✓	
SHARC0_FIR_CH1			✓		
SHARC0_IIR_CH0			✓		
SHARC0_IIR_CH1			✓		
HSMDMA0				✓	
HSMDMA1				✓	
Peripherals					✓

Note the following:

- To reduce the probability of port conflict in a multicore system, FX, A55, HSM, and iDMA are connected on the CL2_0 and CL2_1 ports of L2. CL2_0 has access only to the first 2MB of L2. CL2_1 has access to second 2MB of L2
- For these requesters, access to the initial 2MB of L2 RAM and ROM goes through the CL2_0 port; the later 2MB of L2 RAM goes through the CL2_1 port
- SHARC-FX IPORT and Acceelerators (FIRs and IIRs) are connected over CL2_2 port.
- To increase FIR B/w from L2, FIR_CH0 is connected to DL2_0.

The addressable range for each of the ports is shown in the ***L2 Port Address Allocation*** table. SMPU 2 and 4 support exclusive access over CL2_0 and CL2_1 respectively, between SHARC FX DPORT and A55.

Table 45-2: L2 Port Address Allocation

SMPU Instance	Memory Address Range		EX- Acc	Comments
	Start Address	End Address		
CL2_0	0x2020_0000	0x2022_FFFF	Yes	ROM
	0x2040_0000	0x205F_FFFF	Yes	RAM
DL2_0	0x2000_0000	0x2022_FFFF	No	ROM
	0x2040_0000	0x207F_FFFF	No	RAM
CL2_1	0x2060_0000	0x207F_FFFF	Yes	RAM
CL2_2	0x2020_0000	0x2022_FFFF	No	ROM
	0x2040_0000	0x207F_FFFF	No	RAM
DL2_1	0x2020_0000	0x2022_FFFF	No	ROM
	0x2040_0000	0x207F_FFFF	No	RAM

Table 45-3: SCB Controlled DMA Channel Peripherals

	Requesters	DDE DMA Channels	Non DDE DMA Channels
SCB1	SPORT0, HALF A	DMA0	
	SPORT0, HALF B	DMA1	
	SPORT1, HALF A	DMA2	
	SPORT1, HALF B	DMA3	
	SPORT2, HALF A	DMA4	
	SPORT2, HALF B	DMA5	
	SPORT3, HALF A	DMA6	
	SPORT3, HALF B	DMA7	
SCB2	SPORT4, HALF A	DMA10	
	SPORT4, HALF B	DMA11	
	SPORT5, HALF A	DMA12	
	SPORT5, HALF B	DMA13	
	SPORT6, HALF A	DMA14	
	SPORT6, HALF B	DMA15	
	SPORT7, HALF A	DMA16	
	SPORT7, HALF B	DMA17	
SCB3	UART0, TX	DMA20	

Table 45-3: SCB Controlled DMA Channel Peripherals (Continued)

	Requesters	DDE DMA Channels	Non DDE DMA Channels
	UART0, RX	DMA21	
	UART1, TX	DMA34	
	UART1, RX	DMA35	
	UART2, TX	DMA37	
	UART2, RX	DMA38	
SCB4	SPI0, TX	DMA22	
	SPI0, RX	DMA23	
	SPI1, TX	DMA24	
	SPI1, RX	DMA25	
	SPI2, TX	DMA26	
	SPI2, RX	DMA27	
	SPI5, TX	DMA59	
	SPI5, RX	DMA60	
SCB6	LP0	DMA30	
	LP1	DMA36	
SCB7	CRC0/MDMA0	DMA8, DMA9	
	CRC1/MDMA1	DMA18, DMA19	
SCB8	CRC2/MDMA4	DMA45, DMA46	
	CRC3/MDMA5	DMA47, DMA48	
SCB9	MDMA2	DMA39, DMA40	
	MDMA6	DMA49, DMA50	
SCB10	ETR	N/A	N/A
	CRYPTO	N/A	2 channels (1 read, 1 write)
	EMDMA0	N/A	2 channels (1 read, 1 write)
	EMDMA1	N/A	2 channels (1 read, 1 write)
HSMDMA_SCB	MDMA3	DMA43, DMA44	
	MDMA7	DMA51, DMA52	
	SH0 FIR CH0		
CORES_SCB	A55 M0 Controller	N/A	N/A
	A55 M1 Controller	N/A	N/A
	iDMA	N/A	TBD

Table 45-3: SCB Controlled DMA Channel Peripherals (Continued)

	Requesters	DDE DMA Channels	Non DDE DMA Channels
	SH0 DPORT	N/A	N/A
	SH0 DPORT L2	N/A	N/A
	HSM	N/A	N/A
FIR_IPORT_SCB	SH0 FIR CH1	N/A	CH1
	SH0 IIR CH0	N/A	CH0
	SH0 IIR CH1	N/A	CH1
	SH0 IPORT	N/A	N/A
MMR_SCB	SHARC0 MMR	N/A	N/A
	A55_MRR	N/A	N/A

There are two types of peripherals that use DMA. The first have dedicated DMA channels controlled by the Dedicated DMA Engine (DDE) and have the same operating modes (see DMA Operating Modes) and use the same programming model (DMA Channel Programming Model). The second type is not controlled by the DDE module. These peripherals have their own operating modes and programming models (see the peripheral chapter for this information). The peripheral types are shown in the ***SCB-Controlled DMA Channel Peripherals*** table.

Address Remapping for Backwards Compatibility

To maintain backwards compatibility of the address map to the 21xxx processors, address remapping must occur at the integration level of the SHARC-FX core in SHARC + subsystems.

Address Translation for IPORT to L2 (Boot) ROM 0 and 1

The SHARC-FX reset vector 0x0010_0000 is mapped to the L2 (Boot) ROM. Native addresses issued by the IPORT are remapped/translated to reach the L2 ROM as shown in the following table.

Table 45-4: IPORT to L2 ROM Address Translation

Memory	Native Address Map		Converted Address Map	
	Start	End	Start	End
Boot ROM 0 (64kB	0010_0000	0010_FFFF	2020_0000	2020_FFFF
Boot ROM 1 (64kB)	0011_0000	0011_FFFF	2021_0000	2021_FFFF

S1/S2 to IRAM/DRAM Mapping

Other requesters in a system can access L1 over the bus completer port of the core. The SHARC+ core has two bus completer ports: S1 and S2. The system requester can access the entire L1 over either S1 or S2 port (depending on the grouping). In comparison, the SHARC-FX core has two bus completer ports, I-completer and D-completer, that allow and limit access to IRAM and DRAM, respectively. A bridge has been implemented to allow all system

requesters to access both IRAM and DRAM over I-completer and D-completer. The ***L1 Access Bridge Routing*** table shows the routing source to destination through the bridge. The ***X Bridge to Map Fabric*** figure shows the S1/S2 and accelerator access to IRAM and DRAM through the bridge.

NOTE: Accelerator accesses are also routed through the bridge.

Table 45-5: L1 Access Bridge Routing

Source	Range (SYS_ADDR)		Destination
S1	0x28240000	0x282BFFFF	DRAM/D-Completer
S1	0x282C0000	0x282CFFFF	IRAM/I-Completer
S2	0x28240000	0x282BFFFF	DRAM/D-Completer
S2	0x282C0000	0x282CFFFF	IRAM/I-Completer

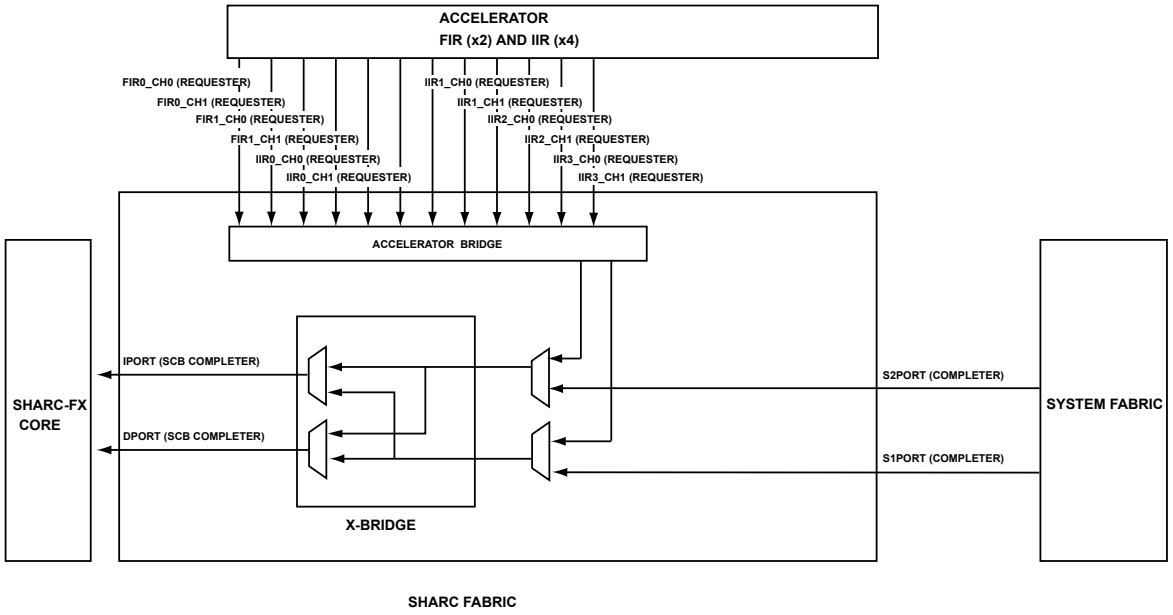


Figure 45-6: X-Bridge to Map Fabric

SHARC-FX Bus Completer Port Remapping

The SHARC-FX core accepts only its native address map on its bus completer port. To maintain SHARC+ backwards compatibility of the L1 bus completer address or multi-processor address, an address remapping or offset adjustment occurs on the incoming addresses to I-completer and D-completer before being forwarded to the SHARC-FX core. The *Bus Completer Port Address Remapping* table shows the remapping of addresses.

Table 45-6: Bus Completer Port Address Remapping

Destination	SYS_ADDR/MP_ADDR		Mapped Native ADDR	
DRAM/D-Completer	0x28240000	0x282BFFFF	0x2F780000	0x2F7FFFFF
IRAM/I-Completer	0x282C0000	0x282CFFFF	0x2F800000	0x2F80FFFF

NOTE:
- Byte and half word accesses to IRAM are not allowed by either the core or DMAs (over I-completer). Only 32-, 64- and 128-bit accesses are allowed. If an 8- or 16-bit access is attempted:
  - the core hits an exception
  - DMAs over I-completer do not receive an error response
- There are no restrictions for DRAM; 8-, 16-, 32-, 64- and128-bit accesses are allowed.
- Unaligned accesses to IRAM are not allowed; the address must always be 32-bit aligned. Unaligned 64- and 128-bit accesses around the 32-bit boundary are valid. The address does not need to be 64- or 28-bit aligned.
- Unaligned accesses to DRAM are allowed for any size.

Port Security

The SHARC-FX core has three requester ports: the instruction requester port, the data requester port, and iDMA. These ports interface with the rest of the system. The SHARC-FX core does not assign any security privileges on accesses from its requester ports. It uses SPU-based security privilege control.

Bus Completer Port Security

The SHARC-FX core has two bus completer ports: IRAM and DRAM. Requesters in the system use these ports to access the IRAM and DRAM of the SHARC-FX processor. These ports interface with the rest of the system. The security privileges of incoming accesses from other requesters in the system are overwritten or passed. The bus completer port security is based on the security privilege of the core (core SPU settings). As secure accesses pass the first stage and enter the second stage, the addresses of incoming accesses are remapped from the system address to the allowed address range supported by SHARC-FX core. If the accesses are not secure, the SHARC-FX processor generates an error response due to an invalid address.

System Crossbars

The System Crossbars (SCB) are the fundamental building blocks of the system bus interconnect. The SCB (often referred to as the system interconnect fabric), is a collection of inter-connection units connecting system requesters to completer memory spaces. The SCB connects one or more requester devices to one or more memory-mapped completer devices. Each connected requester can be a core that originates an SCB transaction, or a requester interface of an upstream SCB cascaded interconnect. Each connected completer can be the final completer of an SCB transaction or a completer interface of a downstream cascaded SCB interconnect (forming a hierarchy of SCBs).

Each SCB that has multiple requesters and completers share the total bandwidth of the SCB. (In a M:N configuration where M requesters are connected to N completers through the SCBx.)

The SCB provides separate channels for reads and writes. Read and write accesses through a given SCB do not share bandwidth. Only SCB0, which is the major SCB in the SCB hierarchy, has the multiple paths between multiple requester and completer interfaces.

All other SCBs in the chip connect to SCB0 through different completer interfaces. Other primary requesters (DMAs, cores, and so on) in the system are distributed across these small SCBs. For a given SCB, all the requester and completers share the total bandwidth of the SCB. (Only SCB0 is the exception). Since different DMA channels are scattered across different SCBs (SCB1, SCB2 SCB3, and so on), they do not conflict for the bandwidth when they are in different SCBs and accessing different completers. SCB0 allows for concurrent data transfer between multiple bus requesters and multiple bus completers, providing flexibility, and full-duplex operation.

If system accesses are carefully architected, SCB has a potential of providing sufficient sustained bandwidth in the end system.

Since the SCBs support burst transfers, it is important to configure the requesting requester appropriately to make best use of available SCB bandwidth. For a DMA requester, choosing the appropriate `DMA_CFG.MSIZE` value, is important from both a functional and a performance perspective. The value in the `DMA_CFG.PSIZE` bit field determines the width of the peripheral bus in use. It can be configured to 1-byte, 2-bytes, or 4-bytes. The `DMA_CFG.MSIZE` value determines the actual size of the SCB bus in use. It also determines the minimum number of bytes which are transferred from or to memory corresponding to a single DMA request or grant. The transfer can be 1-, 2-, 4-, 8-, 16-, or 32-bytes. If the `DMA_CFG.MSIZE` value is greater than the SCB bus width, the SCB performs burst transfers according to the width defined in `DMA_CFG.MSIZE`. When `DMA_CFG.MSIZE` is less than the SCB bus width, bursting is not supported and partial bus use results.

Each of the SCB unit in the fabric consists of N completer interfaces (MSTn). Each of these interfaces has controls for read quality of service, write quality of service, and functional mode. A subset of these matrices includes controls for IB (Interface Block) sync mode, and bus functional mode. For more details on IB, see the clock domain synchronization section.

SCB Bus Requester IDs

The SCB ***Bus Requester IDs*** table indicates which requesters are connected to each of the completer ports of SCB0. The tables also indicate the precise value of the ID as seen by the completer. These values are useful for SWU programming.

NOTE: For an overall diagram of all SCB interconnections, see the SCB Block Diagram.

Table 45-7: Bus Requester IDs

Requester	Hex ID Values	Binary Values
DMA0 (SPORT0, HALF A)	0x0000, 0x0100	13'b0000x00000000
DMA1 (SPORT0, HALF B)	0x0010, 0x0090	13'b00000x0010000

Table 45-7: Bus Requester IDs (Continued)

Requester	Hex ID Values	Binary Values
DMA2 (SPORT1, HALF A)	0x0020, 0x00A0	13'b00000x0100000
DMA3 (SPORT1, HALF B)	0x0030, 0x00B0	13'b00000x0110000
DMA4 (SPORT2, HALF A)	0x0040, 0x00C0	13'b00000x1000000
DMA5 (SPORT2, HALF B)	0x0050, 0x00D0	13'b00000x1010000
DMA6 (SPORT3, HALF A)	0x0060, 0x00E0	13'b00000x1100000
DMA7 (SPORT3, HALF B)	0x0070, 0x00F0	13'b00000x1110000
DMA8 (Enh BW MDMA0 CRC, CH0)	0x0031, 0x00B1	13'b00000x0110001
DMA9 (Enh BW MDMA0 CRC, CH1)	0x0040, 0x00C1	13'b00000x0100001
MLB	0x0042	13'b0000001000010
DMA20 (UART0, TX)	0x0003, 0x0083	13'b00000x0000011
DMA21 (UART0, RX)	0x0043, 0x00C3	13'b00000x1000011
DMA37 (UART2, TX)	0x0033, 0x00B3	13'b00000x0110011
DMA22 (SPI0, TX)	0x0004, 0x0084	13'b00000x0000100
DMA23 (SPI0, RX)	0x0014, 0x0094	13'b00000x0010100
DMA24 (SPI1, TX)	0x0024, 0x0A4	13'b00000x0100100
DMA25 (SPI1, RX)	0x0034, 0x0B4	13'b00000x0110100
DMA26 (SPI2, TX)	0x0054, 0x00D4	13'b00000x1010100
DMA27 (SPI2, RX)	0x0044, 0x00C4	13'b00000x1000100
DMA30 (LP0)	0x0005, 0x0085	13'b00000x0000101
DMA34 (UART1, TX)	0x0013, 0x0093	13'b00000x0010011
DMA35 (UART1, RX)	0x0023, 0x00A3	13'b00000x0100011
DMA36 (LP1)	0x0015, 0x0095	13'b00000x0010101
CRYPTO	0x0056	13'b0000001010110
SH0_FIR_CH0		13'b00xxxx0000111
SH0_FIR_CH1		13'b00xxxx1011010
EMDMA0 (CH0)	0x0006	13'b0000000000110
EMDMA0 (CH1)	0x0016	13'b0000000010110
EMDMA1 (CH0)	0x0026	13'b0000000100110
EMDMA1 (CH1)	0x0036	13'b0000000110110
DMA39 (Enh BW MDMA2, CH0)	0x0008, 0x0088	13'b00000x0001000
DMA40 (Enh BW MDMA2, CH1)	0x0018, 0x098	13'b00000x0011000

Table 45-7: Bus Requester IDs (Continued)

Requester	Hex ID Values	Binary Values
DBG	0x0048	13'b0000001001000
ETR	0x0046	13'b0000001000110
DMA18 (Enh BW MDMA1 CRC1, CH0)	0x0011, 0x0091	13'b00000x0010001
DMA19 (Enh BW MDMA1 CRC1, CH1)	0x0001, 0x0081	13'b00000x0000001
DMA38 (UART2, RX)	0x0053, 0x00D3	13'b00000x1010011
SH0 (DPORT)		13'bxxxxxx0001001
SH0 (IPORT)		13'b00xxxx0100111
DMA43 (High Speed MDMA3, CH0)	0x000A, 0x008A	13'b00000x0001010
DMA44 (High Speed BW MDMA3, CH1)	0x001A, 0x009A	13'b00000x0011010
DMA10 (SPORT4, HALFA)	0x000B, 0x008B	13'b00000x0001011
DMA11 (SPORT4, HALFB)	0x001B, 0x009B	13'b00000x0011011
DMA12 (SPORT5, HALFA)	0x002B, 0x00AB	13'b00000x0101011
DMA13 (SPORT5, HALFB)	0x003B, 0x00BB	13'b00000x0111011
DMA14 (SPORT6, HALFA)	0x004B, 0x00CB	13'b00000x1001011
DMA15 (SPORT6, HALFB)	0x005B, 0x00DB	13'b00000x1011011
DMA16 (SPORT7, HALFA)	0x006B, 0x00EB	13'b00000x1101011
DMA17 (SPORT7, HALFB)	0x007B, 0x00FB	13'b00000x1111011
SH0_MMR		13'bxxxxxx0001100
DMA45 (Enh BW MDMA4 CRC, CH0)	0x001D, 0x009D	13'b00000x0011101
DMA46 (Enh BW MDMA4 CRC, CH1)	0x000D, 0x008D	13'b00000x0001101
DMA47 (Enh BW MDMA5 CRC, CH0)	0x003D, 0x0BD	13'b00000x0111101
DMA48 (Enh BW MDMA5 CRC, CH1)	0x002D, 0x0AD	13'b00000x0101101
DMA49 (Enh BW MDMA6 CRC1, CH0)	0x003D, 0x00BD	13'b00000x0111000
DMA50 (Enh BW MDMA6 CRC1, CH1)	0x0028 0x00A8	13'b00000x0101000
DMA51 (High Speed MDMA3, CH0)	0x002A, 0x00AA	13'b00000x0101010
DMA52 (High Speed BW MDMA3, CH1)	0x003A, 0x00BA	13'b00000x0111010
SH0_IIR_CH0		13'b00xxxx1000111
SH0_IIR_CH1		13'b00xxxx0110111
iDMA		13'b00xxxx0011001
XSPI M0	0x0052	13'b0000001010010
A55 M1 AXI	0x0029	13'b0xxxxx0101001

Table 45-7: Bus Requester IDs (Continued)

Requester	Hex ID Values	Binary Values
GIGE0		13'b00xxxx0011110
DMA57 (SPI5, TX)	0x0064, 0x00E4	13'b00000x1100100
DMA60 (SPI5, RX)	0x0074, 0x00F4	13'b00000x1110100
SH0_DPORT_L2		13'bxxxxxx0111001
GIGE1		13'b00xxxx0101110
XSPI M1		13'b0000001100010
EMMC		13'b00000x0111110
HSM		13'b00xxxx1001001
A55 M0 AXI		13'b0xxxxx1011001
SH0_FIR_CH0		13'b00xxxx1011010
A55_MMR		13'b0xxxxx1101001

SCB Programming Model

The following sections provide information for programming the SCB properly.

Programming SCB Arbitration

Each completer interface has a QoS value (priority) associated with both read and write channels. These values are 4 bits present in the `SCB0_MSTx_RQOS` and `SCB0_MSTx_WQOS` registers. At the entry point to the infrastructure, all transactions are allocated this programmable local QoS value. The arbitration of the transaction throughout the infrastructure uses this QoS. At any arbitration node, a fixed priority exists for transactions with a different QoS. The highest value has the highest priority.

If there are coincident transactions at an arbitration node with the same QoS that require arbitration, then the network uses a Least Recently Granted (LRG) algorithm. At each switch, the requester with the highest QoS acquires access and that switch output takes the QoS value of the winner for that transaction. At the next switch completer interface, the requester uses the QoS value of the winner. QoS can have values from 0 (lowest priority) to 15 (highest priority).

For example in the following figure, ***SCB Arbitration***:

1. At SCB1, requesters (1, 2, 3) have RQOS values of (6, 4, 2)
2. At SCB2, requesters (4, 5, 6) have RQOS values of (12, 13, 1)

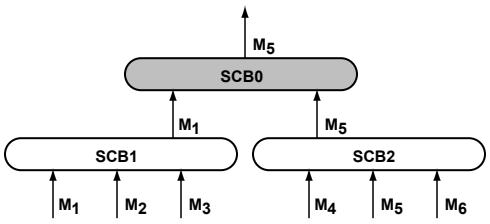


Figure 45-7: SCB Arbitration

In this case, requester 1 wins at SCB1, and requester 5 wins at SCB2. However, in a perfect competition at SCB0, requesters 4 and 5 had the highest overall RQOS values. requesters 4 and 5 would have fought for arbitration directly at SCB0. However, because of the mini*SCBs, requester 1, at a much lower RQOS value, is able to win against requester 4 and make it all the way to SCB0.

Programming Clock Domain Crossing Registers

The ***Clock Domain Crossing Options*** table shows the various clock domain crossings that are available on the processor. The clocking relationships are defined with respect to SYSCLK. The followng clock domains are programmable: CLKO3: SYSCLK, CLKO4: SYSCLK and CLKO8: SYSCLK CDC. The registers to configure the CDC mode are present within the GPV space of the respective fabric. For example: To program the CLKO3: SYSCLK relationship, program the GPV registers within the DMC_CDC fabric and the MMRG fabric.

Table 45-8: Clock Domain Crossing Options

		Clocking Relationships with respect to SYSCLK					
Functional Clocks	**Possible Completers**	**System Fabric**	**MMRG Fabric**	**DMC Fabric**	**SHARC Fabric**	**Arm Fabric**	**SPIF Fabric**
SYSCLK	System Fabric and Infrastructure Modules	1:1	1:1	*1	*1	*1	*1
SCLK0	SCLK0 Clock Doamin	Synchronous (m:1)	Synchronous (m:1)	*1	*1	*1	*1
CLKO0	SHARC0 and its Accelerators	*1	*1	*1	Programmable	*1	*1
CLKO1	A55	*1	*1	*1	*1	Programmable	*1
CLKO3	DMC	*1	Programmable	Programmable	*1	*1	*1
CLKO4	CAN	*1	Programmable	*1	*1	*1	*1
CLKO6	SPI	Synchronous (m:n) (except SPIF completer)	Synchronous (m:n)	*1	*1	*1	Synchronous (m:n)
CLKO8	LP	Programmable	Programmable	*1	*1	*1	*1

*1 Either the interface is not present or CDC is not required.

NOTE: Consider the following points in the *Clock Domain Crossing Options* table:

- For DMC fabric, the sync mode register should be programmed for m:n or async (default). Only for sysclk:CLKO3 = 500:500MHz, sync mode can be 1:1.
- For system MMRG fabric – CLKO8/LP sync mode register should be programmed for m:n or async (default).

Table 45-9: Sync Mode Bit Field Description

Sync Mode	Description
0	sync 1:1
1	sync n:1
2	sync 1:n
3	sync m:n
4	async

To change the clock domain crossing mode, follow the actions described in the ***Changing Clock Domain Crossing Modes*** table.

Table 45-10: Changing Clock Domain Crossing Modes

Original Mode	Required Mode	Action
ASYNC	Any other mode	Change the clocks, then change the MMR register
Any mode	ASYNC	Change the MMR register, then change clocks to ASYNC.
m:n	1:1	Change the clocks, then change the register.
1:1	m:n	Change the register, then change the clocks.

SCB Programming Concepts

The SCB arbitration model among requester or completer SCBs of the processor is fixed (not programmable). But, each completer does have a quality of service (QoS) programmable feature that affects arbitration.

The arbitration of transactions in SCB is based on the QoS value or the priority of the transaction. All requesters with the same priority form a priority group. Arbitration is granted to the highest priority group from which a member is trying to win access, and within that group, to the highest requester at that time. When a requester wins arbitration, it is relegated to the bottom of its group to ensure that it cannot prevent other requesters in its group from accessing the completer.

If you configure all requester priorities to different levels, the arbiter implements a fixed priority scheme. This scheme occurs because each requester is in a group of its own, and therefore, requesters maintain their ordering.

The LRG and fixed priority modes concurrently exist when the requester priority value registers are programmed with a combination of identical and unique values.

NOTE: The SCB arbitration hierarchy is fixed (for example, SCB1 requester to SCB1 completer). However, multiple requester inputs to the same completer permit QoS programming.

The *LRG Arbitration Example* figure shows three groups with different QoS values. Requesters in the same group share a QoS value. The arbitration occurs using an LRG scheme.

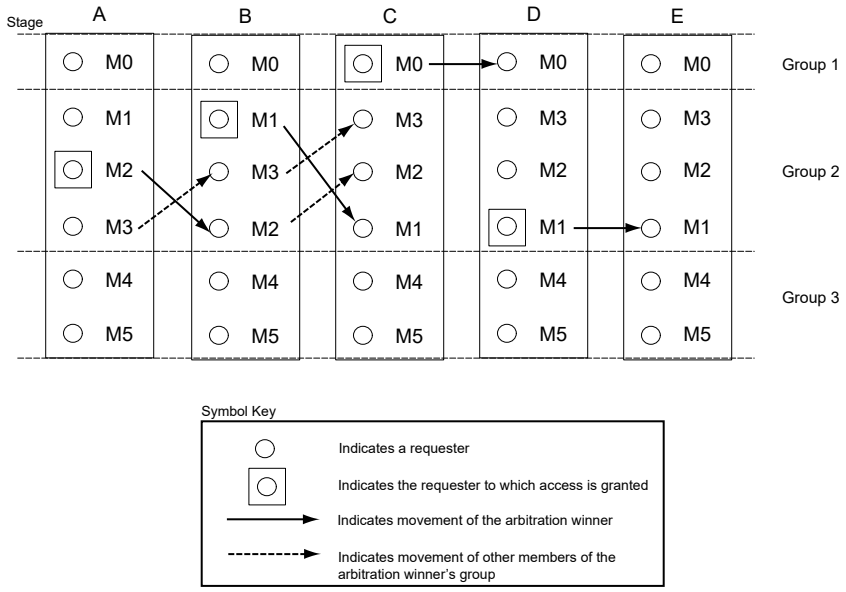


Figure 45-8: LRG Arbitration Example

The QoS value assigned to a transaction at entry point is carried forward by the transaction as it passes through all arbitration stages in the SCB. QoS for all requesters is configured as programmable in the system fabric interconnect.

The priority of the requesters fits into three groups:

- Group A - Peripherals with an external interface, without flow control and with real-time processing requirements
- Group B - Cores, peripherals with flow control, and offload engines
- Group C - MDMAs

Group A: Peripherals with external interface, without flow control and with real-time processing requirements

These requesters are assigned the highest priority. They are latency-critical – a large increase in latency could potentially result in data corruption and catastrophic failure. This high priority SCB group includes requesters connected to SCB1, SCB2, SCB3, SCB4, MLB and XSPI M. Requesters in this group are assigned the QOS reset value of 12.

Group B: Cores, peripherals with flow control, and offload engines

Core is latency-sensitive. Once a core requests data, it typically waits for the data without performing anything else in parallel. An increase in the latency has a direct impact on the performance of the core. This group is assigned medium priority and a QOS reset value of 7. Group B requesters are connected to SCB6, GIGE, CORES_SCB, FIR_IPORT_SCB, and MMR_SCB.

Group C: MDMAs

These requesters are assigned the lowest priority. MDMAs are data intensive and do not have any external interface. Requesters in this group are assigned the QOS reset value of 1. The Group C requesters are connected to SCB7, SCB8, SCB9, and SCB10.

Table 45-11: QoS Register Table

Requester	read_qos Reset Value	write_qos Reset Value
SPORT0_A_DMA	12	12
SPORT0_B_DMA	12	12
SPORT1_A_DMA	12	12
SPORT1_B_DMA	12	12
SPORT2_A_DMA	12	12
SPORT2_B_DMA	12	12
SPORT3_A_DMA	12	12
SPORT3_B_DMA	12	12
SPORT4_A_DMA	12	12
SPORT4_B_DMA	12	12
SPORT5_A_DMA	12	12
SPORT5_B_DMA	12	12
SPORT6_A_DMA	12	12
SPORT6_B_DMA	12	12
SPORT7_A_DMA	12	12
SPORT7_B_DMA	12	12
UART0_TX	12	12
UART0_RX	12	12
UART1_TX	12	12
UART1_RX	12	12
UART2_TX	12	12
UART2_RX	12	12

Table 45-11: QoS Register Table (Continued)

Requester	read_qos Reset Value	write_qos Reset Value
SPI0RX	12	12
SPI0TX	12	12
SPI1TX	12	12
SPI1RX	12	12
SPI2TX	12	12
SPI2RX	12	12
SPI5TX	12	12
SPI5RX	12	12
MLB	12	12
xSPI0	12	12
xSPI1	12	12
LP0	7	7
LP1	7	7
GIGE	7	7
EMMC	7	7
SH0_DPORT	7	7
SH0_DPORT_L2	7	7
iDMA	7	7
SH0_IPORT	7	7
SH0_IIR_CH0	7	7
SH0_IIR_CH1	7	7
SH0_FIR_CH0	7	7
SH0_FIR_CH1	7	7
SH0_MMR	7	7
A55 MMR	7	7
A55 M0 AXI	7	7
A55 M1 AXI	7	7
HSM	7	7
CRC0_CH0	1	1
CRC0_CH1	1	1
CRC1_CH0	1	1

Table 45-11: QoS Register Table (Continued)

Requester	read_qos Reset Value	write_qos Reset Value
CRC1_CH1	1	1
CRC2_CH0	1	1
CRC2_CH1	1	1
CRC3_CH0	1	1
CRC3_CH1	1	1
MDMA2_CH0	1	1
MDMA2_CH1	1	1
MDMA3_CH0	1	1
MDMA3_CH1	1	1
MDMA6_CH0	1	1
MDMA6_CH1	1	1
MDMA7_CH0	1	1
MDMA7_CH1	1	1
EMDMA0_CH0	1	1
EMDMA0_CH1	1	1
EMDMA1_CH0	1	1
EMDMA1_CH1	1	1
CRYPTO	1	1
ETR	1	1
DBG	1	1

QoS Programming

Adhere to the following guidelines if software is modifying the reset value of QoS:

- The highest QoS of any requester in a low priority SCB group must be less than the lowest QoS of any requester in a medium priority SCB group
- The highest QoS of any requester in medium priority SCB group must be less than the lowest QoS of any requester in high priority SCB group

Reset values of QoS have been spaced out to allow software to lower or higher the priority of a few requesters without having to modify the priority all the requesters. For example, the QoS of the SPI can be reduced from 12 to 11 without violating the two guidelines outlined above.

ADSP-2184x SCB0 Register Descriptions

Your module description, here. (SCB0) contains the following registers.

Table 45-12: ADSP-2184x SCB0 Register List

Name	Description
SCB0_HSM_IB_FN_MOD	
SCB0_HSM_IB_FN_MOD2	
SCB0_HSM_IB_READ_QOS	
SCB0_HSM_IB_WRITE_QOS	
SCB0_CL2_1_FN_MOD_ISS_BM	
SCB0_CL2_2_FN_MOD_ISS_BM	
SCB0_COMP_ID_0	
SCB0_COMP_ID_1	
SCB0_COMP_ID_2	
SCB0_COMP_ID_3	
SCB0_CRC0_CH0_IB_FN_MOD	
SCB0_CRC0_CH0_IB_FN_MOD2	
SCB0_CRC0_CH0_IB_READ_QOS	
SCB0_CRC0_CH0_IB_WRITE_QOS	
SCB0_CRC0_CH1_IB_FN_MOD	
SCB0_CRC0_CH1_IB_FN_MOD2	
SCB0_CRC0_CH1_IB_READ_QOS	
SCB0_CRC0_CH1_IB_WRITE_QOS	
SCB0_CRC1_CH0_IB_FN_MOD	
SCB0_CRC1_CH0_IB_FN_MOD2	
SCB0_CRC1_CH0_IB_READ_QOS	
SCB0_CRC1_CH0_IB_WRITE_QOS	
SCB0_CRC1_CH1_IB_FN_MOD	
SCB0_CRC1_CH1_IB_FN_MOD2	
SCB0_CRC1_CH1_IB_READ_QOS	
SCB0_CRC1_CH1_IB_WRITE_QOS	
SCB0_CRC2_CH0_IB_FN_MOD	
SCB0_CRC2_CH0_IB_FN_MOD2	
SCB0_CRC2_CH0_IB_READ_QOS	
SCB0_CRC2_CH0_IB_WRITE_QOS	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_CRC2_CH1_IB_FN_MOD	
SCB0_CRC2_CH1_IB_FN_MOD2	
SCB0_CRC2_CH1_IB_READ_QOS	
SCB0_CRC2_CH1_IB_WRITE_QOS	
SCB0_CRC3_CH0_IB_FN_MOD	
SCB0_CRC3_CH0_IB_FN_MOD2	
SCB0_CRC3_CH0_IB_READ_QOS	
SCB0_CRC3_CH0_IB_WRITE_QOS	
SCB0_CRC3_CH1_IB_FN_MOD	
SCB0_CRC3_CH1_IB_FN_MOD2	
SCB0_CRC3_CH1_IB_READ_QOS	
SCB0_CRC3_CH1_IB_WRITE_QOS	
SCB0_CRYPTO_IB_FN_MOD	
SCB0_CRYPTO_IB_FN_MOD2	
SCB0_CRYPTO_IB_READ_QOS	
SCB0_CRYPTO_IB_WRITE_QOS	
SCB0_DBG_IB_FN_MOD	
SCB0_DBG_IB_FN_MOD2	
SCB0_DBG_IB_READ_QOS	
SCB0_DBG_IB_WRITE_QOS	
SCB0_DL2_0_FN_MOD_ISS_BM	
SCB0_DL2_1_FN_MOD_ISS_BM	
SCB0_DLDMA0_CH0_IB_FN_MOD	
SCB0_DLDMA0_CH0_IB_FN_MOD2	
SCB0_DLDMA0_CH0_IB_READ_QOS	
SCB0_DLDMA0_CH0_IB_WRITE_QOS	
SCB0_DLDMA0_CH1_IB_FN_MOD	
SCB0_DLDMA0_CH1_IB_FN_MOD2	
SCB0_DLDMA0_CH1_IB_READ_QOS	
SCB0_DLDMA0_CH1_IB_WRITE_QOS	
SCB0_DLDMA1_CH0_IB_FN_MOD	
SCB0_DLDMA1_CH0_IB_FN_MOD2	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_DLDMA1_CH0_IB_READ_QOS	
SCB0_DLDMA1_CH0_IB_WRITE_QOS	
SCB0_DLDMA1_CH1_IB_FN_MOD	
SCB0_DLDMA1_CH1_IB_FN_MOD2	
SCB0_DLDMA1_CH1_IB_READ_QOS	
SCB0_DLDMA1_CH1_IB_WRITE_QOS	
SCB0_DMC0_FN_MOD_ISS_BM	
SCB0_EMMC_IB_FN_MOD	
SCB0_EMMC_IB_FN_MOD2	
SCB0_EMMC_IB_READ_QOS	
SCB0_EMMC_IB_WRITE_QOS	
SCB0_ETR_IB_FN_MOD	
SCB0_ETR_IB_FN_MOD2	
SCB0_ETR_IB_READ_QOS	
SCB0_ETR_IB_WRITE_QOS	
SCB0_GIGE1_IB_FN_MOD	
SCB0_GIGE1_IB_FN_MOD2	
SCB0_GIGE1_IB_READ_QOS	
SCB0_GIGE1_IB_WRITE_QOS	
SCB0_GIGE_IB_FN_MOD	
SCB0_GIGE_IB_FN_MOD2	
SCB0_GIGE_IB_READ_QOS	
SCB0_GIGE_IB_WRITE_QOS	
SCB0_HSMDMA1_CH0_IB_FN_MOD	
SCB0_HSMDMA1_CH0_IB_FN_MOD2	
SCB0_HSMDMA1_CH0_IB_READ_QOS	
SCB0_HSMDMA1_CH0_IB_WRITE_QOS	
SCB0_HSMDMA1_CH1_IB_FN_MOD	
SCB0_HSMDMA1_CH1_IB_FN_MOD2	
SCB0_HSMDMA1_CH1_IB_READ_QOS	
SCB0_HSMDMA1_CH1_IB_WRITE_QOS	
SCB0_HSMDMA_CH0_IB_FN_MOD	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_HSMDMA_CH0_IB_FN_MOD2	
SCB0_HSMDMA_CH0_IB_READ_QOS	
SCB0_HSMDMA_CH0_IB_WRITE_QOS	
SCB0_HSMDMA_CH1_IB_FN_MOD	
SCB0_HSMDMA_CH1_IB_FN_MOD2	
SCB0_HSMDMA_CH1_IB_READ_QOS	
SCB0_HSMDMA_CH1_IB_WRITE_QOS	
SCB0_IDMA_FN_MOD	
SCB0_IDMA_READ_QOS	
SCB0_IDMA_WRITE_QOS	
SCB0_LP0_FN_MOD	
SCB0_LP0_READ_QOS	
SCB0_LP0_WRITE_QOS	
SCB0_LP1_FN_MOD	
SCB0_LP1_READ_QOS	
SCB0_LP1_WRITE_QOS	
SCB0_M85_M2_AXI_FN_MOD	
SCB0_M85_M2_AXI_READ_QOS	
SCB0_M85_M2_AXI_WRITE_QOS	
SCB0_M85_M_AXI_FN_MOD	
SCB0_M85_M_AXI_READ_QOS	
SCB0_M85_M_AXI_WRITE_QOS	
SCB0_M85_S2_AXI_IB_FN_MOD	
SCB0_M85_S2_AXI_IB_FN_MOD_ISS_BM	
SCB0_M85_S_AXI_IB_FN_MOD	
SCB0_M85_S_AXI_IB_FN_MOD_ISS_BM	
SCB0_MLB_FN_MOD	
SCB0_MLB_READ_QOS	
SCB0_MLB_WRITE_QOS	
SCB0_MSMDMA1_CH0_IB_FN_MOD	
SCB0_MSMDMA1_CH0_IB_FN_MOD2	
SCB0_MSMDMA1_CH0_IB_READ_QOS	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_MSMDMA1_CH0_IB_WRITE_QOS	
SCB0_MSMDMA1_CH1_IB_FN_MOD	
SCB0_MSMDMA1_CH1_IB_FN_MOD2	
SCB0_MSMDMA1_CH1_IB_READ_QOS	
SCB0_MSMDMA1_CH1_IB_WRITE_QOS	
SCB0_MSMDMA_CH0_IB_FN_MOD	
SCB0_MSMDMA_CH0_IB_FN_MOD2	
SCB0_MSMDMA_CH0_IB_READ_QOS	
SCB0_MSMDMA_CH0_IB_WRITE_QOS	
SCB0_MSMDMA_CH1_IB_FN_MOD	
SCB0_MSMDMA_CH1_IB_FN_MOD2	
SCB0_MSMDMA_CH1_IB_READ_QOS	
SCB0_MSMDMA_CH1_IB_WRITE_QOS	
SCB0_OTP_IB_FN_MOD	
SCB0_OTP_IB_FN_MOD_ISS_BM	
SCB0_PERIPH_ID_0	
SCB0_PERIPH_ID_1	
SCB0_PERIPH_ID_2	
SCB0_PERIPH_ID_3	
SCB0_PERIPH_ID_4	
SCB0_PERIPH_ID_5	
SCB0_PERIPH_ID_6	
SCB0_PERIPH_ID_7	
SCB0_SCB6_IB_FN_MOD	
SCB0_SCB6_IB_FN_MOD2	
SCB0_SCB6_IB_FN_MOD_ISS_BM	
SCB0_SCB6_IB_SYNC_MODE	
SCB0_SH0_DPORT_L2_FN_MOD	
SCB0_SH0_DPORT_L2_READ_QOS	
SCB0_SH0_DPORT_L2_WRITE_QOS	
SCB0_SH0_DPORT_FN_MOD	
SCB0_SH0_DPORT_READ_QOS	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_SH0_DPORT_WRITE_QOS	
SCB0_SH0_FIR_CH0_FN_MOD	
SCB0_SH0_FIR_CH0_READ_QOS	
SCB0_SH0_FIR_CH0_WRITE_QOS	
SCB0_SH0_FIR_CH1_FN_MOD	
SCB0_SH0_FIR_CH1_READ_QOS	
SCB0_SH0_FIR_CH1_WRITE_QOS	
SCB0_SH0_IIR_CH0_FN_MOD	
SCB0_SH0_IIR_CH0_READ_QOS	
SCB0_SH0_IIR_CH0_WRITE_QOS	
SCB0_SH0_IIR_CH1_FN_MOD	
SCB0_SH0_IIR_CH1_READ_QOS	
SCB0_SH0_IIR_CH1_WRITE_QOS	
SCB0_SH0_IPORT_FN_MOD	
SCB0_SH0_IPORT_READ_QOS	
SCB0_SH0_IPORT_WRITE_QOS	
SCB0_SH0_MMR_IB_FN_MOD	
SCB0_SH0_MMR_IB_FN_MOD2	
SCB0_SH0_MMR_IB_READ_QOS	
SCB0_SH0_MMR_IB_WRITE_QOS	
SCB0_SH0_S1PORT_FN_MOD_ISS_BM	
SCB0_SH0_S2PORT_FN_MOD_ISS_BM	
SCB0_SMC_IB_FN_MOD	
SCB0_SMC_IB_FN_MOD_ISS_BM	
SCB0_SMMR_FN_MOD_ISS_BM	
SCB0_SP0A_FN_MOD	
SCB0_SP0A_READ_QOS	
SCB0_SP0A_WRITE_QOS	
SCB0_SP0B_FN_MOD	
SCB0_SP0B_READ_QOS	
SCB0_SP0B_WRITE_QOS	
SCB0_SP1A_FN_MOD	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_SP1A_READ_QOS	
SCB0_SP1A_WRITE_QOS	
SCB0_SP1B_FN_MOD	
SCB0_SP1B_READ_QOS	
SCB0_SP1B_WRITE_QOS	
SCB0_SP2A_FN_MOD	
SCB0_SP2A_READ_QOS	
SCB0_SP2A_WRITE_QOS	
SCB0_SP2B_FN_MOD	
SCB0_SP2B_READ_QOS	
SCB0_SP2B_WRITE_QOS	
SCB0_SP3A_FN_MOD	
SCB0_SP3A_READ_QOS	
SCB0_SP3A_WRITE_QOS	
SCB0_SP3B_FN_MOD	
SCB0_SP3B_READ_QOS	
SCB0_SP3B_WRITE_QOS	
SCB0_SP4A_FN_MOD	
SCB0_SP4A_READ_QOS	
SCB0_SP4A_WRITE_QOS	
SCB0_SP4B_FN_MOD	
SCB0_SP4B_READ_QOS	
SCB0_SP4B_WRITE_QOS	
SCB0_SP5A_FN_MOD	
SCB0_SP5A_READ_QOS	
SCB0_SP5A_WRITE_QOS	
SCB0_SP5B_FN_MOD	
SCB0_SP5B_READ_QOS	
SCB0_SP5B_WRITE_QOS	
SCB0_SP6A_FN_MOD	
SCB0_SP6A_READ_QOS	
SCB0_SP6A_WRITE_QOS	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_SP6B_FN_MOD	
SCB0_SP6B_READ_QOS	
SCB0_SP6B_WRITE_QOS	
SCB0_SP7A_FN_MOD	
SCB0_SP7A_READ_QOS	
SCB0_SP7A_WRITE_QOS	
SCB0_SP7B_FN_MOD	
SCB0_SP7B_READ_QOS	
SCB0_SP7B_WRITE_QOS	
SCB0_SPI0RX_FN_MOD	
SCB0_SPI0RX_READ_QOS	
SCB0_SPI0RX_WRITE_QOS	
SCB0_SPI0TX_FN_MOD	
SCB0_SPI0TX_READ_QOS	
SCB0_SPI0TX_WRITE_QOS	
SCB0_SPI1RX_FN_MOD	
SCB0_SPI1RX_READ_QOS	
SCB0_SPI1RX_WRITE_QOS	
SCB0_SPI1TX_FN_MOD	
SCB0_SPI1TX_READ_QOS	
SCB0_SPI1TX_WRITE_QOS	
SCB0_SPI2RX_FN_MOD	
SCB0_SPI2RX_READ_QOS	
SCB0_SPI2RX_WRITE_QOS	
SCB0_SPI2TX_FN_MOD	
SCB0_SPI2TX_READ_QOS	
SCB0_SPI2TX_WRITE_QOS	
SCB0_SPI5RX_FN_MOD	
SCB0_SPI5RX_READ_QOS	
SCB0_SPI5RX_WRITE_QOS	
SCB0_SPI5TX_FN_MOD	
SCB0_SPI5TX_READ_QOS	

Table 45-12: ADSP-2184x SCB0 Register List (Continued)

Name	Description
SCB0_SPI5TX_WRITE_QOS	
SCB0_SPIF_FN_MOD_ISS_BM	
SCB0_UART0_RX_FN_MOD	
SCB0_UART0_RX_READ_QOS	
SCB0_UART0_RX_WRITE_QOS	
SCB0_UART0_TX_FN_MOD	
SCB0_UART0_TX_READ_QOS	
SCB0_UART0_TX_WRITE_QOS	
SCB0_UART1_RX_FN_MOD	
SCB0_UART1_RX_READ_QOS	
SCB0_UART1_RX_WRITE_QOS	
SCB0_UART1_TX_FN_MOD	
SCB0_UART1_TX_READ_QOS	
SCB0_UART1_TX_WRITE_QOS	
SCB0_UART2_RX_FN_MOD	
SCB0_UART2_RX_READ_QOS	
SCB0_UART2_RX_WRITE_QOS	
SCB0_UART2_TX_FN_MOD	
SCB0_UART2_TX_READ_QOS	
SCB0_UART2_TX_WRITE_QOS	
SCB0_XSPI_M1_IB_FN_MOD	
SCB0_XSPI_M1_IB_FN_MOD2	
SCB0_XSPI_M_IB_FN_MOD	
SCB0_XSPI_M_IB_FN_MOD2	

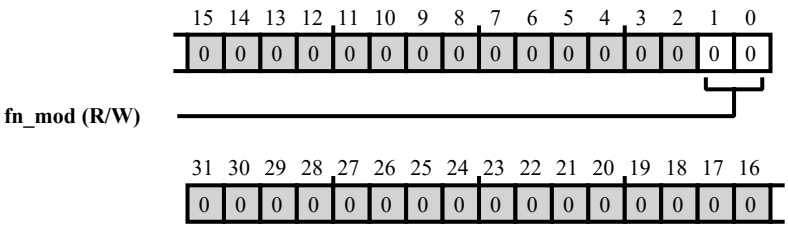


Figure 45-9: SCB0_HSM_IB_FN_MOD Register Diagram

Table 45-13: SCB0_HSM_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

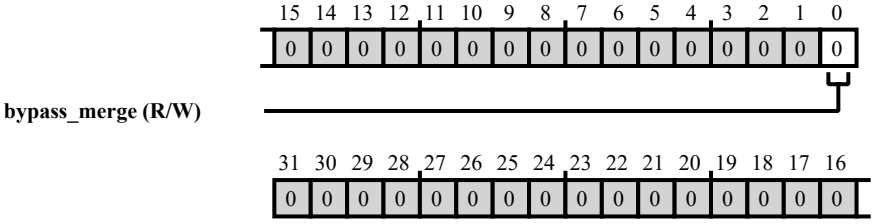


Figure 45-10: SCB0_HSM_IB_FN_MOD2 Register Diagram

Table 45-14: SCB0_HSM_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

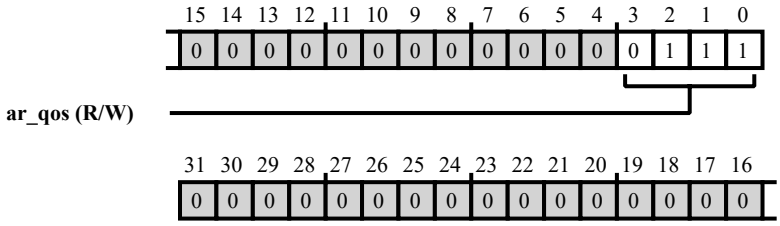


Figure 45-11: SCB0_HSM_IB_READ_QOS Register Diagram

Table 45-15: SCB0_HSM_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

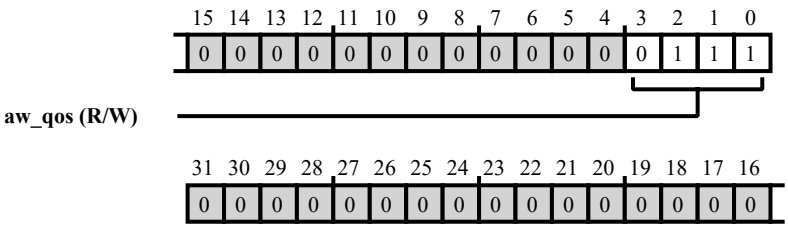


Figure 45-12: SCB0_HSM_IB_WRITE_QOS Register Diagram

Table 45-16: SCB0_HSM_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

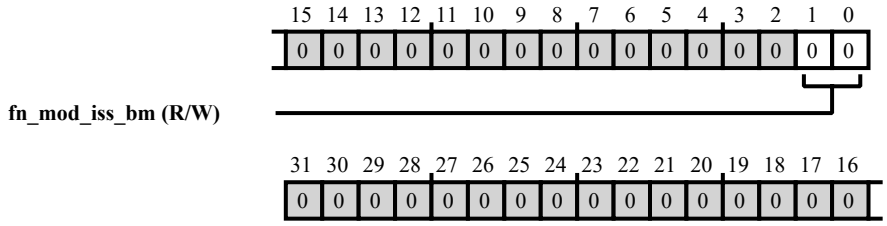


Figure 45-13: SCB0_CL2_1_FN_MOD_ISS_BM Register Diagram

Table 45-17: SCB0_CL2_1_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

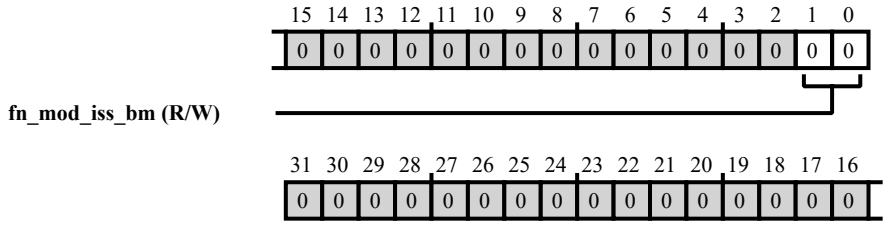


Figure 45-14: SCB0_CL2_2_FN_MOD_ISS_BM Register Diagram

Table 45-18: SCB0_CL2_2_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

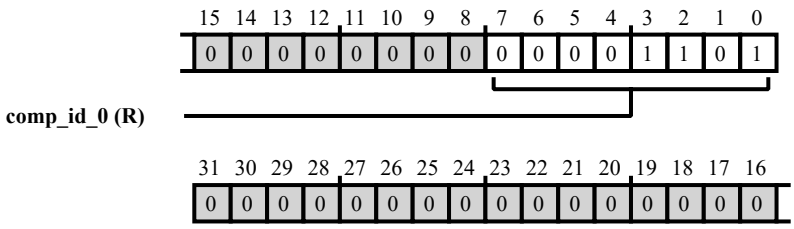


Figure 45-15: SCB0_COMP_ID_0 Register Diagram

Table 45-19: SCB0_COMP_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_0	.

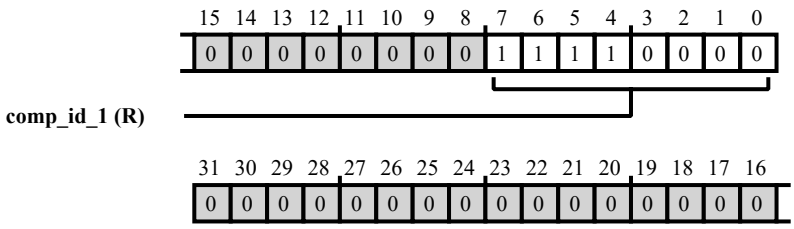


Figure 45-16: SCB0_COMP_ID_1 Register Diagram

Table 45-20: SCB0_COMP_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_1	.

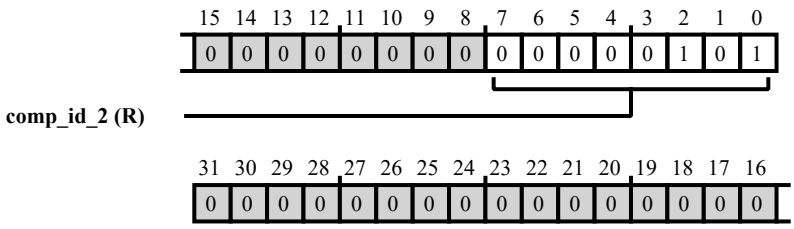


Figure 45-17: SCB0_COMP_ID_2 Register Diagram

Table 45-21: SCB0_COMP_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_2	.

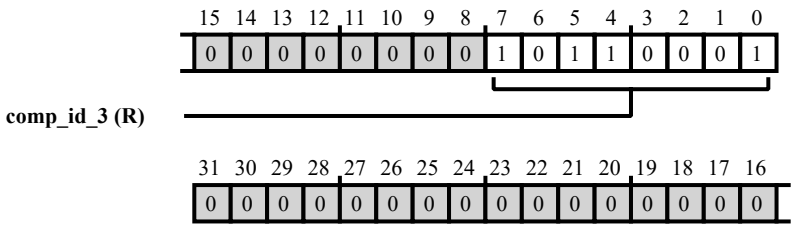


Figure 45-18: SCB0_COMP_ID_3 Register Diagram

Table 45-22: SCB0_COMP_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_3	.

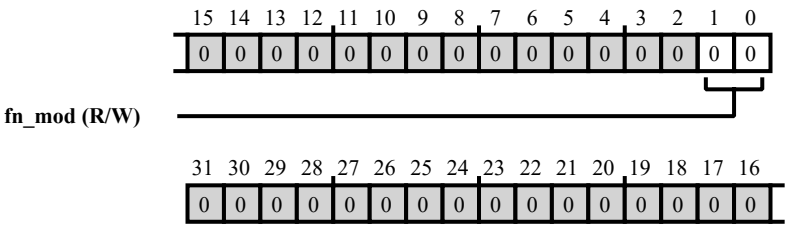


Figure 45-19: SCB0_CRC0_CH0_IB_FN_MOD Register Diagram

Table 45-23: SCB0_CRC0_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

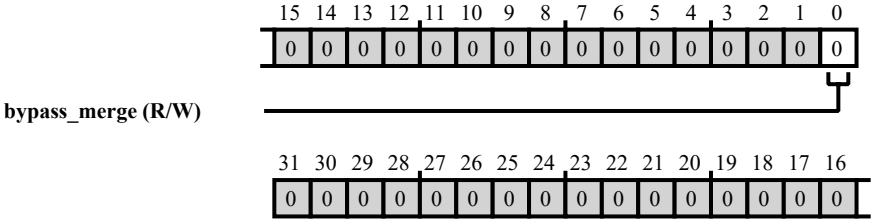


Figure 45-20: SCB0_CRC0_CH0_IB_FN_MOD2 Register Diagram

Table 45-24: SCB0_CRC0_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

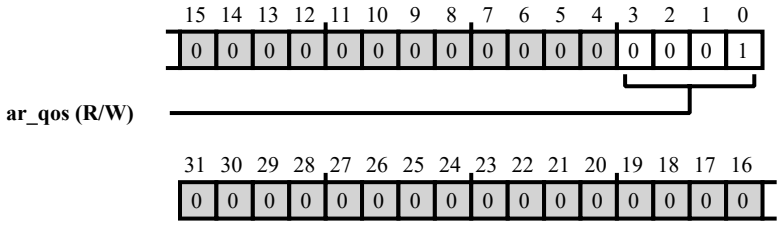


Figure 45-21: SCB0_CRC0_CH0_IB_READ_QOS Register Diagram

Table 45-25: SCB0_CRC0_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

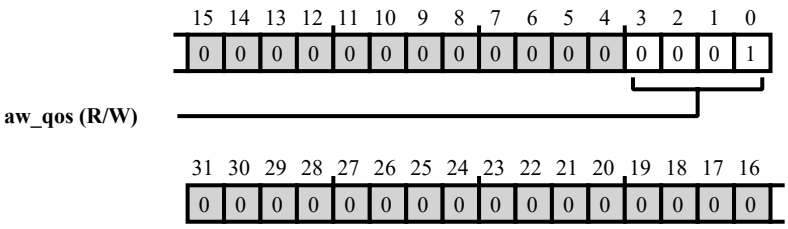


Figure 45-22: SCB0_CRC0_CH0_IB_WRITE_QOS Register Diagram

Table 45-26: SCB0_CRC0_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

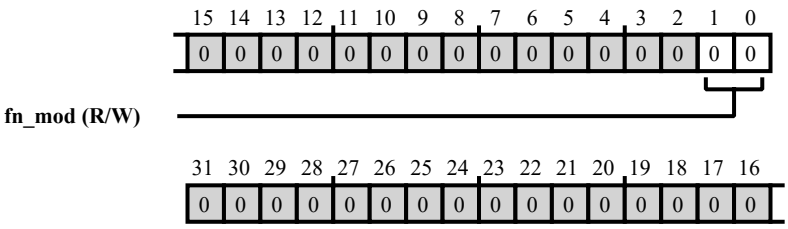


Figure 45-23: SCB0_CRC0_CH1_IB_FN_MOD Register Diagram

Table 45-27: SCB0_CRC0_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

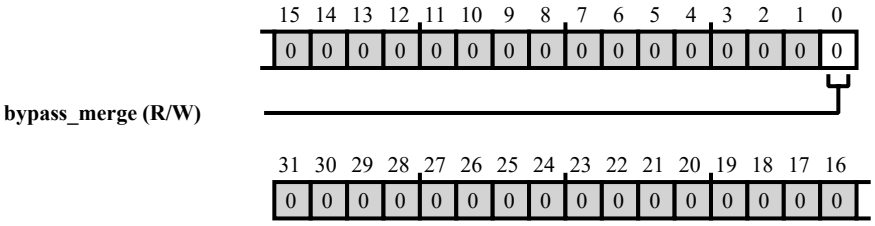


Figure 45-24: SCB0_CRC0_CH1_IB_FN_MOD2 Register Diagram

Table 45-28: SCB0_CRC0_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

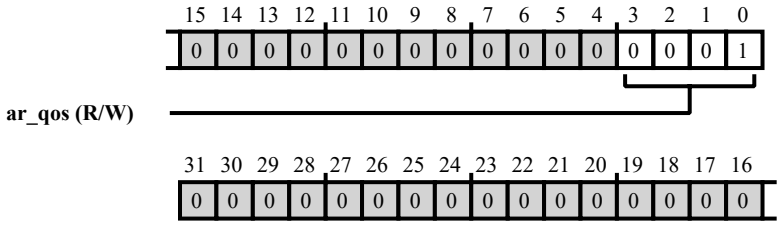


Figure 45-25: SCB0_CRC0_CH1_IB_READ_QOS Register Diagram

Table 45-29: SCB0_CRC0_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

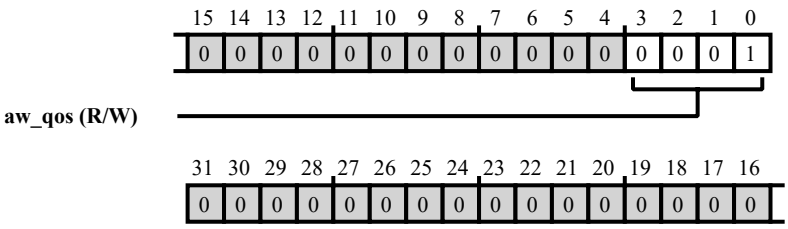


Figure 45-26: SCB0_CRC0_CH1_IB_WRITE_QOS Register Diagram

Table 45-30: SCB0_CRC0_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

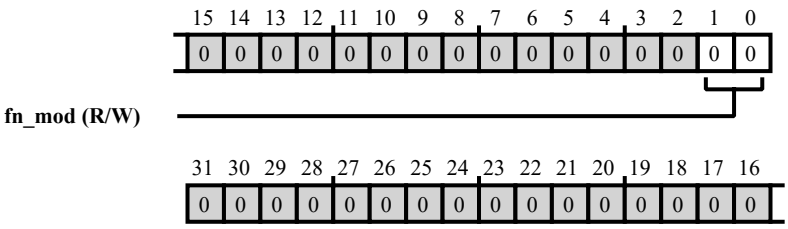


Figure 45-27: SCB0_CRC1_CH0_IB_FN_MOD Register Diagram

Table 45-31: SCB0_CRC1_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

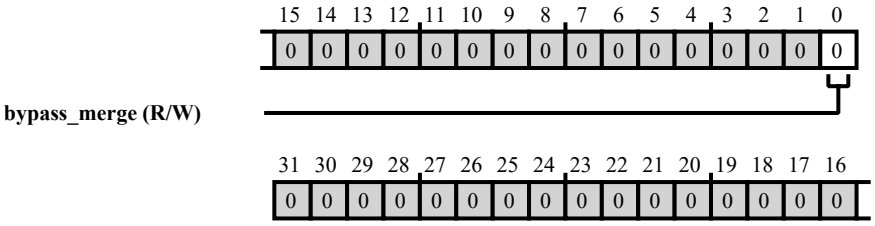


Figure 45-28: SCB0_CRC1_CH0_IB_FN_MOD2 Register Diagram

Table 45-32: SCB0_CRC1_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

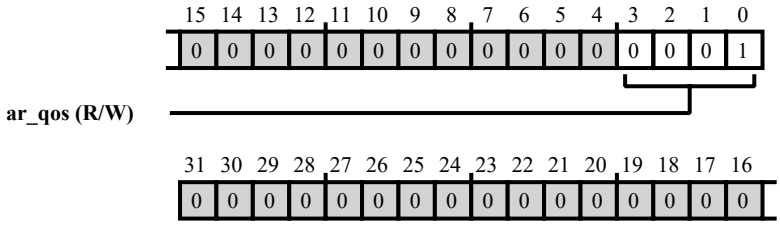


Figure 45-29: SCB0_CRC1_CH0_IB_READ_QOS Register Diagram

Table 45-33: SCB0_CRC1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

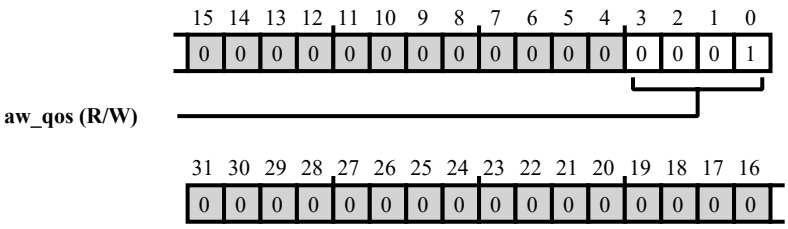


Figure 45-30: SCB0_CRC1_CH0_IB_WRITE_QOS Register Diagram

Table 45-34: SCB0_CRC1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

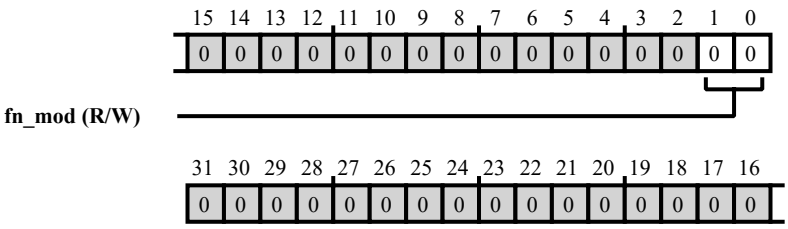


Figure 45-31: SCB0_CRC1_CH1_IB_FN_MOD Register Diagram

Table 45-35: SCB0_CRC1_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

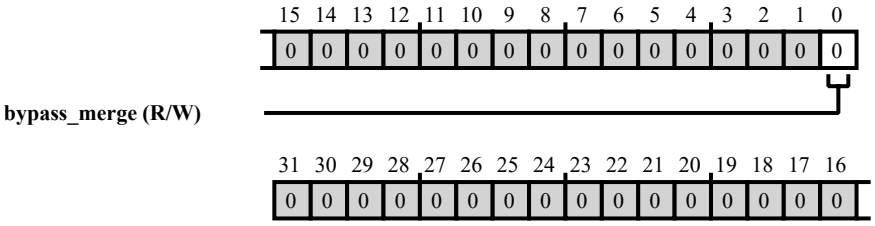


Figure 45-32: SCB0_CRC1_CH1_IB_FN_MOD2 Register Diagram

Table 45-36: SCB0_CRC1_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

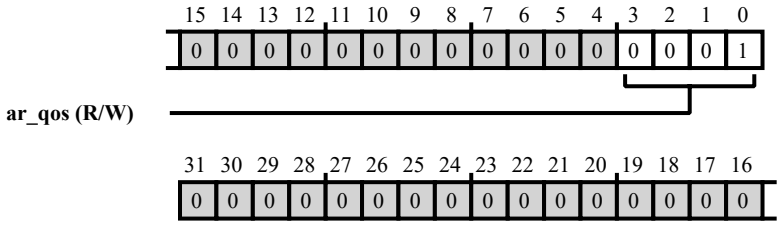


Figure 45-33: SCB0_CRC1_CH1_IB_READ_QOS Register Diagram

Table 45-37: SCB0_CRC1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

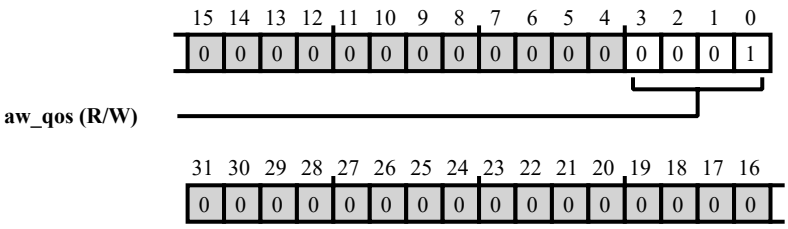


Figure 45-34: SCB0_CRC1_CH1_IB_WRITE_QOS Register Diagram

Table 45-38: SCB0_CRC1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

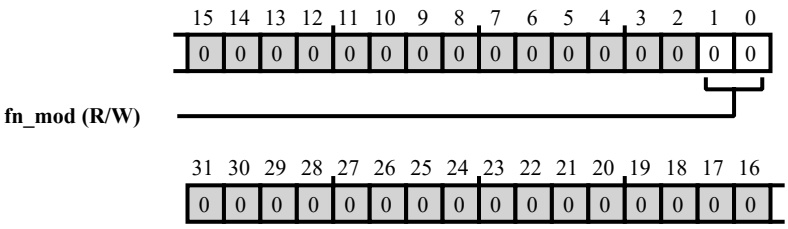


Figure 45-35: SCB0_CRC2_CH0_IB_FN_MOD Register Diagram

Table 45-39: SCB0_CRC2_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

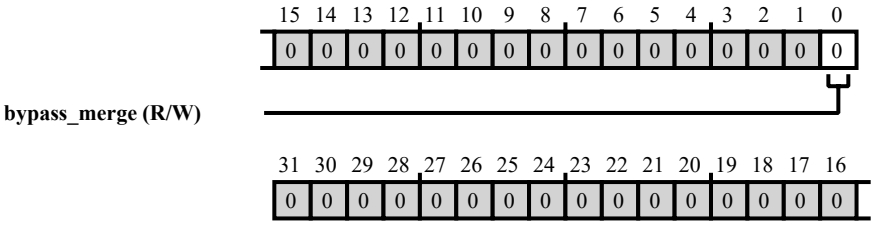


Figure 45-36: SCB0_CRC2_CH0_IB_FN_MOD2 Register Diagram

Table 45-40: SCB0_CRC2_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

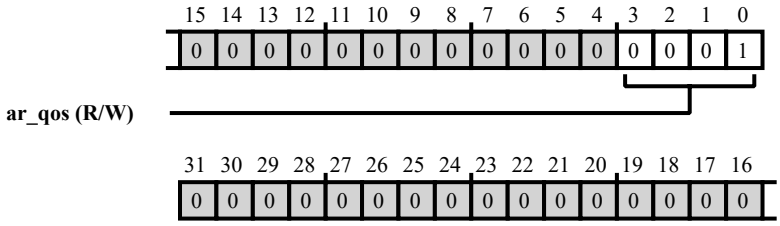


Figure 45-37: SCB0_CRC2_CH0_IB_READ_QOS Register Diagram

Table 45-41: SCB0_CRC2_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

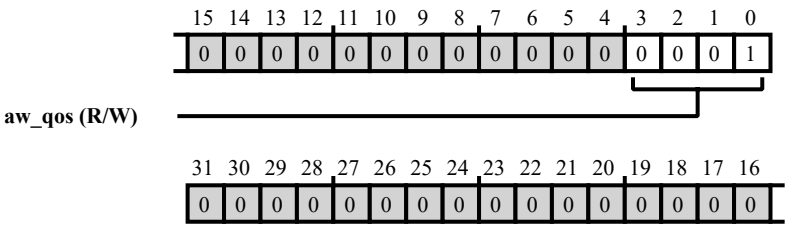


Figure 45-38: SCB0_CRC2_CH0_IB_WRITE_QOS Register Diagram

Table 45-42: SCB0_CRC2_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

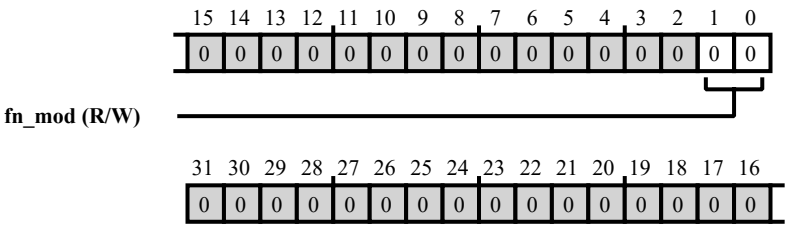


Figure 45-39: SCB0_CRC2_CH1_IB_FN_MOD Register Diagram

Table 45-43: SCB0_CRC2_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

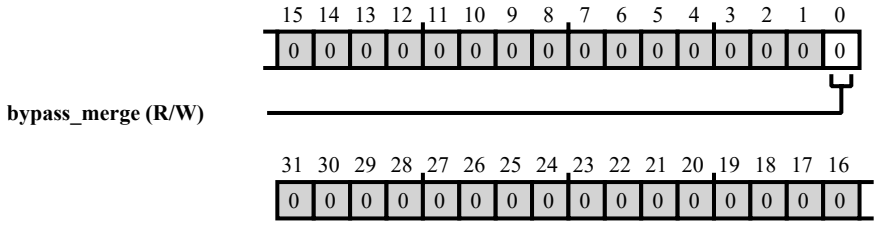


Figure 45-40: SCB0_CRC2_CH1_IB_FN_MOD2 Register Diagram

Table 45-44: SCB0_CRC2_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

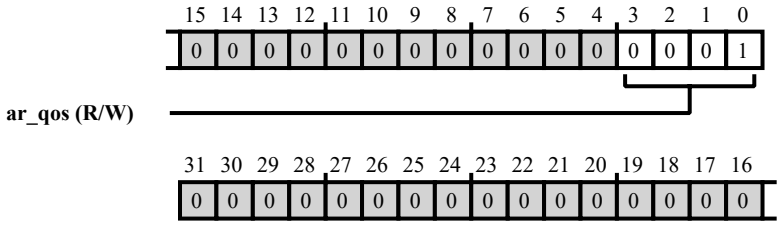


Figure 45-41: SCB0_CRC2_CH1_IB_READ_QOS Register Diagram

Table 45-45: SCB0_CRC2_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

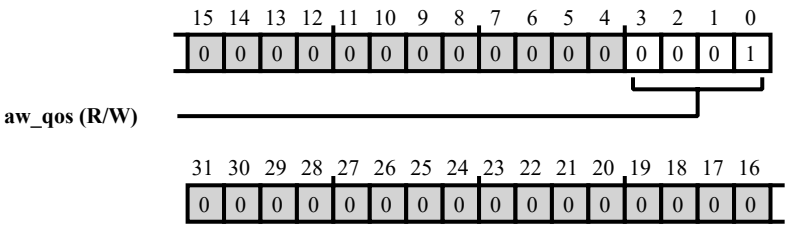


Figure 45-42: SCB0_CRC2_CH1_IB_WRITE_QOS Register Diagram

Table 45-46: SCB0_CRC2_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

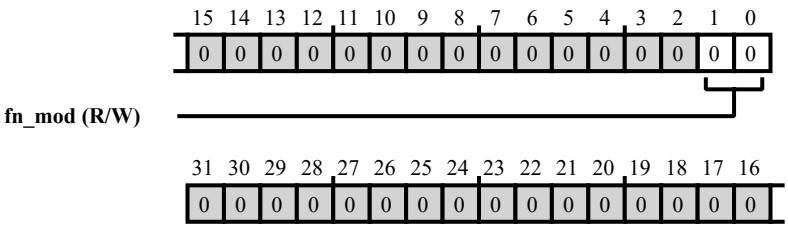


Figure 45-43: SCB0_CRC3_CH0_IB_FN_MOD Register Diagram

Table 45-47: SCB0_CRC3_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

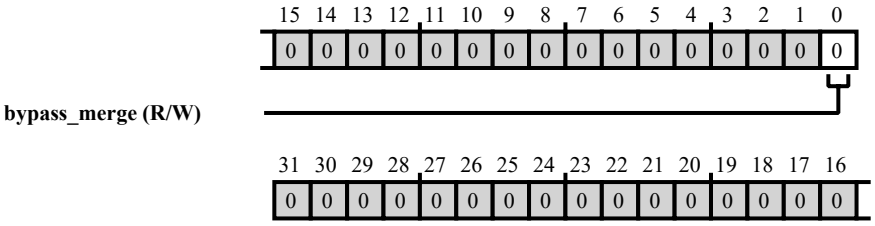


Figure 45-44: SCB0_CRC3_CH0_IB_FN_MOD2 Register Diagram

Table 45-48: SCB0_CRC3_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

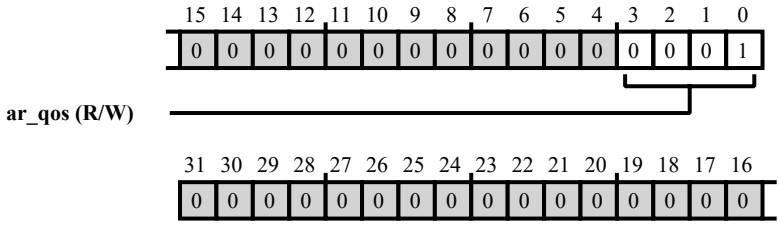


Figure 45-45: SCB0_CRC3_CH0_IB_READ_QOS Register Diagram

Table 45-49: SCB0_CRC3_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

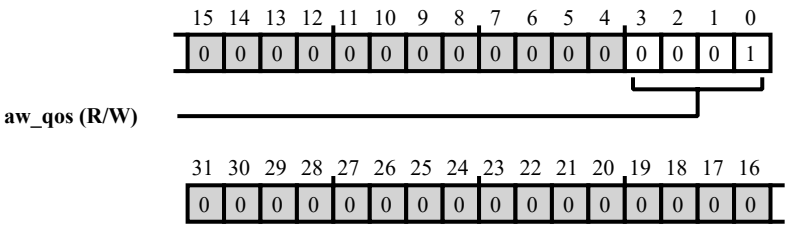


Figure 45-46: SCB0_CRC3_CH0_IB_WRITE_QOS Register Diagram

Table 45-50: SCB0_CRC3_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

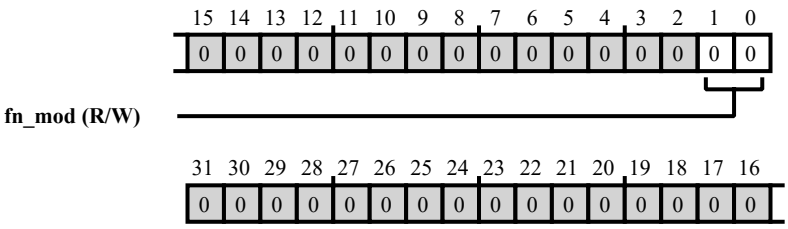


Figure 45-47: SCB0_CRC3_CH1_IB_FN_MOD Register Diagram

Table 45-51: SCB0_CRC3_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

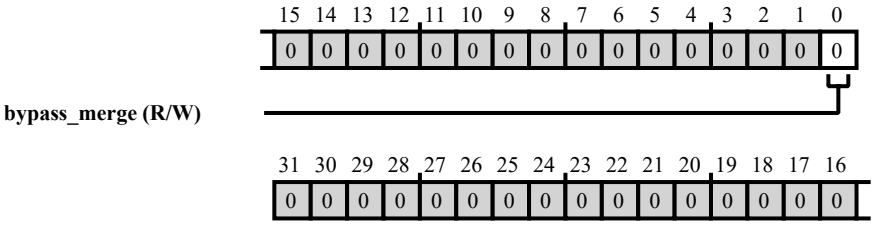


Figure 45-48: SCB0_CRC3_CH1_IB_FN_MOD2 Register Diagram

Table 45-52: SCB0_CRC3_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

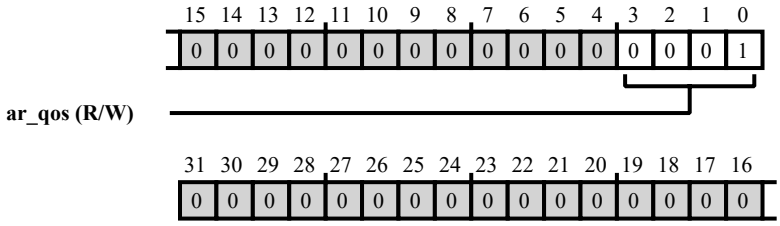


Figure 45-49: SCB0_CRC3_CH1_IB_READ_QOS Register Diagram

Table 45-53: SCB0_CRC3_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

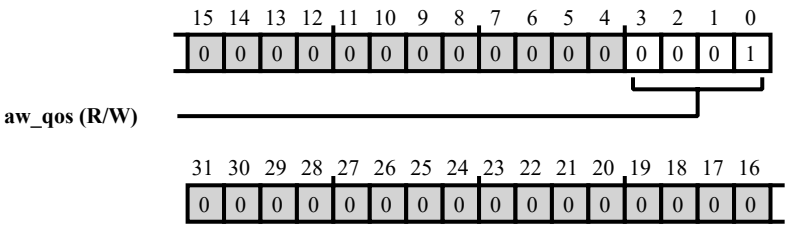


Figure 45-50: SCB0_CRC3_CH1_IB_WRITE_QOS Register Diagram

Table 45-54: SCB0_CRC3_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

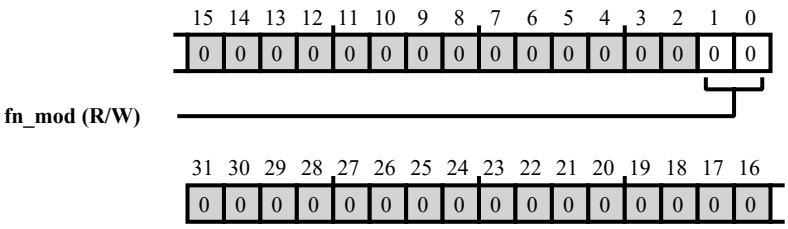


Figure 45-51: SCB0_CRYPTO_IB_FN_MOD Register Diagram

Table 45-55: SCB0_CRYPTO_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

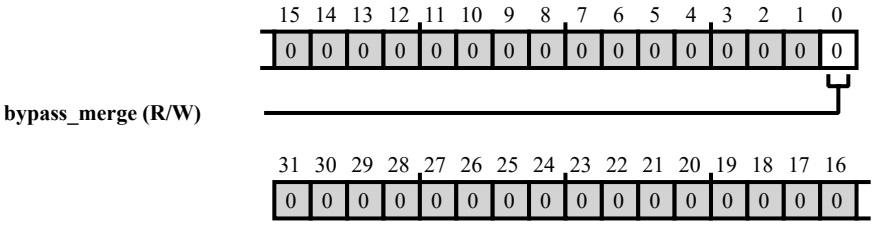


Figure 45-52: SCB0_CRYPTO_IB_FN_MOD2 Register Diagram

Table 45-56: SCB0_CRYPTO_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

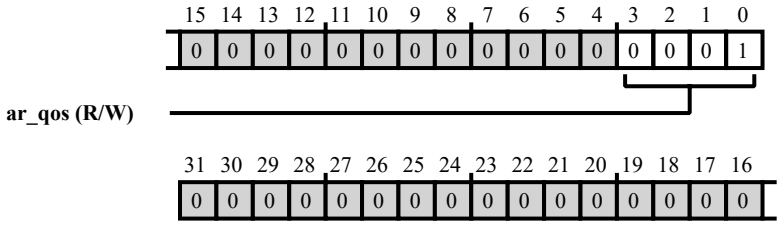


Figure 45-53: SCB0_CRYPTO_IB_READ_QOS Register Diagram

Table 45-57: SCB0_CRYPTO_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

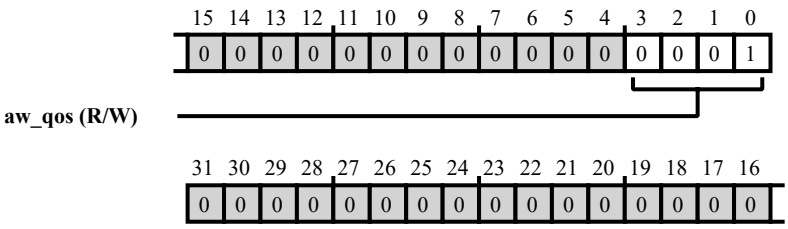


Figure 45-54: SCB0_CRYPTO_IB_WRITE_QOS Register Diagram

Table 45-58: SCB0_CRYPTO_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

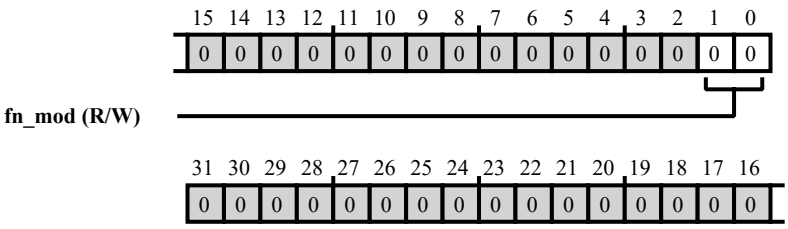


Figure 45-55: SCB0_DBG_IB_FN_MOD Register Diagram

Table 45-59: SCB0_DBG_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

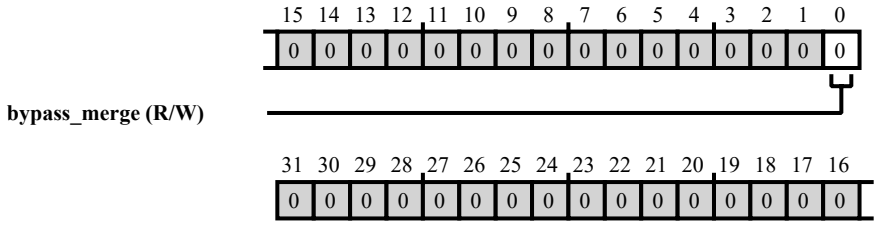


Figure 45-56: SCB0_DBG_IB_FN_MOD2 Register Diagram

Table 45-60: SCB0_DBG_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

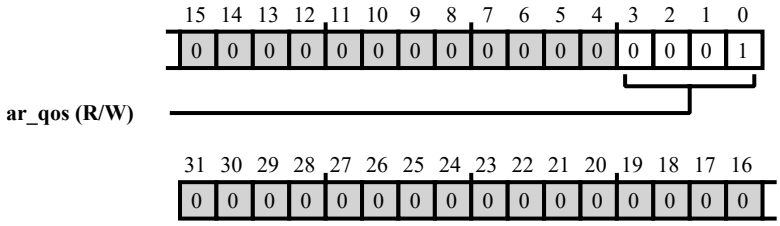


Figure 45-57: SCB0_DBG_IB_READ_QOS Register Diagram

Table 45-61: SCB0_DBG_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

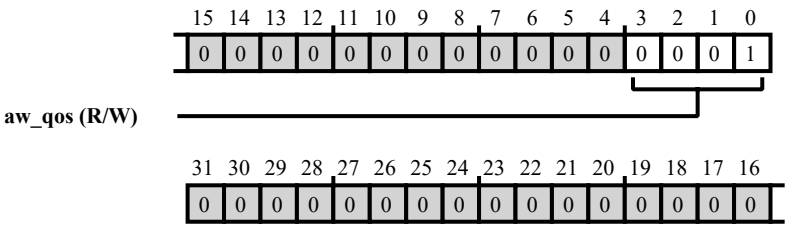


Figure 45-58: SCB0_DBG_IB_WRITE_QOS Register Diagram

Table 45-62: SCB0_DBG_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

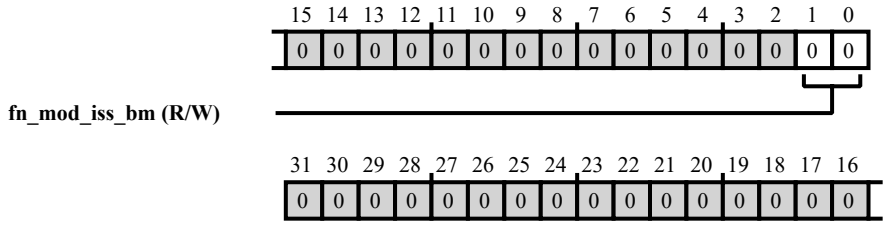


Figure 45-59: SCB0_DL2_0_FN_MOD_ISS_BM Register Diagram

Table 45-63: SCB0_DL2_0_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

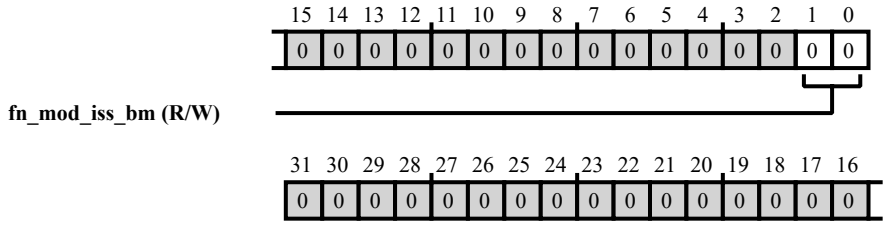


Figure 45-60: SCB0_DL2_1_FN_MOD_ISS_BM Register Diagram

Table 45-64: SCB0_DL2_1_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

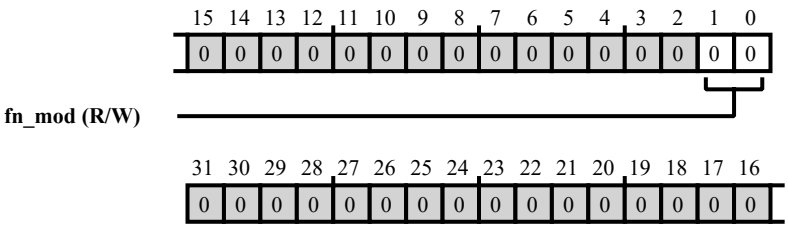


Figure 45-61: SCB0_DLDMA0_CH0_IB_FN_MOD Register Diagram

Table 45-65: SCB0_DLDMA0_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

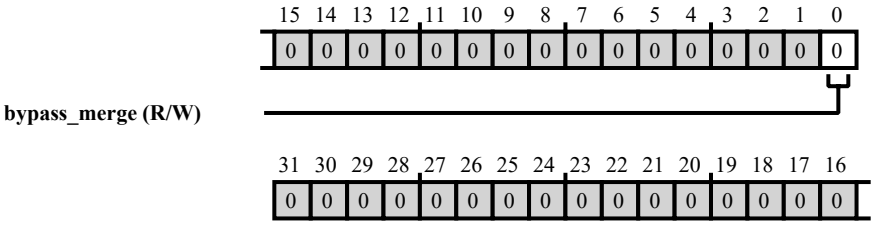


Figure 45-62: SCB0_DLDMA0_CH0_IB_FN_MOD2 Register Diagram

Table 45-66: SCB0_DLDMA0_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

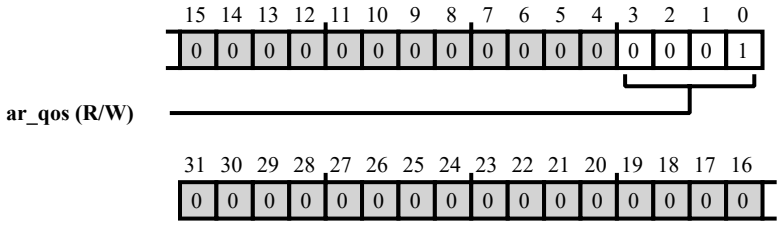


Figure 45-63: SCB0_DLDMA0_CH0_IB_READ_QOS Register Diagram

Table 45-67: SCB0_DLDMA0_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

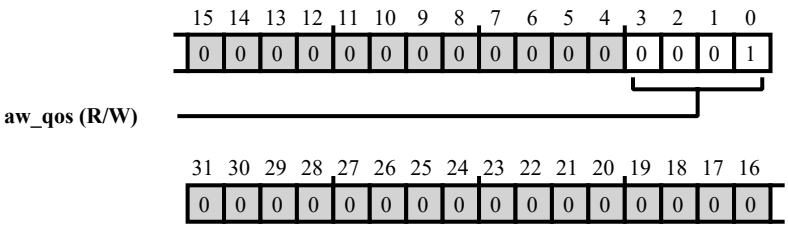


Figure 45-64: SCB0_DLDMA0_CH0_IB_WRITE_QOS Register Diagram

Table 45-68: SCB0_DLDMA0_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

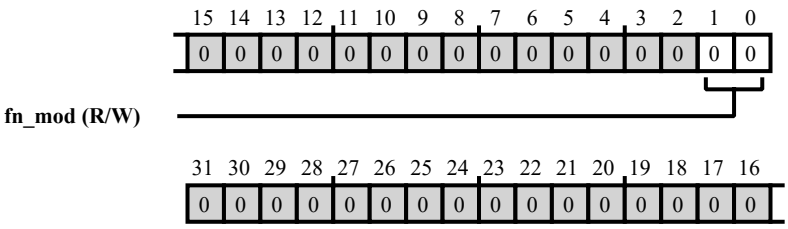


Figure 45-65: SCB0_DLDMA0_CH1_IB_FN_MOD Register Diagram

Table 45-69: SCB0_DLDMA0_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

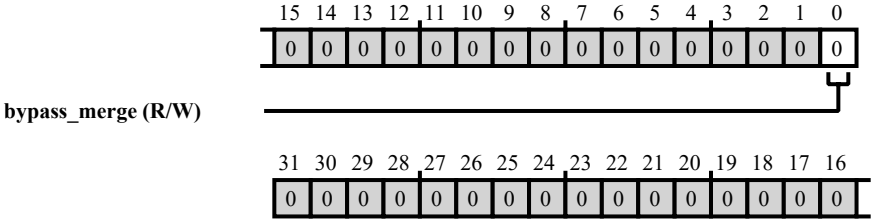


Figure 45-66: SCB0_DLDMA0_CH1_IB_FN_MOD2 Register Diagram

Table 45-70: SCB0_DLDMA0_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

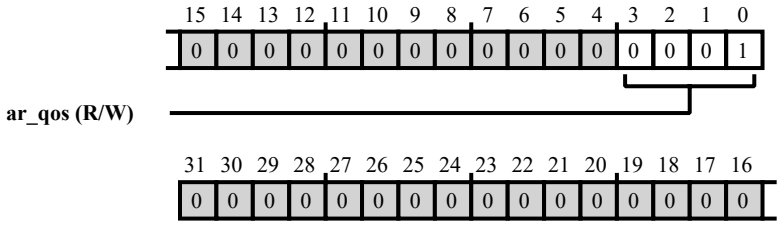


Figure 45-67: SCB0_DLDMA0_CH1_IB_READ_QOS Register Diagram

Table 45-71: SCB0_DLDMA0_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

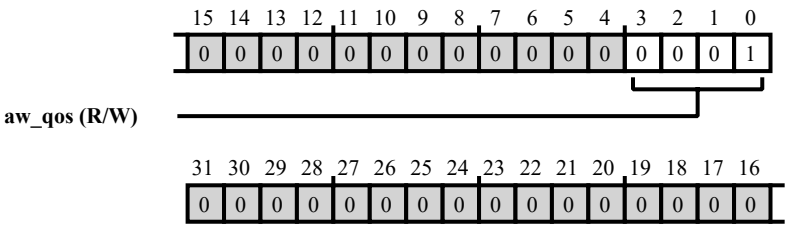


Figure 45-68: SCB0_DLDMA0_CH1_IB_WRITE_QOS Register Diagram

Table 45-72: SCB0_DLDMA0_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

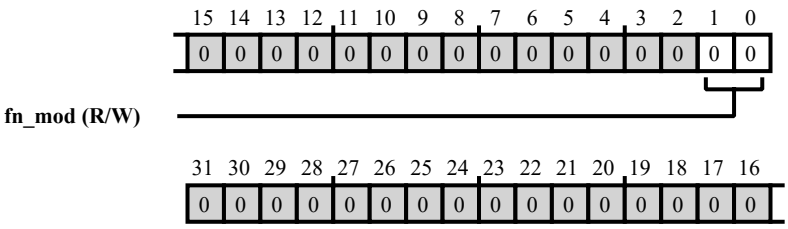


Figure 45-69: SCB0_DLDMA1_CH0_IB_FN_MOD Register Diagram

Table 45-73: SCB0_DLDMA1_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

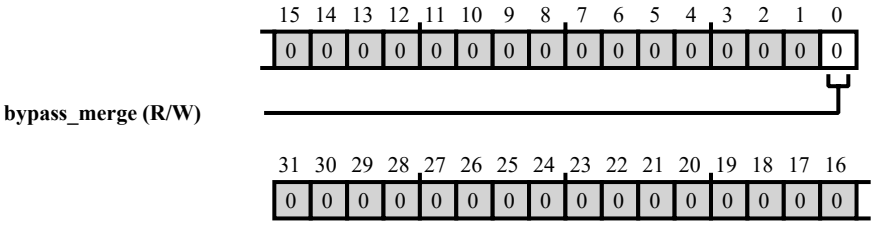


Figure 45-70: SCB0_DLDMA1_CH0_IB_FN_MOD2 Register Diagram

Table 45-74: SCB0_DLDMA1_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

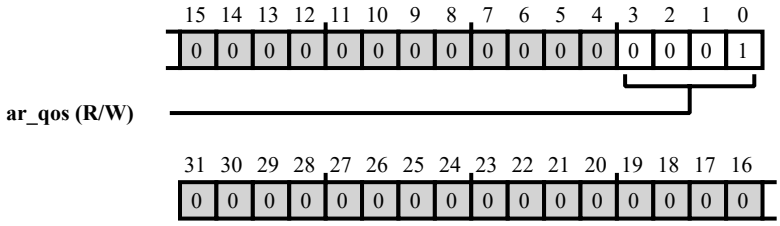


Figure 45-71: SCB0_DLDMA1_CH0_IB_READ_QOS Register Diagram

Table 45-75: SCB0_DLDMA1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

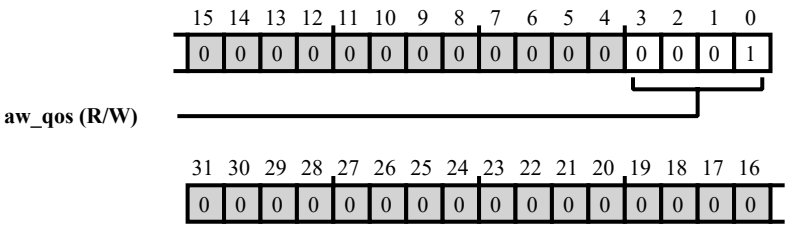


Figure 45-72: SCB0_DLDMA1_CH0_IB_WRITE_QOS Register Diagram

Table 45-76: SCB0_DLDMA1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

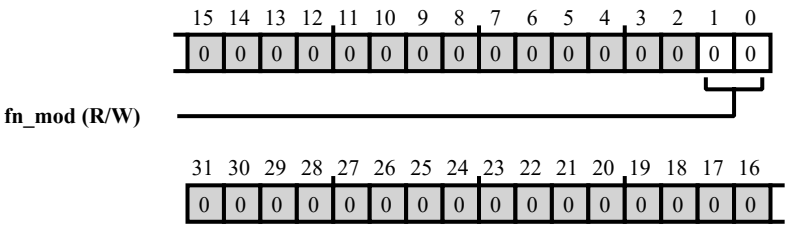


Figure 45-73: SCB0_DLDMA1_CH1_IB_FN_MOD Register Diagram

Table 45-77: SCB0_DLDMA1_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

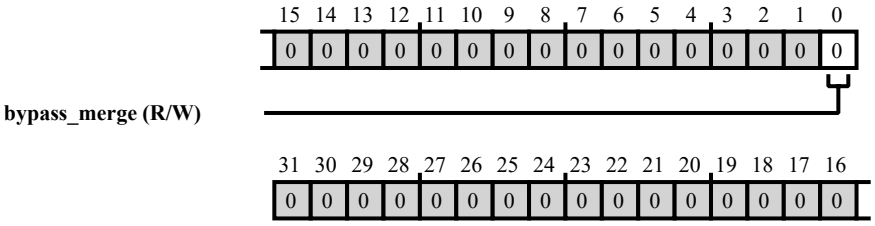


Figure 45-74: SCB0_DLDMA1_CH1_IB_FN_MOD2 Register Diagram

Table 45-78: SCB0_DLDMA1_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

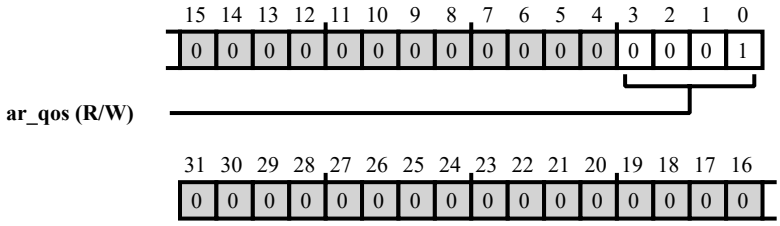


Figure 45-75: SCB0_DLDMA1_CH1_IB_READ_QOS Register Diagram

Table 45-79: SCB0_DLDMA1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

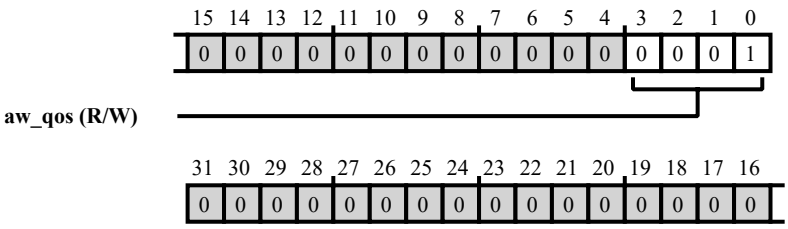


Figure 45-76: SCB0_DLDMA1_CH1_IB_WRITE_QOS Register Diagram

Table 45-80: SCB0_DLDMA1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

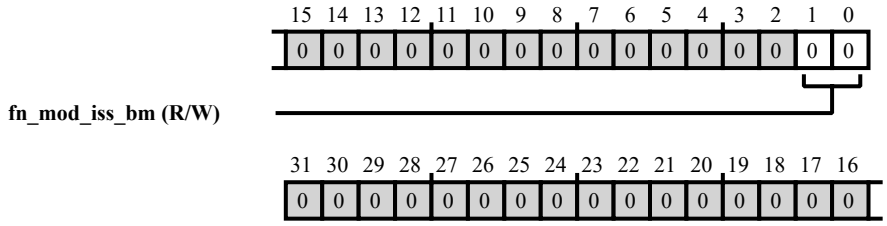


Figure 45-77: SCB0_DMC0_FN_MOD_ISS_BM Register Diagram

Table 45-81: SCB0_DMC0_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

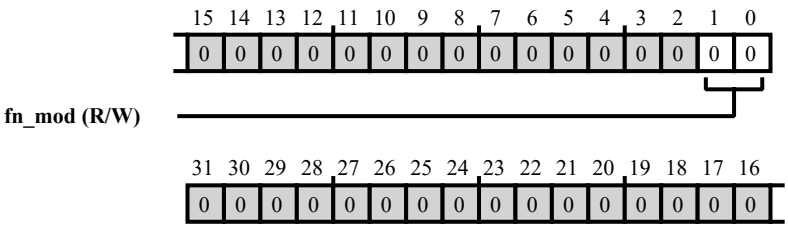


Figure 45-78: SCB0_EMMC_IB_FN_MOD Register Diagram

Table 45-82: SCB0_EMMC_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

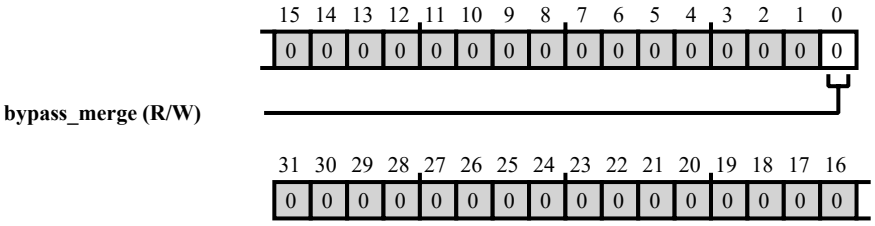


Figure 45-79: SCB0_EMMC_IB_FN_MOD2 Register Diagram

Table 45-83: SCB0_EMMC_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

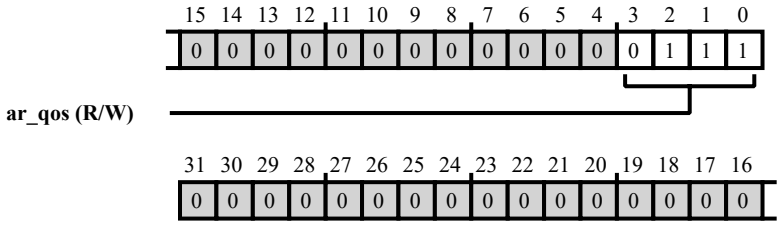


Figure 45-80: SCB0_EMMC_IB_READ_QOS Register Diagram

Table 45-84: SCB0_EMMC_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

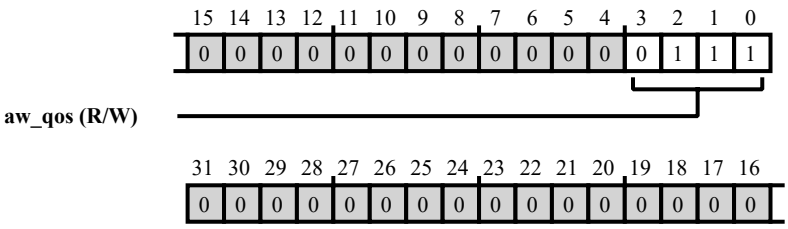


Figure 45-81: SCB0_EMMC_IB_WRITE_QOS Register Diagram

Table 45-85: SCB0_EMMC_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

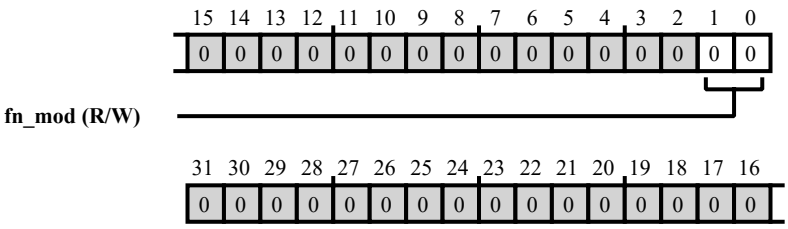


Figure 45-82: SCB0_ETR_IB_FN_MOD Register Diagram

Table 45-86: SCB0_ETR_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

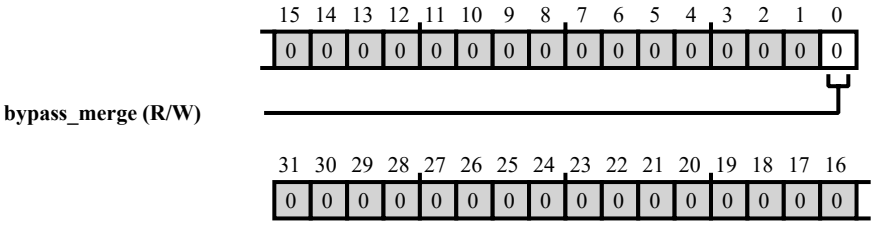


Figure 45-83: SCB0_ETR_IB_FN_MOD2 Register Diagram

Table 45-87: SCB0_ETR_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

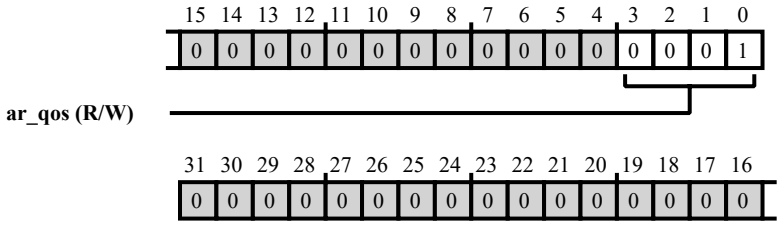


Figure 45-84: SCB0_ETR_IB_READ_QOS Register Diagram

Table 45-88: SCB0_ETR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

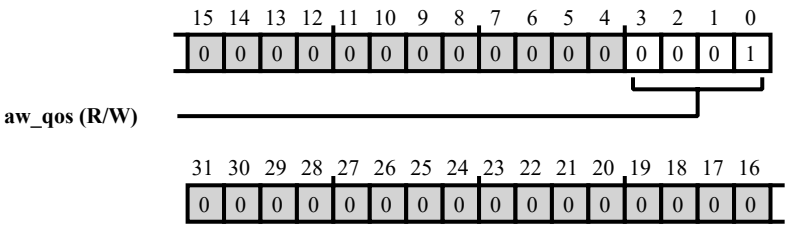


Figure 45-85: SCB0_ETR_IB_WRITE_QOS Register Diagram

Table 45-89: SCB0_ETR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

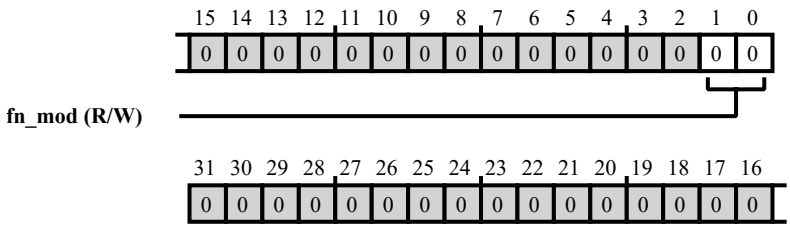


Figure 45-86: SCB0_GIGE1_IB_FN_MOD Register Diagram

Table 45-90: SCB0_GIGE1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

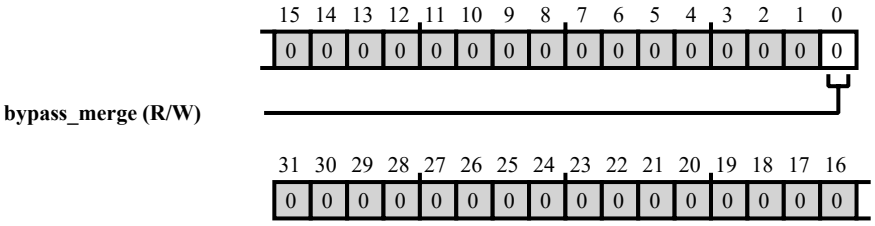


Figure 45-87: SCB0_GIGE1_IB_FN_MOD2 Register Diagram

Table 45-91: SCB0_GIGE1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

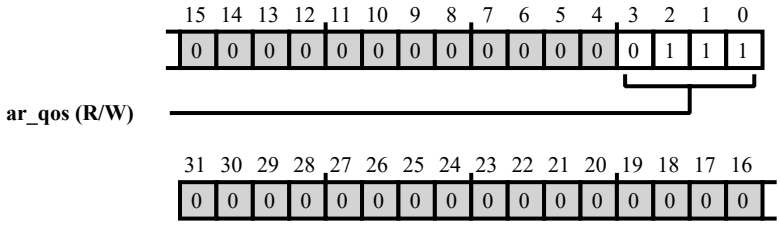


Figure 45-88: SCB0_GIGE1_IB_READ_QOS Register Diagram

Table 45-92: SCB0_GIGE1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

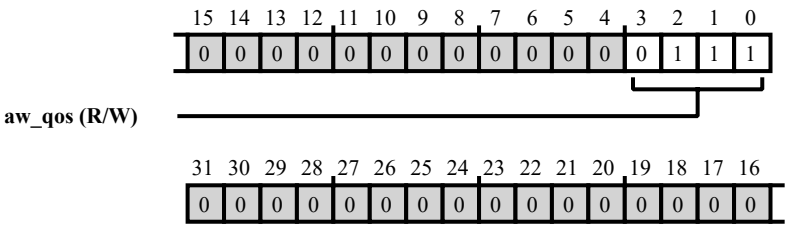


Figure 45-89: SCB0_GIGE1_IB_WRITE_QOS Register Diagram

Table 45-93: SCB0_GIGE1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

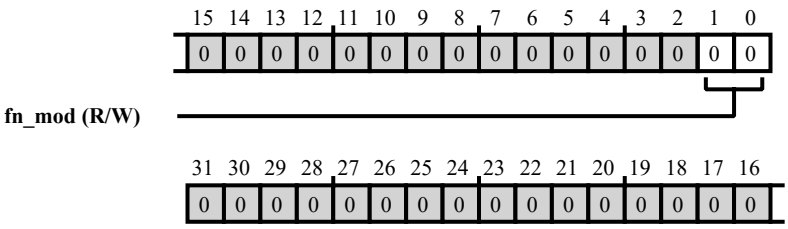


Figure 45-90: SCB0_GIGE_IB_FN_MOD Register Diagram

Table 45-94: SCB0_GIGE_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

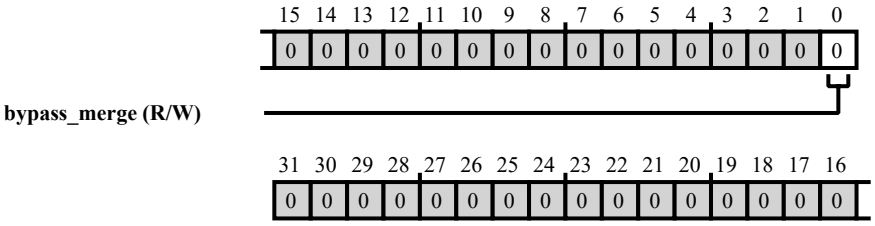


Figure 45-91: SCB0_GIGE_IB_FN_MOD2 Register Diagram

Table 45-95: SCB0_GIGE_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

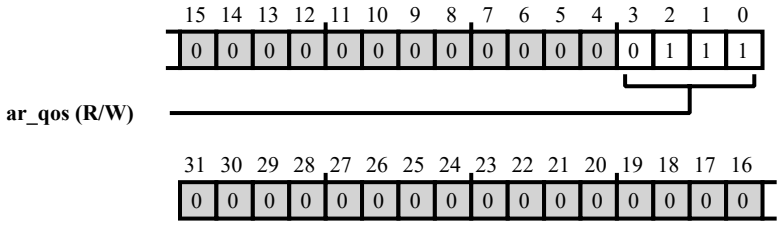


Figure 45-92: SCB0_GIGE_IB_READ_QOS Register Diagram

Table 45-96: SCB0_GIGE_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

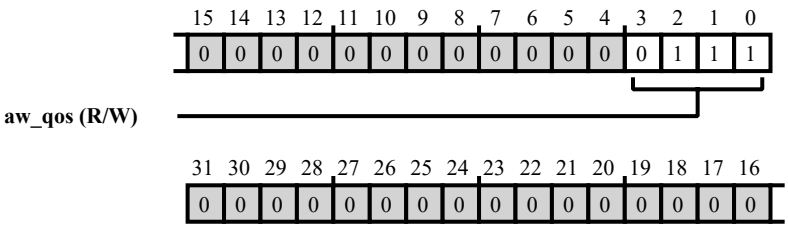


Figure 45-93: SCB0_GIGE_IB_WRITE_QOS Register Diagram

Table 45-97: SCB0_GIGE_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

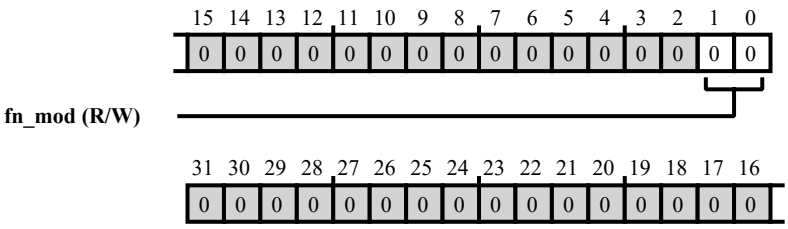


Figure 45-94: SCB0_HSMDMA1_CH0_IB_FN_MOD Register Diagram

Table 45-98: SCB0_HSMDMA1_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

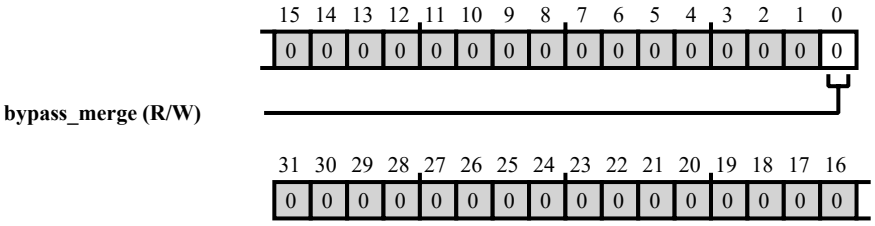


Figure 45-95: SCB0_HSMDMA1_CH0_IB_FN_MOD2 Register Diagram

Table 45-99: SCB0_HSMDMA1_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

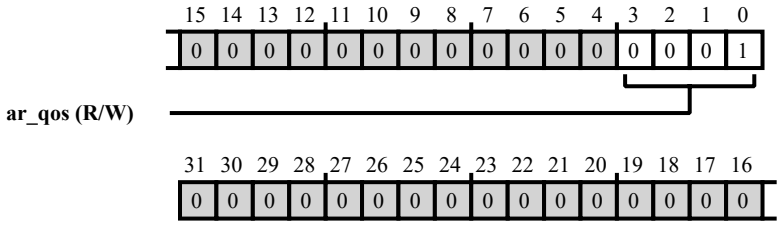


Figure 45-96: SCB0_HSMDMA1_CH0_IB_READ_QOS Register Diagram

Table 45-100: SCB0_HSMDMA1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

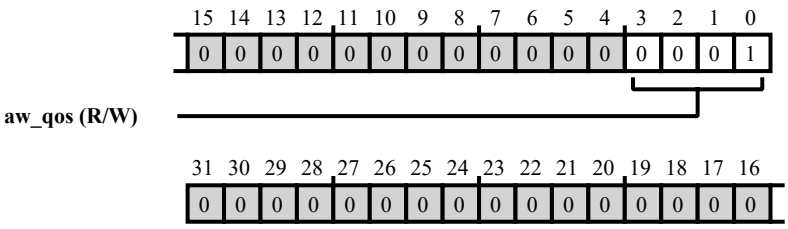


Figure 45-97: SCB0_HSMDMA1_CH0_IB_WRITE_QOS Register Diagram

Table 45-101: SCB0_HSMDMA1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

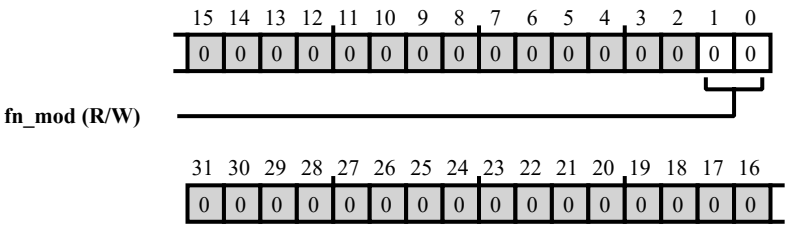


Figure 45-98: SCB0_HSMDMA1_CH1_IB_FN_MOD Register Diagram

Table 45-102: SCB0_HSMDMA1_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

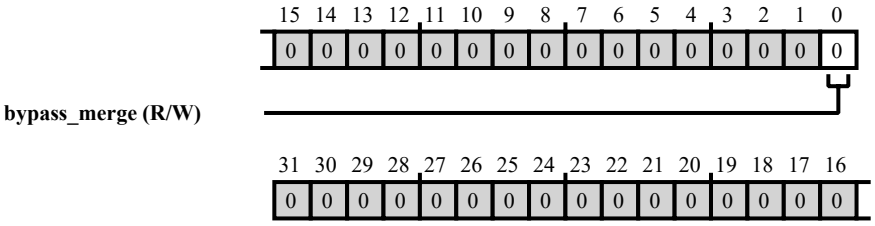


Figure 45-99: SCB0_HSMDMA1_CH1_IB_FN_MOD2 Register Diagram

Table 45-103: SCB0_HSMDMA1_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

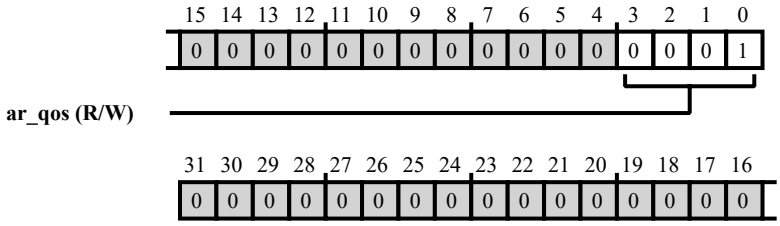


Figure 45-100: SCB0_HSMDMA1_CH1_IB_READ_QOS Register Diagram

Table 45-104: SCB0_HSMDMA1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

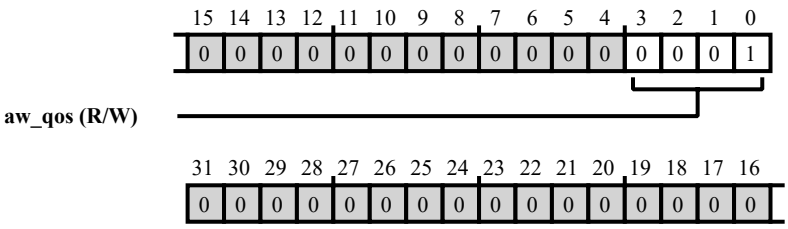


Figure 45-101: SCB0_HSMDMA1_CH1_IB_WRITE_QOS Register Diagram

Table 45-105: SCB0_HSMDMA1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

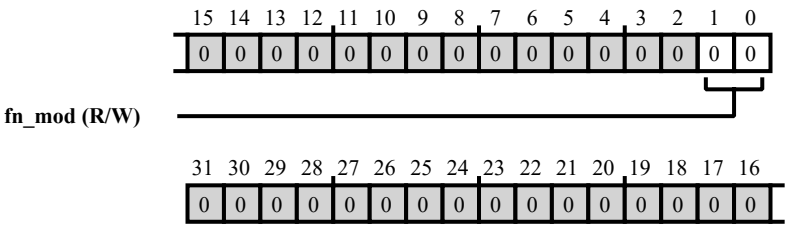


Figure 45-102: SCB0_HSMDMA_CH0_IB_FN_MOD Register Diagram

Table 45-106: SCB0_HSMDMA_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

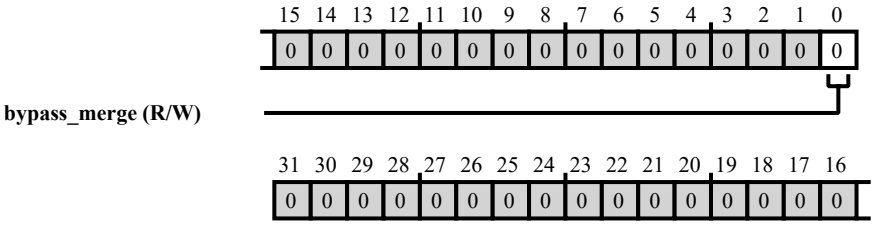


Figure 45-103: SCB0_HSMDMA_CH0_IB_FN_MOD2 Register Diagram

Table 45-107: SCB0_HSMDMA_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

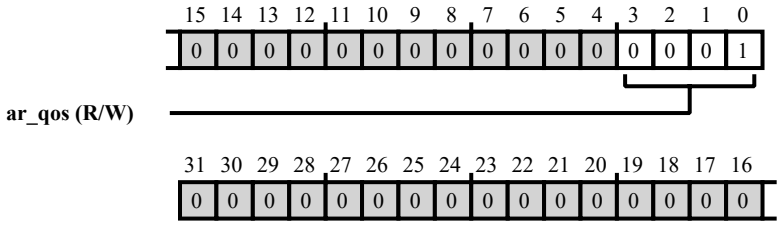


Figure 45-104: SCB0_HSMDMA_CH0_IB_READ_QOS Register Diagram

Table 45-108: SCB0_HSMDMA_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

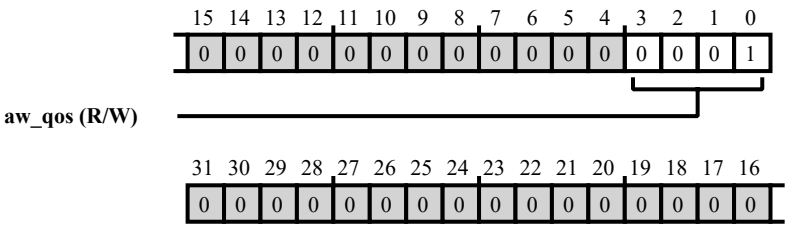


Figure 45-105: SCB0_HSMDMA_CH0_IB_WRITE_QOS Register Diagram

Table 45-109: SCB0_HSMDMA_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

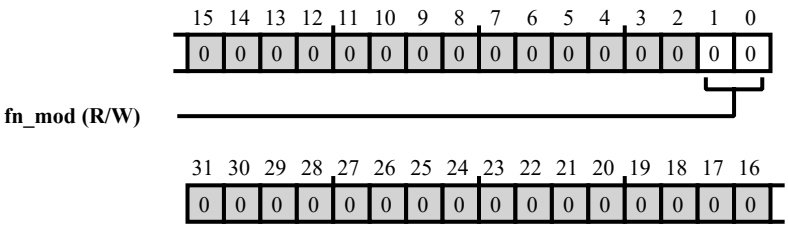


Figure 45-106: SCB0_HSMDMA_CH1_IB_FN_MOD Register Diagram

Table 45-110: SCB0_HSMDMA_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

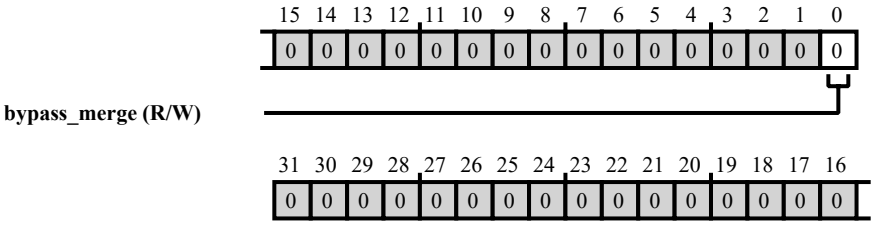


Figure 45-107: SCB0_HSMDMA_CH1_IB_FN_MOD2 Register Diagram

Table 45-111: SCB0_HSMDMA_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

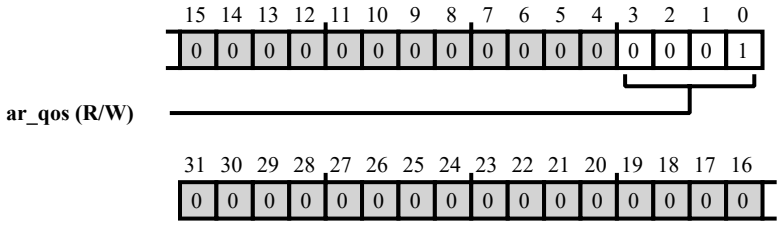


Figure 45-108: SCB0_HSMDMA_CH1_IB_READ_QOS Register Diagram

Table 45-112: SCB0_HSMDMA_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

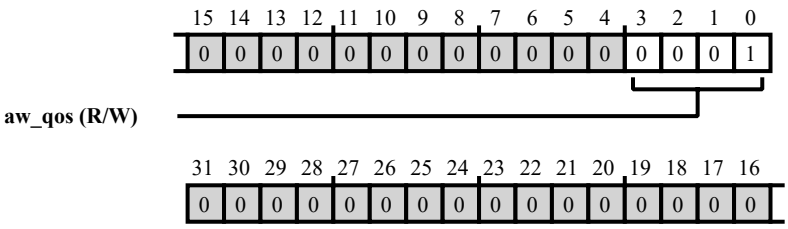


Figure 45-109: SCB0_HSMDMA_CH1_IB_WRITE_QOS Register Diagram

Table 45-113: SCB0_HSMDMA_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

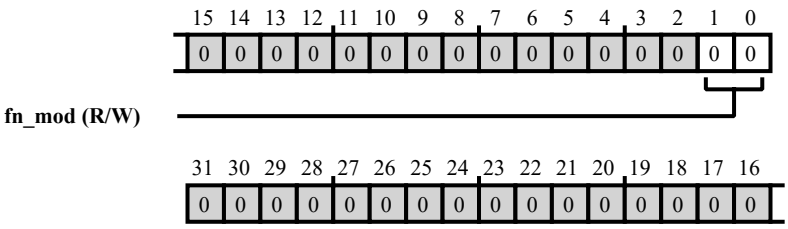


Figure 45-110: SCB0_IDMA_FN_MOD Register Diagram

Table 45-114: SCB0_IDMA_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

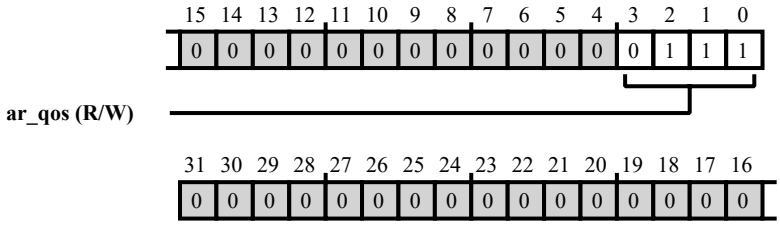


Figure 45-111: SCB0_IDMA_READ_QOS Register Diagram

Table 45-115: SCB0_IDMA_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

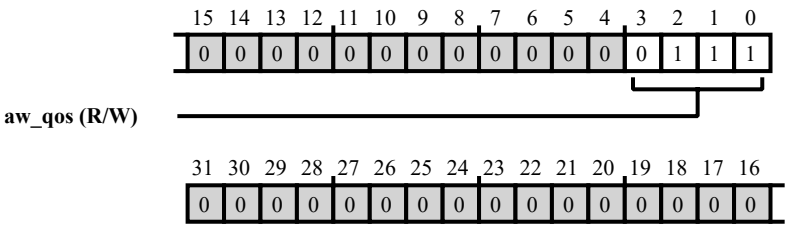


Figure 45-112: SCB0_IDMA_WRITE_QOS Register Diagram

Table 45-116: SCB0_IDMA_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

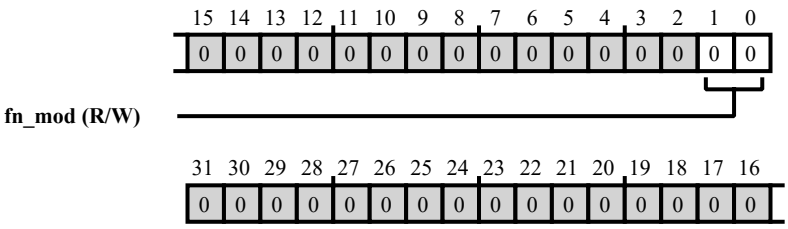


Figure 45-113: SCB0_LP0_FN_MOD Register Diagram

Table 45-117: SCB0_LP0_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

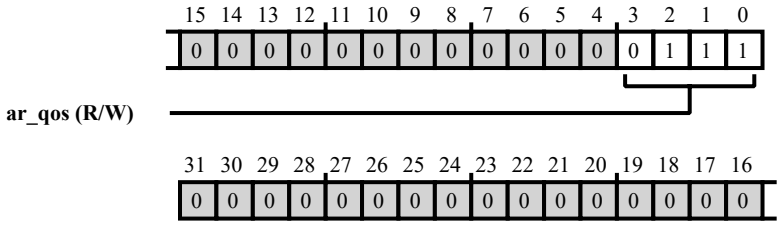


Figure 45-114: SCB0_LP0_READ_QOS Register Diagram

Table 45-118: SCB0_LP0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

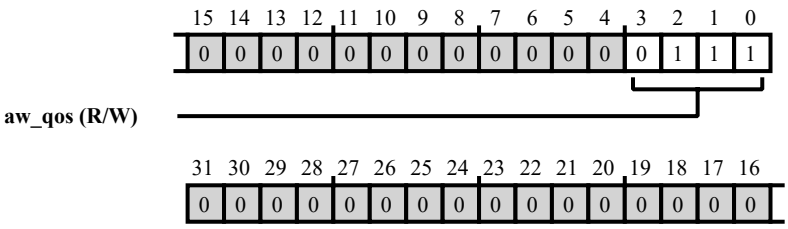


Figure 45-115: SCB0_LP0_WRITE_QOS Register Diagram

Table 45-119: SCB0_LP0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

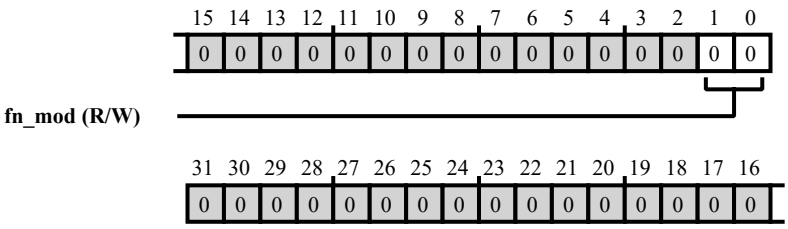


Figure 45-116: SCB0_LP1_FN_MOD Register Diagram

Table 45-120: SCB0_LP1_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

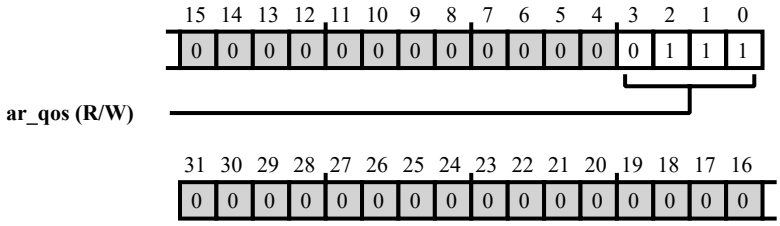


Figure 45-117: SCB0_LP1_READ_QOS Register Diagram

Table 45-121: SCB0_LP1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

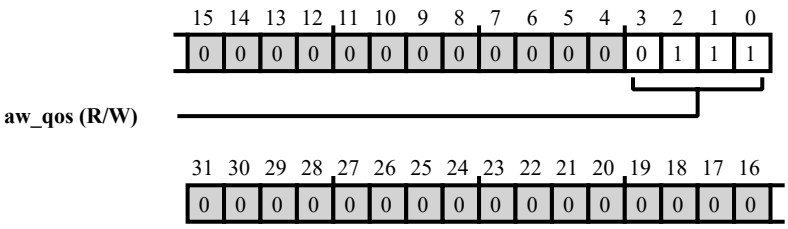


Figure 45-118: SCB0_LP1_WRITE_QOS Register Diagram

Table 45-122: SCB0_LP1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

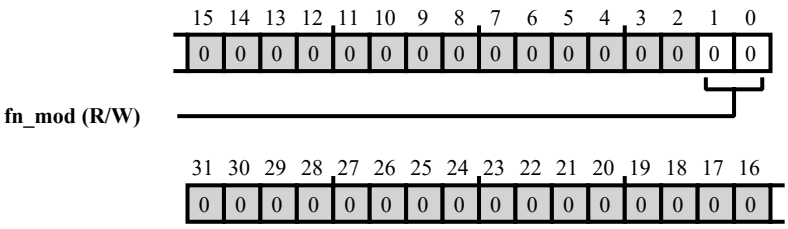


Figure 45-119: SCB0_M85_M2_AXI_FN_MOD Register Diagram

Table 45-123: SCB0_M85_M2_AXI_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

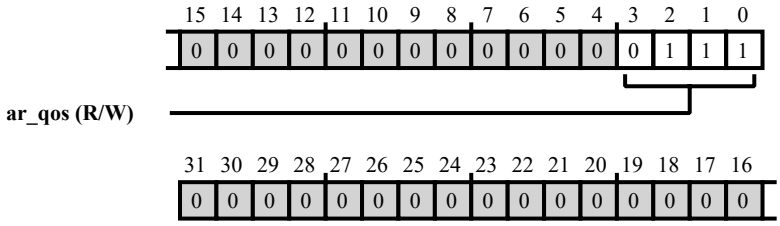


Figure 45-120: SCB0_M85_M2_AXI_READ_QOS Register Diagram

Table 45-124: SCB0_M85_M2_AXI_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

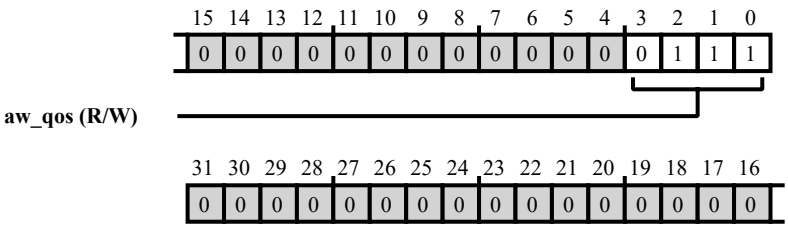


Figure 45-121: SCB0_M85_M2_AXI_WRITE_QOS Register Diagram

Table 45-125: SCB0_M85_M2_AXI_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

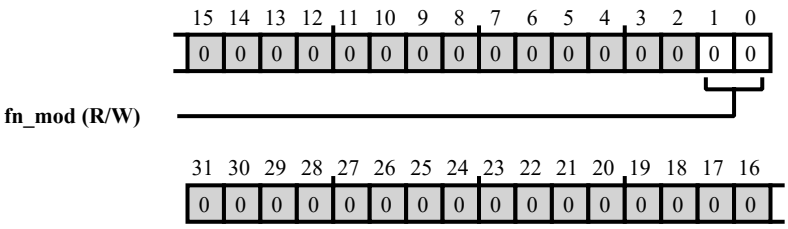


Figure 45-122: SCB0_M85_M_AXI_FN_MOD Register Diagram

Table 45-126: SCB0_M85_M_AXI_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

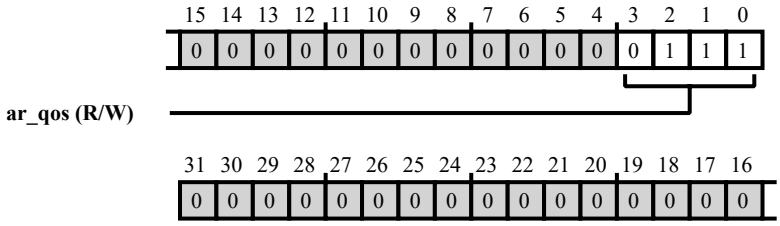


Figure 45-123: SCB0_M85_M_AXI_READ_QOS Register Diagram

Table 45-127: SCB0_M85_M_AXI_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

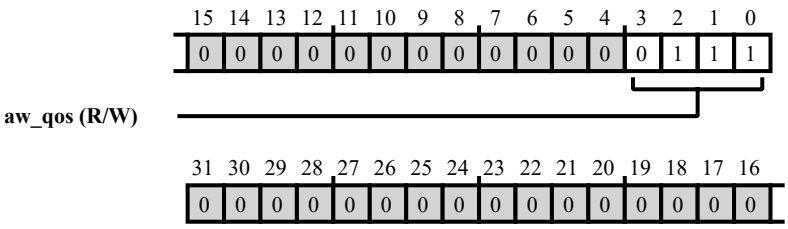


Figure 45-124: SCB0_M85_M_AXI_WRITE_QOS Register Diagram

Table 45-128: SCB0_M85_M_AXI_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

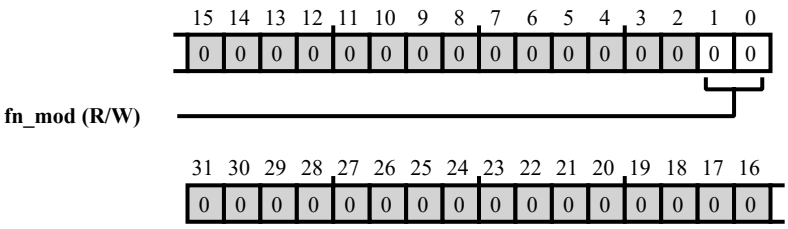


Figure 45-125: SCB0_M85_S2_AXI_IB_FN_MOD Register Diagram

Table 45-129: SCB0_M85_S2_AXI_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

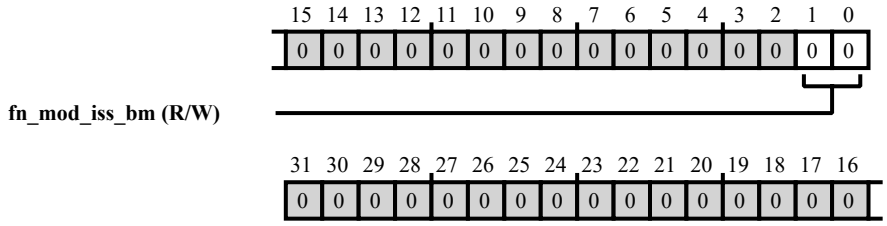


Figure 45-126: SCB0_M85_S2_AXI_IB_FN_MOD_ISS_BM Register Diagram

Table 45-130: SCB0_M85_S2_AXI_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

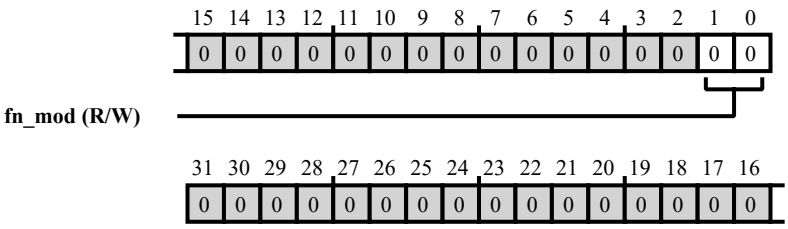


Figure 45-127: SCB0_M85_S_AXI_IB_FN_MOD Register Diagram

Table 45-131: SCB0_M85_S_AXI_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

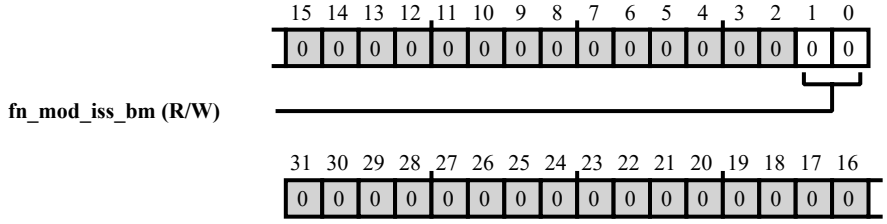


Figure 45-128: SCB0_M85_S_AXI_IB_FN_MOD_ISS_BM Register Diagram

Table 45-132: SCB0_M85_S_AXI_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

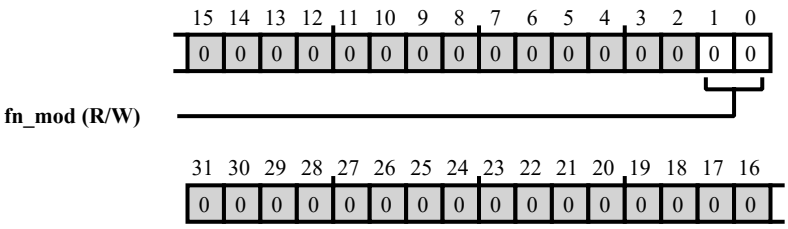


Figure 45-129: SCB0_MLB_FN_MOD Register Diagram

Table 45-133: SCB0_MLB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

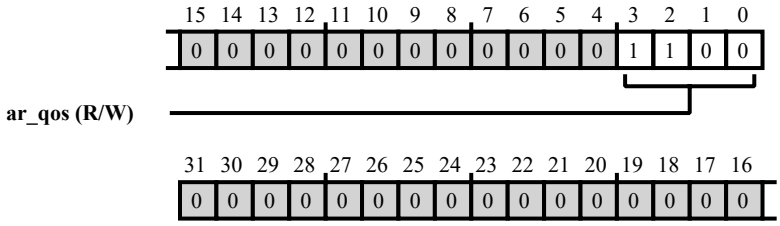


Figure 45-130: SCB0_MLB_READ_QOS Register Diagram

Table 45-134: SCB0_MLB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

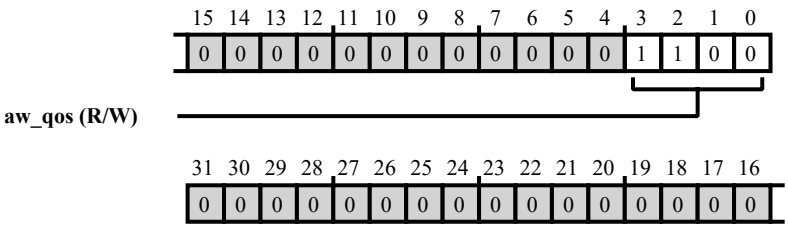


Figure 45-131: SCB0_MLB_WRITE_QOS Register Diagram

Table 45-135: SCB0_MLB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

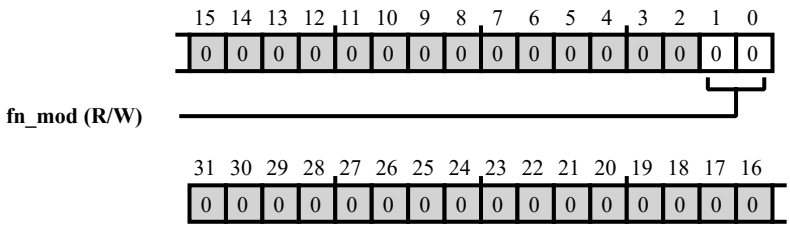


Figure 45-132: SCB0_MSMDMA1_CH0_IB_FN_MOD Register Diagram

Table 45-136: SCB0_MSMDMA1_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

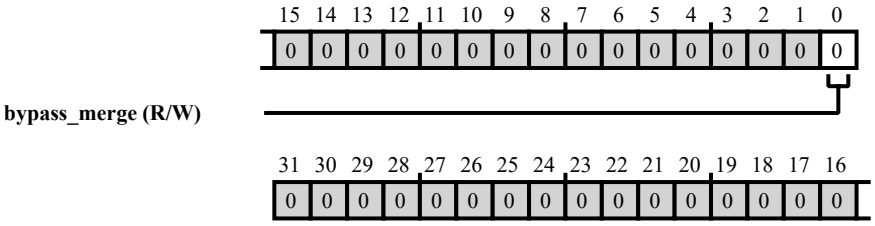


Figure 45-133: SCB0_MSMDMA1_CH0_IB_FN_MOD2 Register Diagram

Table 45-137: SCB0_MSMDMA1_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

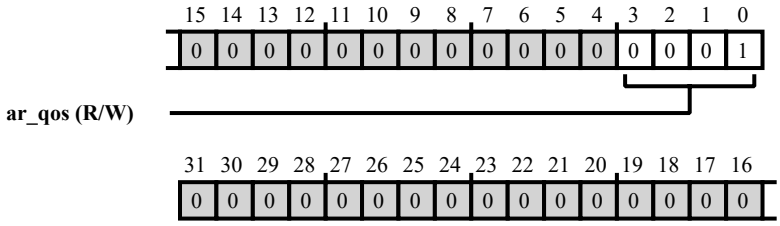


Figure 45-134: SCB0_MSMDMA1_CH0_IB_READ_QOS Register Diagram

Table 45-138: SCB0_MSMDMA1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

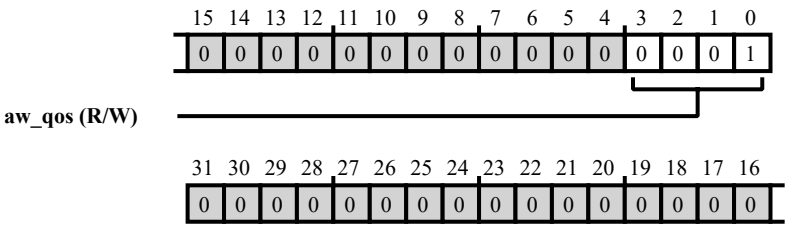


Figure 45-135: SCB0_MSMDMA1_CH0_IB_WRITE_QOS Register Diagram

Table 45-139: SCB0_MSMDMA1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

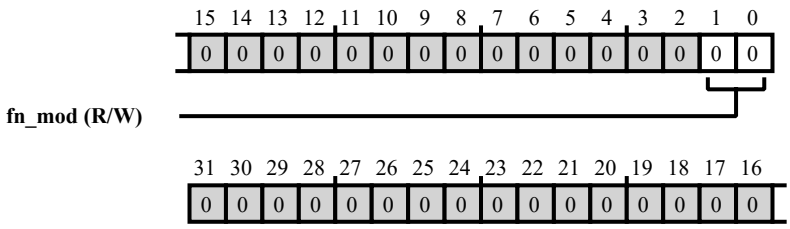


Figure 45-136: SCB0_MSMDMA1_CH1_IB_FN_MOD Register Diagram

Table 45-140: SCB0_MSMDMA1_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

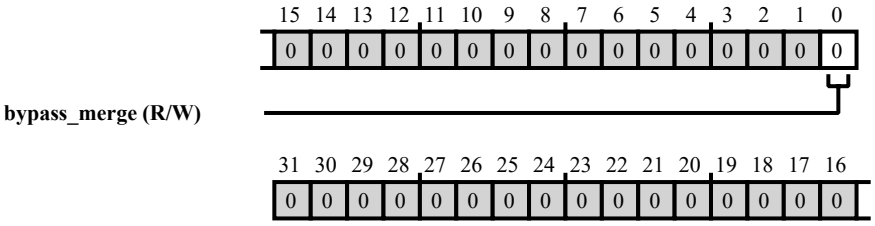


Figure 45-137: SCB0_MSMDMA1_CH1_IB_FN_MOD2 Register Diagram

Table 45-141: SCB0_MSMDMA1_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

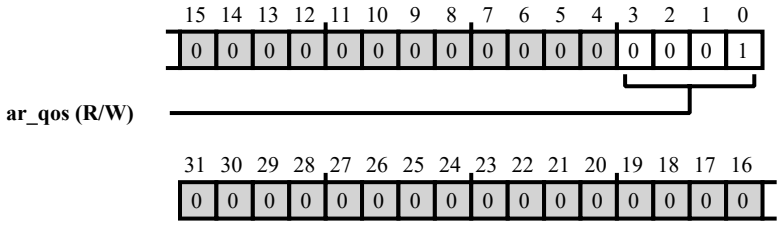


Figure 45-138: SCB0_MSMDMA1_CH1_IB_READ_QOS Register Diagram

Table 45-142: SCB0_MSMDMA1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

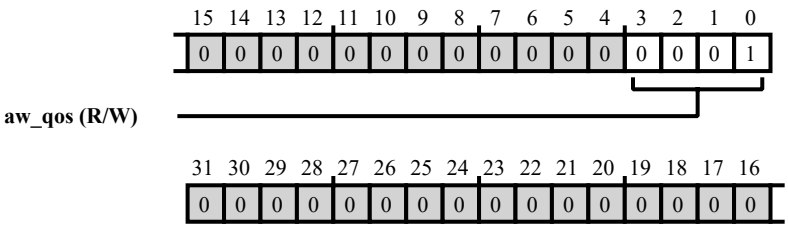


Figure 45-139: SCB0_MSMDMA1_CH1_IB_WRITE_QOS Register Diagram

Table 45-143: SCB0_MSMDMA1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

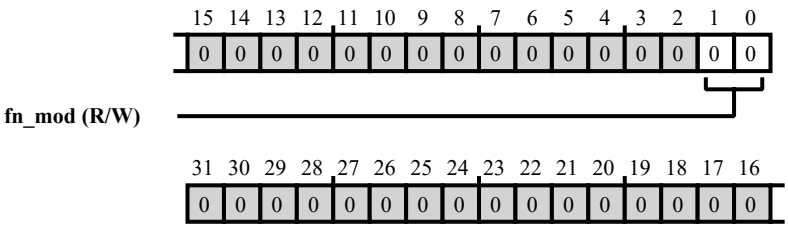


Figure 45-140: SCB0_MSMDMA_CH0_IB_FN_MOD Register Diagram

Table 45-144: SCB0_MSMDMA_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

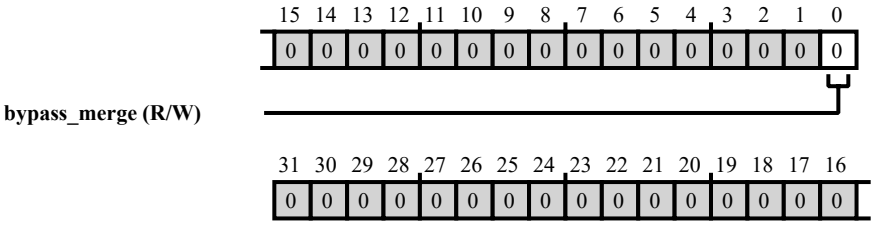


Figure 45-141: SCB0_MSMDMA_CH0_IB_FN_MOD2 Register Diagram

Table 45-145: SCB0_MSMDMA_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

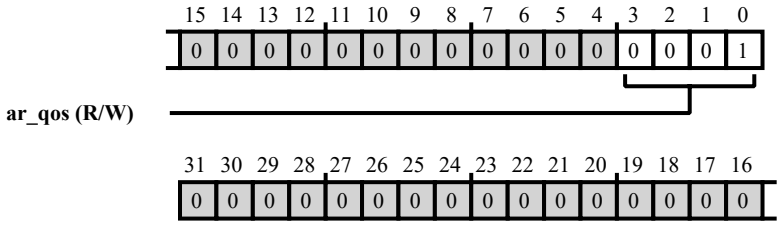


Figure 45-142: SCB0_MSMDMA_CH0_IB_READ_QOS Register Diagram

Table 45-146: SCB0_MSMDMA_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

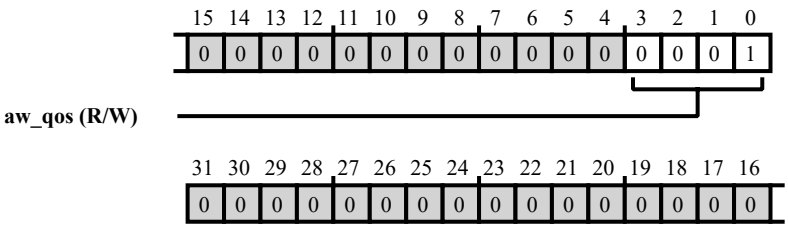


Figure 45-143: SCB0_MSMDMA_CH0_IB_WRITE_QOS Register Diagram

Table 45-147: SCB0_MSMDMA_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

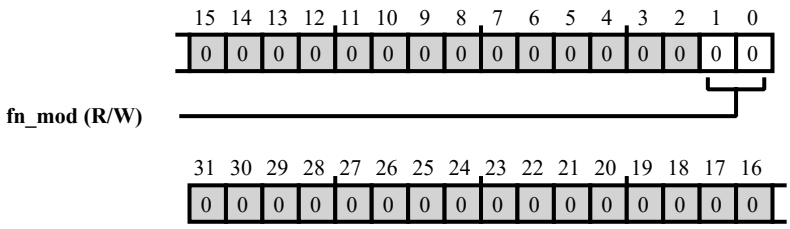


Figure 45-144: SCB0_MSMDMA_CH1_IB_FN_MOD Register Diagram

Table 45-148: SCB0_MSMDMA_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

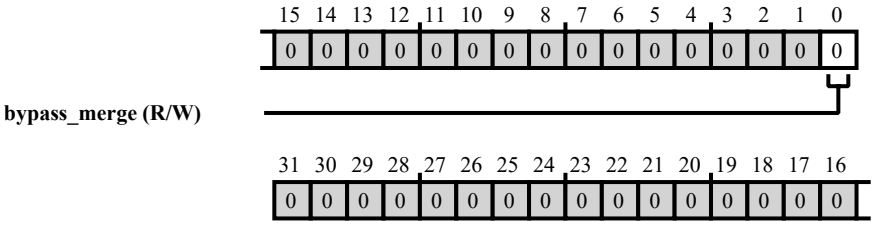


Figure 45-145: SCB0_MSMDMA_CH1_IB_FN_MOD2 Register Diagram

Table 45-149: SCB0_MSMDMA_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

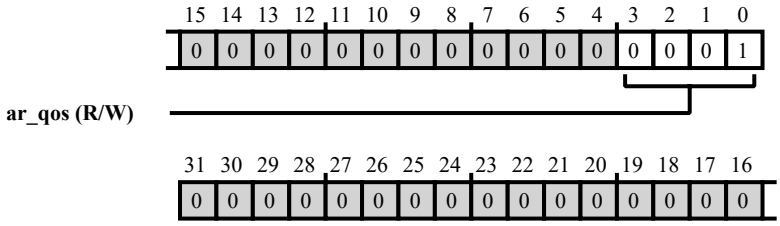


Figure 45-146: SCB0_MSMDMA_CH1_IB_READ_QOS Register Diagram

Table 45-150: SCB0_MSMDMA_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

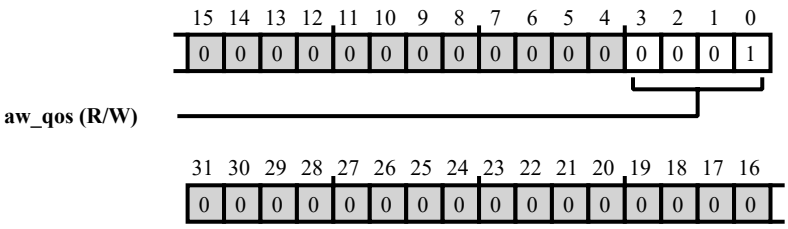


Figure 45-147: SCB0_MSMDMA_CH1_IB_WRITE_QOS Register Diagram

Table 45-151: SCB0_MSMDMA_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

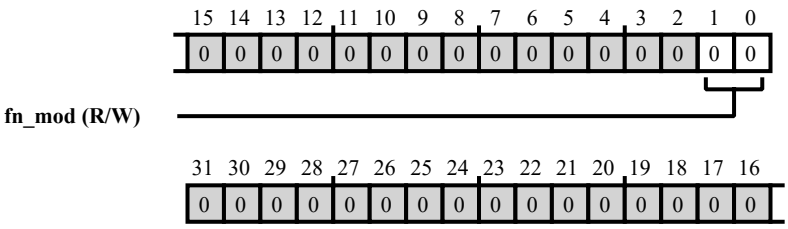


Figure 45-148: SCB0_OTP_IB_FN_MOD Register Diagram

Table 45-152: SCB0_OTP_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

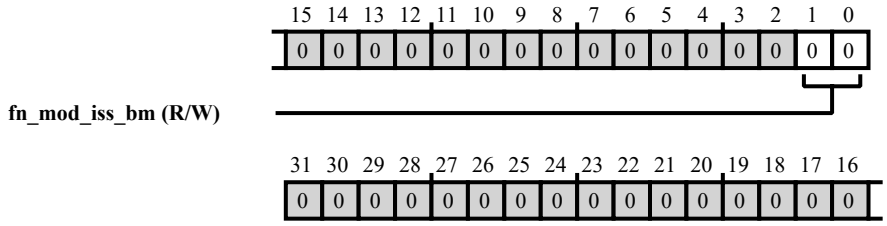


Figure 45-149: SCB0_OTP_IB_FN_MOD_ISS_BM Register Diagram

Table 45-153: SCB0_OTP_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

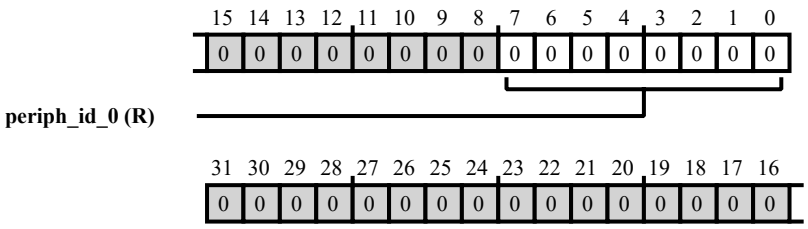


Figure 45-150: SCB0_PERIPH_ID_0 Register Diagram

Table 45-154: SCB0_PERIPH_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_0	.

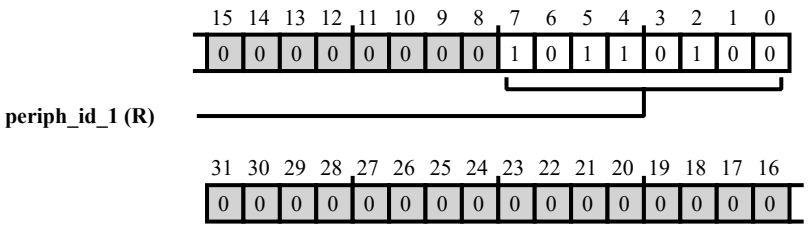


Figure 45-151: SCB0_PERIPH_ID_1 Register Diagram

Table 45-155: SCB0_PERIPH_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_1	.

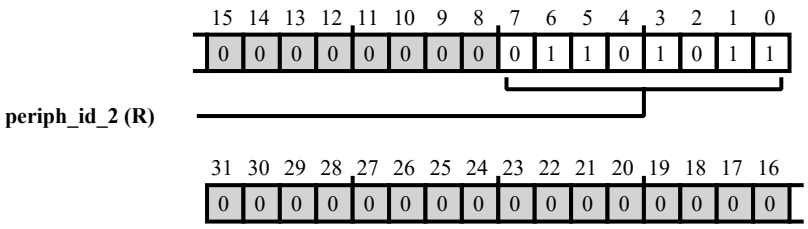


Figure 45-152: SCB0_PERIPH_ID_2 Register Diagram

Table 45-156: SCB0_PERIPH_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_2	.

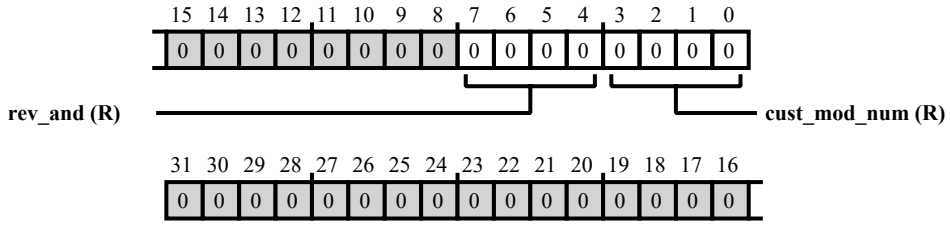


Figure 45-153: SCB0_PERIPH_ID_3 Register Diagram

Table 45-157: SCB0_PERIPH_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV_AND	.
3:0 (R/NW)	CUST_MOD_NUM	.

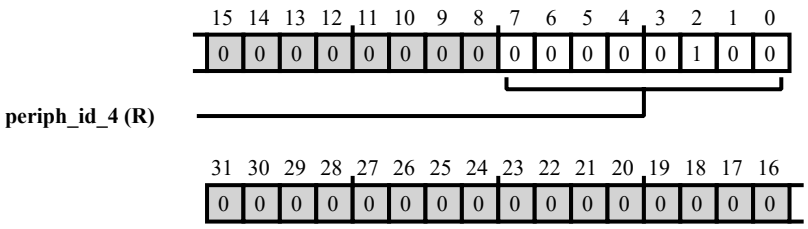


Figure 45-154: SCB0_PERIPH_ID_4 Register Diagram

Table 45-158: SCB0_PERIPH_ID_4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_4	.

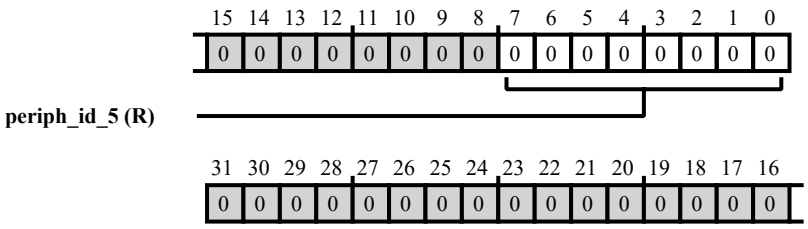


Figure 45-155: SCB0_PERIPH_ID_5 Register Diagram

Table 45-159: SCB0_PERIPH_ID_5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_5	.

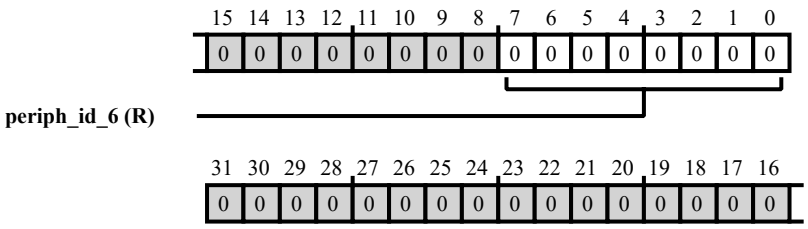


Figure 45-156: SCB0_PERIPH_ID_6 Register Diagram

Table 45-160: SCB0_PERIPH_ID_6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_6	.

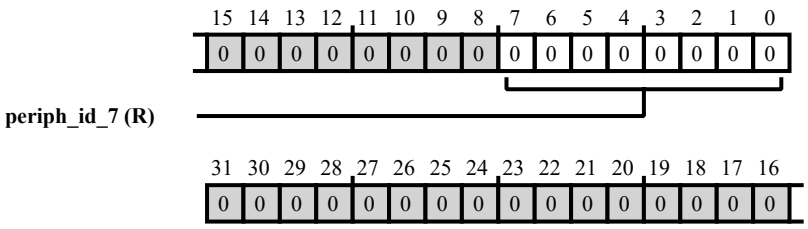


Figure 45-157: SCB0_PERIPH_ID_7 Register Diagram

Table 45-161: SCB0_PERIPH_ID_7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_7	.

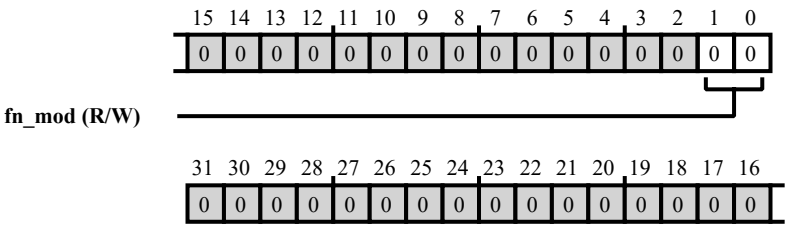


Figure 45-158: SCB0_SCB6_IB_FN_MOD Register Diagram

Table 45-162: SCB0_SCB6_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

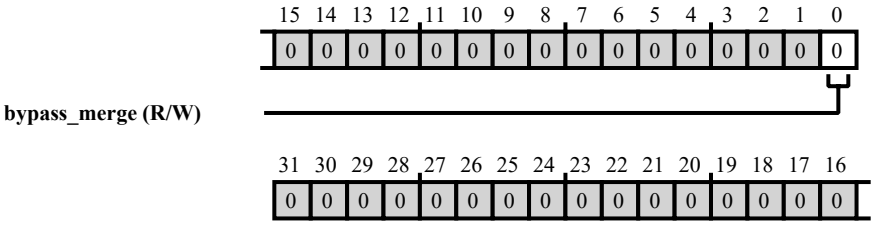


Figure 45-159: SCB0_SCB6_IB_FN_MOD2 Register Diagram

Table 45-163: SCB0_SCB6_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

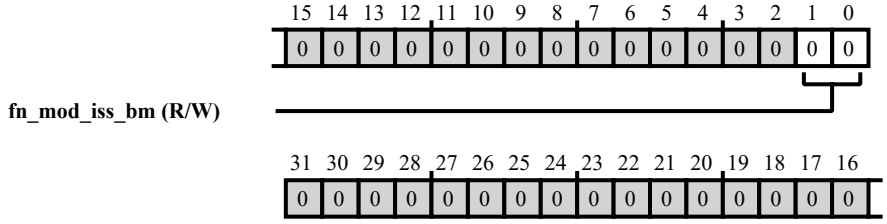


Figure 45-160: SCB0_SCB6_IB_FN_MOD_ISS_BM Register Diagram

Table 45-164: SCB0_SCB6_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

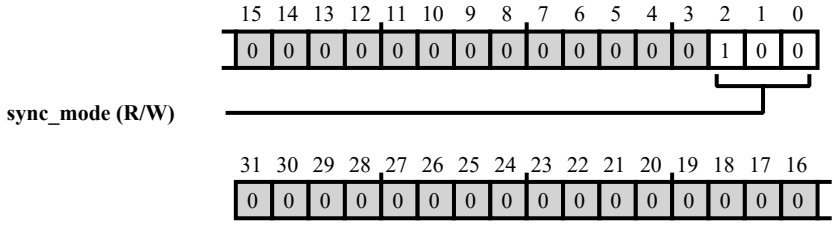


Figure 45-161: SCB0_SCB6_IB_SYNC_MODE Register Diagram

Table 45-165: SCB0_SCB6_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

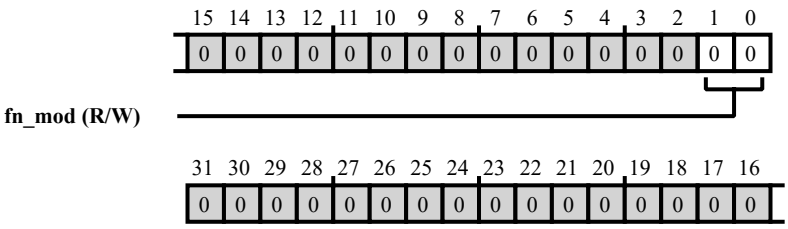


Figure 45-162: SCB0_SH0_DPORT_L2_FN_MOD Register Diagram

Table 45-166: SCB0_SH0_DPORT_L2_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

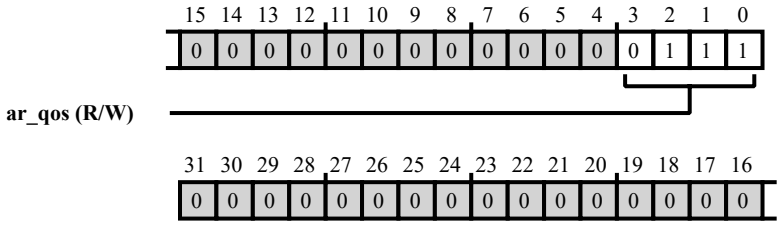


Figure 45-163: SCB0_SH0_DPORT_L2_READ_QOS Register Diagram

Table 45-167: SCB0_SH0_DPORT_L2_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

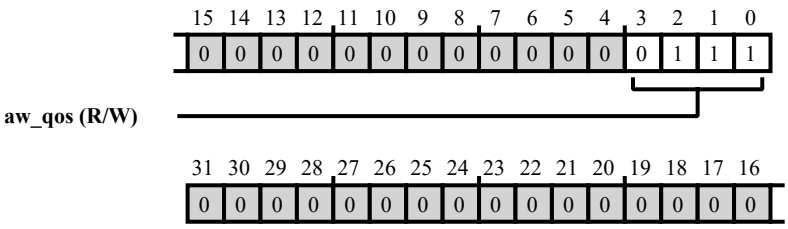


Figure 45-164: SCB0_SH0_DPORT_L2_WRITE_QOS Register Diagram

Table 45-168: SCB0_SH0_DPORT_L2_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

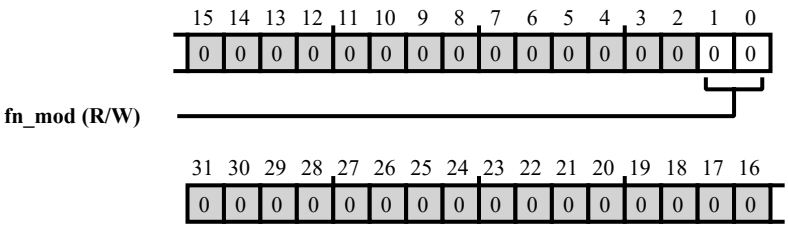


Figure 45-165: SCB0_SH0_DPORT_FN_MOD Register Diagram

Table 45-169: SCB0_SH0_DPORT_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

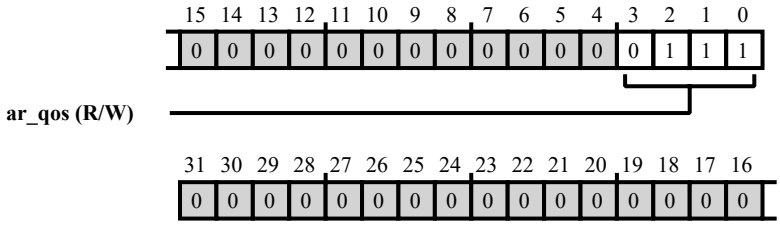


Figure 45-166: SCB0_SH0_DPORT_READ_QOS Register Diagram

Table 45-170: SCB0_SH0_DPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

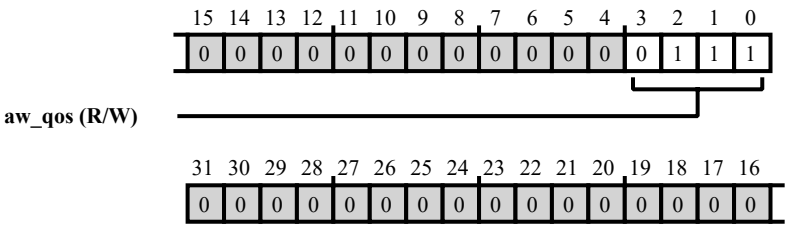


Figure 45-167: SCB0_SH0_DPORT_WRITE_QOS Register Diagram

Table 45-171: SCB0_SH0_DPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

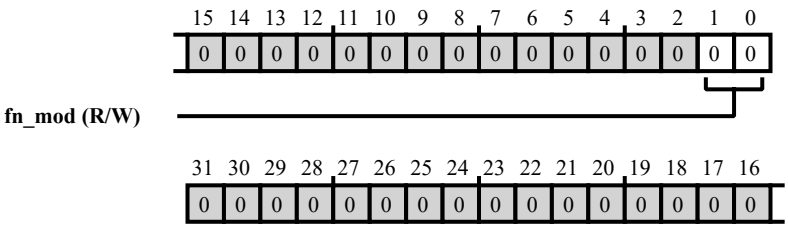


Figure 45-168: SCB0_SH0_FIR_CH0_FN_MOD Register Diagram

Table 45-172: SCB0_SH0_FIR_CH0_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

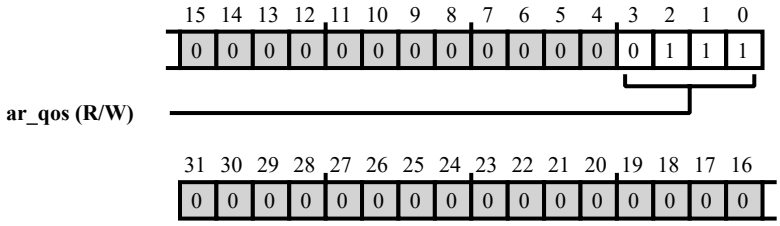


Figure 45-169: SCB0_SH0_FIR_CH0_READ_QOS Register Diagram

Table 45-173: SCB0_SH0_FIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

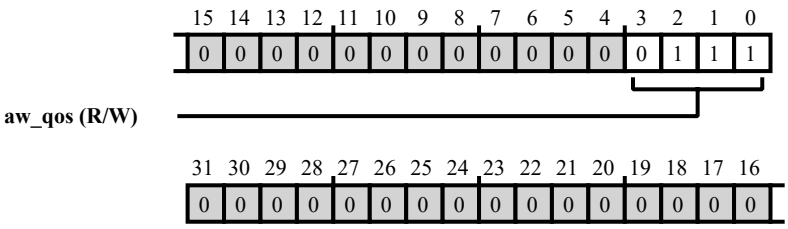


Figure 45-170: SCB0_SH0_FIR_CH0_WRITE_QOS Register Diagram

Table 45-174: SCB0_SH0_FIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

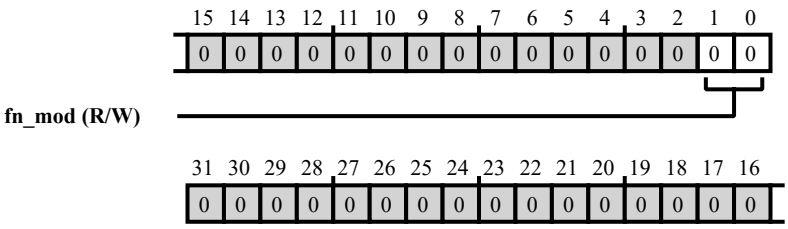


Figure 45-171: SCB0_SH0_FIR_CH1_FN_MOD Register Diagram

Table 45-175: SCB0_SH0_FIR_CH1_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

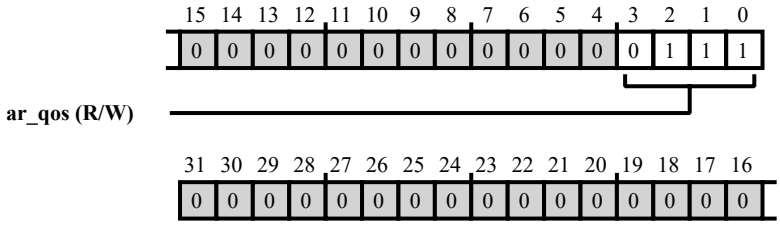


Figure 45-172: SCB0_SH0_FIR_CH1_READ_QOS Register Diagram

Table 45-176: SCB0_SH0_FIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

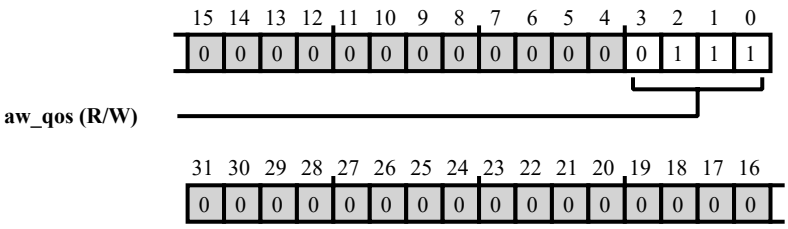


Figure 45-173: SCB0_SH0_FIR_CH1_WRITE_QOS Register Diagram

Table 45-177: SCB0_SH0_FIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

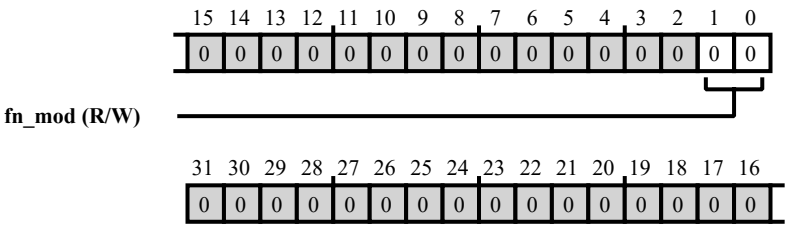


Figure 45-174: SCB0_SH0_IIR_CH0_FN_MOD Register Diagram

Table 45-178: SCB0_SH0_IIR_CH0_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

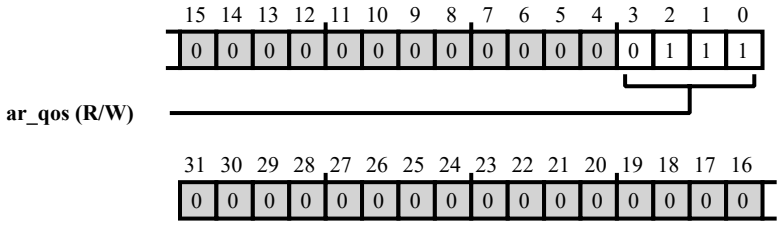


Figure 45-175: SCB0_SH0_IIR_CH0_READ_QOS Register Diagram

Table 45-179: SCB0_SH0_IIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

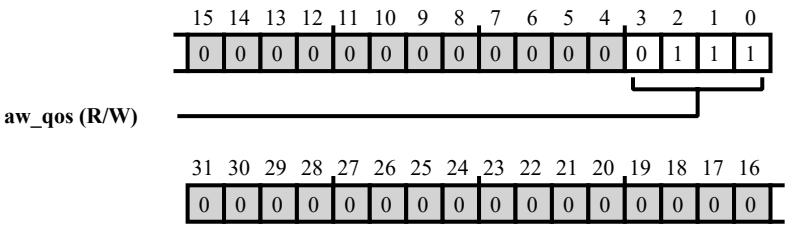


Figure 45-176: SCB0_SH0_IIR_CH0_WRITE_QOS Register Diagram

Table 45-180: SCB0_SH0_IIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

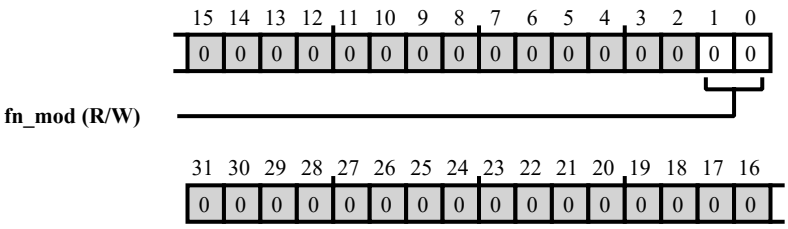


Figure 45-177: SCB0_SH0_IIR_CH1_FN_MOD Register Diagram

Table 45-181: SCB0_SH0_IIR_CH1_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

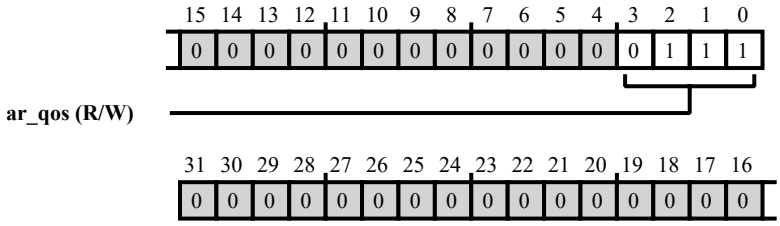


Figure 45-178: SCB0_SH0_IIR_CH1_READ_QOS Register Diagram

Table 45-182: SCB0_SH0_IIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

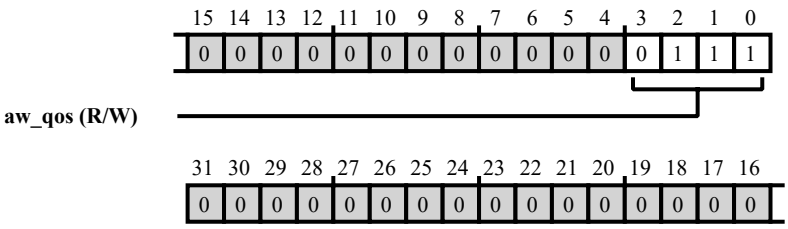


Figure 45-179: SCB0_SH0_IIR_CH1_WRITE_QOS Register Diagram

Table 45-183: SCB0_SH0_IIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

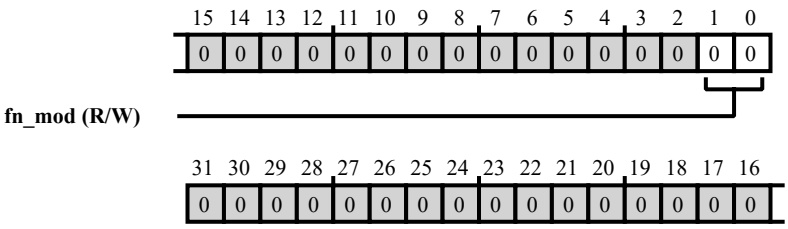


Figure 45-180: SCB0_SH0_IPORT_FN_MOD Register Diagram

Table 45-184: SCB0_SH0_IPORT_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

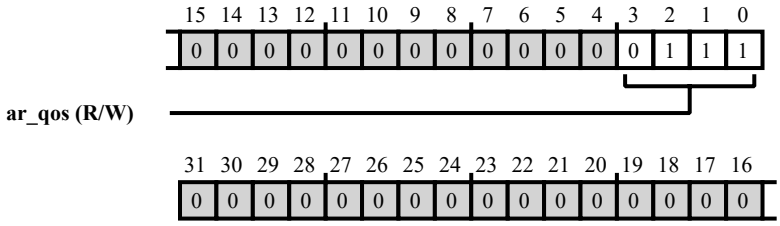


Figure 45-181: SCB0_SH0_IPORT_READ_QOS Register Diagram

Table 45-185: SCB0_SH0_IPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

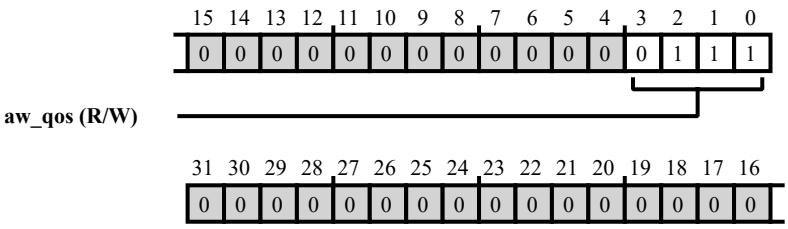


Figure 45-182: SCB0_SH0_IPORT_WRITE_QOS Register Diagram

Table 45-186: SCB0_SH0_IPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

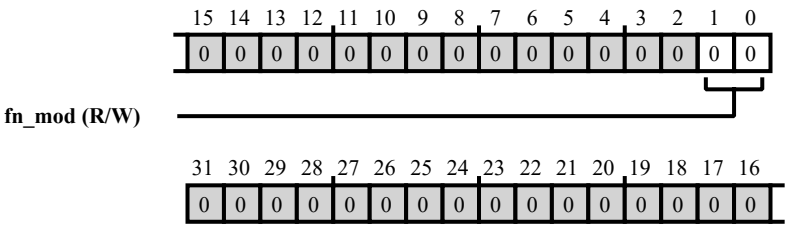


Figure 45-183: SCB0_SH0_MMR_IB_FN_MOD Register Diagram

Table 45-187: SCB0_SH0_MMR_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

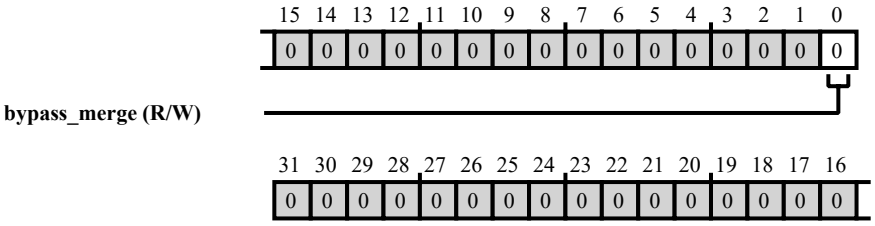


Figure 45-184: SCB0_SH0_MMR_IB_FN_MOD2 Register Diagram

Table 45-188: SCB0_SH0_MMR_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

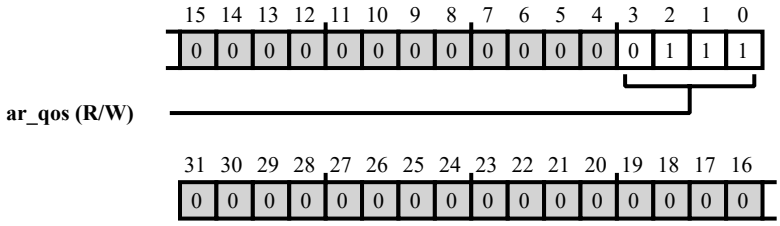


Figure 45-185: SCB0_SH0_MMR_IB_READ_QOS Register Diagram

Table 45-189: SCB0_SH0_MMR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

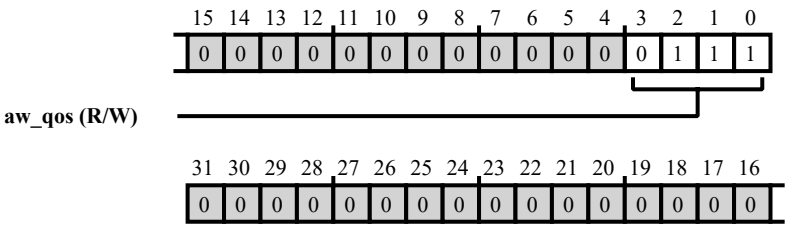


Figure 45-186: SCB0_SH0_MMR_IB_WRITE_QOS Register Diagram

Table 45-190: SCB0_SH0_MMR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

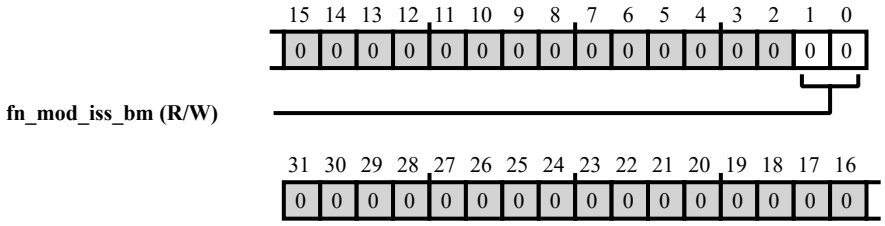


Figure 45-187: SCB0_SH0_S1PORT_FN_MOD_ISS_BM Register Diagram

Table 45-191: SCB0_SH0_S1PORT_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

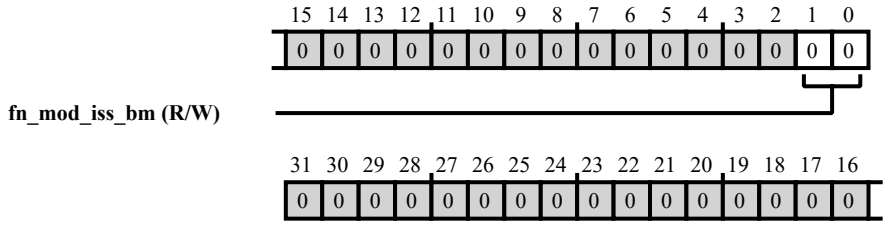


Figure 45-188: SCB0_SH0_S2PORT_FN_MOD_ISS_BM Register Diagram

Table 45-192: SCB0_SH0_S2PORT_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

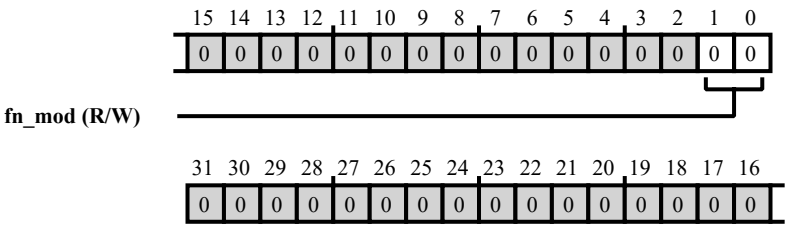


Figure 45-189: SCB0_SMC_IB_FN_MOD Register Diagram

Table 45-193: SCB0_SMC_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

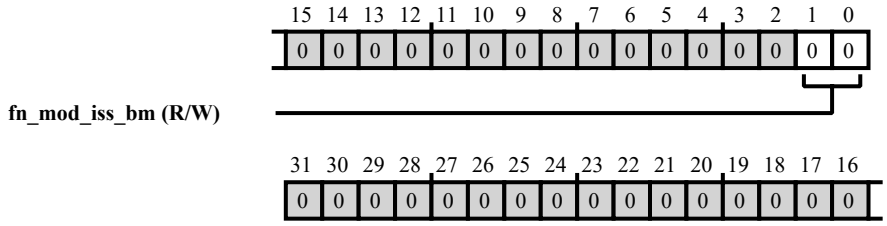


Figure 45-190: SCB0_SMC_IB_FN_MOD_ISS_BM Register Diagram

Table 45-194: SCB0_SMC_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

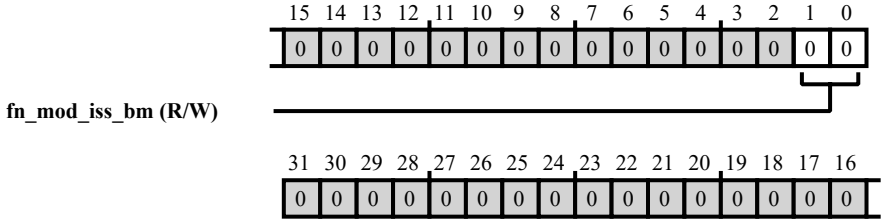


Figure 45-191: SCB0_SMMR_FN_MOD_ISS_BM Register Diagram

Table 45-195: SCB0_SMMR_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

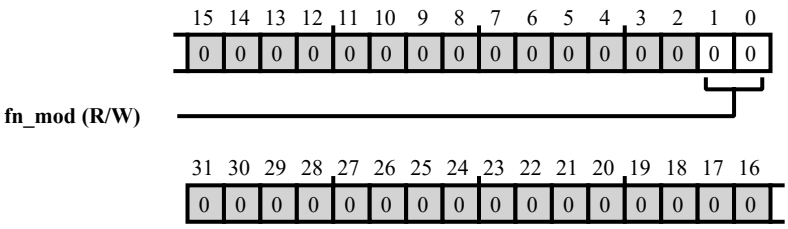


Figure 45-192: SCB0_SP0A_FN_MOD Register Diagram

Table 45-196: SCB0_SP0A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

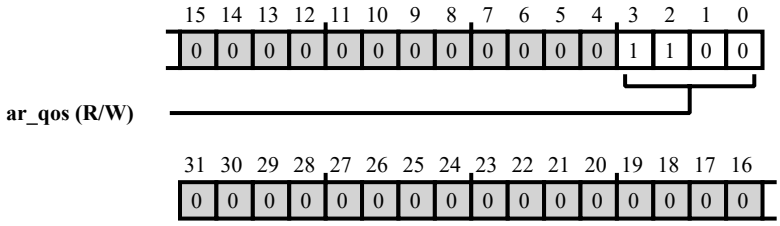


Figure 45-193: SCB0_SP0A_READ_QOS Register Diagram

Table 45-197: SCB0_SP0A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

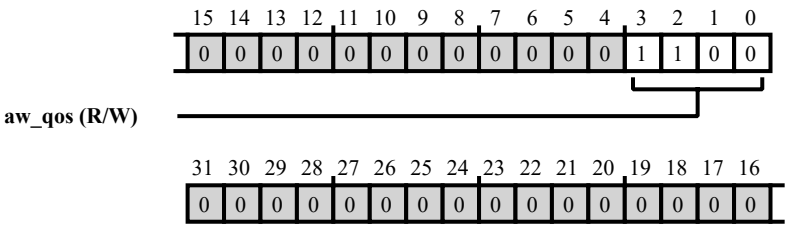


Figure 45-194: SCB0_SP0A_WRITE_QOS Register Diagram

Table 45-198: SCB0_SP0A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

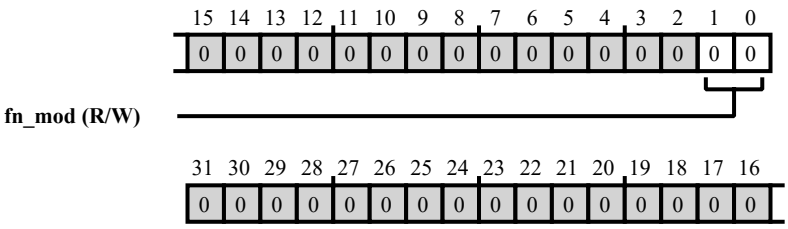


Figure 45-195: SCB0_SP0B_FN_MOD Register Diagram

Table 45-199: SCB0_SP0B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

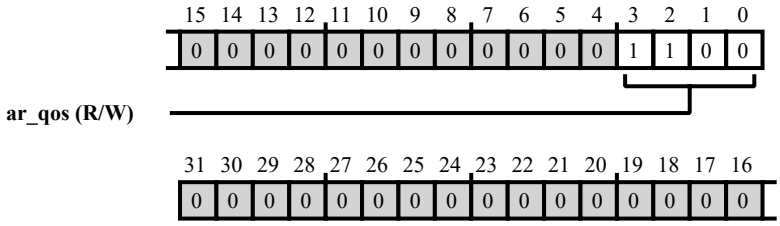


Figure 45-196: SCB0_SP0B_READ_QOS Register Diagram

Table 45-200: SCB0_SP0B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

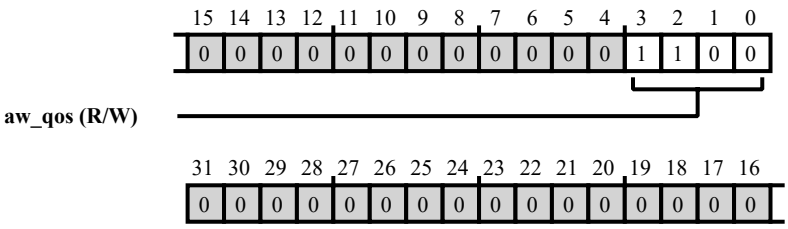


Figure 45-197: SCB0_SP0B_WRITE_QOS Register Diagram

Table 45-201: SCB0_SP0B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

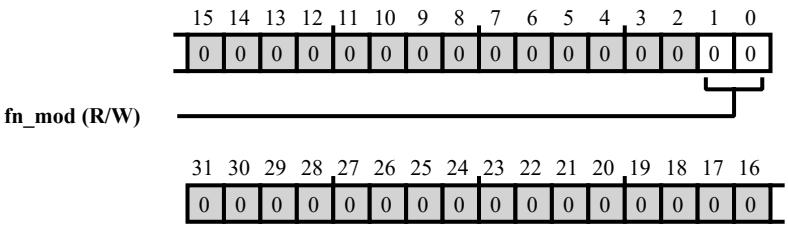


Figure 45-198: SCB0_SP1A_FN_MOD Register Diagram

Table 45-202: SCB0_SP1A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

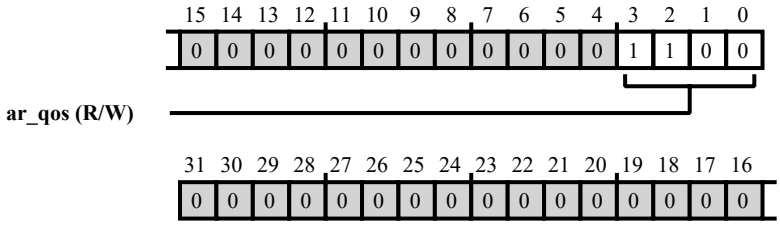


Figure 45-199: SCB0_SP1A_READ_QOS Register Diagram

Table 45-203: SCB0_SP1A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

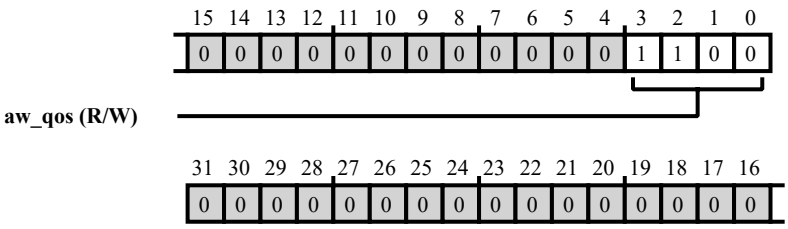


Figure 45-200: SCB0_SP1A_WRITE_QOS Register Diagram

Table 45-204: SCB0_SP1A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

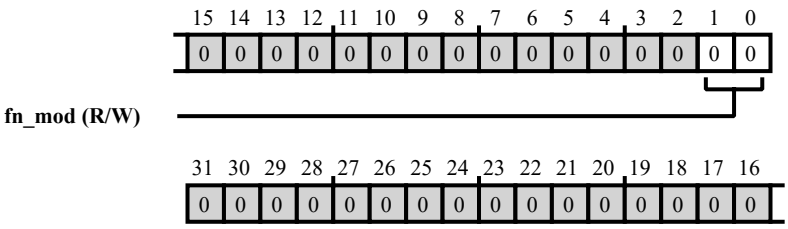


Figure 45-201: SCB0_SP1B_FN_MOD Register Diagram

Table 45-205: SCB0_SP1B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

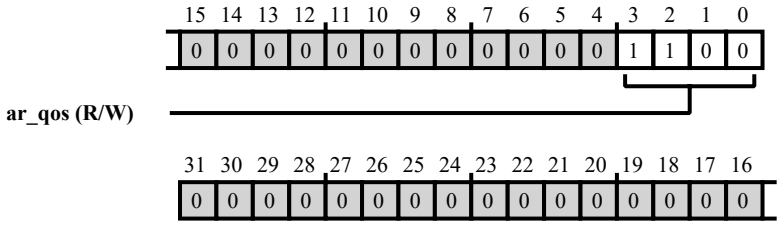


Figure 45-202: SCB0_SP1B_READ_QOS Register Diagram

Table 45-206: SCB0_SP1B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

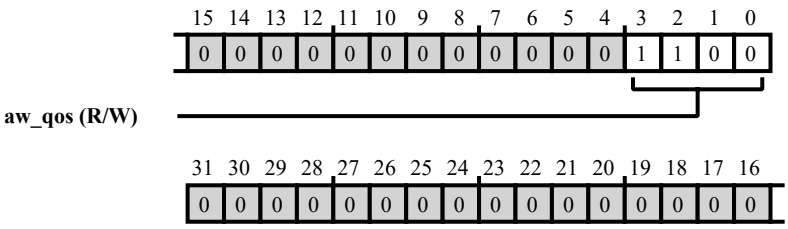


Figure 45-203: SCB0_SP1B_WRITE_QOS Register Diagram

Table 45-207: SCB0_SP1B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

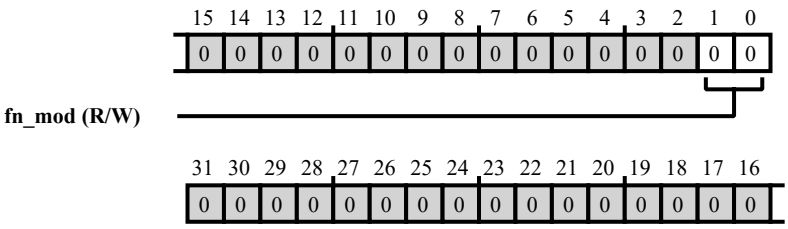


Figure 45-204: SCB0_SP2A_FN_MOD Register Diagram

Table 45-208: SCB0_SP2A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

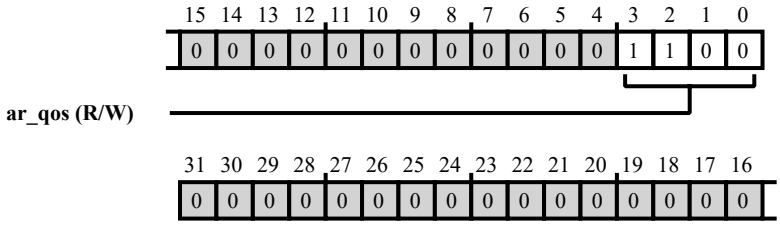


Figure 45-205: SCB0_SP2A_READ_QOS Register Diagram

Table 45-209: SCB0_SP2A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

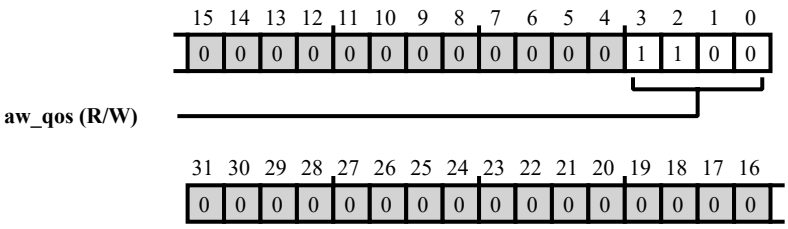


Figure 45-206: SCB0_SP2A_WRITE_QOS Register Diagram

Table 45-210: SCB0_SP2A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

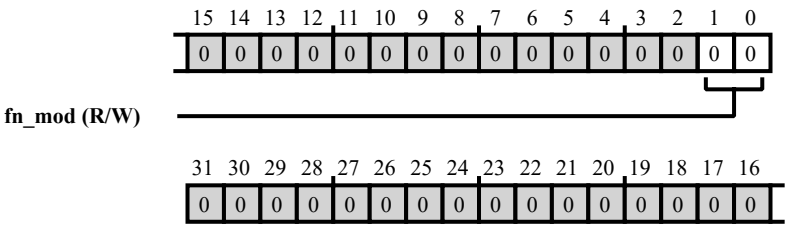


Figure 45-207: SCB0_SP2B_FN_MOD Register Diagram

Table 45-211: SCB0_SP2B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

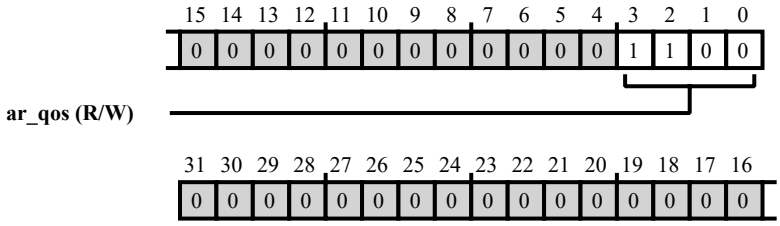


Figure 45-208: SCB0_SP2B_READ_QOS Register Diagram

Table 45-212: SCB0_SP2B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

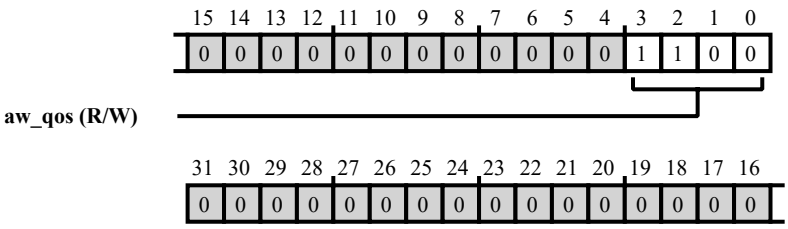


Figure 45-209: SCB0_SP2B_WRITE_QOS Register Diagram

Table 45-213: SCB0_SP2B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

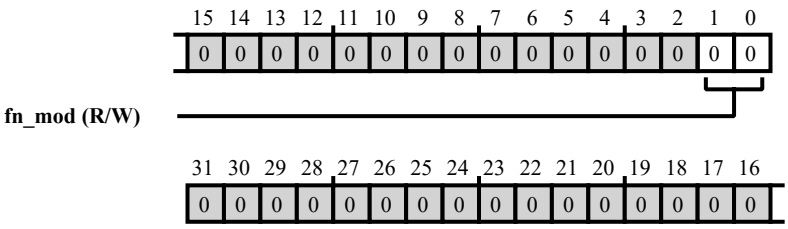


Figure 45-210: SCB0_SP3A_FN_MOD Register Diagram

Table 45-214: SCB0_SP3A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

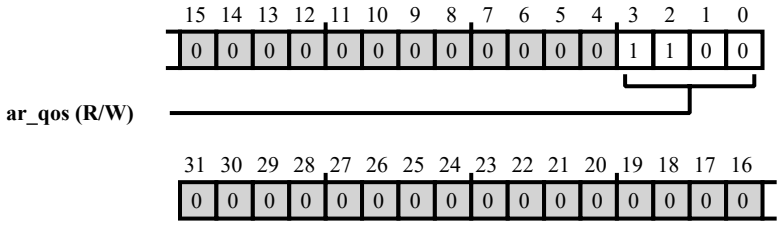


Figure 45-211: SCB0_SP3A_READ_QOS Register Diagram

Table 45-215: SCB0_SP3A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

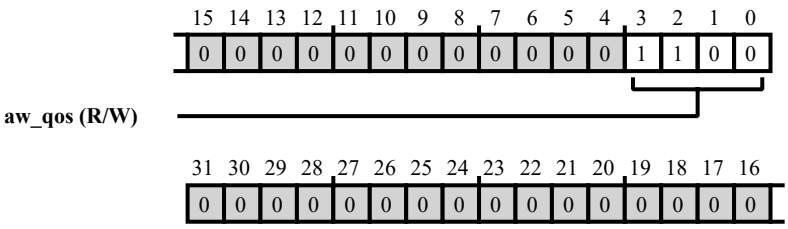


Figure 45-212: SCB0_SP3A_WRITE_QOS Register Diagram

Table 45-216: SCB0_SP3A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

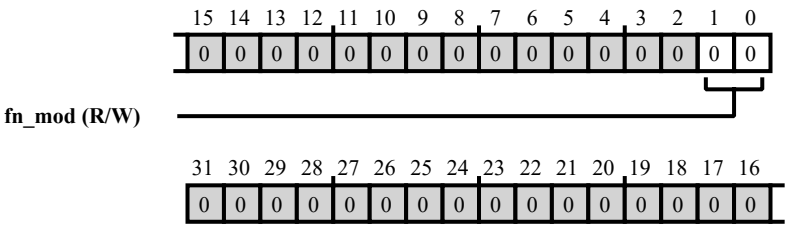


Figure 45-213: SCB0_SP3B_FN_MOD Register Diagram

Table 45-217: SCB0_SP3B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

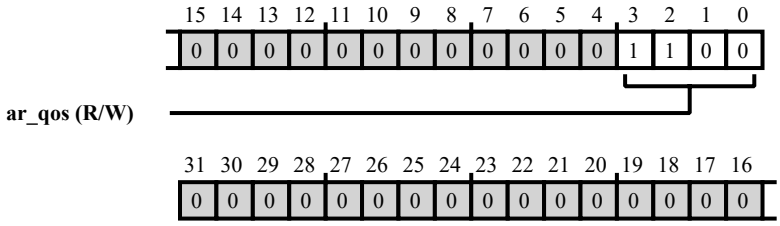


Figure 45-214: SCB0_SP3B_READ_QOS Register Diagram

Table 45-218: SCB0_SP3B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

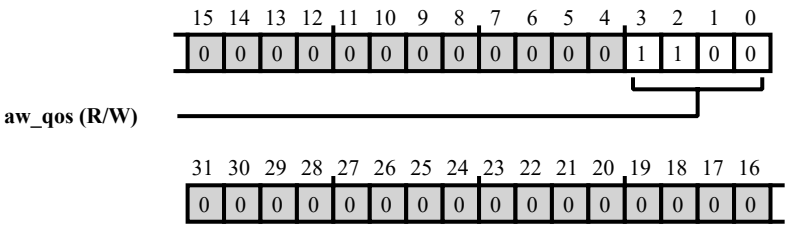


Figure 45-215: SCB0_SP3B_WRITE_QOS Register Diagram

Table 45-219: SCB0_SP3B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

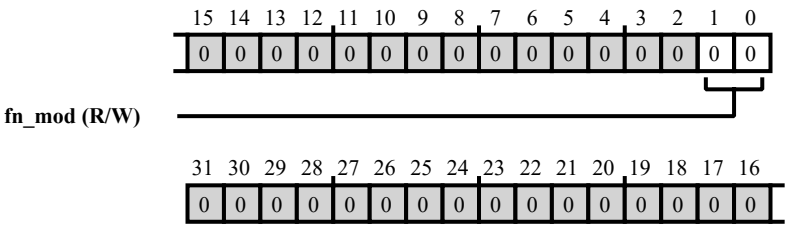


Figure 45-216: SCB0_SP4A_FN_MOD Register Diagram

Table 45-220: SCB0_SP4A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

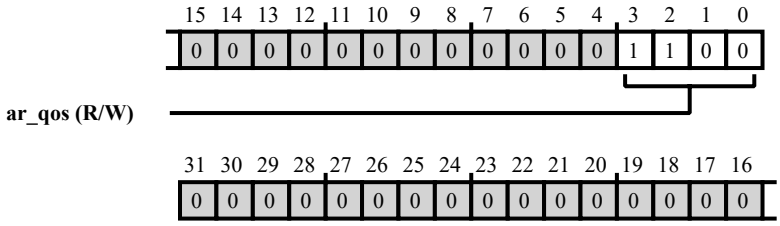


Figure 45-217: SCB0_SP4A_READ_QOS Register Diagram

Table 45-221: SCB0_SP4A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

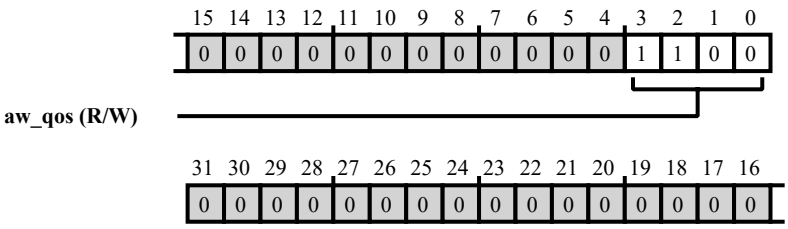


Figure 45-218: SCB0_SP4A_WRITE_QOS Register Diagram

Table 45-222: SCB0_SP4A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

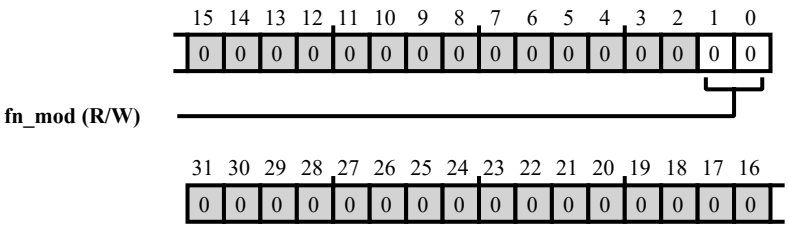


Figure 45-219: SCB0_SP4B_FN_MOD Register Diagram

Table 45-223: SCB0_SP4B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

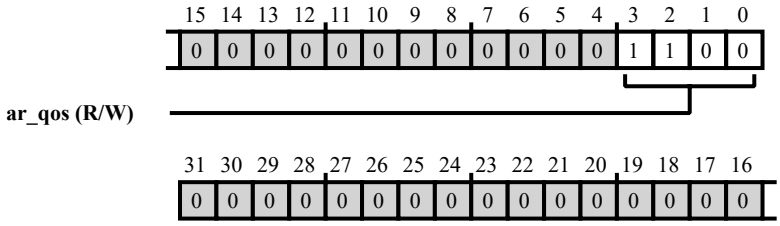


Figure 45-220: SCB0_SP4B_READ_QOS Register Diagram

Table 45-224: SCB0_SP4B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

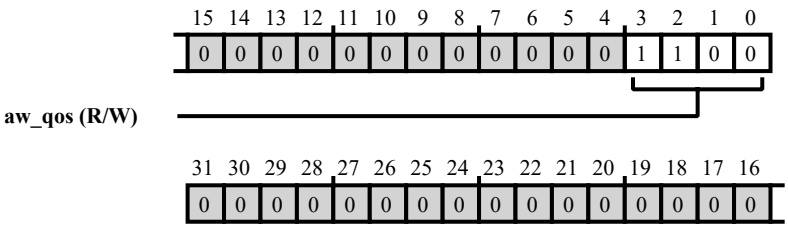


Figure 45-221: SCB0_SP4B_WRITE_QOS Register Diagram

Table 45-225: SCB0_SP4B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

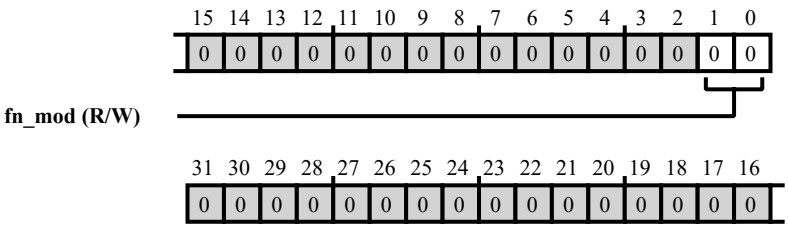


Figure 45-222: SCB0_SP5A_FN_MOD Register Diagram

Table 45-226: SCB0_SP5A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

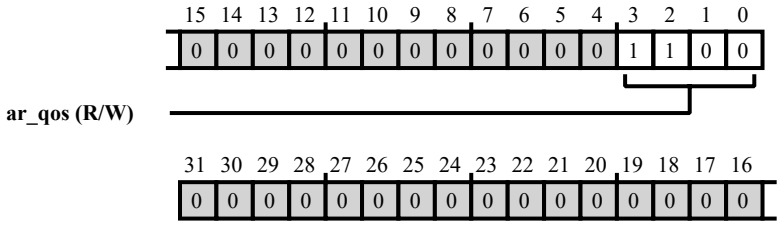


Figure 45-223: SCB0_SP5A_READ_QOS Register Diagram

Table 45-227: SCB0_SP5A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

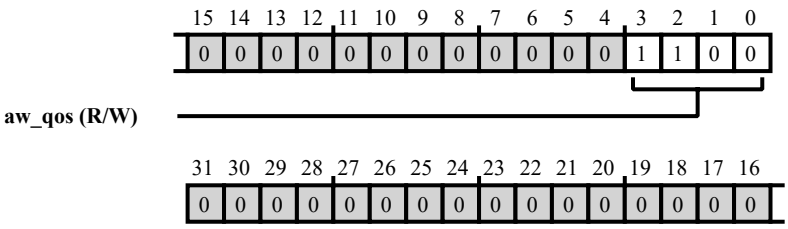


Figure 45-224: SCB0_SP5A_WRITE_QOS Register Diagram

Table 45-228: SCB0_SP5A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

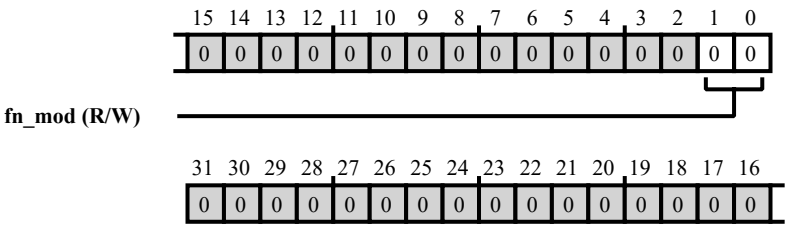


Figure 45-225: SCB0_SP5B_FN_MOD Register Diagram

Table 45-229: SCB0_SP5B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

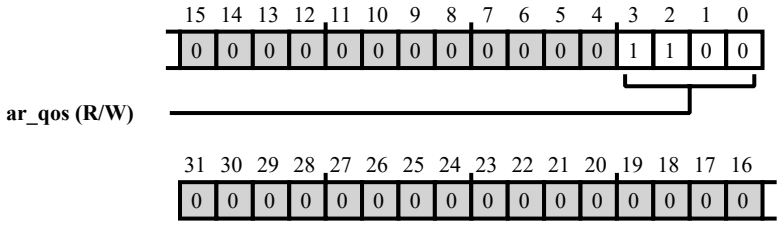


Figure 45-226: SCB0_SP5B_READ_QOS Register Diagram

Table 45-230: SCB0_SP5B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

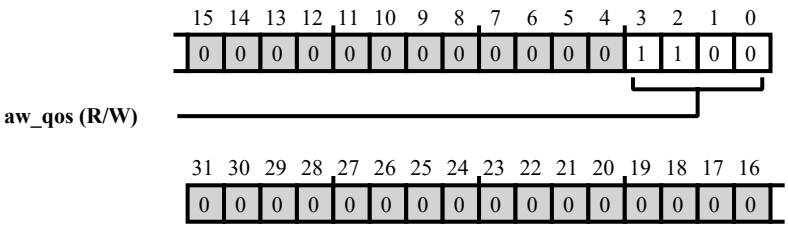


Figure 45-227: SCB0_SP5B_WRITE_QOS Register Diagram

Table 45-231: SCB0_SP5B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

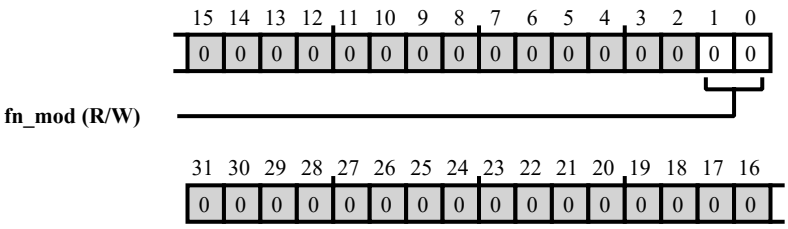


Figure 45-228: SCB0_SP6A_FN_MOD Register Diagram

Table 45-232: SCB0_SP6A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

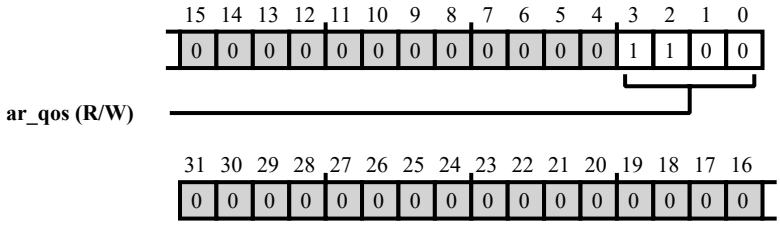


Figure 45-229: SCB0_SP6A_READ_QOS Register Diagram

Table 45-233: SCB0_SP6A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

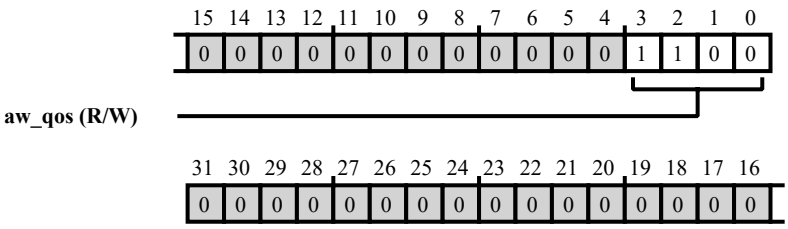


Figure 45-230: SCB0_SP6A_WRITE_QOS Register Diagram

Table 45-234: SCB0_SP6A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

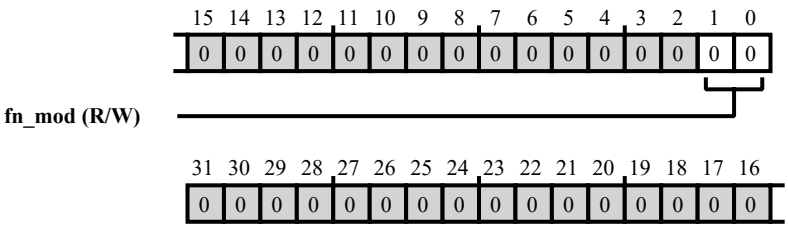


Figure 45-231: SCB0_SP6B_FN_MOD Register Diagram

Table 45-235: SCB0_SP6B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

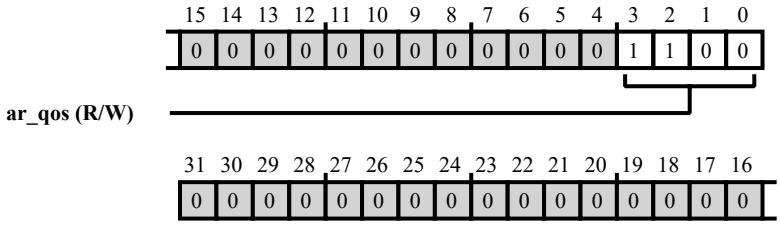


Figure 45-232: SCB0_SP6B_READ_QOS Register Diagram

Table 45-236: SCB0_SP6B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

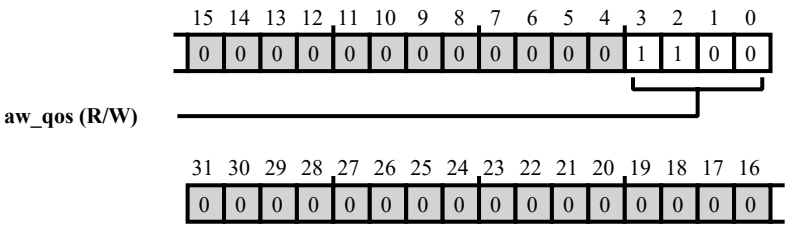


Figure 45-233: SCB0_SP6B_WRITE_QOS Register Diagram

Table 45-237: SCB0_SP6B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

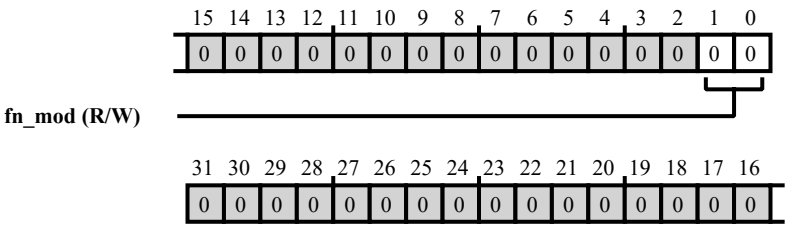


Figure 45-234: SCB0_SP7A_FN_MOD Register Diagram

Table 45-238: SCB0_SP7A_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

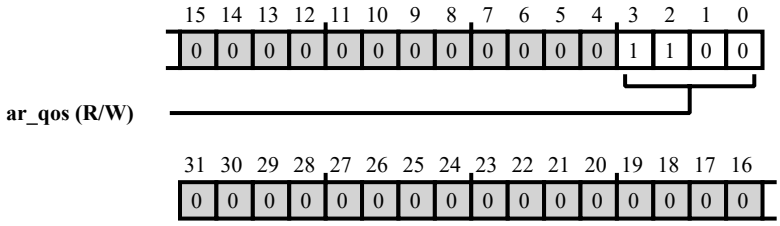


Figure 45-235: SCB0_SP7A_READ_QOS Register Diagram

Table 45-239: SCB0_SP7A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

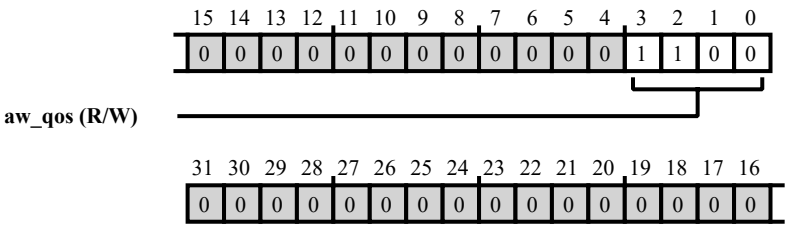


Figure 45-236: SCB0_SP7A_WRITE_QOS Register Diagram

Table 45-240: SCB0_SP7A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

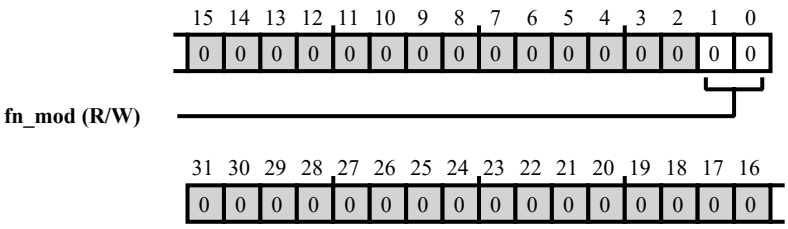


Figure 45-237: SCB0_SP7B_FN_MOD Register Diagram

Table 45-241: SCB0_SP7B_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

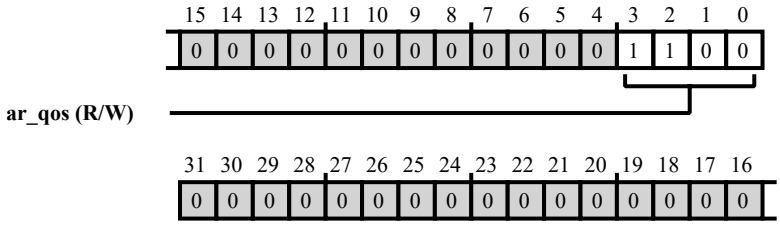


Figure 45-238: SCB0_SP7B_READ_QOS Register Diagram

Table 45-242: SCB0_SP7B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

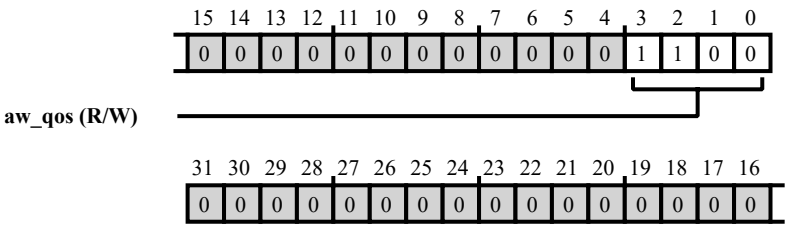


Figure 45-239: SCB0_SP7B_WRITE_QOS Register Diagram

Table 45-243: SCB0_SP7B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

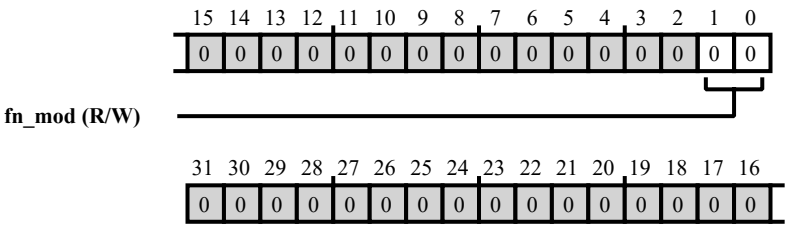


Figure 45-240: SCB0_SPI0RX_FN_MOD Register Diagram

Table 45-244: SCB0_SPI0RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

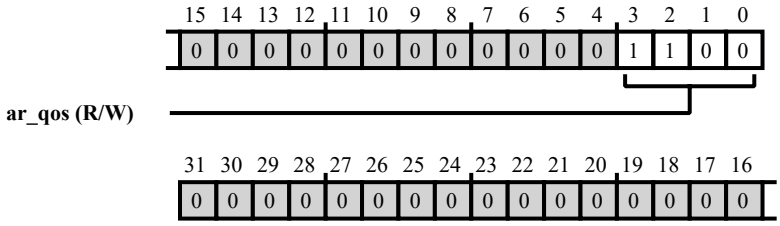


Figure 45-241: SCB0_SPI0RX_READ_QOS Register Diagram

Table 45-245: SCB0_SPI0RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

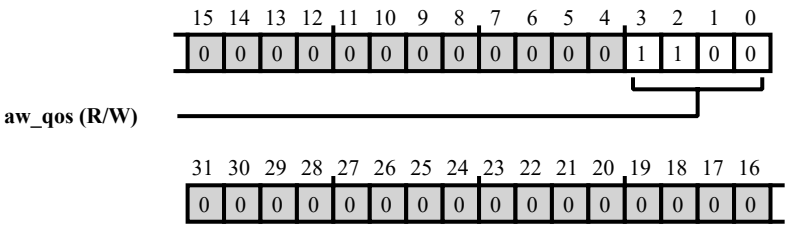


Figure 45-242: SCB0_SPI0RX_WRITE_QOS Register Diagram

Table 45-246: SCB0_SPI0RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

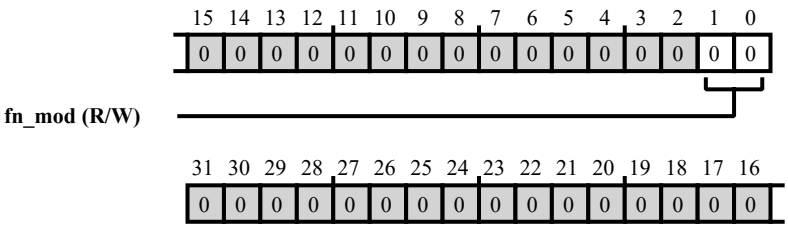


Figure 45-243: SCB0_SPI0TX_FN_MOD Register Diagram

Table 45-247: SCB0_SPI0TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

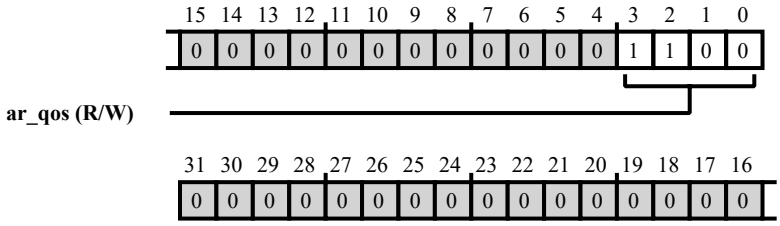


Figure 45-244: SCB0_SPI0TX_READ_QOS Register Diagram

Table 45-248: SCB0_SPI0TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

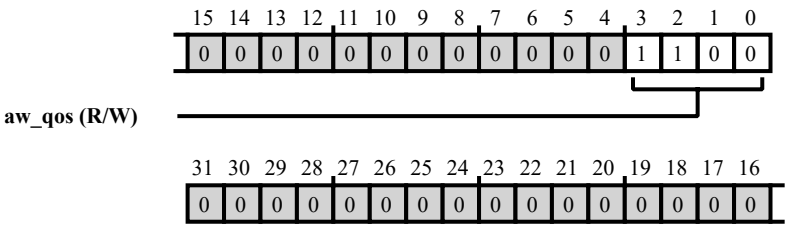


Figure 45-245: SCB0_SPI0TX_WRITE_QOS Register Diagram

Table 45-249: SCB0_SPI0TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

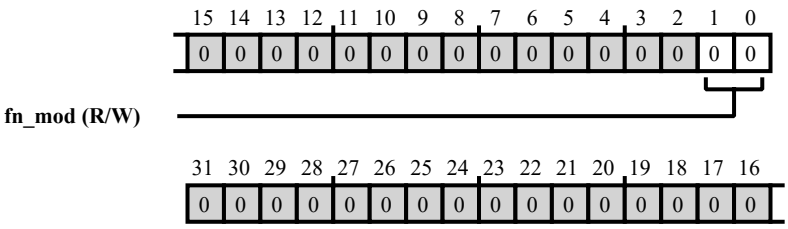


Figure 45-246: SCB0_SPI1RX_FN_MOD Register Diagram

Table 45-250: SCB0_SPI1RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

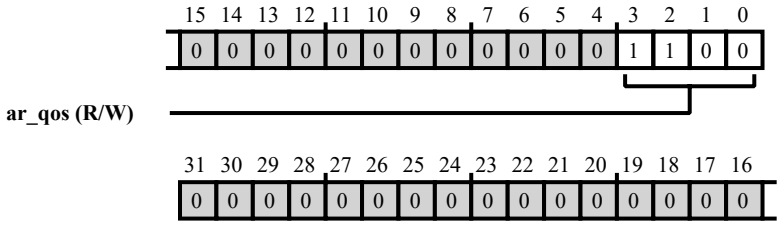


Figure 45-247: SCB0_SPI1RX_READ_QOS Register Diagram

Table 45-251: SCB0_SPI1RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

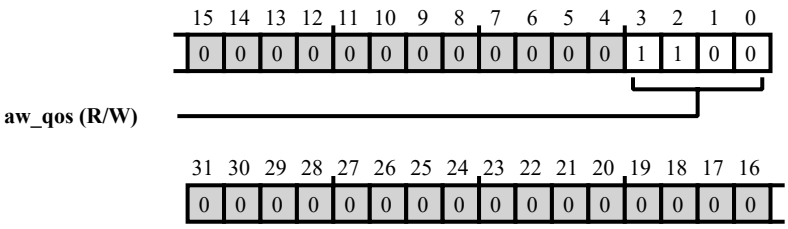


Figure 45-248: SCB0_SPI1RX_WRITE_QOS Register Diagram

Table 45-252: SCB0_SPI1RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

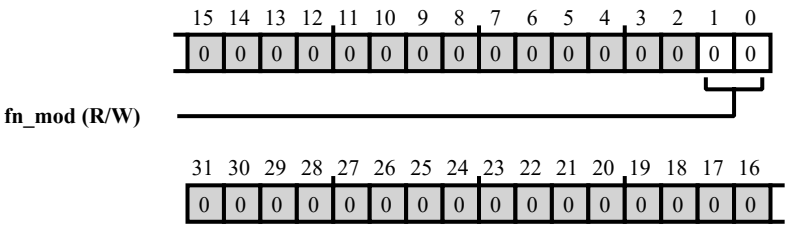


Figure 45-249: SCB0_SPI1TX_FN_MOD Register Diagram

Table 45-253: SCB0_SPI1TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

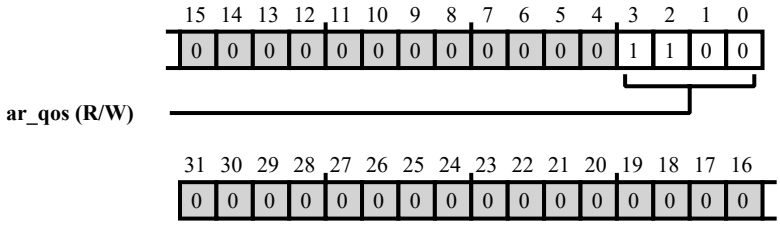


Figure 45-250: SCB0_SPI1TX_READ_QOS Register Diagram

Table 45-254: SCB0_SPI1TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

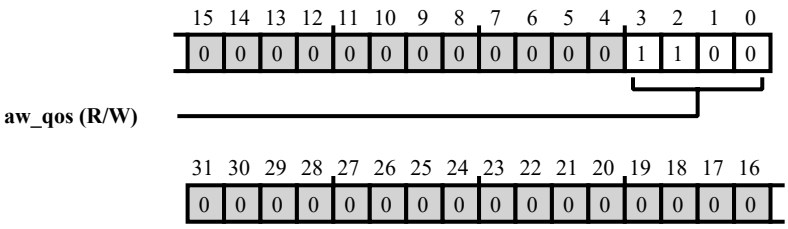


Figure 45-251: SCB0_SPI1TX_WRITE_QOS Register Diagram

Table 45-255: SCB0_SPI1TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

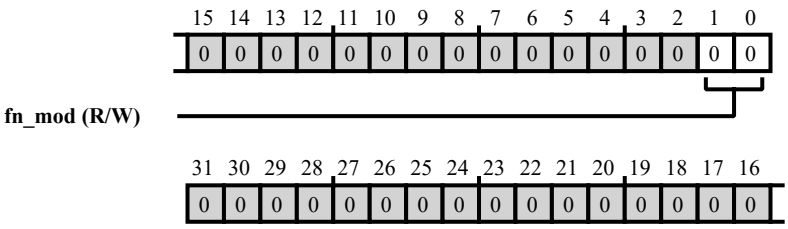


Figure 45-252: SCB0_SPI2RX_FN_MOD Register Diagram

Table 45-256: SCB0_SPI2RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

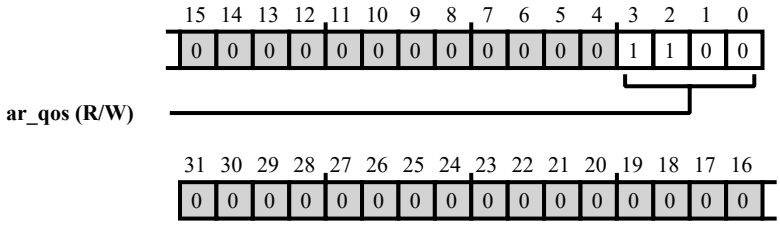


Figure 45-253: SCB0_SPI2RX_READ_QOS Register Diagram

Table 45-257: SCB0_SPI2RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

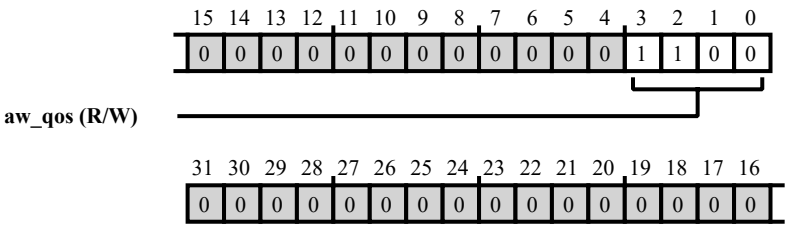


Figure 45-254: SCB0_SPI2RX_WRITE_QOS Register Diagram

Table 45-258: SCB0_SPI2RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

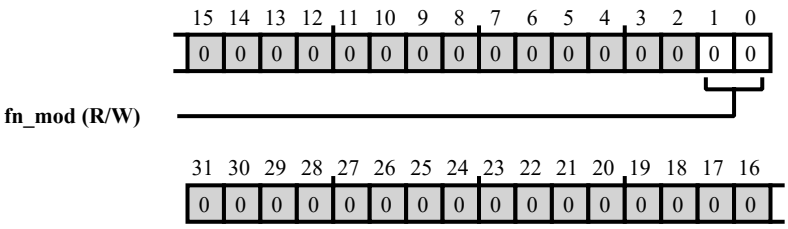


Figure 45-255: SCB0_SPI2TX_FN_MOD Register Diagram

Table 45-259: SCB0_SPI2TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

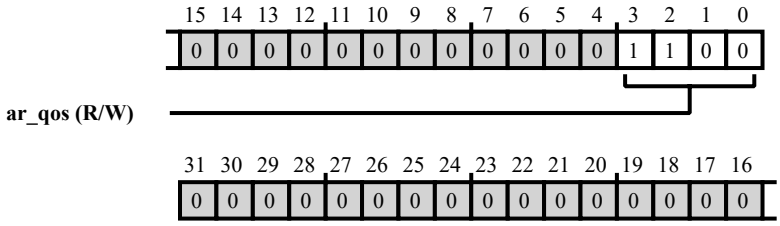


Figure 45-256: SCB0_SPI2TX_READ_QOS Register Diagram

Table 45-260: SCB0_SPI2TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

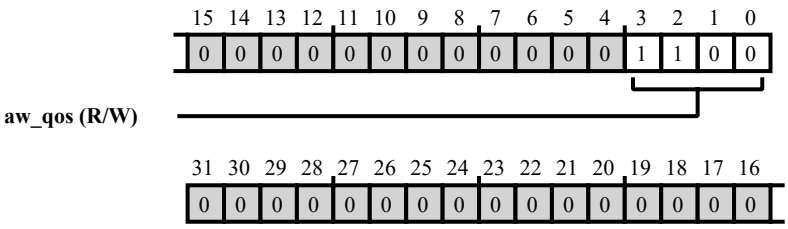


Figure 45-257: SCB0_SPI2TX_WRITE_QOS Register Diagram

Table 45-261: SCB0_SPI2TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

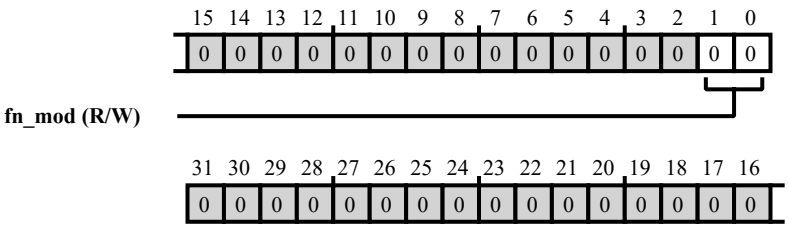


Figure 45-258: SCB0_SPI5RX_FN_MOD Register Diagram

Table 45-262: SCB0_SPI5RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

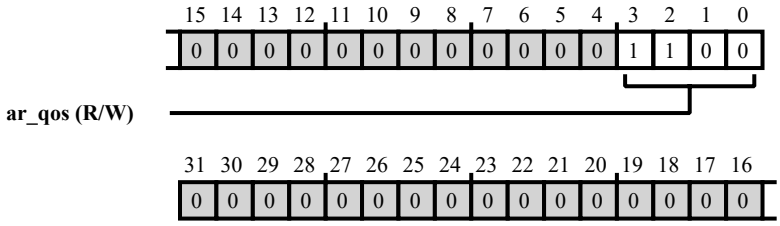


Figure 45-259: SCB0_SPI5RX_READ_QOS Register Diagram

Table 45-263: SCB0_SPI5RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

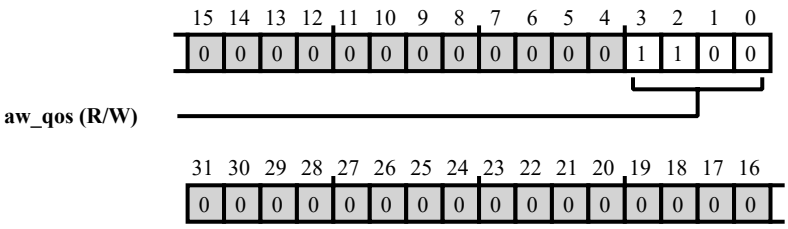


Figure 45-260: SCB0_SPI5RX_WRITE_QOS Register Diagram

Table 45-264: SCB0_SPI5RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

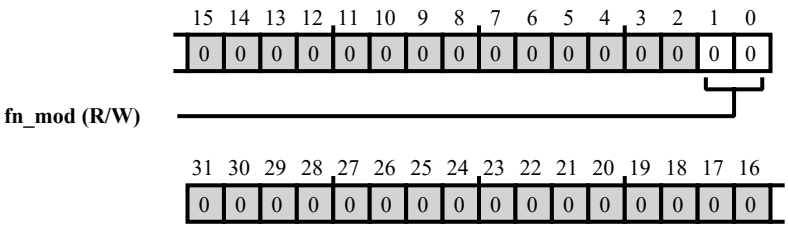


Figure 45-261: SCB0_SPI5TX_FN_MOD Register Diagram

Table 45-265: SCB0_SPI5TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

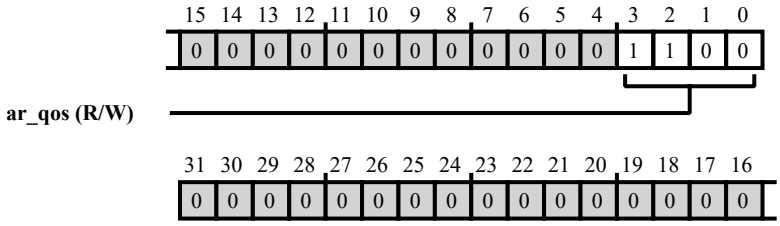


Figure 45-262: SCB0_SPI5TX_READ_QOS Register Diagram

Table 45-266: SCB0_SPI5TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

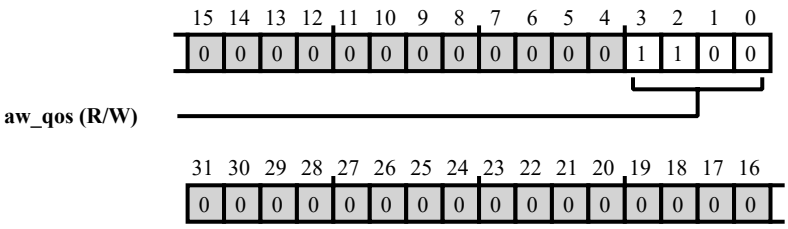


Figure 45-263: SCB0_SPI5TX_WRITE_QOS Register Diagram

Table 45-267: SCB0_SPI5TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

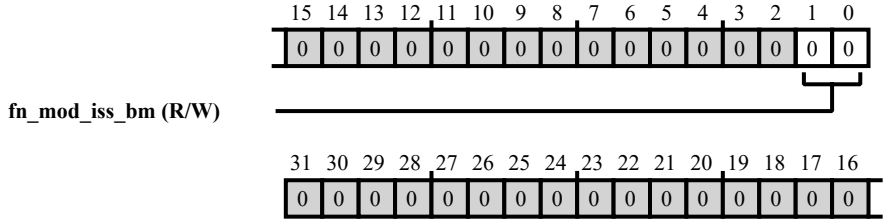


Figure 45-264: SCB0_SPIF_FN_MOD_ISS_BM Register Diagram

Table 45-268: SCB0_SPIF_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

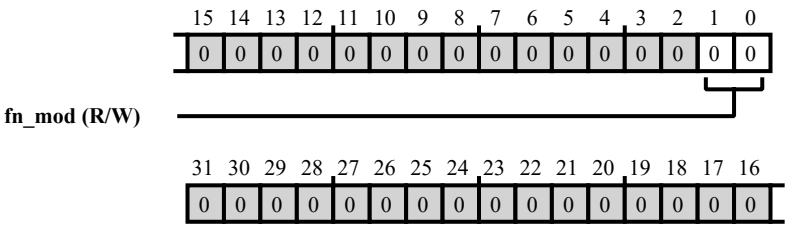


Figure 45-265: SCB0_UART0_RX_FN_MOD Register Diagram

Table 45-269: SCB0_UART0_RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

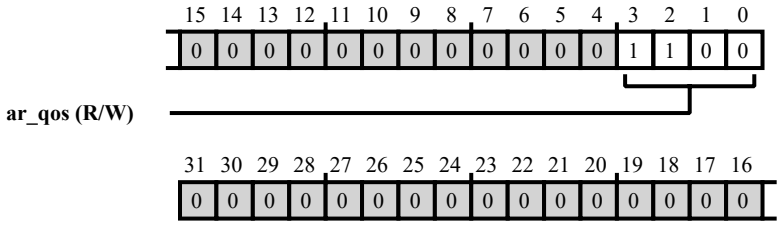


Figure 45-266: SCB0_UART0_RX_READ_QOS Register Diagram

Table 45-270: SCB0_UART0_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

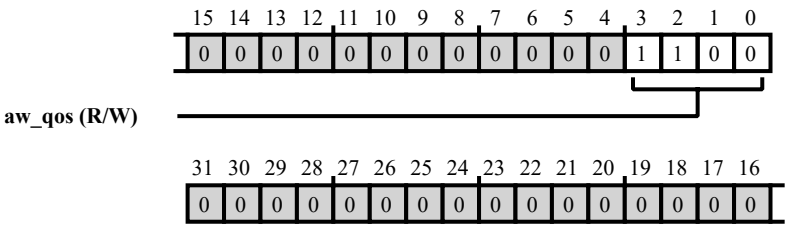


Figure 45-267: SCB0_UART0_RX_WRITE_QOS Register Diagram

Table 45-271: SCB0_UART0_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

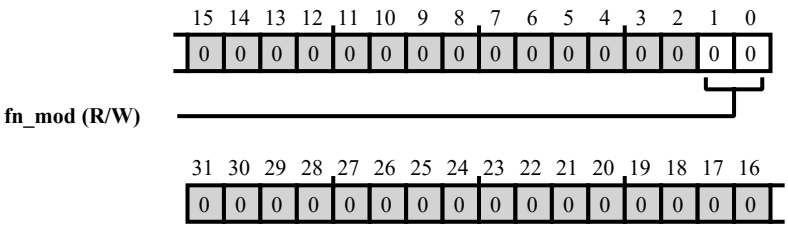


Figure 45-268: SCB0_UART0_TX_FN_MOD Register Diagram

Table 45-272: SCB0_UART0_TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

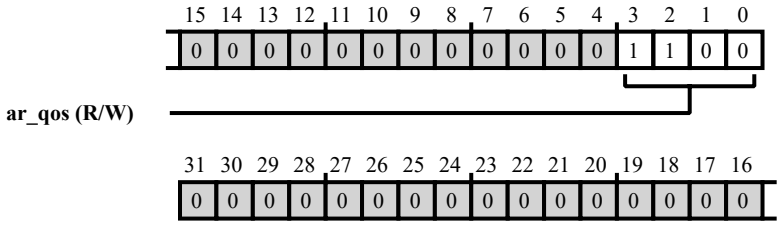


Figure 45-269: SCB0_UART0_TX_READ_QOS Register Diagram

Table 45-273: SCB0_UART0_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

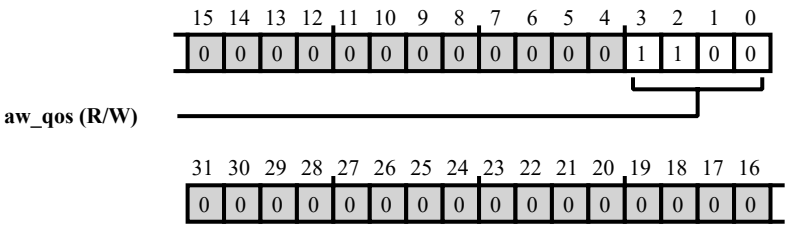


Figure 45-270: SCB0_UART0_TX_WRITE_QOS Register Diagram

Table 45-274: SCB0_UART0_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

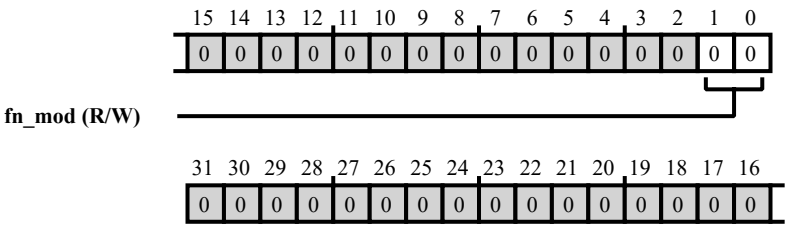


Figure 45-271: SCB0_UART1_RX_FN_MOD Register Diagram

Table 45-275: SCB0_UART1_RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

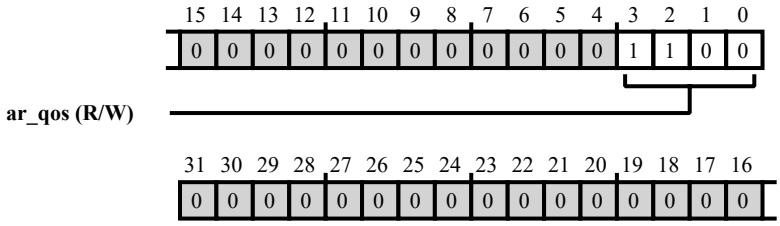


Figure 45-272: SCB0_UART1_RX_READ_QOS Register Diagram

Table 45-276: SCB0_UART1_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

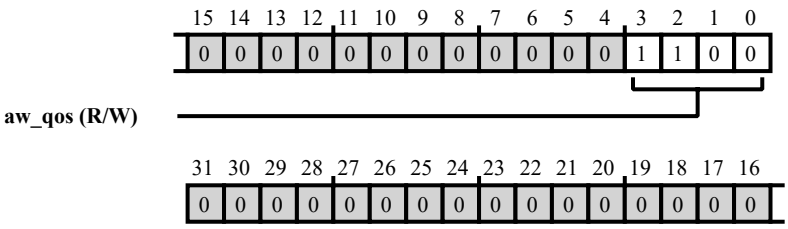


Figure 45-273: SCB0_UART1_RX_WRITE_QOS Register Diagram

Table 45-277: SCB0_UART1_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

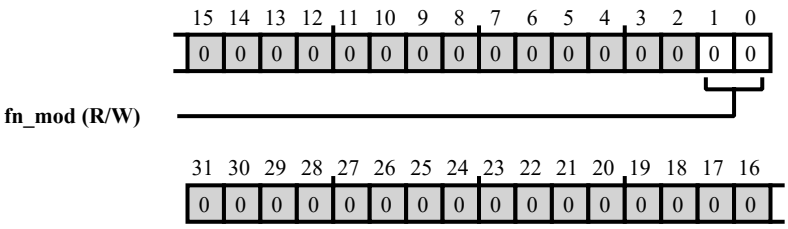


Figure 45-274: SCB0_UART1_TX_FN_MOD Register Diagram

Table 45-278: SCB0_UART1_TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

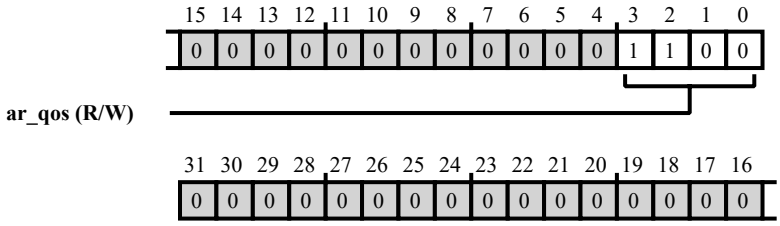


Figure 45-275: SCB0_UART1_TX_READ_QOS Register Diagram

Table 45-279: SCB0_UART1_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

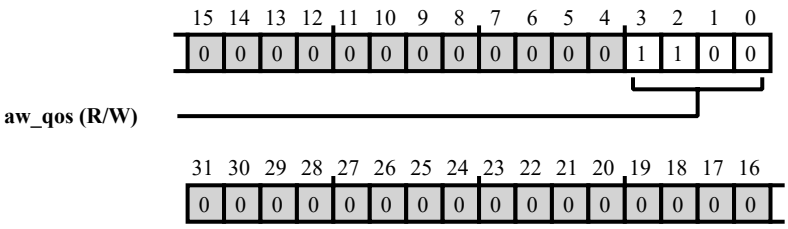


Figure 45-276: SCB0_UART1_TX_WRITE_QOS Register Diagram

Table 45-280: SCB0_UART1_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

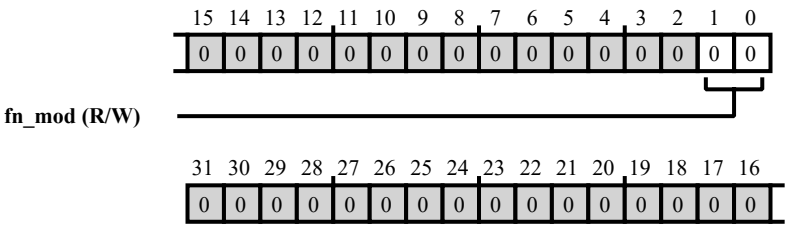


Figure 45-277: SCB0_UART2_RX_FN_MOD Register Diagram

Table 45-281: SCB0_UART2_RX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

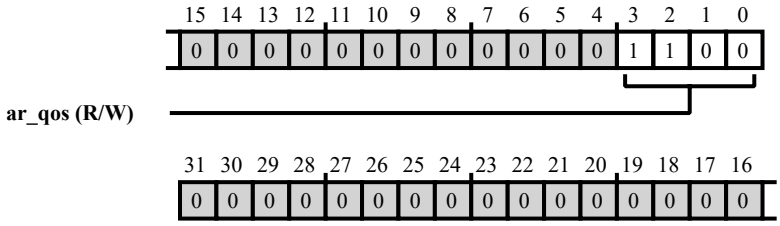


Figure 45-278: SCB0_UART2_RX_READ_QOS Register Diagram

Table 45-282: SCB0_UART2_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

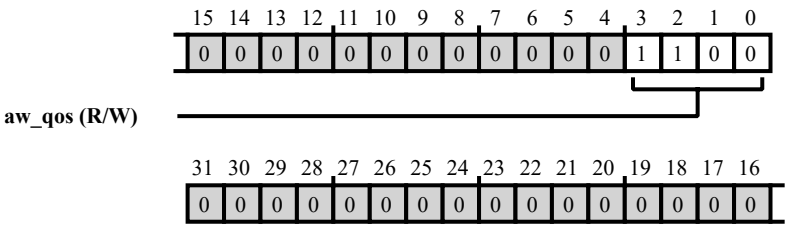


Figure 45-279: SCB0_UART2_RX_WRITE_QOS Register Diagram

Table 45-283: SCB0_UART2_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

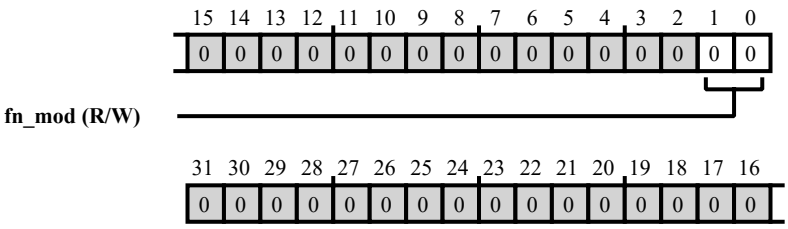


Figure 45-280: SCB0_UART2_TX_FN_MOD Register Diagram

Table 45-284: SCB0_UART2_TX_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

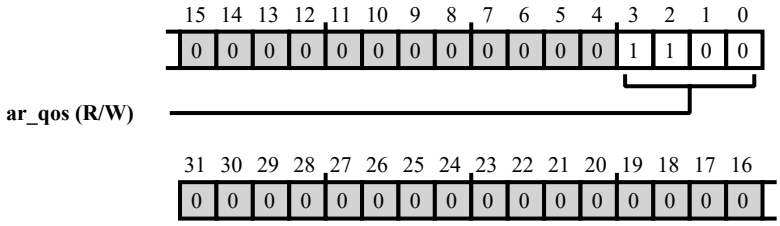


Figure 45-281: SCB0_UART2_TX_READ_QOS Register Diagram

Table 45-285: SCB0_UART2_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

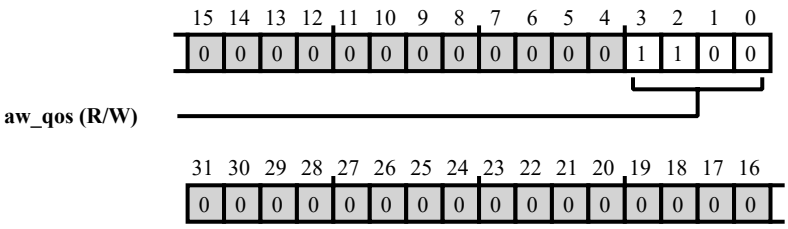


Figure 45-282: SCB0_UART2_TX_WRITE_QOS Register Diagram

Table 45-286: SCB0_UART2_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

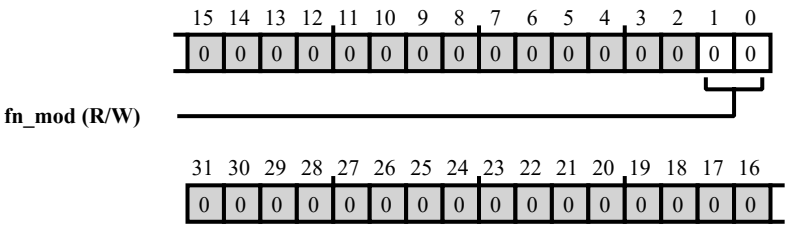


Figure 45-283: SCB0_XSPI_M1_IB_FN_MOD Register Diagram

Table 45-287: SCB0_XSPI_M1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

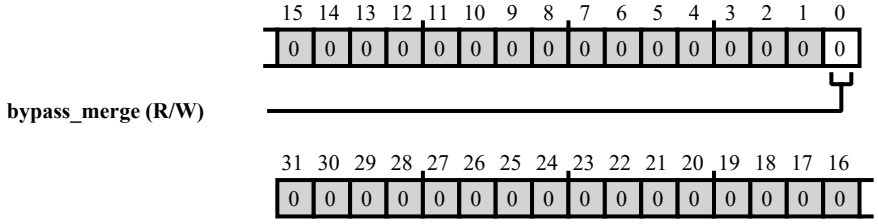


Figure 45-284: SCB0_XSPI_M1_IB_FN_MOD2 Register Diagram

Table 45-288: SCB0_XSPI_M1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

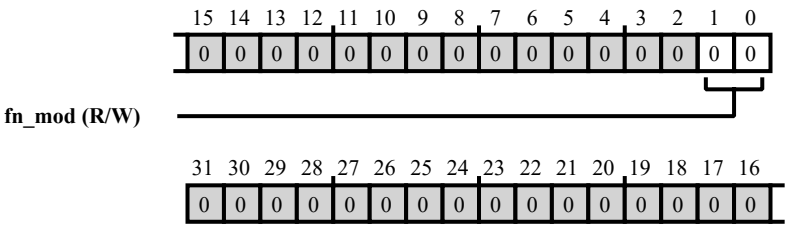


Figure 45-285: SCB0_XSPI_M_IB_FN_MOD Register Diagram

Table 45-289: SCB0_XSPI_M_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

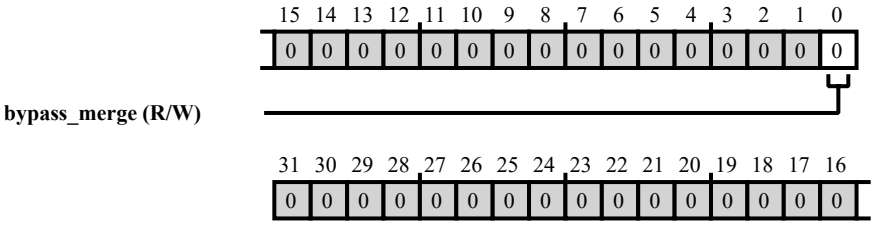


Figure 45-286: SCB0_XSPI_M_IB_FN_MOD2 Register Diagram

Table 45-290: SCB0_XSPI_M_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

ADSP-2184x SCB1 Register Descriptions

Your module description, here. (SCB1) contains the following registers.

Table 45-291: ADSP-2184x SCB1 Register List

Name	Description
SCB1_MST_IB_SYNC_MODE	

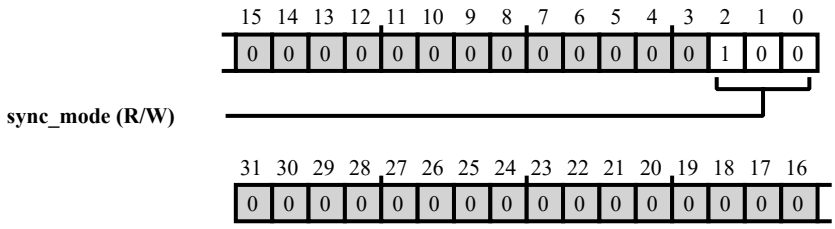


Figure 45-287: SCB1_MST_IB_SYNC_MODE Register Diagram

Table 45-292: SCB1_MST_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

ADSP-2184x SCB3 Register Descriptions

Your module description, here. (SCB3) contains the following registers.

Table 45-293: ADSP-2184x SCB3 Register List

Name	Description
SCB3_APB_CLKO3_IB_FN_MOD_ISS_BM	
SCB3_APB_CLKO3_IB_SYNC_MODE	
SCB3_APB_CLKO4_IB_FN_MOD_ISS_BM	
SCB3_APB_CLKO4_IB_SYNC_MODE	
SCB3_APB_CLKO6_IB_FN_MOD_ISS_BM	
SCB3_APB_CLKO8_IB_FN_MOD_ISS_BM	
SCB3_APB_CLKO8_IB_SYNC_MODE	
SCB3_APB_CLKPWM_IB_FN_MOD_ISS_BM	
SCB3_APB_CLKPWM_IB_SYNC_MODE	
SCB3_APB_SCLK0_0_IB_FN_MOD_ISS_BM	
SCB3_APB_SCLK0_1_IB_FN_MOD_ISS_BM	
SCB3_COMP_ID_0	
SCB3_COMP_ID_1	
SCB3_COMP_ID_2	
SCB3_COMP_ID_3	
SCB3_CRYPTO_0_AHB_CNTL	

Table 45-293: ADSP-2184x SCB3 Register List (Continued)

Name	Description
SCB3_CRYPTO_0_FN_MOD_ISS_BM	
SCB3_CRYPTO_1_AHB_CNTL	
SCB3_CRYPTO_1_FN_MOD_ISS_BM	
SCB3_DAI0_AHB_CNTL	
SCB3_DAI0_FN_MOD_ISS_BM	
SCB3_DAI1_AHB_CNTL	
SCB3_DAI1_FN_MOD_ISS_BM	
SCB3_DLMDMA0_AHB_CNTL	
SCB3_DLMDMA0_FN_MOD_ISS_BM	
SCB3_PERIPH_ID_0	
SCB3_PERIPH_ID_1	
SCB3_PERIPH_ID_2	
SCB3_PERIPH_ID_3	
SCB3_PERIPH_ID_4	
SCB3_PERIPH_ID_5	
SCB3_PERIPH_ID_6	
SCB3_PERIPH_ID_7	
SCB3_SMMR_FN_MOD	
SCB3_SPIF_FABRIC_FN_MOD_ISS_BM	

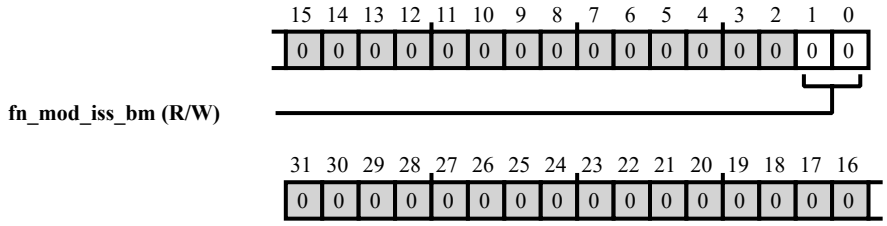


Figure 45-288: SCB3_APB_CLKO3_IB_FN_MOD_ISS_BM Register Diagram

Table 45-294: SCB3_APB_CLKO3_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

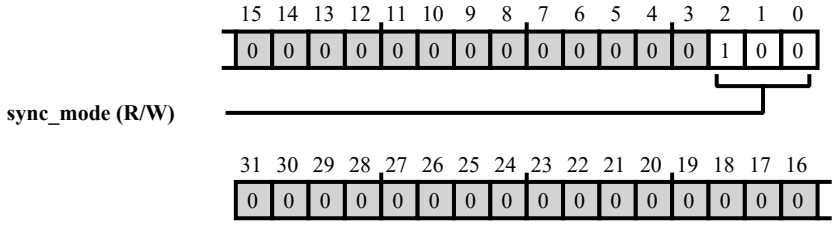


Figure 45-289: SCB3_APB_CLKO3_IB_SYNC_MODE Register Diagram

Table 45-295: SCB3_APB_CLKO3_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

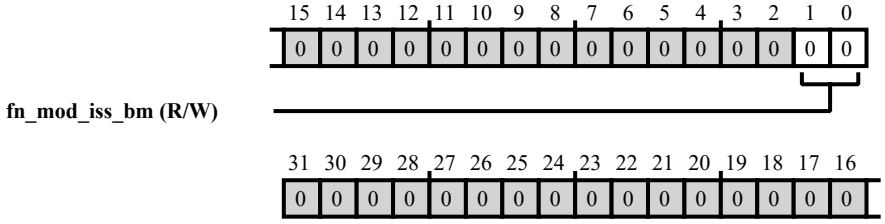


Figure 45-290: SCB3_APB_CLKO4_IB_FN_MOD_ISS_BM Register Diagram

Table 45-296: SCB3_APB_CLKO4_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

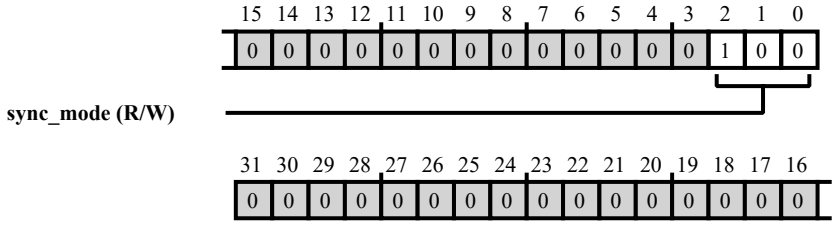


Figure 45-291: SCB3_APB_CLKO4_IB_SYNC_MODE Register Diagram

Table 45-297: SCB3_APB_CLKO4_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

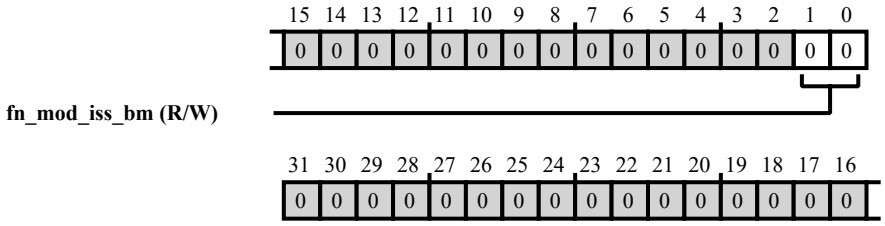


Figure 45-292: SCB3_APB_CLKO6_IB_FN_MOD_ISS_BM Register Diagram

Table 45-298: SCB3_APB_CLKO6_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

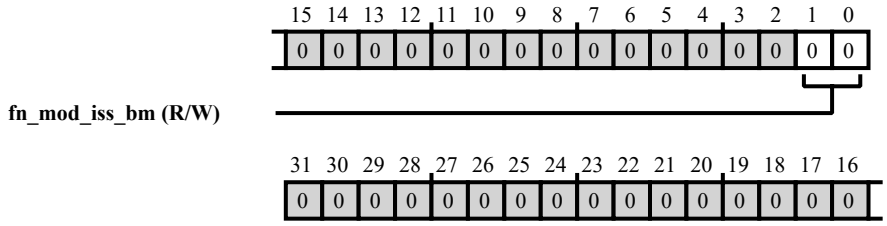


Figure 45-293: SCB3_APB_CLKO8_IB_FN_MOD_ISS_BM Register Diagram

Table 45-299: SCB3_APB_CLKO8_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

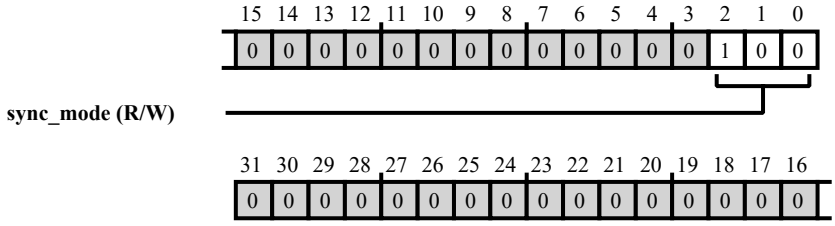


Figure 45-294: SCB3_APB_CLKO8_IB_SYNC_MODE Register Diagram

Table 45-300: SCB3_APB_CLKO8_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

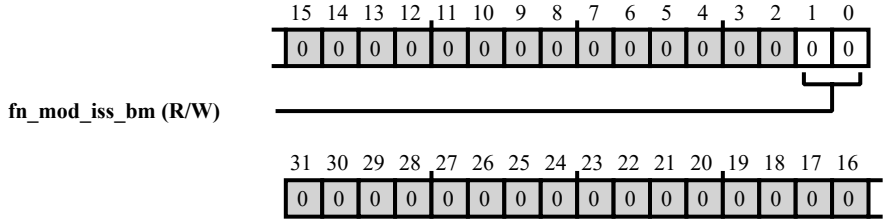


Figure 45-295: SCB3_APB_CLKPWM_IB_FN_MOD_ISS_BM Register Diagram

Table 45-301: SCB3_APB_CLKPWM_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

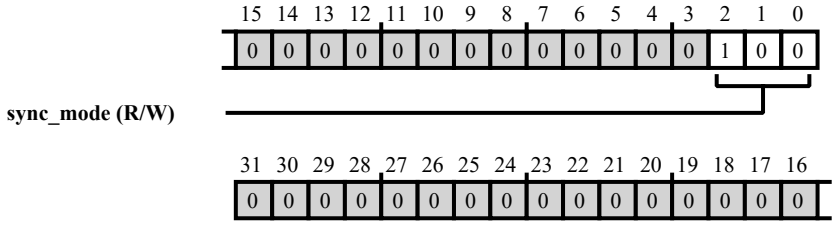


Figure 45-296: SCB3_APB_CLKPWM_IB_SYNC_MODE Register Diagram

Table 45-302: SCB3_APB_CLKPWM_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

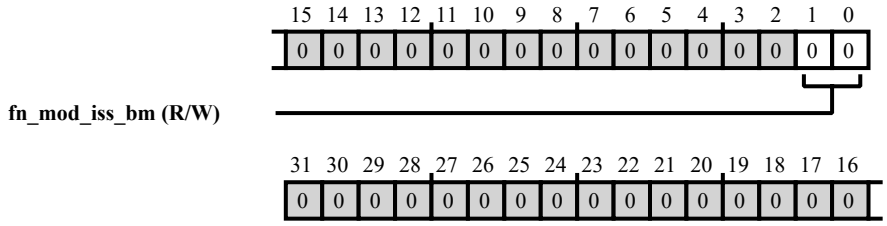


Figure 45-297: SCB3_APB_SCLK0_0_IB_FN_MOD_ISS_BM Register Diagram

Table 45-303: SCB3_APB_SCLK0_0_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

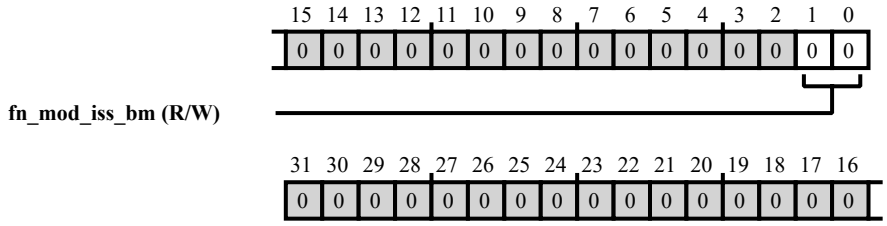


Figure 45-298: SCB3_APB_SCLK0_1_IB_FN_MOD_ISS_BM Register Diagram

Table 45-304: SCB3_APB_SCLK0_1_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

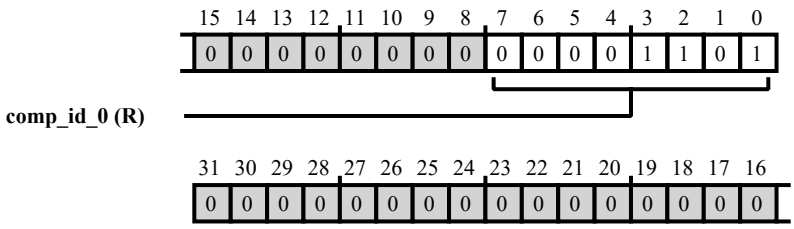


Figure 45-299: SCB3_COMP_ID_0 Register Diagram

Table 45-305: SCB3_COMP_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_0	.

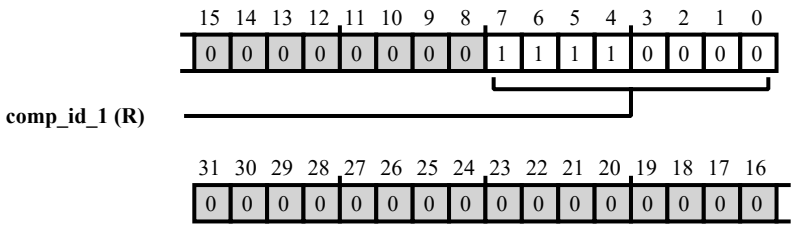


Figure 45-300: SCB3_COMP_ID_1 Register Diagram

Table 45-306: SCB3_COMP_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_1	.

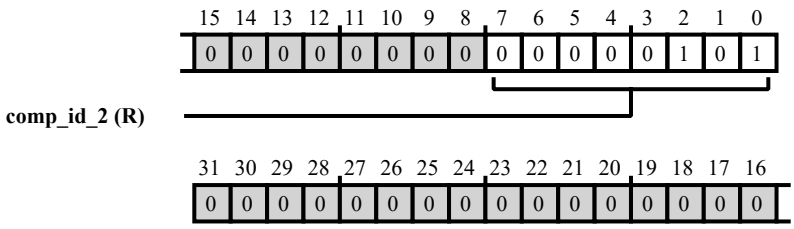


Figure 45-301: SCB3_COMP_ID_2 Register Diagram

Table 45-307: SCB3_COMP_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_2	.

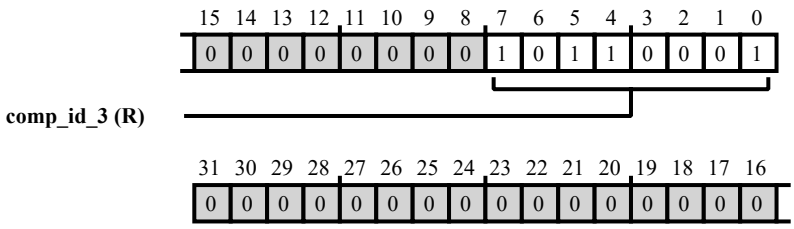


Figure 45-302: SCB3_COMP_ID_3 Register Diagram

Table 45-308: SCB3_COMP_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_3	.

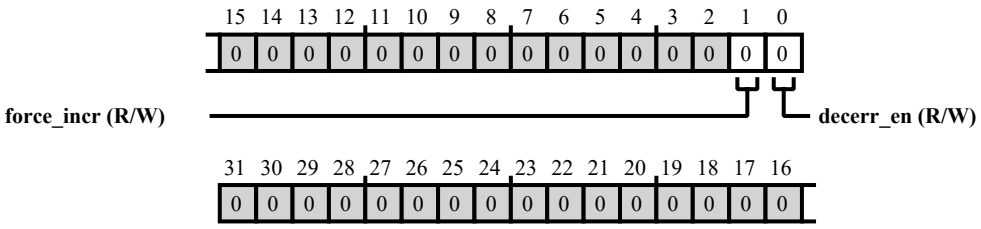


Figure 45-303: SCB3_CRYPTO_0_AHB_CNTL Register Diagram

Table 45-309: SCB3_CRYPTO_0_AHB_CNTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	FORCE_INCR	.
0 (R/W)	DECERR_EN	.

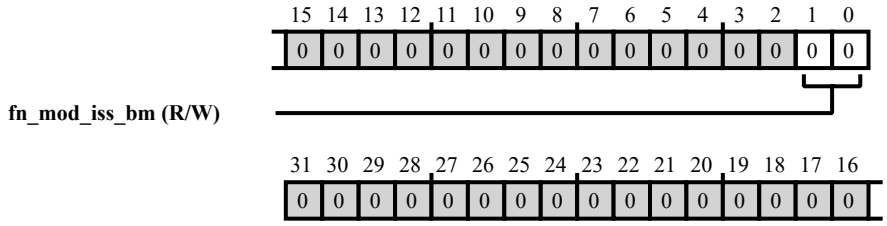


Figure 45-304: SCB3_CRYPTO_0_FN_MOD_ISS_BM Register Diagram

Table 45-310: SCB3_CRYPTO_0_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

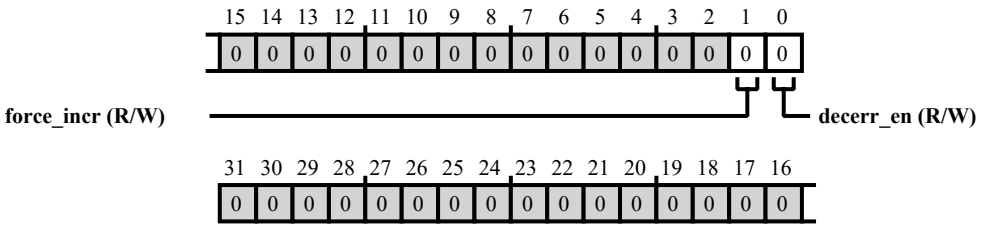


Figure 45-305: SCB3_CRYPTO_1_AHB_CNTL Register Diagram

Table 45-311: SCB3_CRYPTO_1_AHB_CNTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	FORCE_INCR	.
0 (R/W)	DECERR_EN	.

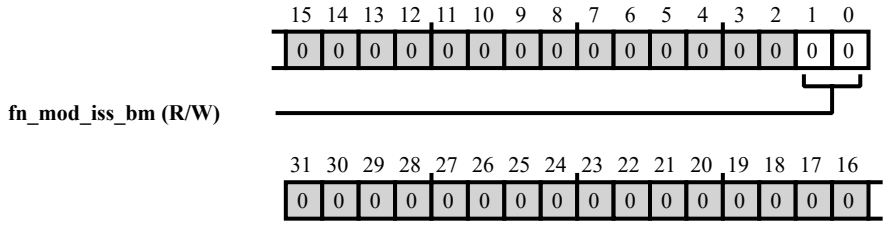


Figure 45-306: SCB3_CRYPTO_1_FN_MOD_ISS_BM Register Diagram

Table 45-312: SCB3_CRYPTO_1_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

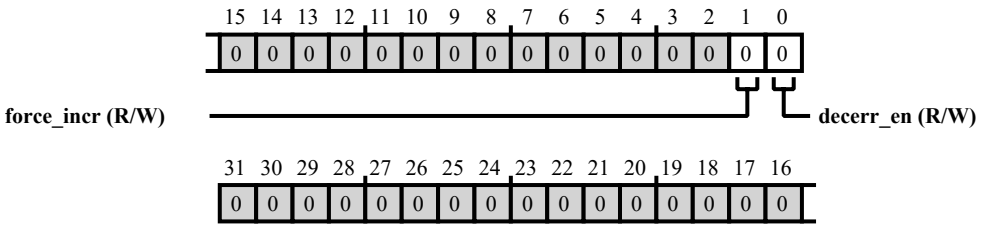


Figure 45-307: SCB3_DAI0_AHB_CNTL Register Diagram

Table 45-313: SCB3_DAI0_AHB_CNTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	FORCE_INCR	.
0 (R/W)	DECERR_EN	.

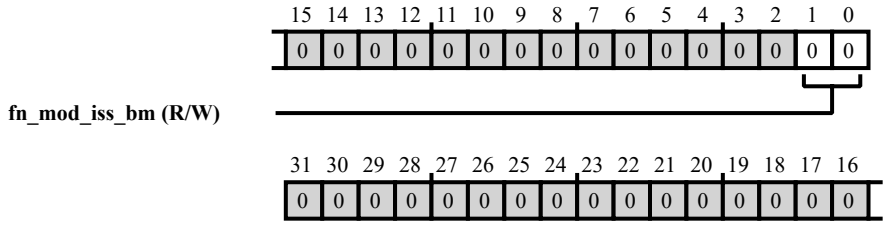


Figure 45-308: SCB3_DAI0_FN_MOD_ISS_BM Register Diagram

Table 45-314: SCB3_DAI0_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

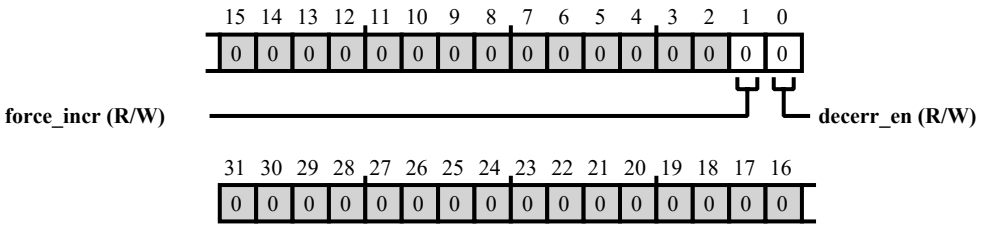


Figure 45-309: SCB3_DAI1_AHB_CNTL Register Diagram

Table 45-315: SCB3_DAI1_AHB_CNTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	FORCE_INCR	.
0 (R/W)	DECERR_EN	.

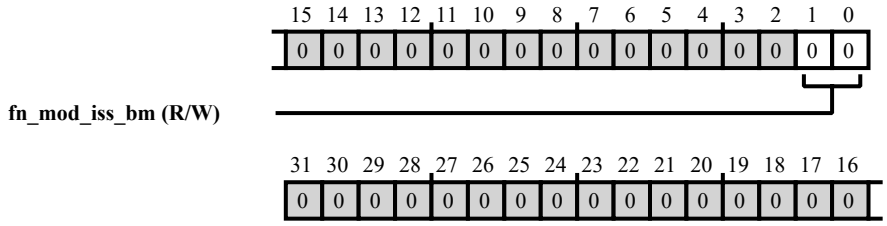


Figure 45-310: SCB3_DAI1_FN_MOD_ISS_BM Register Diagram

Table 45-316: SCB3_DAI1_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

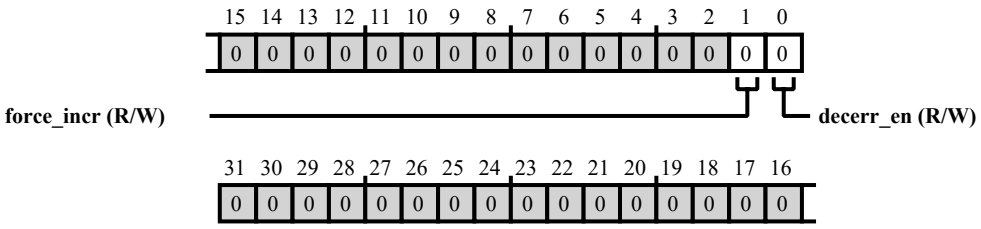


Figure 45-311: SCB3_DLMDMA0_AHB_CNTL Register Diagram

Table 45-317: SCB3_DLMDMA0_AHB_CNTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	FORCE_INCR	.
0 (R/W)	DECERR_EN	.

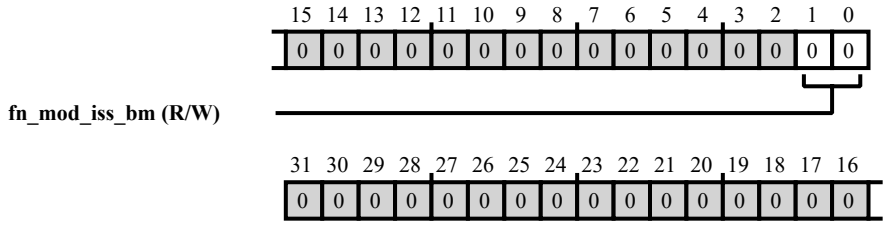


Figure 45-312: SCB3_DLMDMA0_FN_MOD_ISS_BM Register Diagram

Table 45-318: SCB3_DLMDMA0_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

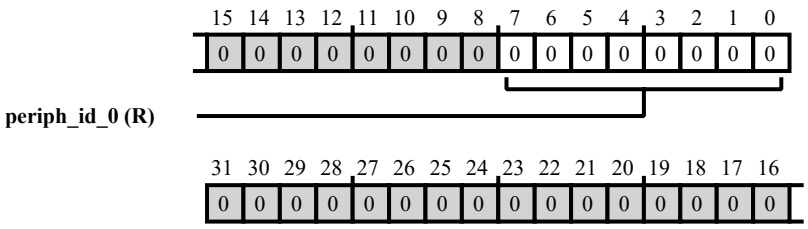


Figure 45-313: SCB3_PERIPH_ID_0 Register Diagram

Table 45-319: SCB3_PERIPH_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_0	.

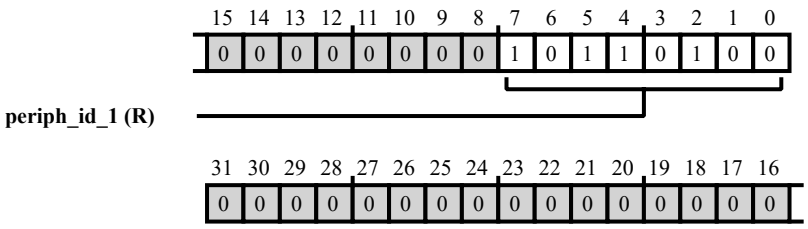


Figure 45-314: SCB3_PERIPH_ID_1 Register Diagram

Table 45-320: SCB3_PERIPH_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_1	.

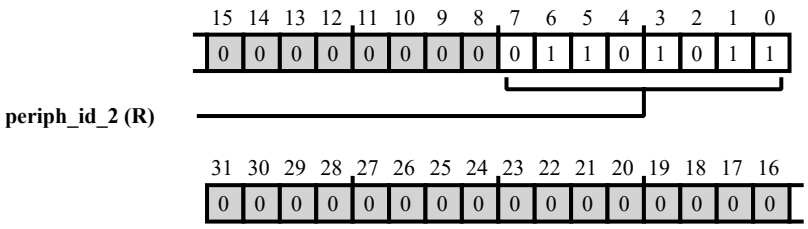


Figure 45-315: SCB3_PERIPH_ID_2 Register Diagram

Table 45-321: SCB3_PERIPH_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_2	.

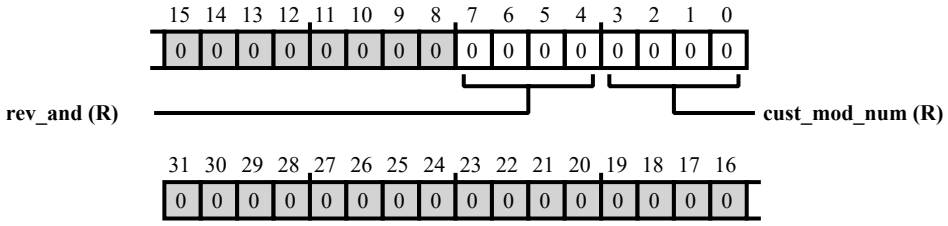


Figure 45-316: SCB3_PERIPH_ID_3 Register Diagram

Table 45-322: SCB3_PERIPH_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV_AND	.
3:0 (R/NW)	CUST_MOD_NUM	.

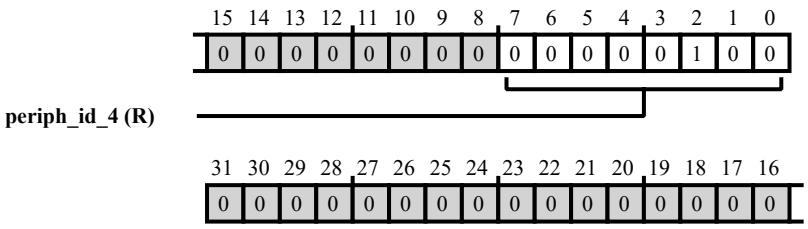


Figure 45-317: SCB3_PERIPH_ID_4 Register Diagram

Table 45-323: SCB3_PERIPH_ID_4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_4	.

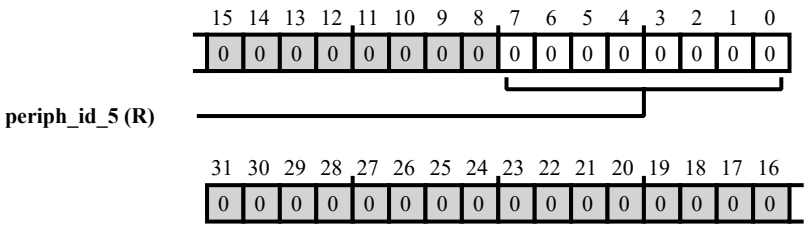


Figure 45-318: SCB3_PERIPH_ID_5 Register Diagram

Table 45-324: SCB3_PERIPH_ID_5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_5	.

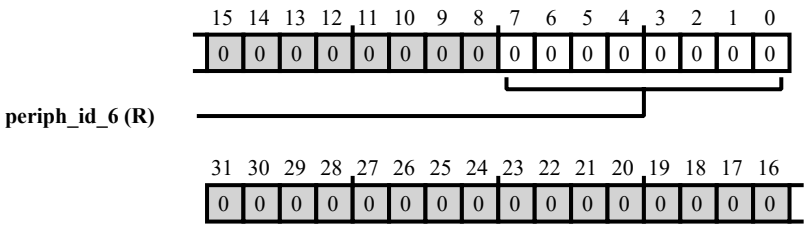


Figure 45-319: SCB3_PERIPH_ID_6 Register Diagram

Table 45-325: SCB3_PERIPH_ID_6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_6	.

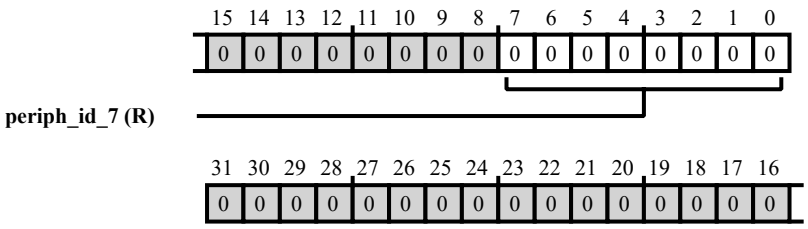


Figure 45-320: SCB3_PERIPH_ID_7 Register Diagram

Table 45-326: SCB3_PERIPH_ID_7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_7	.

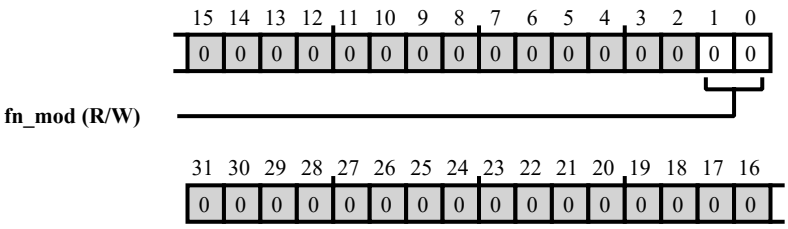


Figure 45-321: SCB3_SMMR_FN_MOD Register Diagram

Table 45-327: SCB3_SMMR_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

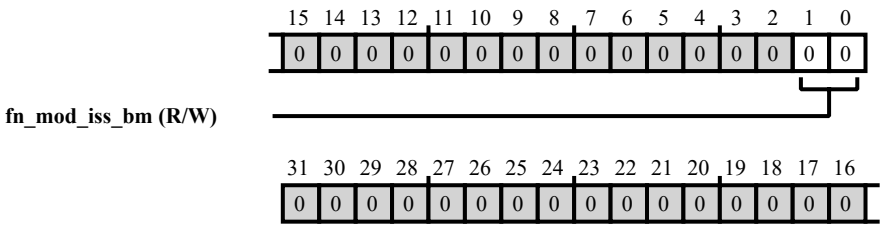


Figure 45-322: SCB3_SPIF_FABRIC_FN_MOD_ISS_BM Register Diagram

Table 45-328: SCB3_SPIF_FABRIC_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

ADSP-2184x SCB4 Register Descriptions

Your module description, here. (SCB4) contains the following registers.

Table 45-329: ADSP-2184x SCB4 Register List

Name	Description
SCB4_COMP_ID_0	
SCB4_COMP_ID_1	
SCB4_COMP_ID_2	
SCB4_COMP_ID_3	
SCB4_FABRIC_FIR_CH0_IB_FN_MOD	
SCB4_FABRIC_FIR_CH0_IB_FN_MOD_ISS_BM	
SCB4_FABRIC_FIR_CH0_IB_SYNC_MODE	
SCB4_FABRIC_FIR_CH1_IB_FN_MOD	
SCB4_FABRIC_FIR_CH1_IB_FN_MOD_ISS_BM	
SCB4_FABRIC_FIR_CH1_IB_SYNC_MODE	
SCB4_FABRIC_IIR_CH0_IB_FN_MOD	
SCB4_FABRIC_IIR_CH0_IB_FN_MOD_ISS_BM	
SCB4_FABRIC_IIR_CH0_IB_SYNC_MODE	
SCB4_FABRIC_IIR_CH1_IB_FN_MOD	
SCB4_FABRIC_IIR_CH1_IB_FN_MOD_ISS_BM	
SCB4_FABRIC_IIR_CH1_IB_SYNC_MODE	

Table 45-329: ADSP-2184x SCB4 Register List (Continued)

Name	Description
SCB4_FABRIC_MMR_IB_FN_MOD	
SCB4_FABRIC_MMR_IB_FN_MOD2	
SCB4_FABRIC_MMR_IB_FN_MOD_ISS_BM	
SCB4_FABRIC_MMR_IB_SYNC_MODE	
SCB4_FABRIC_S1PORT_IB_FN_MOD	
SCB4_FABRIC_S1PORT_IB_READ_QOS	
SCB4_FABRIC_S1PORT_IB_SYNC_MODE	
SCB4_FABRIC_S1PORT_IB_WRITE_QOS	
SCB4_FABRIC_S2PORT_IB_FN_MOD	
SCB4_FABRIC_S2PORT_IB_READ_QOS	
SCB4_FABRIC_S2PORT_IB_SYNC_MODE	
SCB4_FABRIC_S2PORT_IB_WRITE_QOS	
SCB4_FIR1_CH0_FN_MOD	
SCB4_FIR1_CH0_READ_QOS	
SCB4_FIR1_CH0_WRITE_QOS	
SCB4_FIR1_CH1_FN_MOD	
SCB4_FIR1_CH1_READ_QOS	
SCB4_FIR1_CH1_WRITE_QOS	
SCB4_FIR_CH0_FN_MOD	
SCB4_FIR_CH0_READ_QOS	
SCB4_FIR_CH0_WRITE_QOS	
SCB4_FIR_CH1_FN_MOD	
SCB4_FIR_CH1_READ_QOS	
SCB4_FIR_CH1_WRITE_QOS	
SCB4_IDMA_IB_FN_MOD2	
SCB4_IDMA_IB_SYNC_MODE	
SCB4_IIR0_CH0_IB_FN_MOD	
SCB4_IIR0_CH0_IB_FN_MOD2	
SCB4_IIR0_CH0_IB_READ_QOS	
SCB4_IIR0_CH0_IB_WRITE_QOS	
SCB4_IIR0_CH1_IB_FN_MOD	
SCB4_IIR0_CH1_IB_FN_MOD2	

Table 45-329: ADSP-2184x SCB4 Register List (Continued)

Name	Description
SCB4_IIR0_CH1_IB_READ_QOS	
SCB4_IIR0_CH1_IB_WRITE_QOS	
SCB4_IIR1_CH0_IB_FN_MOD	
SCB4_IIR1_CH0_IB_FN_MOD2	
SCB4_IIR1_CH0_IB_READ_QOS	
SCB4_IIR1_CH0_IB_WRITE_QOS	
SCB4_IIR1_CH1_IB_FN_MOD	
SCB4_IIR1_CH1_IB_FN_MOD2	
SCB4_IIR1_CH1_IB_READ_QOS	
SCB4_IIR1_CH1_IB_WRITE_QOS	
SCB4_IIR2_CH0_IB_FN_MOD	
SCB4_IIR2_CH0_IB_FN_MOD2	
SCB4_IIR2_CH0_IB_READ_QOS	
SCB4_IIR2_CH0_IB_WRITE_QOS	
SCB4_IIR2_CH1_IB_FN_MOD	
SCB4_IIR2_CH1_IB_FN_MOD2	
SCB4_IIR2_CH1_IB_READ_QOS	
SCB4_IIR2_CH1_IB_WRITE_QOS	
SCB4_IIR3_CH0_IB_FN_MOD	
SCB4_IIR3_CH0_IB_FN_MOD2	
SCB4_IIR3_CH0_IB_READ_QOS	
SCB4_IIR3_CH0_IB_WRITE_QOS	
SCB4_IIR3_CH1_IB_FN_MOD	
SCB4_IIR3_CH1_IB_FN_MOD2	
SCB4_IIR3_CH1_IB_READ_QOS	
SCB4_IIR3_CH1_IB_WRITE_QOS	
SCB4_PERIPH_ID_0	
SCB4_PERIPH_ID_1	
SCB4_PERIPH_ID_2	
SCB4_PERIPH_ID_3	
SCB4_PERIPH_ID_4	
SCB4_PERIPH_ID_5	

Table 45-329: ADSP-2184x SCB4 Register List (Continued)

Name	Description
SCB4_PERIPH_ID_6	
SCB4_PERIPH_ID_7	
SCB4_REMAP	
SCB4_SHARC_DPORT_FN_MOD	
SCB4_SHARC_DPORT_READ_QOS	
SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD	
SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD_ISS_BM	
SCB4_SHARC_DPORT_TO_SF_L2_IB_SYNC_MODE	
SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD	
SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD_ISS_BM	
SCB4_SHARC_DPORT_TO_SF_IB_SYNC_MODE	
SCB4_SHARC_DPORT_WRITE_QOS	
SCB4_SHARC_IPORT_IB_FN_MOD2	
SCB4_SHARC_IPORT_IB_SYNC_MODE	

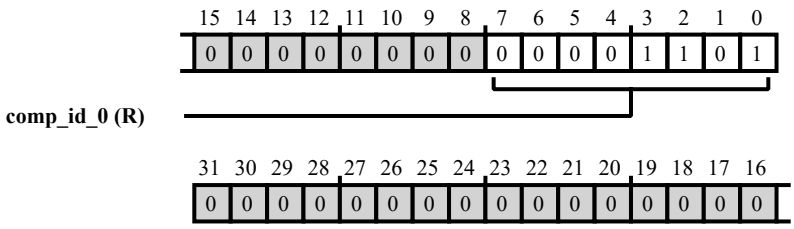


Figure 45-323: SCB4_COMP_ID_0 Register Diagram

Table 45-330: SCB4_COMP_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_0	.

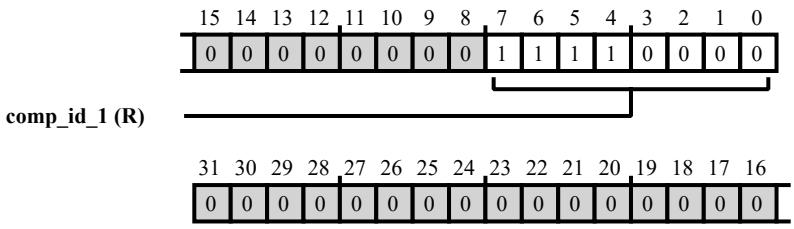


Figure 45-324: SCB4_COMP_ID_1 Register Diagram

Table 45-331: SCB4_COMP_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_1	.

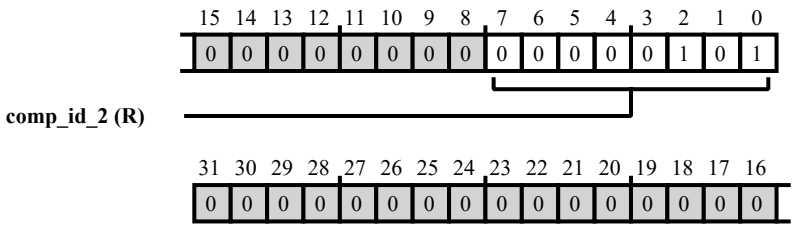


Figure 45-325: SCB4_COMP_ID_2 Register Diagram

Table 45-332: SCB4_COMP_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_2	.

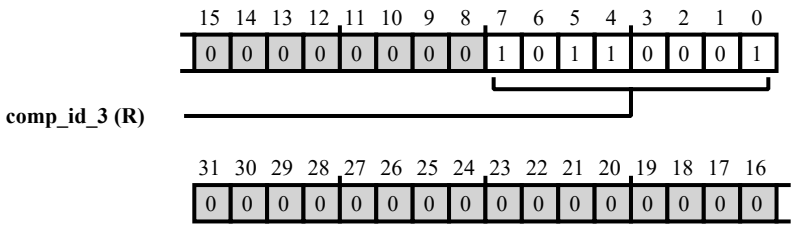


Figure 45-326: SCB4_COMP_ID_3 Register Diagram

Table 45-333: SCB4_COMP_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_3	.

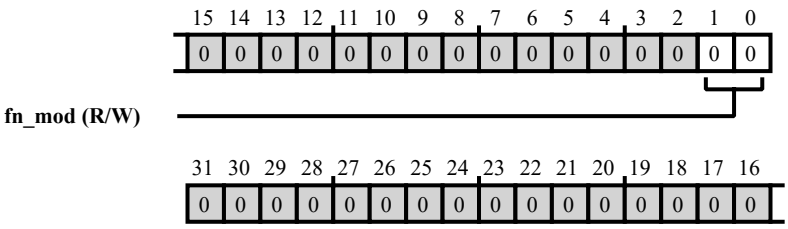


Figure 45-327: SCB4_FABRIC_FIR_CH0_IB_FN_MOD Register Diagram

Table 45-334: SCB4_FABRIC_FIR_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

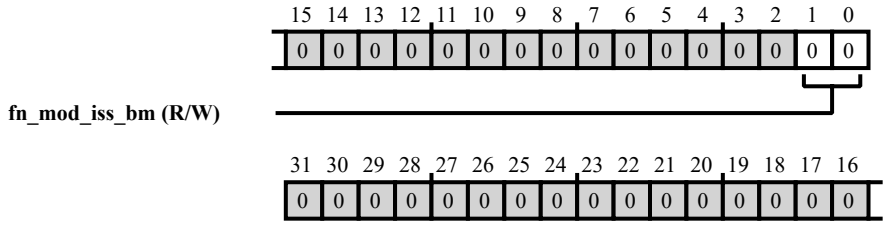


Figure 45-328: SCB4_FABRIC_FIR_CH0_IB_FN_MOD_ISS_BM Register Diagram

Table 45-335: SCB4_FABRIC_FIR_CH0_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

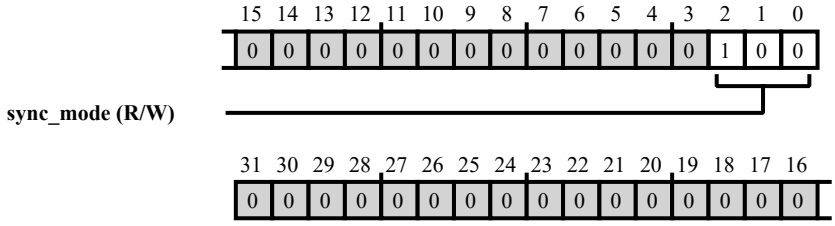


Figure 45-329: SCB4_FABRIC_FIR_CH0_IB_SYNC_MODE Register Diagram

Table 45-336: SCB4_FABRIC_FIR_CH0_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

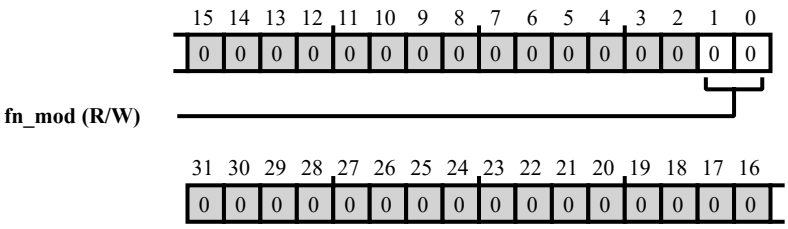


Figure 45-330: SCB4_FABRIC_FIR_CH1_IB_FN_MOD Register Diagram

Table 45-337: SCB4_FABRIC_FIR_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

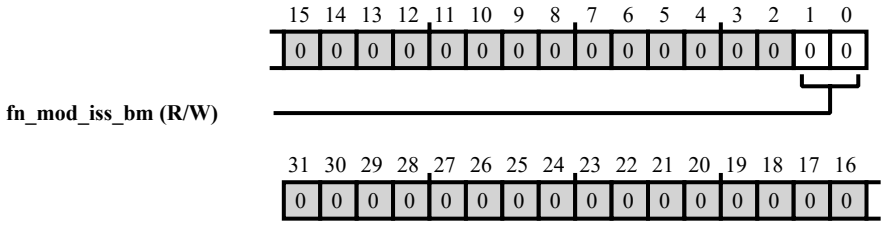


Figure 45-331: SCB4_FABRIC_FIR_CH1_IB_FN_MOD_ISS_BM Register Diagram

Table 45-338: SCB4_FABRIC_FIR_CH1_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

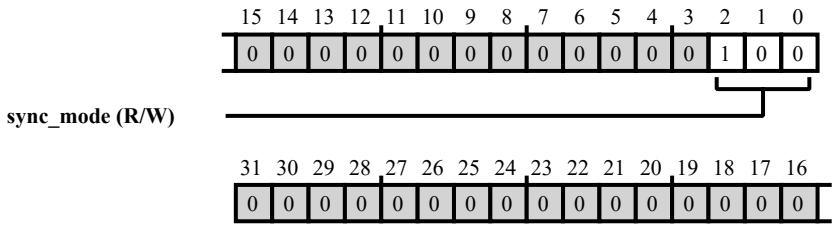


Figure 45-332: SCB4_FABRIC_FIR_CH1_IB_SYNC_MODE Register Diagram

Table 45-339: SCB4_FABRIC_FIR_CH1_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

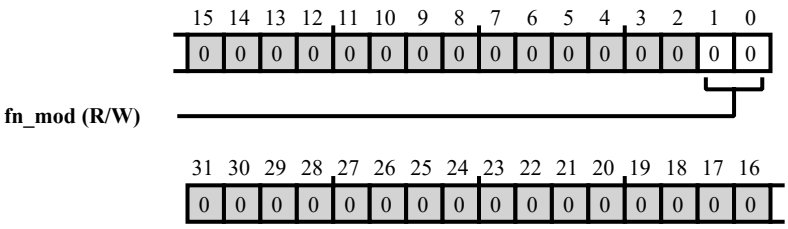


Figure 45-333: SCB4_FABRIC_IIR_CH0_IB_FN_MOD Register Diagram

Table 45-340: SCB4_FABRIC_IIR_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

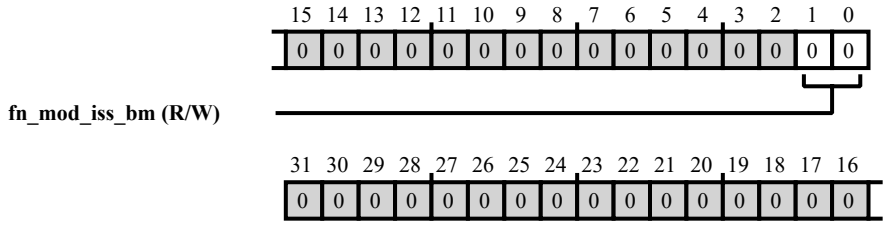


Figure 45-334: SCB4_FABRIC_IIR_CH0_IB_FN_MOD_ISS_BM Register Diagram

Table 45-341: SCB4_FABRIC_IIR_CH0_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

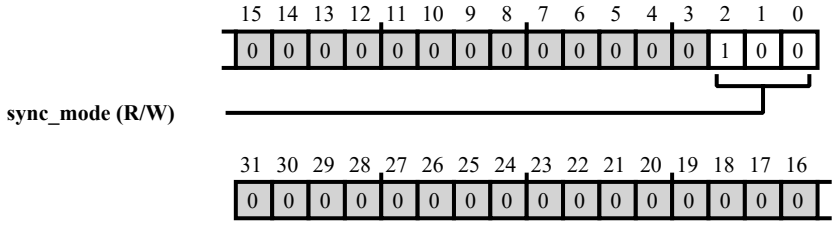


Figure 45-335: SCB4_FABRIC_IIR_CH0_IB_SYNC_MODE Register Diagram

Table 45-342: SCB4_FABRIC_IIR_CH0_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

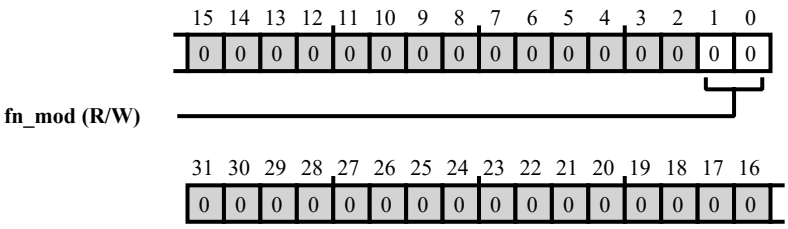


Figure 45-336: SCB4_FABRIC_IIR_CH1_IB_FN_MOD Register Diagram

Table 45-343: SCB4_FABRIC_IIR_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

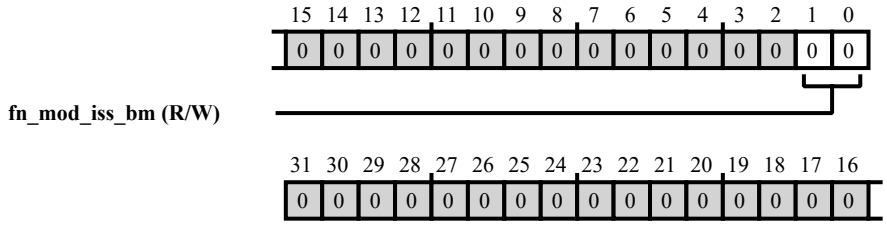


Figure 45-337: SCB4_FABRIC_IIR_CH1_IB_FN_MOD_ISS_BM Register Diagram

Table 45-344: SCB4_FABRIC_IIR_CH1_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

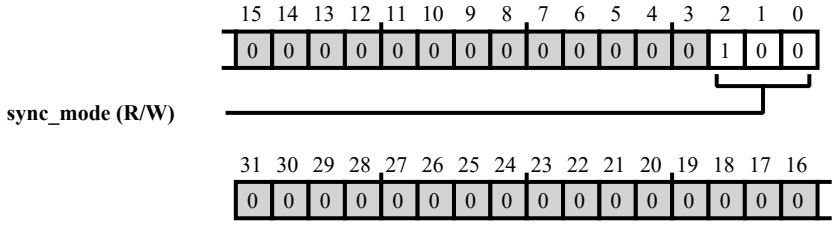


Figure 45-338: SCB4_FABRIC_IIR_CH1_IB_SYNC_MODE Register Diagram

Table 45-345: SCB4_FABRIC_IIR_CH1_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

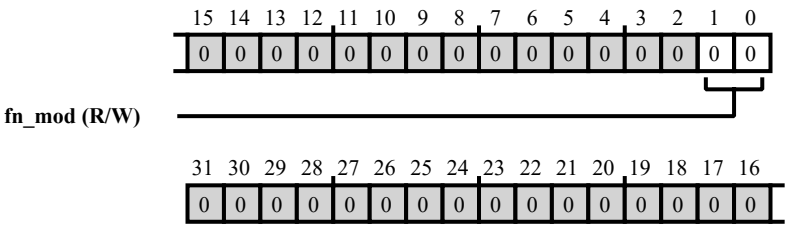


Figure 45-339: SCB4_FABRIC_MMR_IB_FN_MOD Register Diagram

Table 45-346: SCB4_FABRIC_MMR_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

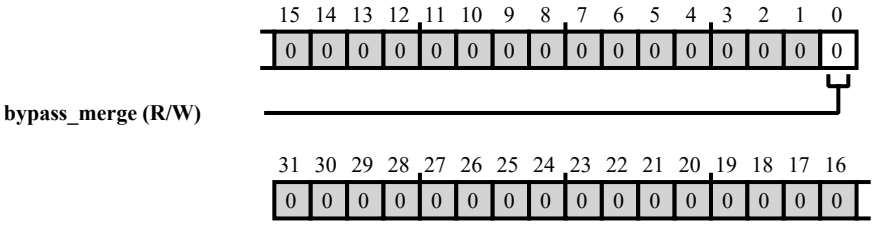


Figure 45-340: SCB4_FABRIC_MMR_IB_FN_MOD2 Register Diagram

Table 45-347: SCB4_FABRIC_MMR_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

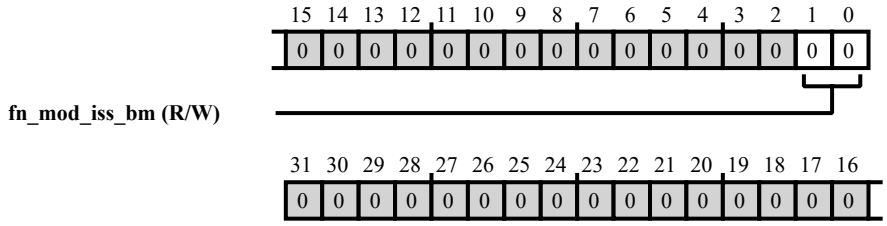


Figure 45-341: SCB4_FABRIC_MMR_IB_FN_MOD_ISS_BM Register Diagram

Table 45-348: SCB4_FABRIC_MMR_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

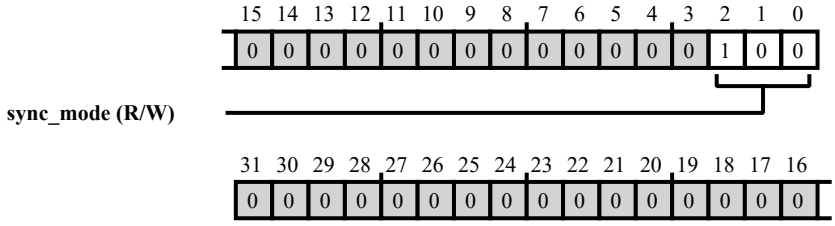


Figure 45-342: SCB4_FABRIC_MMR_IB_SYNC_MODE Register Diagram

Table 45-349: SCB4_FABRIC_MMR_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

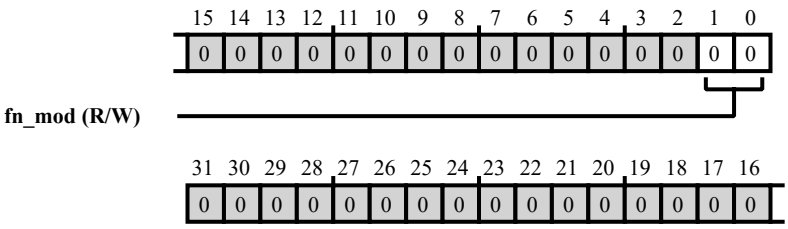


Figure 45-343: SCB4_FABRIC_S1PORT_IB_FN_MOD Register Diagram

Table 45-350: SCB4_FABRIC_S1PORT_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

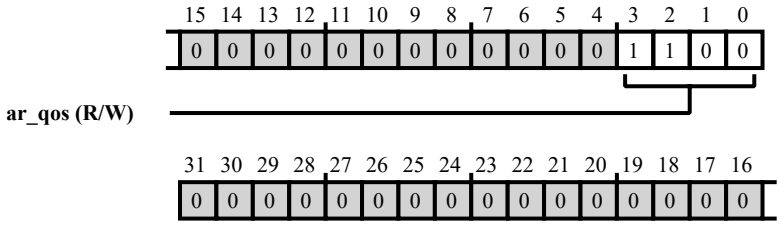


Figure 45-344: SCB4_FABRIC_S1PORT_IB_READ_QOS Register Diagram

Table 45-351: SCB4_FABRIC_S1PORT_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

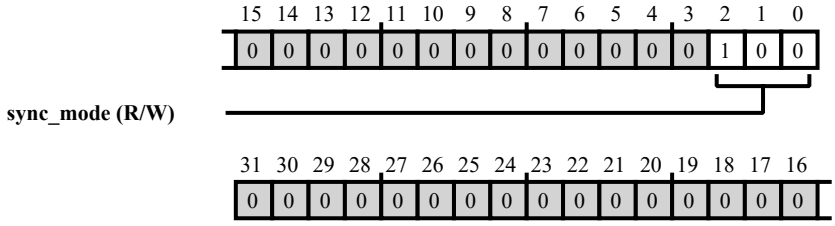


Figure 45-345: SCB4_FABRIC_S1PORT_IB_SYNC_MODE Register Diagram

Table 45-352: SCB4_FABRIC_S1PORT_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

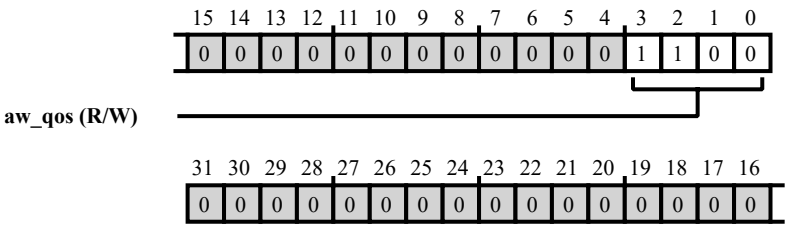


Figure 45-346: SCB4_FABRIC_S1PORT_IB_WRITE_QOS Register Diagram

Table 45-353: SCB4_FABRIC_S1PORT_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

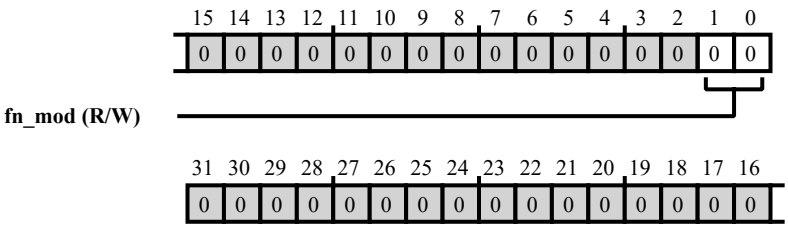


Figure 45-347: SCB4_FABRIC_S2PORT_IB_FN_MOD Register Diagram

Table 45-354: SCB4_FABRIC_S2PORT_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

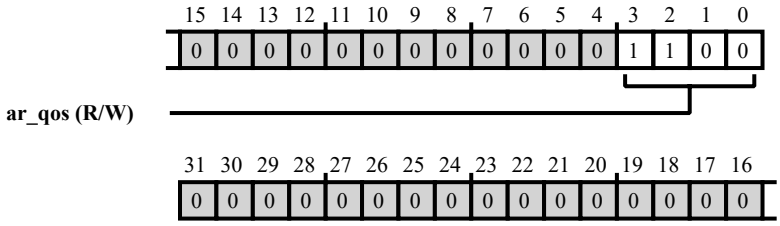


Figure 45-348: SCB4_FABRIC_S2PORT_IB_READ_QOS Register Diagram

Table 45-355: SCB4_FABRIC_S2PORT_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

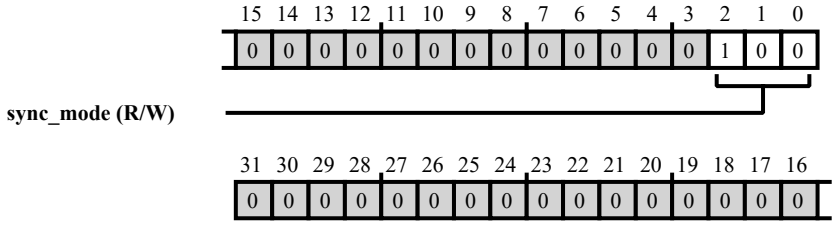


Figure 45-349: SCB4_FABRIC_S2PORT_IB_SYNC_MODE Register Diagram

Table 45-356: SCB4_FABRIC_S2PORT_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

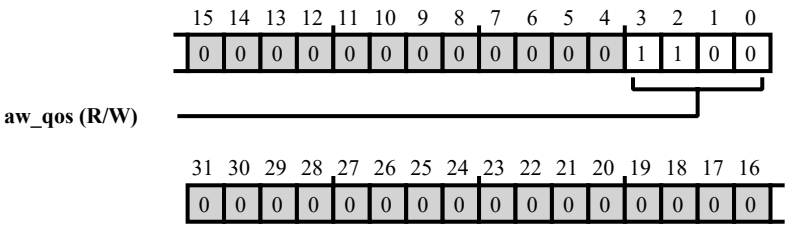


Figure 45-350: SCB4_FABRIC_S2PORT_IB_WRITE_QOS Register Diagram

Table 45-357: SCB4_FABRIC_S2PORT_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

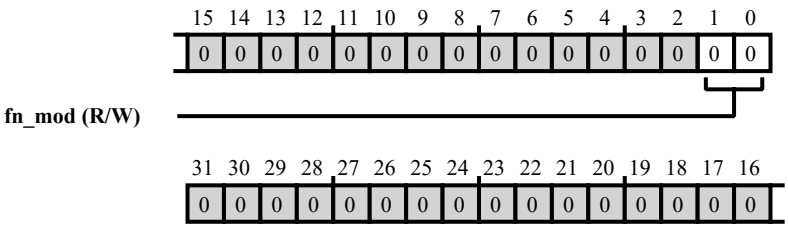


Figure 45-351: SCB4_FIR1_CH0_FN_MOD Register Diagram

Table 45-358: SCB4_FIR1_CH0_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

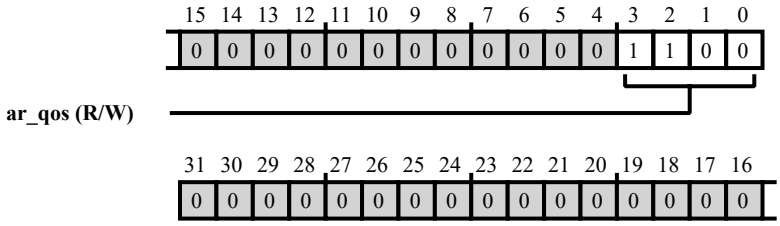


Figure 45-352: SCB4_FIR1_CH0_READ_QOS Register Diagram

Table 45-359: SCB4_FIR1_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

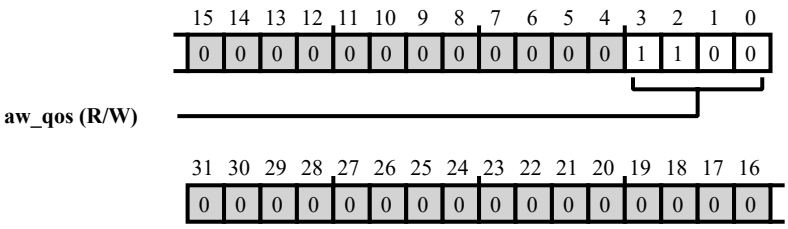


Figure 45-353: SCB4_FIR1_CH0_WRITE_QOS Register Diagram

Table 45-360: SCB4_FIR1_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

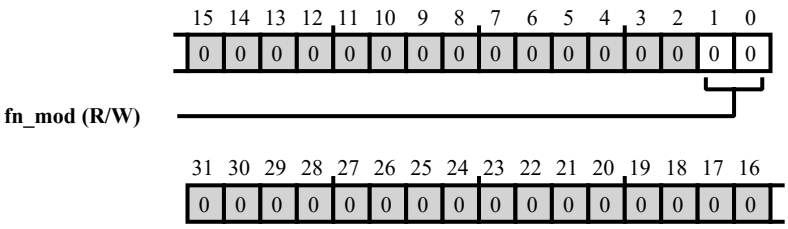


Figure 45-354: SCB4_FIR1_CH1_FN_MOD Register Diagram

Table 45-361: SCB4_FIR1_CH1_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

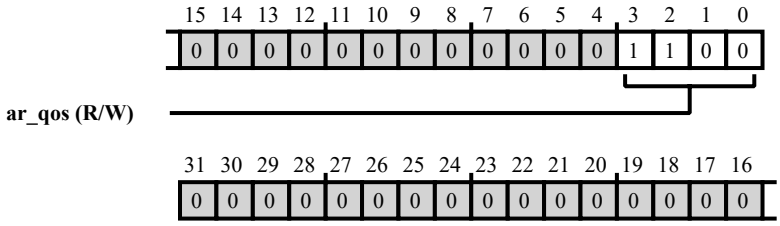


Figure 45-355: SCB4_FIR1_CH1_READ_QOS Register Diagram

Table 45-362: SCB4_FIR1_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

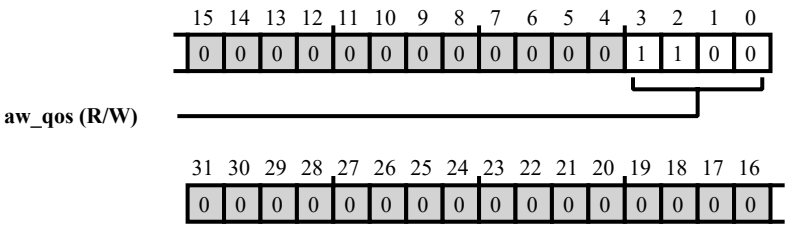


Figure 45-356: SCB4_FIR1_CH1_WRITE_QOS Register Diagram

Table 45-363: SCB4_FIR1_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

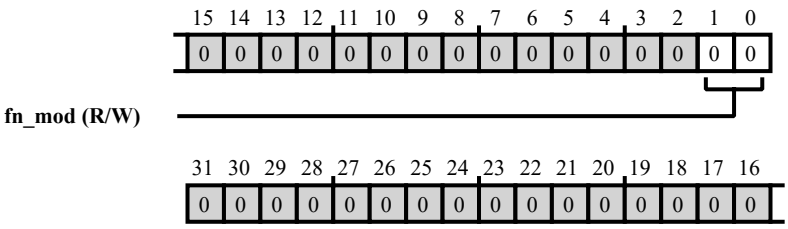


Figure 45-357: SCB4_FIR_CH0_FN_MOD Register Diagram

Table 45-364: SCB4_FIR_CH0_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

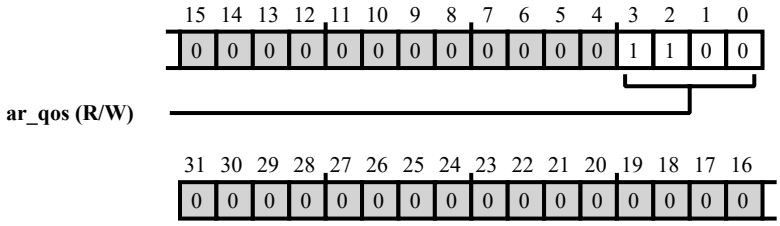


Figure 45-358: SCB4_FIR_CH0_READ_QOS Register Diagram

Table 45-365: SCB4_FIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

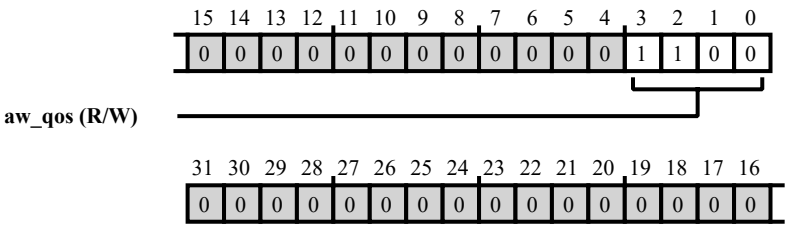


Figure 45-359: SCB4_FIR_CH0_WRITE_QOS Register Diagram

Table 45-366: SCB4_FIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

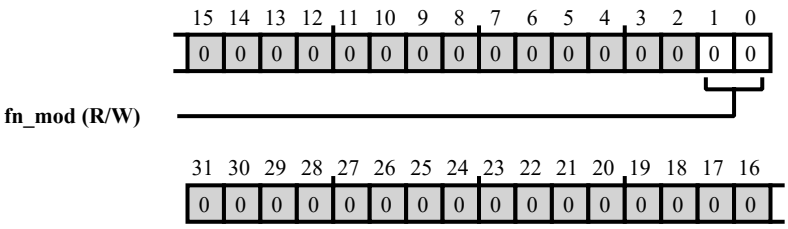


Figure 45-360: SCB4_FIR_CH1_FN_MOD Register Diagram

Table 45-367: SCB4_FIR_CH1_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

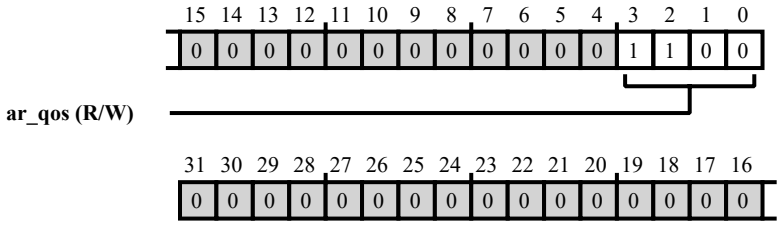


Figure 45-361: SCB4_FIR_CH1_READ_QOS Register Diagram

Table 45-368: SCB4_FIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

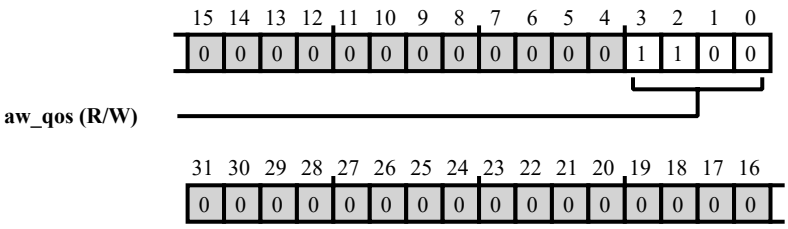


Figure 45-362: SCB4_FIR_CH1_WRITE_QOS Register Diagram

Table 45-369: SCB4_FIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

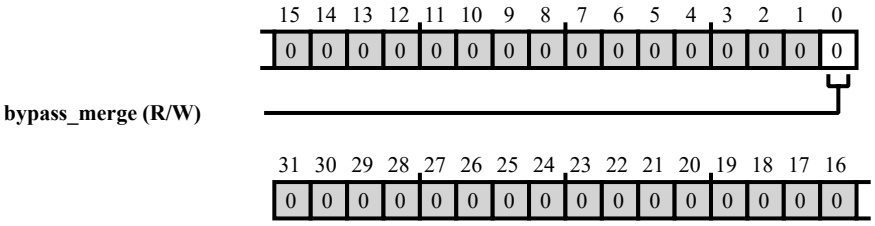


Figure 45-363: SCB4_IDMA_IB_FN_MOD2 Register Diagram

Table 45-370: SCB4_IDMA_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

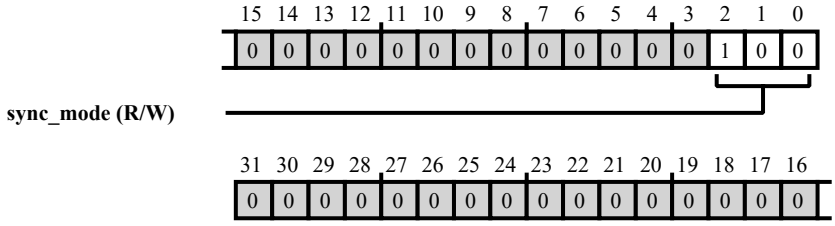


Figure 45-364: SCB4_IDMA_IB_SYNC_MODE Register Diagram

Table 45-371: SCB4_IDMA_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

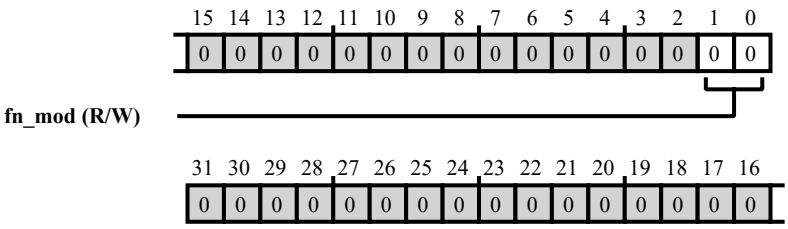


Figure 45-365: SCB4_IIR0_CH0_IB_FN_MOD Register Diagram

Table 45-372: SCB4_IIR0_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

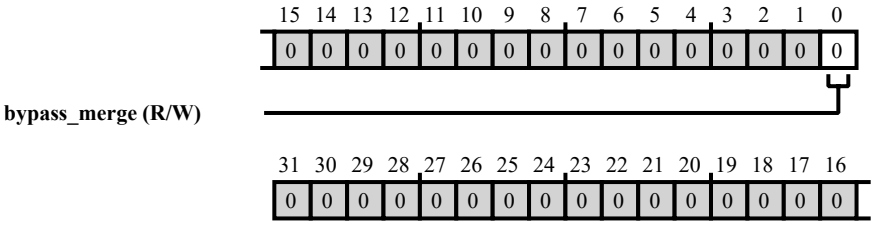


Figure 45-366: SCB4_IIR0_CH0_IB_FN_MOD2 Register Diagram

Table 45-373: SCB4_IIR0_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

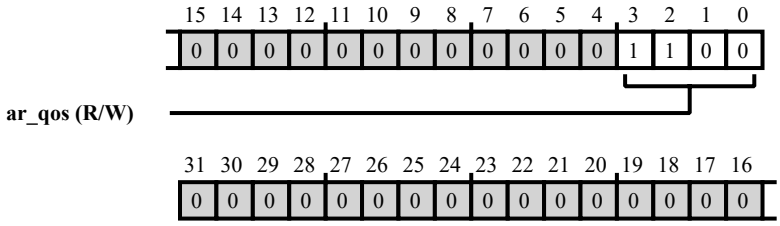


Figure 45-367: SCB4_IIR0_CH0_IB_READ_QOS Register Diagram

Table 45-374: SCB4_IIR0_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

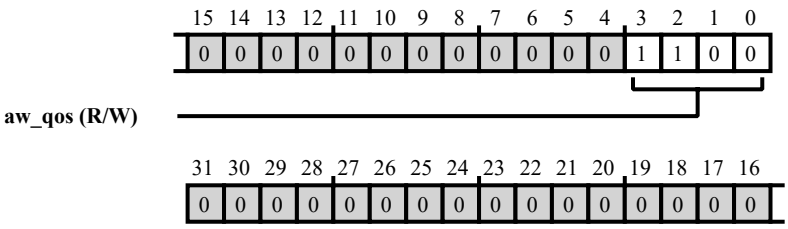


Figure 45-368: SCB4_IIR0_CH0_IB_WRITE_QOS Register Diagram

Table 45-375: SCB4_IIR0_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

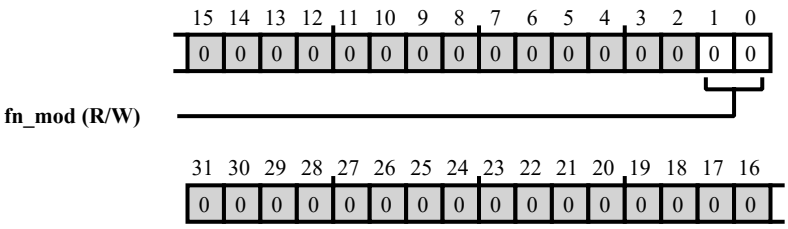


Figure 45-369: SCB4_IIR0_CH1_IB_FN_MOD Register Diagram

Table 45-376: SCB4_IIR0_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

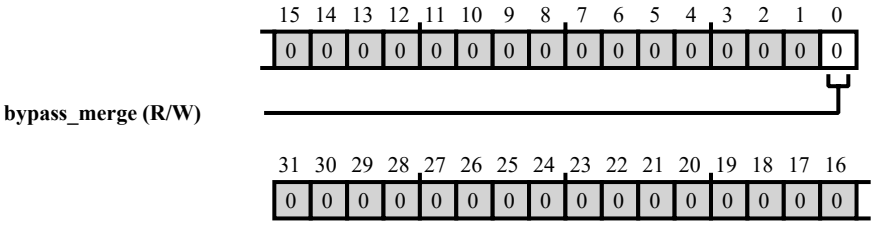


Figure 45-370: SCB4_IIR0_CH1_IB_FN_MOD2 Register Diagram

Table 45-377: SCB4_IIR0_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

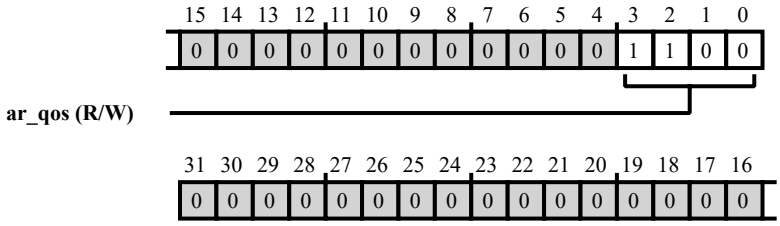


Figure 45-371: SCB4_IIR0_CH1_IB_READ_QOS Register Diagram

Table 45-378: SCB4_IIR0_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

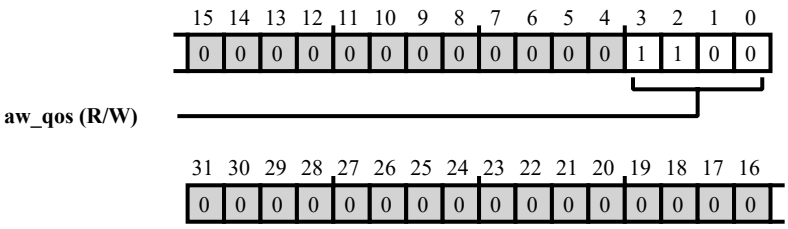


Figure 45-372: SCB4_IIR0_CH1_IB_WRITE_QOS Register Diagram

Table 45-379: SCB4_IIR0_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

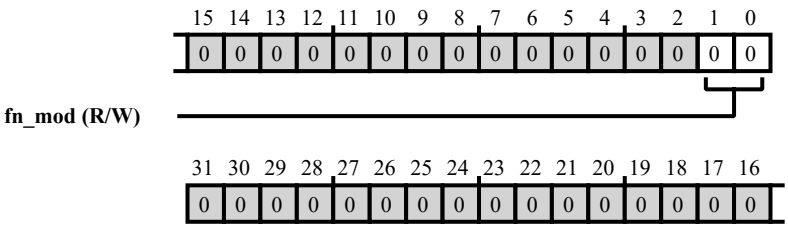


Figure 45-373: SCB4_IIR1_CH0_IB_FN_MOD Register Diagram

Table 45-380: SCB4_IIR1_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

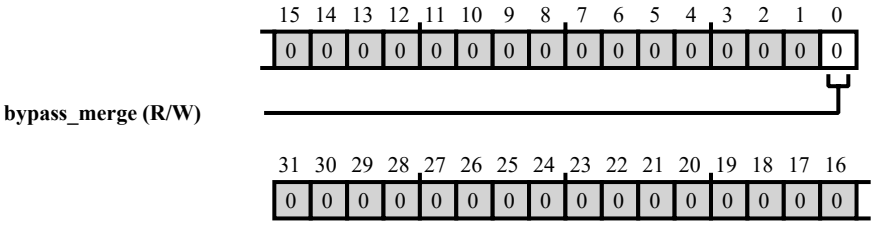


Figure 45-374: SCB4_IIR1_CH0_IB_FN_MOD2 Register Diagram

Table 45-381: SCB4_IIR1_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

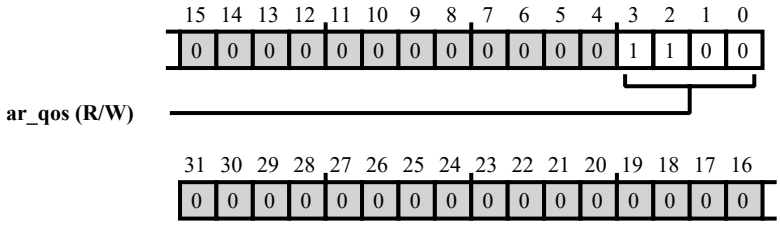


Figure 45-375: SCB4_IIR1_CH0_IB_READ_QOS Register Diagram

Table 45-382: SCB4_IIR1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

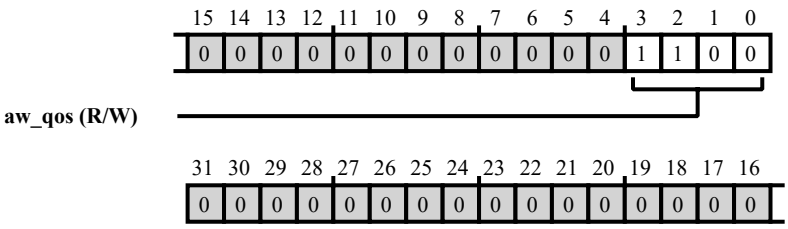


Figure 45-376: SCB4_IIR1_CH0_IB_WRITE_QOS Register Diagram

Table 45-383: SCB4_IIR1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

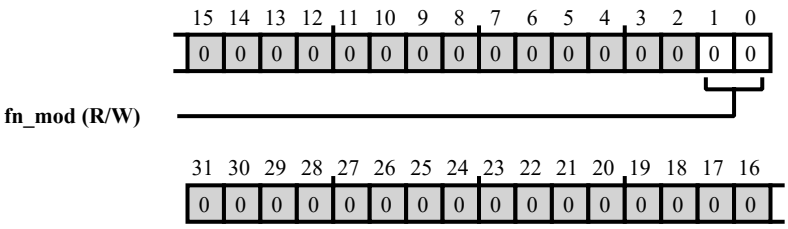


Figure 45-377: SCB4_IIR1_CH1_IB_FN_MOD Register Diagram

Table 45-384: SCB4_IIR1_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

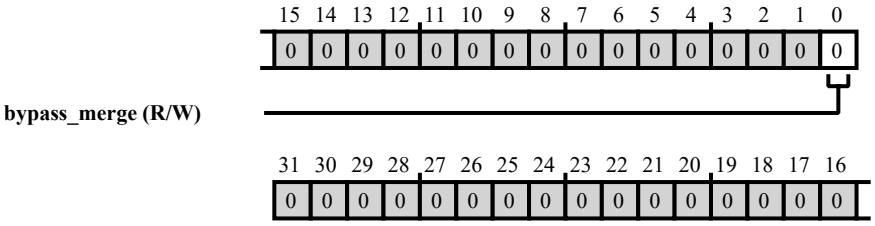


Figure 45-378: SCB4_IIR1_CH1_IB_FN_MOD2 Register Diagram

Table 45-385: SCB4_IIR1_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

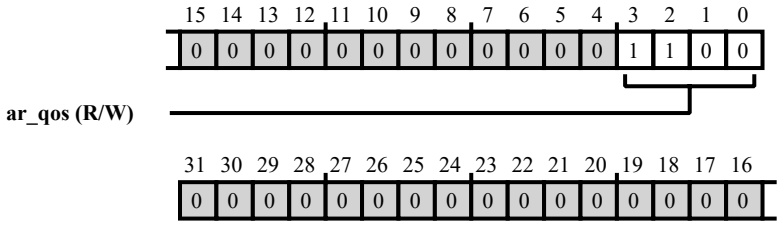


Figure 45-379: SCB4_IIR1_CH1_IB_READ_QOS Register Diagram

Table 45-386: SCB4_IIR1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

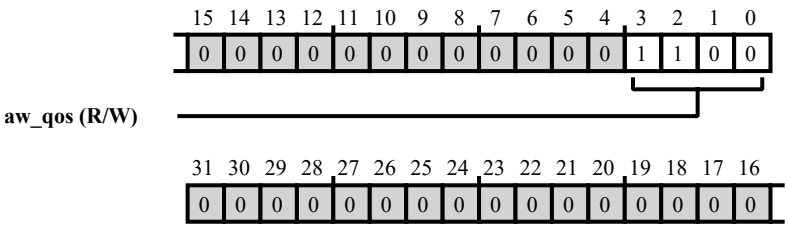


Figure 45-380: SCB4_IIR1_CH1_IB_WRITE_QOS Register Diagram

Table 45-387: SCB4_IIR1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

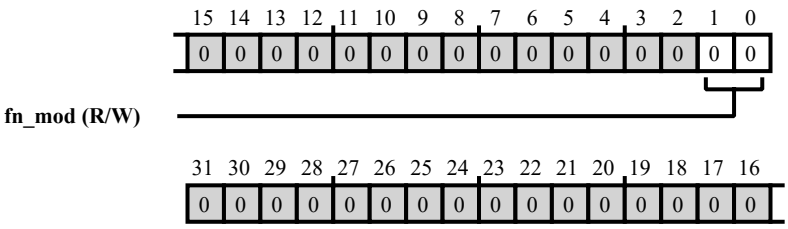


Figure 45-381: SCB4_IIR2_CH0_IB_FN_MOD Register Diagram

Table 45-388: SCB4_IIR2_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

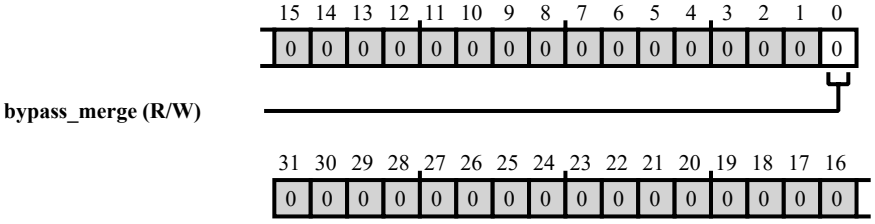


Figure 45-382: SCB4_IIR2_CH0_IB_FN_MOD2 Register Diagram

Table 45-389: SCB4_IIR2_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

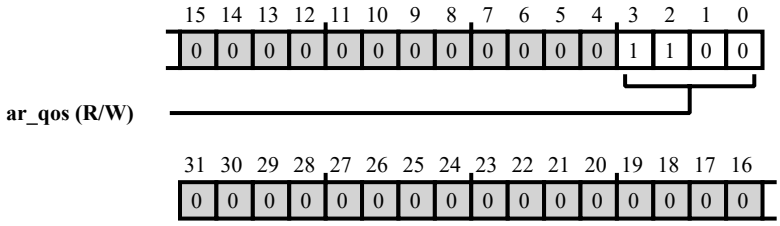


Figure 45-383: SCB4_IIR2_CH0_IB_READ_QOS Register Diagram

Table 45-390: SCB4_IIR2_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

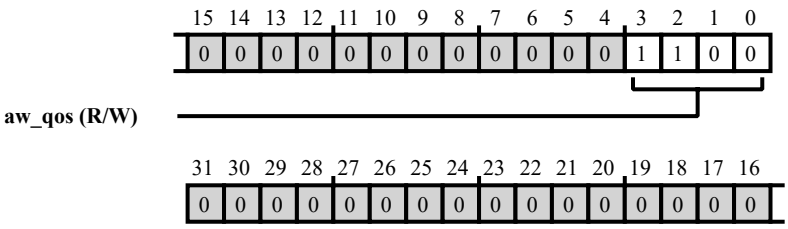


Figure 45-384: SCB4_IIR2_CH0_IB_WRITE_QOS Register Diagram

Table 45-391: SCB4_IIR2_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

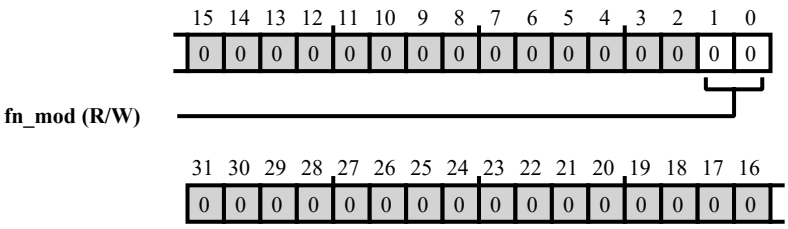


Figure 45-385: SCB4_IIR2_CH1_IB_FN_MOD Register Diagram

Table 45-392: SCB4_IIR2_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

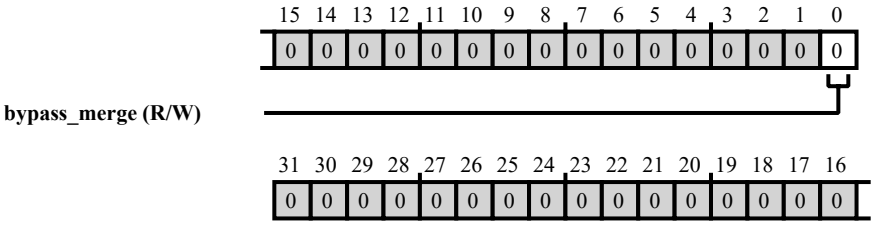


Figure 45-386: SCB4_IIR2_CH1_IB_FN_MOD2 Register Diagram

Table 45-393: SCB4_IIR2_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

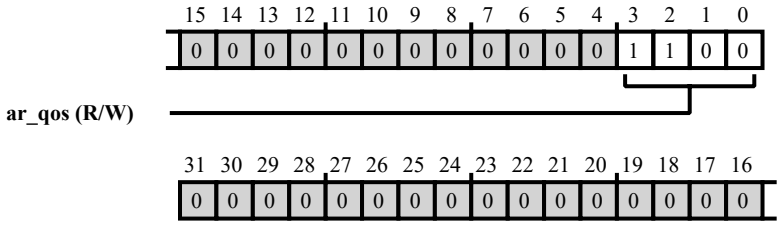


Figure 45-387: SCB4_IIR2_CH1_IB_READ_QOS Register Diagram

Table 45-394: SCB4_IIR2_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

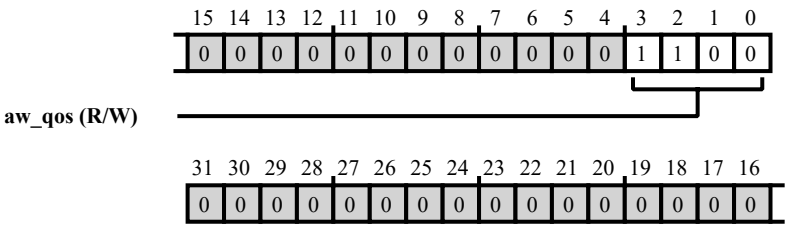


Figure 45-388: SCB4_IIR2_CH1_IB_WRITE_QOS Register Diagram

Table 45-395: SCB4_IIR2_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

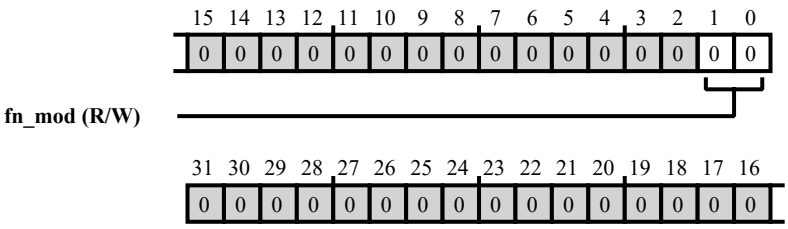


Figure 45-389: SCB4_IIR3_CH0_IB_FN_MOD Register Diagram

Table 45-396: SCB4_IIR3_CH0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

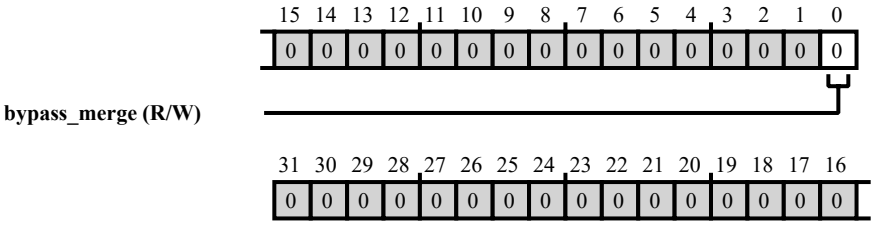


Figure 45-390: SCB4_IIR3_CH0_IB_FN_MOD2 Register Diagram

Table 45-397: SCB4_IIR3_CH0_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

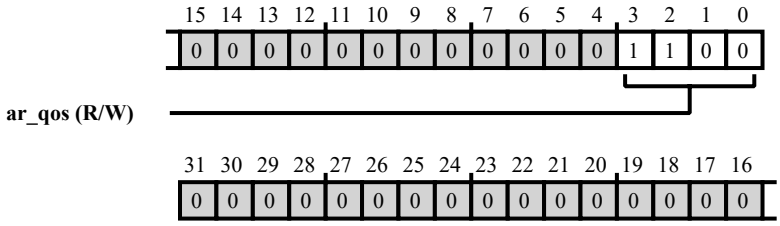


Figure 45-391: SCB4_IIR3_CH0_IB_READ_QOS Register Diagram

Table 45-398: SCB4_IIR3_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

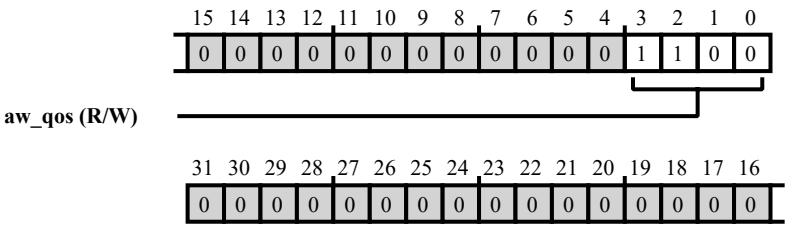


Figure 45-392: SCB4_IIR3_CH0_IB_WRITE_QOS Register Diagram

Table 45-399: SCB4_IIR3_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

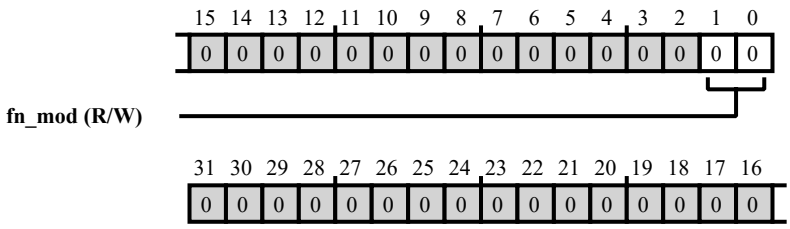


Figure 45-393: SCB4_IIR3_CH1_IB_FN_MOD Register Diagram

Table 45-400: SCB4_IIR3_CH1_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

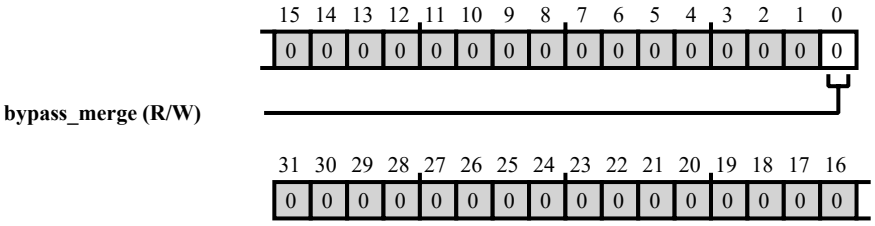


Figure 45-394: SCB4_IIR3_CH1_IB_FN_MOD2 Register Diagram

Table 45-401: SCB4_IIR3_CH1_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

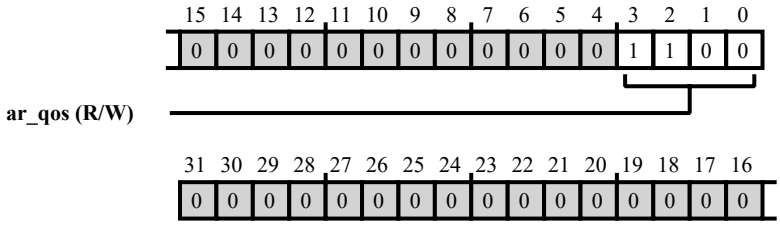


Figure 45-395: SCB4_IIR3_CH1_IB_READ_QOS Register Diagram

Table 45-402: SCB4_IIR3_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

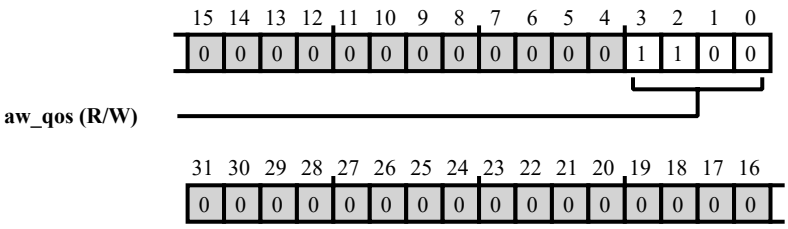


Figure 45-396: SCB4_IIR3_CH1_IB_WRITE_QOS Register Diagram

Table 45-403: SCB4_IIR3_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

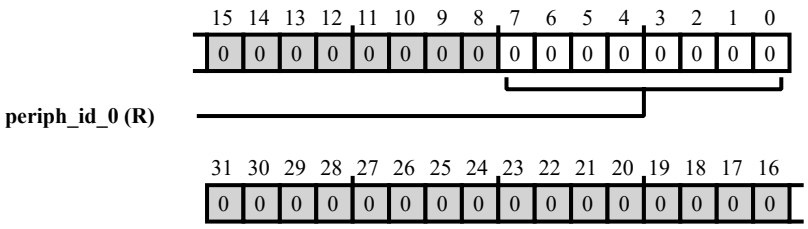


Figure 45-397: SCB4_PERIPH_ID_0 Register Diagram

Table 45-404: SCB4_PERIPH_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_0	.

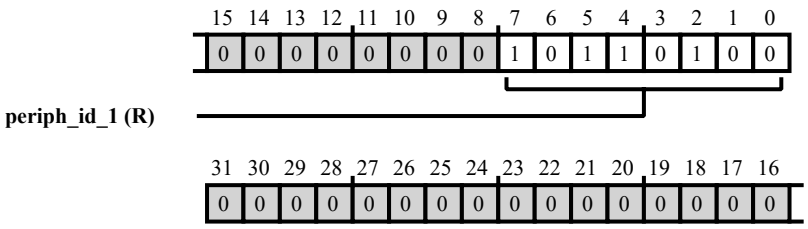


Figure 45-398: SCB4_PERIPH_ID_1 Register Diagram

Table 45-405: SCB4_PERIPH_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_1	.

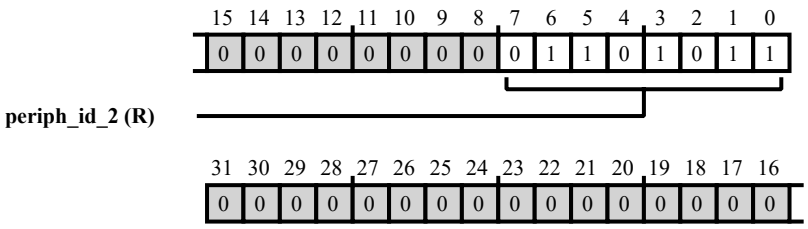


Figure 45-399: SCB4_PERIPH_ID_2 Register Diagram

Table 45-406: SCB4_PERIPH_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_2	.

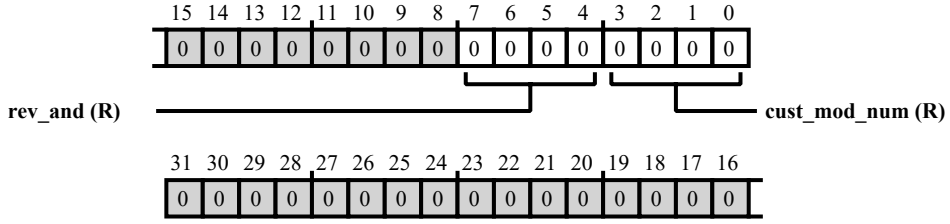


Figure 45-400: SCB4_PERIPH_ID_3 Register Diagram

Table 45-407: SCB4_PERIPH_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV_AND	.
3:0 (R/NW)	CUST_MOD_NUM	.

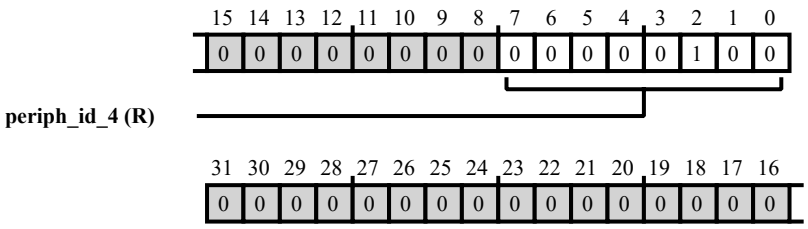


Figure 45-401: SCB4_PERIPH_ID_4 Register Diagram

Table 45-408: SCB4_PERIPH_ID_4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_4	.

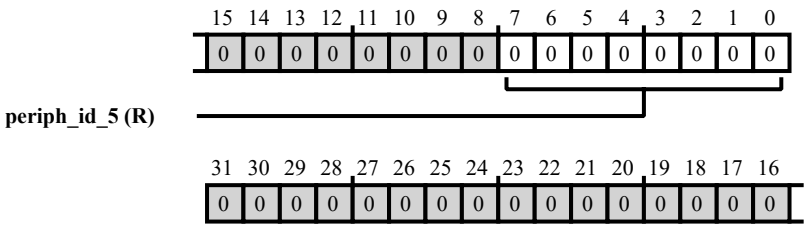


Figure 45-402: SCB4_PERIPH_ID_5 Register Diagram

Table 45-409: SCB4_PERIPH_ID_5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_5	.

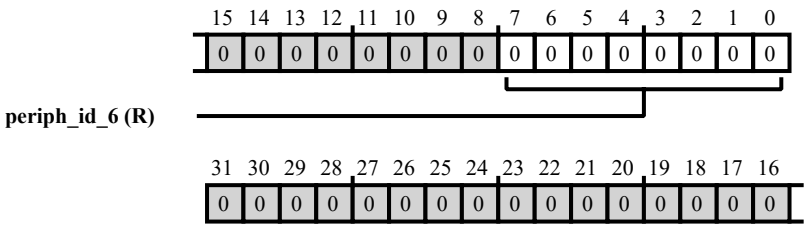


Figure 45-403: SCB4_PERIPH_ID_6 Register Diagram

Table 45-410: SCB4_PERIPH_ID_6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_6	.

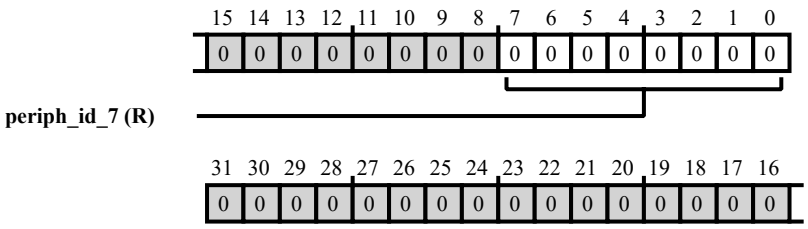


Figure 45-404: SCB4_PERIPH_ID_7 Register Diagram

Table 45-411: SCB4_PERIPH_ID_7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_7	.

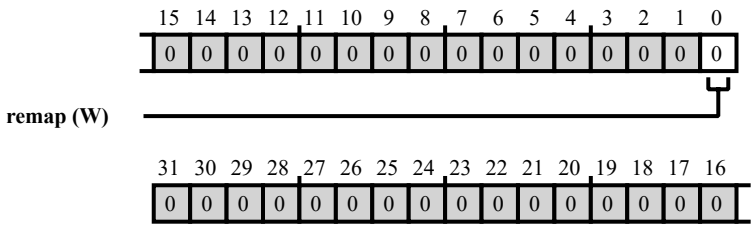


Figure 45-405: SCB4_REMAP Register Diagram

Table 45-412: SCB4_REMAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (RX/W)	REMAP	.

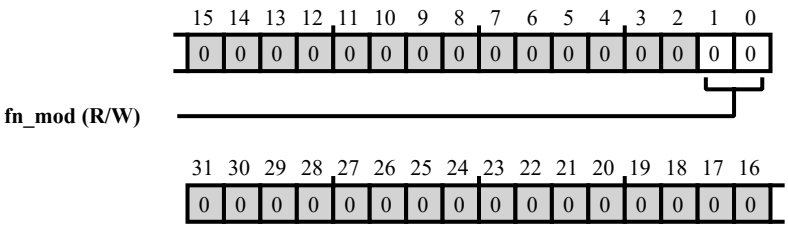


Figure 45-406: SCB4_SHARC_DPORT_FN_MOD Register Diagram

Table 45-413: SCB4_SHARC_DPORT_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

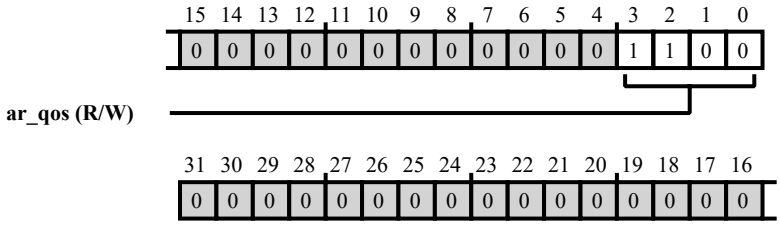


Figure 45-407: SCB4_SHARC_DPORT_READ_QOS Register Diagram

Table 45-414: SCB4_SHARC_DPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	.

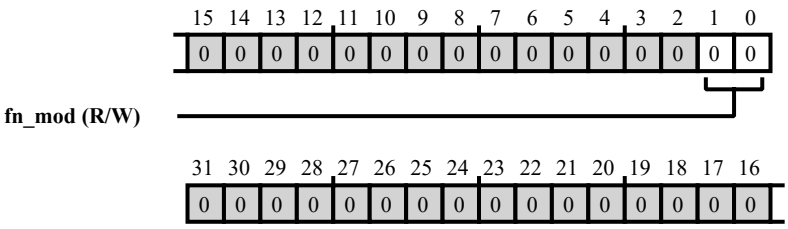


Figure 45-408: SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD Register Diagram

Table 45-415: SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

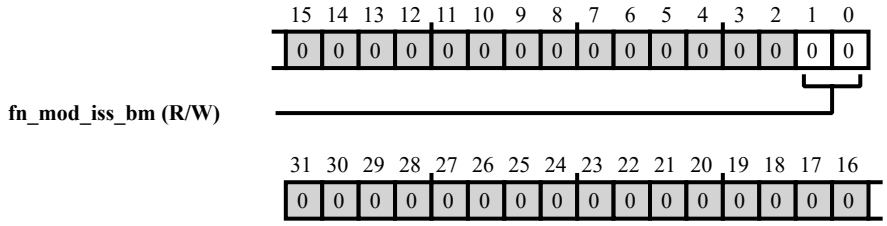


Figure 45-409: SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD_ISS_BM Register Diagram

Table 45-416: SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

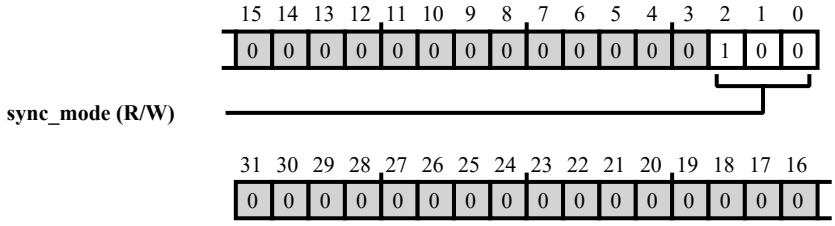


Figure 45-410: SCB4_SHARC_DPORT_TO_SF_L2_IB_SYNC_MODE Register Diagram

Table 45-417: SCB4_SHARC_DPORT_TO_SF_L2_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

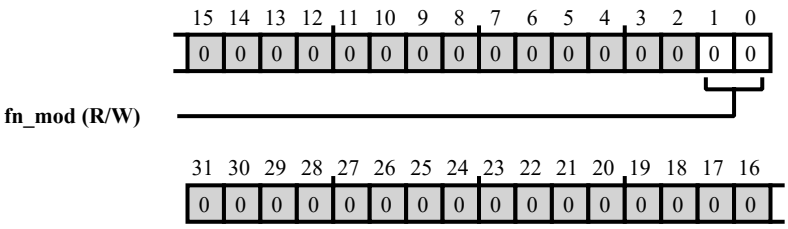


Figure 45-411: SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD Register Diagram

Table 45-418: SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

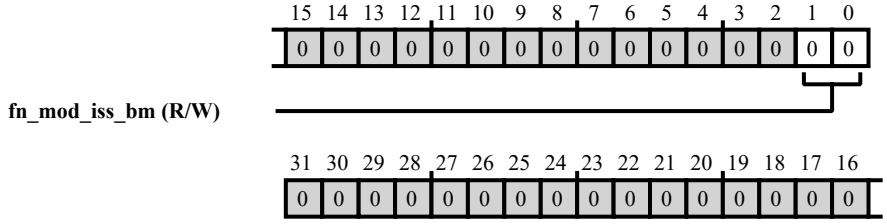


Figure 45-412: SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD_ISS_BM Register Diagram

Table 45-419: SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

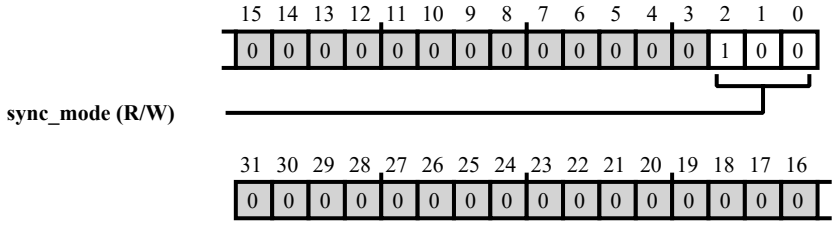


Figure 45-413: SCB4_SHARC_DPORT_TO_SF_IB_SYNC_MODE Register Diagram

Table 45-420: SCB4_SHARC_DPORT_TO_SF_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

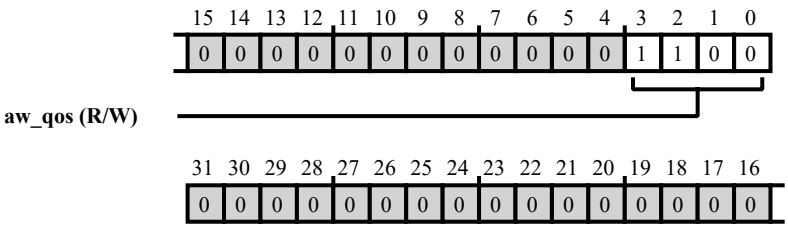


Figure 45-414: SCB4_SHARC_DPORT_WRITE_QOS Register Diagram

Table 45-421: SCB4_SHARC_DPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	.

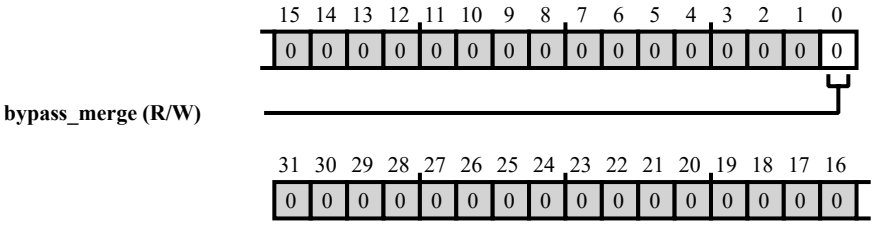


Figure 45-415: SCB4_SHARC_IPORT_IB_FN_MOD2 Register Diagram

Table 45-422: SCB4_SHARC_IPORT_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

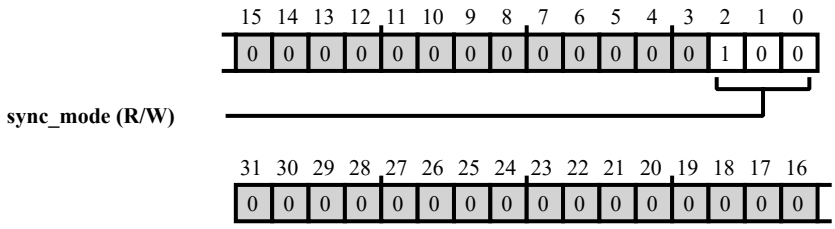


Figure 45-416: SCB4_SHARC_IPORT_IB_SYNC_MODE Register Diagram

Table 45-423: SCB4_SHARC_IPORT_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	.

ADSP-2184x INST1 Register Descriptions

Your module description, here. (INST1) contains the following registers.

Table 45-424: ADSP-2184x INST1 Register List

Name	Description
INST1_COMP_ID_0	
INST1_COMP_ID_1	
INST1_COMP_ID_2	
INST1_COMP_ID_3	
INST1_OSPI_S_IB_FN_MOD	
INST1_OSPI_S_IB_FN_MOD2	
INST1_OSPI_S_IB_FN_MOD_ISS_BM	
INST1_OSPI_S_IB_FN_MOD_LB	
INST1_PERIPH_ID_0	
INST1_PERIPH_ID_1	
INST1_PERIPH_ID_2	
INST1_PERIPH_ID_3	
INST1_PERIPH_ID_4	
INST1_PERIPH_ID_5	
INST1_PERIPH_ID_6	
INST1_PERIPH_ID_7	

Table 45-424: ADSP-2184x INST1 Register List (Continued)

Name	Description
INST1_REMAP	
INST1_SPI2_S_IB_FN_MOD	
INST1_SPI2_S_IB_FN_MOD2	
INST1_SPI2_S_IB_FN_MOD_ISS_BM	
INST1_SPIF_M_FN_MOD	

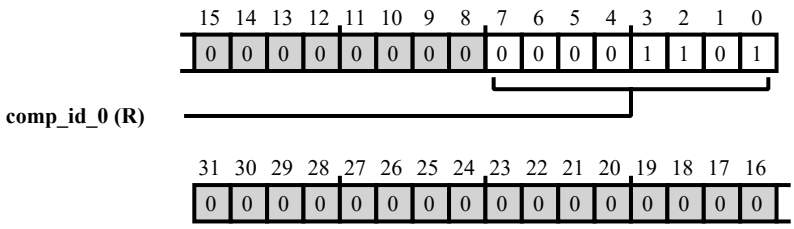


Figure 45-417: INST1_COMP_ID_0 Register Diagram

Table 45-425: INST1_COMP_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_0	.

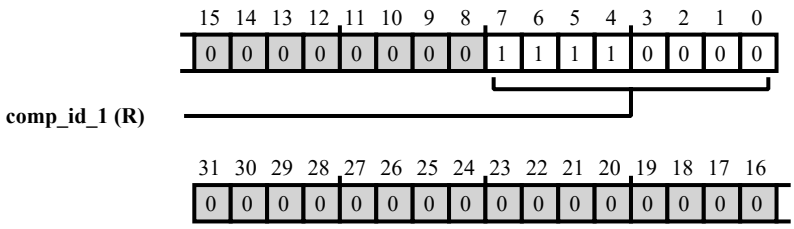


Figure 45-418: INST1_COMP_ID_1 Register Diagram

Table 45-426: INST1_COMP_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_1	.

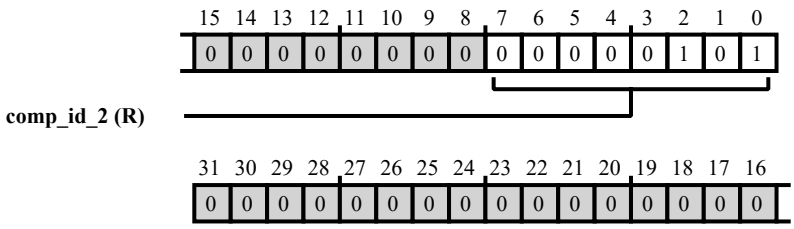


Figure 45-419: INST1_COMP_ID_2 Register Diagram

Table 45-427: INST1_COMP_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_2	.

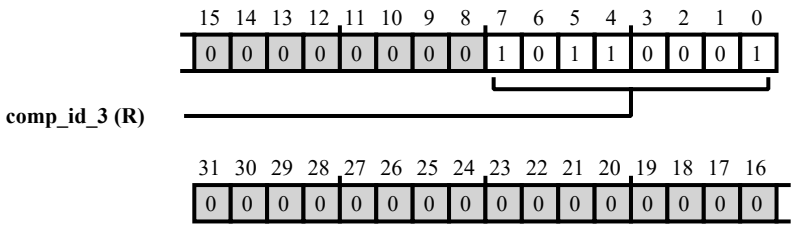


Figure 45-420: INST1_COMP_ID_3 Register Diagram

Table 45-428: INST1_COMP_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMP_ID_3	.

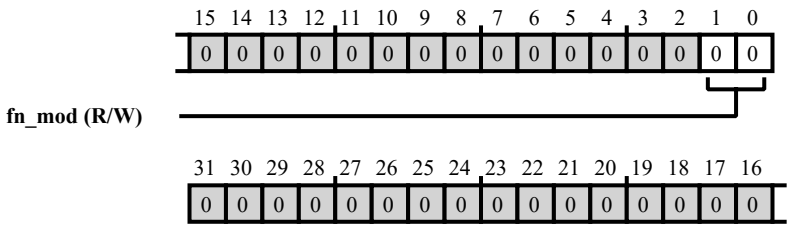


Figure 45-421: INST1_OSPI_S_IB_FN_MOD Register Diagram

Table 45-429: INST1_OSPI_S_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

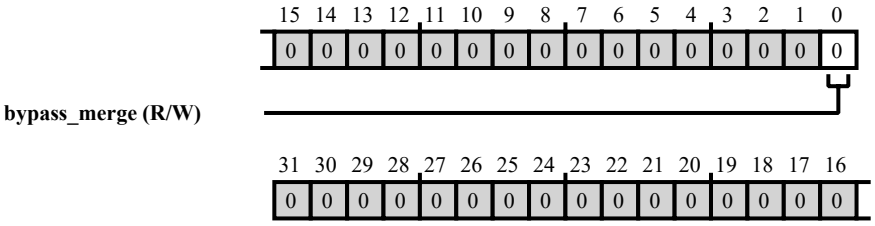


Figure 45-422: INST1_OSPI_S_IB_FN_MOD2 Register Diagram

Table 45-430: INST1_OSPI_S_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

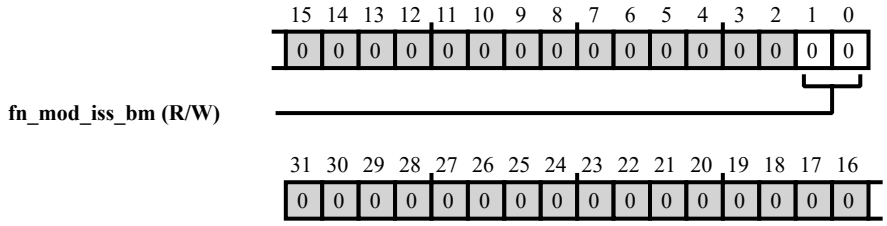


Figure 45-423: INST1_OSPI_S_IB_FN_MOD_ISS_BM Register Diagram

Table 45-431: INST1_OSPI_S_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

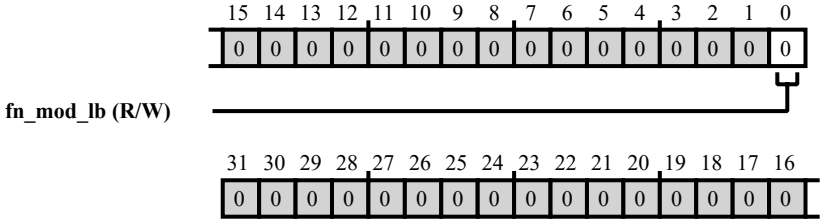


Figure 45-424: INST1_OSPI_S_IB_FN_MOD_LB Register Diagram

Table 45-432: INST1_OSPI_S_IB_FN_MOD_LB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	FN_MOD_LB	.

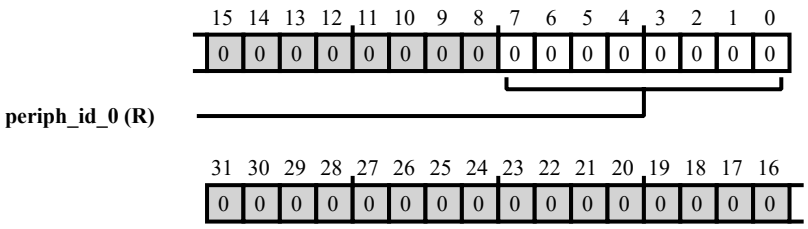


Figure 45-425: INST1_PERIPH_ID_0 Register Diagram

Table 45-433: INST1_PERIPH_ID_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_0	.

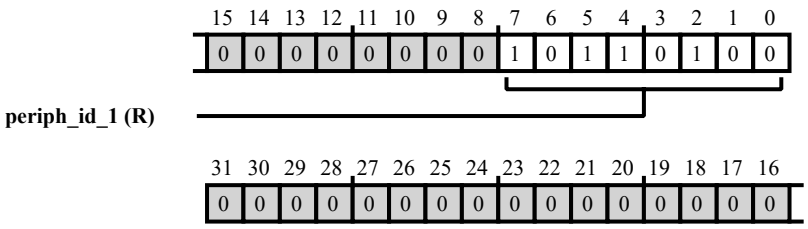


Figure 45-426: INST1_PERIPH_ID_1 Register Diagram

Table 45-434: INST1_PERIPH_ID_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_1	.

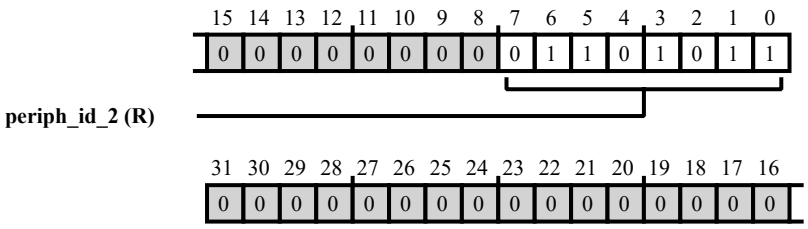


Figure 45-427: INST1_PERIPH_ID_2 Register Diagram

Table 45-435: INST1_PERIPH_ID_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_2	.

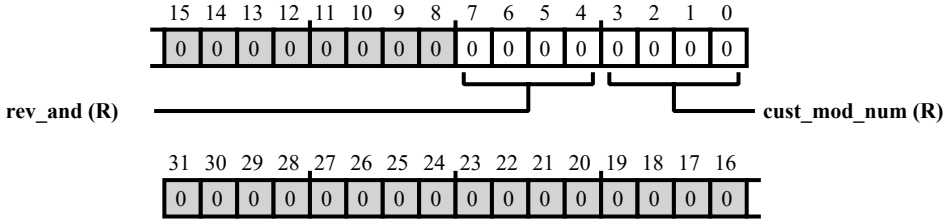


Figure 45-428: INST1_PERIPH_ID_3 Register Diagram

Table 45-436: INST1_PERIPH_ID_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV_AND	.
3:0 (R/NW)	CUST_MOD_NUM	.

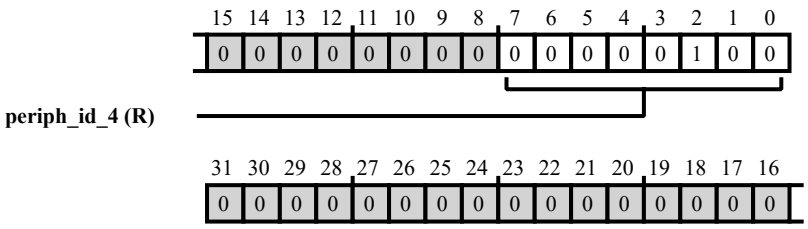


Figure 45-429: INST1_PERIPH_ID_4 Register Diagram

Table 45-437: INST1_PERIPH_ID_4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_4	.

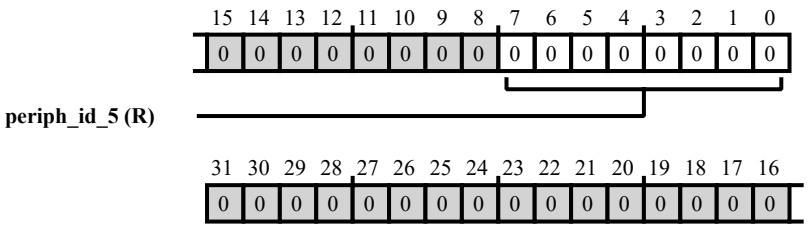


Figure 45-430: INST1_PERIPH_ID_5 Register Diagram

Table 45-438: INST1_PERIPH_ID_5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_5	.

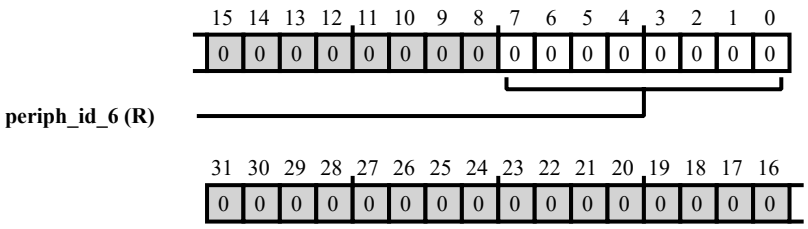


Figure 45-431: INST1_PERIPH_ID_6 Register Diagram

Table 45-439: INST1_PERIPH_ID_6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_6	.

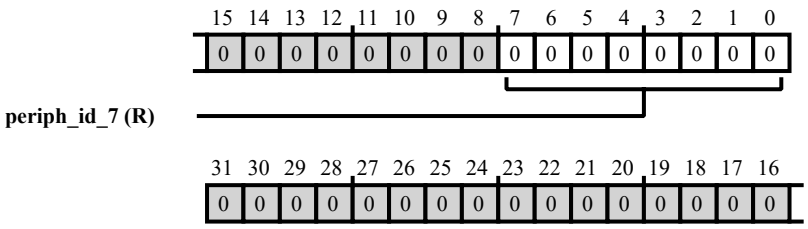


Figure 45-432: INST1_PERIPH_ID_7 Register Diagram

Table 45-440: INST1_PERIPH_ID_7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PERIPH_ID_7	.

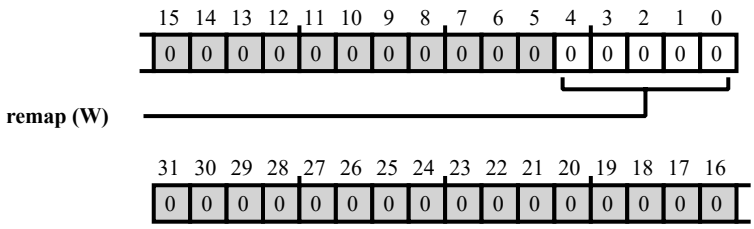


Figure 45-433: INST1_REMAP Register Diagram

Table 45-441: INST1_REMAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (RX/W)	REMAP	.

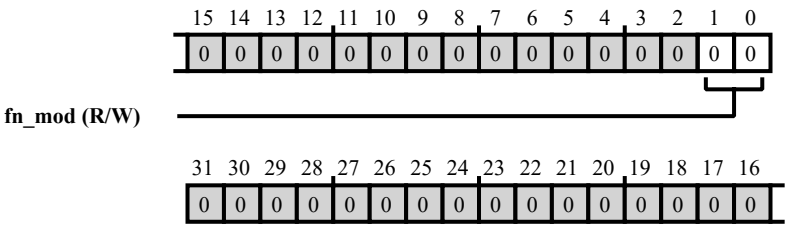


Figure 45-434: INST1_SPI2_S_IB_FN_MOD Register Diagram

Table 45-442: INST1_SPI2_S_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

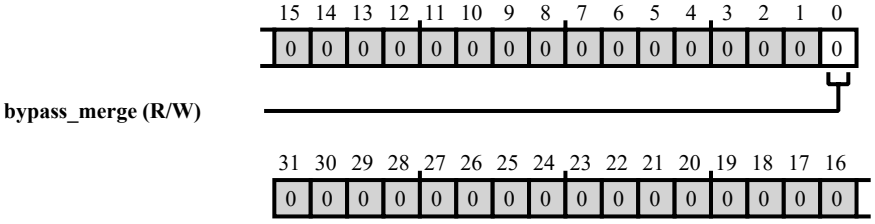


Figure 45-435: INST1_SPI2_S_IB_FN_MOD2 Register Diagram

Table 45-443: INST1_SPI2_S_IB_FN_MOD2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	BYPASS_MERGE	.

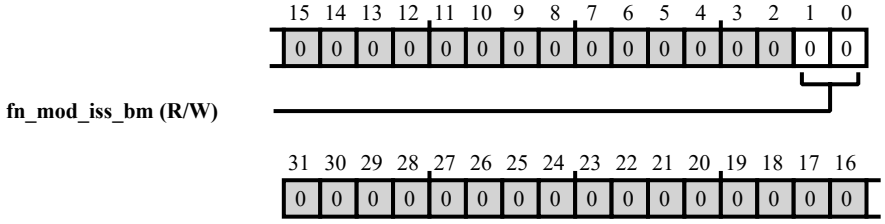


Figure 45-436: INST1_SPI2_S_IB_FN_MOD_ISS_BM Register Diagram

Table 45-444: INST1_SPI2_S_IB_FN_MOD_ISS_BM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD_ISS_BM	.

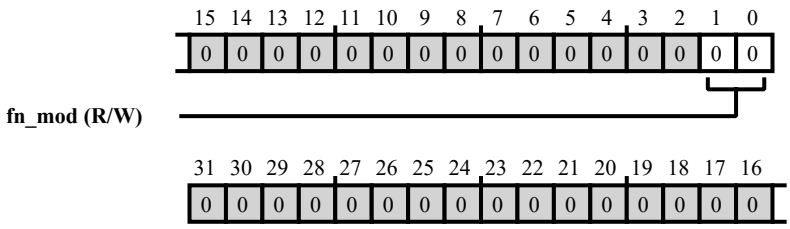


Figure 45-437: INST1_SPIF_M_FN_MOD Register Diagram

Table 45-445: INST1_SPIF_M_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	.

46 System Watchpoint Unit (SWU)

The system watchpoint unit (SWU) is a single module used for transaction monitoring. The SWU is attached to each system completer through the system crossbar interface and provides ports for all address channel signals for the system crossbar. The SWU does not have ports for the read/write data channel signals or the low-power interface signals.

Each SWU contains four match groups of registers with associated hardware. These four SWU match groups operate independently, but share common event (interrupt and trigger) outputs. Each match group can monitor either the write or read address channel and can operate in either watchpoint mode or bandwidth mode.

In this chapter, the naming convention for registers omit the numeric reference to any of the SWUs or it's groups. For example, SWU_GCTL represents registers SWU1_GCTL, SWU2_GCTL, and so on. Similarly, SWU_CUR[n] represents SWU1_CUR0/CUR1/CUR2/CUR3, SWU2_CUR0/CUR1/CUR2/CUR3 and so on.

SWU Features

The system watchpoint unit has the following features.

- Four independent match groups for each SWU
- Each match group can operate in either bandwidth mode or watchpoint mode

SWU Functional Description

This section describes the function of the SWU match block, interface block, and MMR block.

NOTE: In the *SWU Interrupt List* table, there are three SWUs, SWU1 (CL2_0), SWU3 (CL2_1) and SWU5 (CL2_2) for L2 Memory Core access and two SWUs, SWU2 (DL2_0) and SWU4 (DL2_1) for L2 Memory DMA access. CL2_2, DL2_0 and DL2_1 have access to the entire L2 range. CL2_0 has access to first 1MB and CL2_1 has access to second 1MB of L2.

ADSP-2184x SWU Register List

The System Watchpoint Unit (SWU) provides debug and development support through flexible transaction level and bandwidth monitoring and associated event triggering. The SWU can generate events based on monitoring transactions at the system targets through watchpoint-match groups. The SWU also provides watchpoint event

status reporting, a global lock, and processor reset capability. A set of registers governs SWU operations. For more information on SWU functionality, see the SWU register descriptions.

Table 46-1: ADSP-2184x SWU Register List

Name	Description
SWU_CNT[n]	Count Register n
SWU_CTL[n]	Control Register n
SWU_CUR[n]	Current Register n
SWU_GCTL	Global Control Register
SWU_GSTAT	Global Status Register
SWU_HIST[n]	Bandwidth History Register n
SWU_ID[n]	ID Register n
SWU_LA[n]	Lower Address Register n
SWU_TARG[n]	Target Register n
SWU_UA[n]	Upper Address Register n

ADSP-2184x SWU Interrupt List

Table 46-2: ADSP-2184x SWU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
307	SWU0_EVT	SWU0 Event SMC	None	
308	SWU1_EVT	SWU1 Event L2 Memory DMA Port 0	None	
309	SWU2_EVT	SWU2 Event L2 Memory Core Port 0	None	
310	SWU7_EVT	SWU7 Event SHARC0 Target Port 1	None	
311	SWU8_EVT	SWU8 Event SHARC0 Target Port 2	None	
312	SWU11_EVT	SWU11 Event SMMR	None	
313	SWU12_EVT	SWU12 Event SPI2/XSPI	None	
314	SWU13_EVT	SWU13 Event DMC0	None	
315	SWU3_EVT	SWU3 Event CL2_1	None	
316	SWU4_EVT	SWU4 Event DL2_1	None	
317	SWU5_EVT	SWU5 Event CL2_2	None	

ADSP-2184x SWU Trigger List

Table 46-3: ADSP-2184x SWU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
145		Reserved	
146	SWU1_EVT	SWU1 Event	None
147	SWU2_EVT	SWU2 Event	None
148	SWU7_EVT	SWU7 Event	None
149	SWU8_EVT	SWU8 Event	None
150	SWU11_EVT	SWU11 Event	None
151	SWU12_EVT	SWU12 Event	None
152	SWU13_EVT	SWU13 Event	None
153	SWU3_EVT	SWU3 Event	None
154	SWU4_EVT	SWU4 Event	None
155	SWU5_EVT	SWU5 Event	None
156		Reserved	
157	SWU1_DBG	SWU1 Debug	Edge
158	SWU2_DBG	SWU2 Debug	Edge
159	SWU7_DBG	SWU7 Debug	Edge
160	SWU8_DBG	SWU8 Debug	Edge
161	SWU11_DBG	SWU11 Debug	Edge
162	SWU12_DBG	SWU12 Debug	Edge
163	SWU13_DBG	SWU13 Debug	Edge
164	SWU3_DBG	SWU3 Debug	Edge
165	SWU4_DBG	SWU4 Debug	Edge
166	SWU5_DBG	SWU5 Debug	Edge

Table 46-4: ADSP-2184x SWU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
177		Reserved	
178	SWU1_EN	SWU1 Enable	Pulse
179	SWU2_EN	SWU2 Enable	Pulse
180	SWU7_EN	SWU7 Enable	Pulse
181	SWU8_EN	SWU8 Enable	Pulse

Table 46-4: ADSP-2184x SWU Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
182	SWU11_EN	SWU11 Enable	Pulse
183	SWU12_EN	SWU12 Enable	Pulse
184	SWU13_EN	SWU13 Enable	Pulse
185	SWU3_EN	SWU3 Enable Event CL2_1	Pulse
186	SWU4_EN	SWU4 Enable Event DL2_1	Pulse
187	SWU5_EN	SWU5 Enable Event CL2_2	Pulse

SWU Definitions

The following definitions are helpful when using the SWU module.

Watchpoint Mode

Mode in which transactions are recognized on an exact match. Actions can be configured to be taken after a specified number of matches have occurred.

Bandwidth Mode

Mode in which transactions are recognized and counted inside sampling window.

SWU Architectural Concepts

The information in this section provides basic module design concepts.

SWU-to-SCB Interface

The SWU system crossbar interface block latches all transactions on the system crossbar read and write address channels when the SWU_GCTL.EN register enable bit is set.

SWU Block Diagram

The *System Watchpoint Unit Top-Level Block Diagram* figure shows the SWU block diagram.

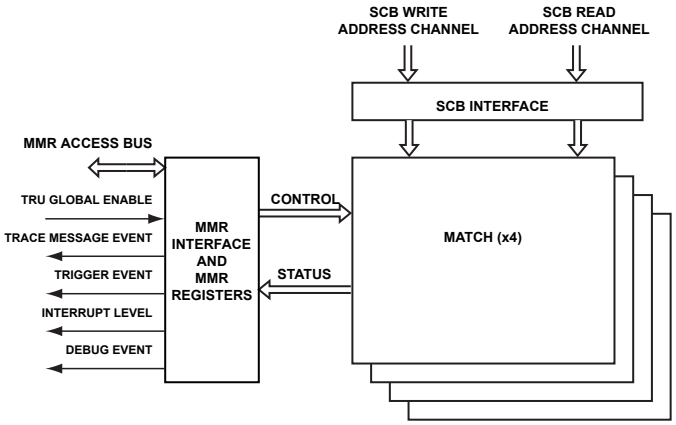


Figure 46-1: System Watchpoint Unit Top-Level Block Diagram

SCB Interface Block

The SWU system crossbar (SCB) latches all transactions on the SCB read and write address channels when the `SWU_GCTL.EN` bit is set.

MMR Interface Block

The SWU MMR block contains the peripheral bus interface and the SWU MMR registers. It also merges all interrupt requests and events from each match block into common outputs.

SWU Operating Modes

There are two operating modes supported by the SWU: bandwidth mode and watchpoint mode.

Bandwidth Mode

In bandwidth mode, the SWU module counts transactions which match the properties specified in the `SWU_CTL[n]` register during a sampling window determined by the respective `SWU_CNT[n]` register. At the end of the sampling window, the SWU stores results in the `SWU_HIST[n]` register. If the sampled bandwidth falls outside a programmed range, then the programmed action occurs.

Watchpoint Mode

In watchpoint mode, if the `SWU_CTL[n].CNTEN` bit is set, the SWU module decrements the `SWU_CUR[n]` register for each match, until it equals zero, at which point any programmed actions occur. The `SWU_CUR[n]` register is then reloaded from the `SWU_CNT[n]` register (if the `SWU_CTL[n].CNTRPTEN` bit is set), and the cycle repeats. If the `SWU_CTL[n].CNTRPTEN` bit is not set, any programmed actions happen on every match.

Match Block

There are four match blocks for each SWU. Each SWU match block can monitor either the read or write address channel, selected by the `SWU_CTL[n].DIR` bit. The SWU match block can operate in either watchpoint or bandwidth mode, as selected by the `SWU_CTL[n].BWEN` bit.

Scaling

Scaling allows the SWU to count more transactions by scaling the number of transactions and the number of clock cycles in bandwidth window (CNTn register) by 10,100 or 1000. This functionality is applicable only in bandwidth mode (SWU_CTL[n].BWEN==1).

Consider a case where the SWU_TARG[n].BWMAX bit field is programmed to 2. In the absence of any scaling, bandwidth overflow occurs when the SWU_CUR[n].CURBW bit field value > the SWU_TARG[n].BWMAX bit field value (2). With scaling set to 1:100 and after 275 transactions, the SWU_CUR[n].CURBW value is still 2 and equal to SWU_TARG[n].BWMAX. This event triggers a bandwidth overflow as the actual number of transactions is greater than 200 (2 × 100). The code can be rerun with a smaller scaling selected to analyze the cause for the overflow.

The counter increments after every group of N transactions with scaling enabled, where N is either 10,100 or 1000. (For N = 10, 0–9 transactions == 0 scaled transaction, 10–19 transactions == 1 scaled transaction, 20–29 transactions == 2 scaled transactions, and so on).

Fractional counts with scaling enabled are discarded and not rolled over from one bandwidth window to the next. For example, consider a case where scaling by 1000 is configured and the first window has 1200 transactions. The second window has 2800 transactions. The bandwidth for the first window is read as 1 and the second as 2. The 200 transactions from the first window do not get carried forward to the next window.

Do not use scaling by 10 with the SWU_CTL[n].BLENINC bit enabled if any of the requesters accessing the completer can launch a transaction of burst length 16.

SWU Event Control

The SWU can generate the following events when a match occurs and when the event is enabled by configuring the proper bits in the control register.

1. Trace Message
2. Trigger
3. Interrupt request
4. Debug

SWU Interrupts

All interrupt requests and events from each match block are merged into common outputs.

SWU Status and Errors

SWU status and errors are reported in the SWU_GSTAT register. The SWU records an address error when a write or read attempt is made to the MMR address space of the SWU and the register does not exist. This error is the only one the SWU records. The register contains bits that perform the following functions.

- Indicate whether a particular match group sampled a transaction that is below a minimum target or above a maximum target in bandwidth mode.
- Indicate whether a watchpoint match occurred for each match group.
- Indicate whether an interrupt request was triggered due to a match event from one of the match groups.

Triggers

The SWU can be either a trigger generator or a trigger receiver depending on the trigger routing unit (TRU) configuration. As a trigger generator, programs must set the `SWU_CTL[n].TRGEN` bit so that when a match condition is met, a trigger event is generated. Each SWU in the system can also be a trigger receiver when mapped as one in the TRU.

When the SWU is a receiver, a trigger event activates the SWU by automatically setting the `SWU_GCTL.EN` bit. Since the SWU can be automatically enabled through a trigger event, programs must pre-configure the SWU before enabling the TRU. Furthermore, although a trigger event can enable the SWU as a generator, to disable the SWU, programs must manually clear the `SWU_GCTL.EN` bit.

SWU Programming Model

Program the appropriate registers to use the SWU. Each control register configures aspects such as:

- The direction of monitoring (reads or writes)
- Whether SWU uses bandwidth mode or watchpoint mode
- The setup of events that are triggered when a condition is met while monitoring using the SWU

Configure supplemental registers such as the lower (`SWU_LA[n]`) and upper (`SWU_UA[n]`) address boundaries before enabling the SWU.

Once the SWU has been enabled and the monitoring conditions are met, events are generated when configured.

The global status register (`SWU_GSTAT`) can be read to observe the status of the units.

The ***SWU Logical Flow*** diagram shows the logical program flow of the SWU.

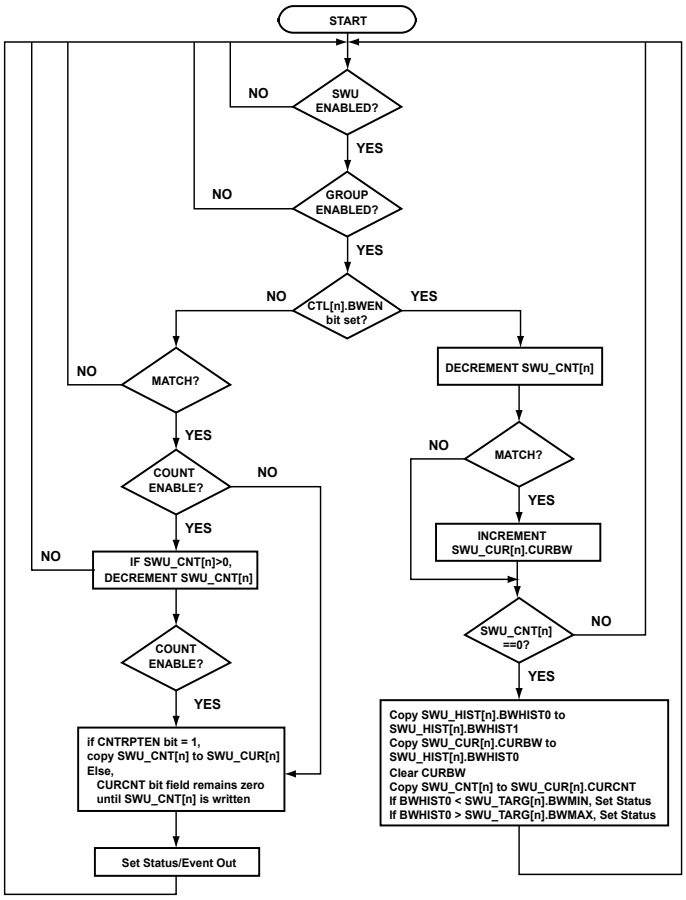


Figure 46-2: SWU Logical Flow

SWU Mode Configuration

The following sections show the steps for configuring SWU bandwidth mode and watchpoint mode.

Configuring the SWU for Bandwidth Mode

In bandwidth mode, the SWU counts transactions which match during a sampling window. At the end of the sampling window, the SWU stores the results. An action can be taken if the sampled bandwidth goes above or falls below a programmed range.

1. Configure the `SWU_CTL[n].DIR` bit to test the match on writes or reads.
2. Configure the `SWU_CTL[n].ACMPM` bits to address comparisons, exact match, matches inside a range or matches outside a range.
3. If ID comparison is desired, set the `SWU_CTL[n].IDCMPEN` bit.
4. Set the `SWU_CTL[n].BLENINC` bit to increment by burst length or clear it to increment by 1.
5. Configure the `SWU_CTL[n].MAXACT` and `SWU_CTL[n].MINACT` bits to enable actions taken when the bandwidth goes above the maximum, or falls below the minimum, respectively.

6. Set the SWU_CTL[n].BWEN=1 to enable bandwidth mode.
7. Program the lower address register, SWU_LA[n], and upper address register, SWU_UA[n], to define the memory range for comparison.
8. If ID comparison is enabled, program the ID register, SWU_ID[n].
9. Program the count register, SWU_CNT[n], with the number of clock cycles for which the SWU counts the number of matches.
10. If the SWU is set to respond when the bandwidth measurement underflows or overflows, program the min and max values into the SWU_TARG[n] register.
11. Enable the SWU

The SWU counts the number of matches in a pre-defined number of clock cycles as programmed. As an option, it can define lower and upper limits. If the matches fall outside the limits, an action can be taken.

Configuring the SWU for Watchpoint Mode

In watchpoint mode, the SWU can trigger a programmed action after every match or after a number of matches. This sequence can be automatically reset.

1. Set the SWU_CTL[n].DIR bit to test the match on writes or reads.
2. Configure the SWU_CTL[n].ACMPM bits for address comparisons, exact match, matches inside a range or matches outside a range.
3. If ID comparison is desired, set the SWU_CTL[n].IDCMPEN.
4. Set the SWU_CTL[n].CNTEN bit to enable the events to be triggered when the count decrements to zero.
5. If needed, set the SWU_CTL[n].CNTRPTEN bit to automatically reload the counter after it has decremented to zero to start another match sequence.
6. Clear the SWU_CTL[n].BWEN = 0 to configure watchpoint mode.
7. Configure the lower address register, SWU_LA[n], and upper address register, SWU_UA[n], to define the memory range for comparison.
8. If ID comparison is enabled, configure the ID register, SWU_ID[n].
9. Configure the count register, SWU_CNT[n], to determine how many matches occur before the watchpoint group responds.
10. Enable the SWU.

The SWU detects and counts down the number of match occurrences. When the counter expires, an action is taken.

ADSP-2184x SWU Register Descriptions

System Watchpoint Unit (SWU) contains the following registers.

Table 46-5: ADSP-2184x SWU Register List

Name	Description
SWU_CNT[n]	Count Register n
SWU_CTL[n]	Control Register n
SWU_CUR[n]	Current Register n
SWU_GCTL	Global Control Register
SWU_GSTAT	Global Status Register
SWU_HIST[n]	Bandwidth History Register n
SWU_ID[n]	ID Register n
SWU_LA[n]	Lower Address Register n
SWU_TARG[n]	Target Register n
SWU_UA[n]	Upper Address Register n

Count Register n

The SWU count registers (SWU_CNT[n]) contain a 16-bit count field (SWU_CNT[n].COUNT) whose usage differs depending on the mode of the watchpoint group. In bandwidth mode, the SWU_CNT[n].COUNT field value defines the number of clock cycles in a bandwidth period. In watchpoint mode, when the cycle count is enabled, the SWU_CNT[n].COUNT field value determines how many matches occur before the watchpoint group takes action.

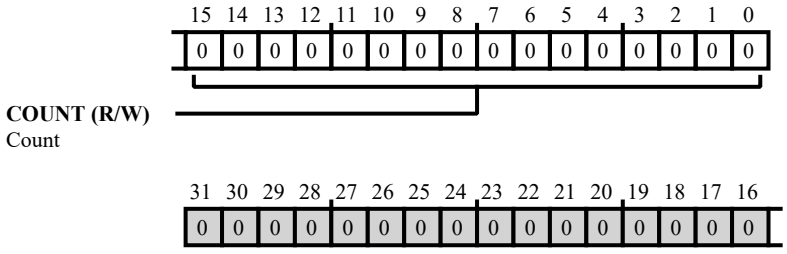


Figure 46-3: SWU_CNT[n] Register Diagram

Table 46-6: SWU_CNT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	COUNT	Count. The SWU_CNT[n].COUNT field value defines the number of clock cycles in a bandwidth period. In watchpoint mode, when the cycle count is enabled, the SWU_CNT[n].COUNT field value determines how many matches occur before the watchpoint group takes action.

Control Register n

The SWU control registers (SWU_CTL[n]) contain watchpoint attribute controls for all four watchpoint groups. These controls include enabling watchpoints, selecting the transaction direction for match, selecting address comparison mode, enabling ID comparison, enabling security comparison, enabling cycle count, enabling count repeat, enabling debug events, enabling interrupts, enabling triggers, enabling trace messages, enabling bandwidth mode, selecting the burst length increment, and enabling bandwidth underflow and overflow detection.

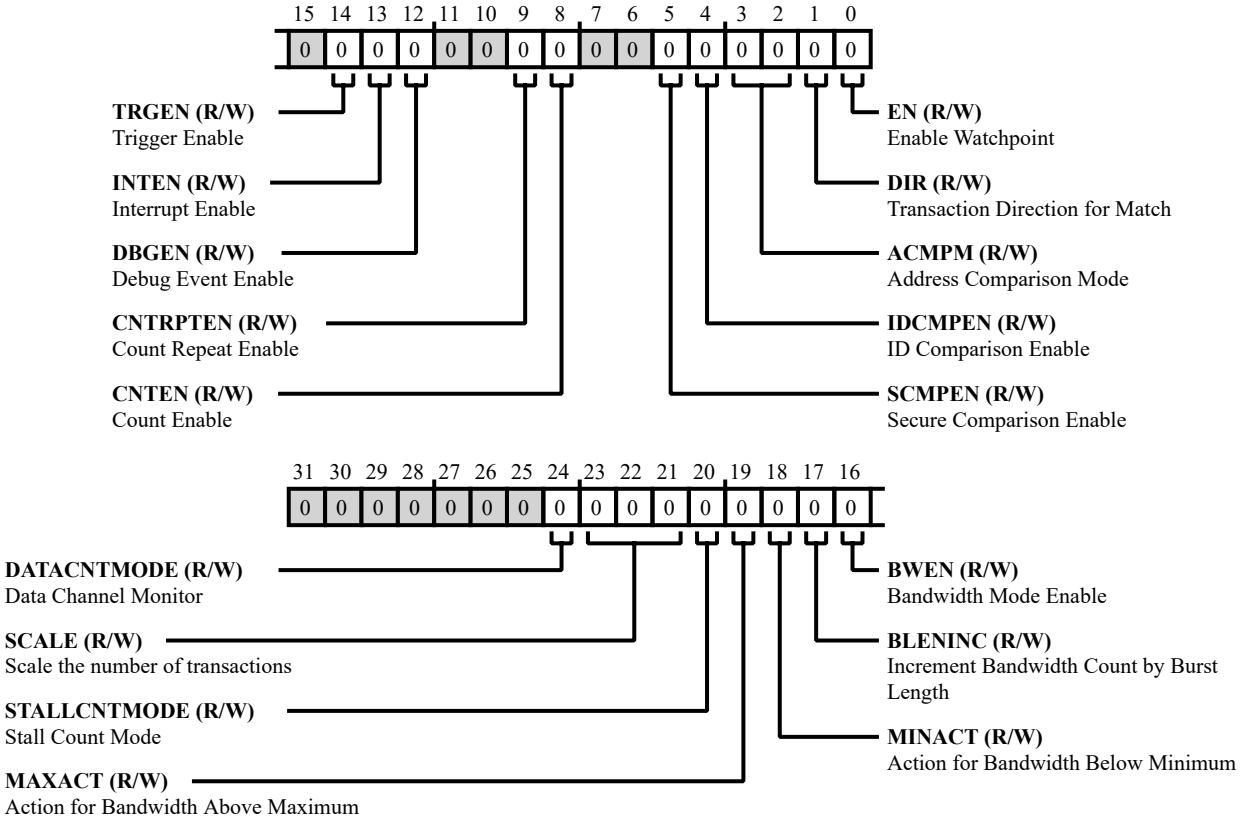


Figure 46-4: SWU_CTL[n] Register Diagram

Table 46-7: SWU_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	DATACNTMODE	Data Channel Monitor. The SWU_CTL[n].DATACNTMODE bit determines whether an address channel or a data channel is monitored. Note that in data channel only ID,READY and VALID signals are monitored and hence other comparisons (Address, Lock, Secure) will be ignored even if enabled. SWU_CTL[n].STALLCNTMODE and SWU_CTL[n].DIR can be used in conjunction with this bit	
		0	Monitor address channel
		1	Monitor data channel
23:21 (R/W)	SCALE	Scale the number of transactions. The SWU_CTL[n].SCALE bit field allows a program to count more transactions by scaling the number of transactions and also the number of clock cycles in the bandwidth window (SWU_CNT[n] register) by 10,100 or 1000. This is applicable only in bandwidth mode (SWU_CTL[n].BWEN==1).	
		0	No scaling
		1	1:10
		2	1:100
		3	Reserved
		4	1:1000
		5-7	Reserved
20 (R/W)	STALLCNTMODE	Stall Count Mode. The SWU_CTL[n].STALLCNTMODE bit determines whether the number of stalls are counted or whether the number of transactions are counted. This feature is only valid in bandwidth mode.	
		0	Count number of transactions
		1	Count number of stalls
19 (R/W)	MAXACT	Action for Bandwidth Above Maximum. Each SWU_CTL[n].MAXACT bit determines whether a watchpoint group takes action on bandwidth overflow. This feature is only valid in bandwidth mode.	
		0	No Action
		1	Take Action
18 (R/W)	MINACT	Action for Bandwidth Below Minimum. Each SWU_CTL[n].MINACT bit determines whether a watchpoint group takes action on bandwidth underflow. This feature is only valid in bandwidth mode.	
		0	No Action
		1	Take Action

Table 46-7: SWU_CTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	BLENINC	Increment Bandwidth Count by Burst Length. Each SWU_CTL[n].BLENINC bit controls how a watchpoint group's bandwidth count is incremented in the SWU_CUR[n] register's SWU_CUR[n].CURBW field. If the SWU_CTL[n].BLENINC bit is cleared (= 0), the SWU increments the bandwidth count by 1 for each matching transaction. If the SWU_CTL[n].BLENINC bit is set (=1), the SWU increments the bandwidth count by the burst length of the transaction for each matching transaction. This feature is only valid for bandwidth mode (SWU_CTL[n].BWEN bit == 1). Note that if the address range match is enabled (SWU_CTL[n].ACMPM bits) and if any address of a burst falls within the address range, the SWU_CUR[n].CURBW field is incremented by the burst length even if some of the burst address fall outside of the range. Also, note that the burst size of the transaction is not included in the increment, only the burst length of the transaction. This increment operation provides an approximate (not exact) number of bus cycles consumed during the bandwidth.	
		0	Increment by 1
		1	Burst Length Increment for Bandwidth Count
16 (R/W)	BWEN	Bandwidth Mode Enable. Each SWU_CTL[n].BWEN bit controls whether a watchpoint group operates in watchpoint mode or bandwidth mode. In watchpoint mode, the SWU_CTL[n].CNTEN and (optionally) SWU_CTL[n].CNTRPTEN registers control usage of the cycle count for watchpoint group operations. In bandwidth mode, the SWU_CTL[n].BLENINC, SWU_TARG[n], and SWU_HIST[n] registers control usage of watchpoint matches for watchpoint group operations.	
		0	Watchpoint Mode
		1	Bandwidth Mode
14 (R/W)	TRGEN	Trigger Enable. Each SWU_CTL[n].TRGEN bit controls whether a match for a watchpoint group generates a trigger event. This feature is valid in both bandwidth and watchpoint modes.	
		0	Disable
		1	Enable
13 (R/W)	INTEN	Interrupt Enable. Each SWU_CTL[n].INTEN bit controls whether a match for a watchpoint group generates an interrupt. This feature is valid in both bandwidth and watchpoint modes.	
		0	Disable
		1	Enable

Table 46-7: SWU_CTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	DBGEN	Debug Event Enable. Each SWU_CTL[n].DBGEN bit controls debug event comparison for a watchpoint group, permitting matches based on debug status.	
		0	Disable
		1	Enable
9 (R/W)	CNTRPTEN	Count Repeat Enable. Each SWU_CTL[n].CNTRPTEN bit controls whether the watchpoint group's cycle count is reloaded and repeated after cycle countdown. If the SWU_CTL[n] register's SWU_CTL[n].CNTRPTEN bit is set, the SWU_CUR[n] register's SWU_CUR[n].CURCNT field is reloaded from SWU_CNT[n] register's SWU_CNT[n].COUNT field, and the countdown starts again. If SWU_CTL[n].CNTRPTEN bit is cleared, the expired count remains zero, and no further events are signalled. (See the SWU_CTL[n].CNTEN bit description for information regarding the countdown setup.)	
		0	Disable
		1	Enable
8 (R/W)	CNTEN	Count Enable. Each SWU_CTL[n].CNTEN bit controls whether the cycle count in the watchpoint group's SWU_CNT[n] register is decremented each cycle until it reaches zero. This feature is only valid in watchpoint mode (SWU_CTL[n].BWEN bit == 0).When the count reaches zero, any enabled watchpoint events are triggered. (See the SWU_CTL[n].CNTRPTEN bit description for optional actions at that may occur at the end of the countdown.)	
		0	Disable
		1	Enable
5 (R/W)	SCMPEN	Secure Comparison Enable. Each SWU_CTL[n].SCMPEN bit controls secure transaction comparison operation of an SWU watchpoint group, permitting matches based on transaction security.	
		0	Match on all transaction
		1	Match only secure transactions
4 (R/W)	IDCMPEN	ID Comparison Enable. Each SWU_CTL[n].IDCMPEN bit controls the ID comparison operation of an SWU watchpoint group. The ID match is based on comparison with the value in the SWU_ID[n] register.	

Table 46-7: SWU_CTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	ACMPM	Address Comparison Mode. Each set of `SWU_CTL[n].ACMPM` bits control the address comparison operation of an SWU watchpoint group. The address within range for comparison is defined as (`SWU_LA[n]` register <= address < `SWU_UA[n]` register). The address outside range for comparison is defined as (address < `SWU_LA[n]`) or (`SWU_UA[n]` <= address).	
		0	No address comparison
		1	Exact match on LAn
		2	Match on address within range
		3	Match on address outside range
1 (R/W)	DIR	Transaction Direction for Match. Each `SWU_CTL[n].DIR` bit determines whether the SWU check reads or writes for watchpoint matches.	
		0	Match on reads only
		1	Match on writes only
0 (R/W)	EN	Enable Watchpoint. Each `SWU_CTL[n].EN` bit controls the operation of one SWU watchpoint group. Clearing the `SWU_CTL[n].EN` bit halts the execution of watchpoint or bandwidth tracking operations in the watchpoint group without resetting status or configuration registers. Setting the `SWU_CTL[n].EN` bit enables the SWU watchpoint group to begin or resume operation with the current configuration and status.	
		0	Disable
		1	Enable

Current Register n

The SWU current register (SWU_CUR[n]) operation varies depending whether the watchpoint group is in bandwidth mode or watchpoint mode. In both modes, the watchpoint count begins when the SWU loads the register's SWU_CUR[n].CURCNT field from the SWU_CNT[n] register's SWU_CNT[n].COUNT field when the watchpoint count is enabled (SWU_CTL[n] register, SWU_CTL[n].CNTEN bit =1).

In bandwidth mode, the current count field (SWU_CUR[n].CURCNT) contains the cycle count remaining within the current watchpoint period. The SWU decrements this value every cycle until the count reaches zero. At that point, the SWU reloads the SWU_CUR[n].CURCNT field from SWU_CNT[n] register's SWU_CNT[n].COUNT field. In bandwidth mode, the current bandwidth field (SWU_CUR[n].CURBW) contains the count of watchpoint matches (bandwidth) accumulated in the current watchpoint period.

In watchpoint mode, the current count field (SWU_CUR[n].CURCNT) contains the watchpoint match count remaining within the current watchpoint period. The SWU decrements this value with every watchpoint match until the count reaches zero. At that point, the SWU reloads the SWU_CUR[n].CURCNT field from SWU_CNT[n] register's SWU_CNT[n].COUNT field if the SWU_CTL[n] register's SWU_CTL[n].CNTRPTEN bit is set (=1). In watchpoint mode, the current bandwidth field (SWU_CUR[n].CURBW) is undefined.

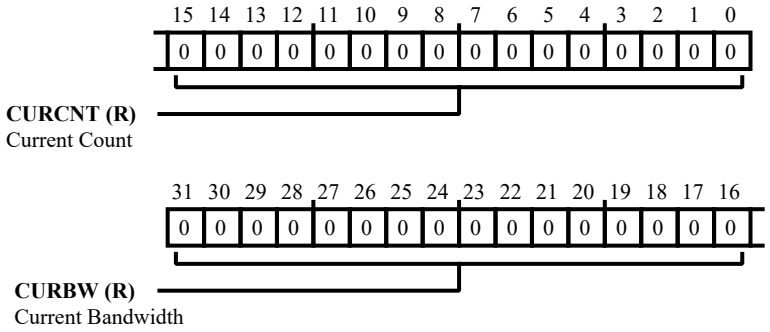


Figure 46-5: SWU_CUR[n] Register Diagram

Table 46-8: SWU_CUR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	CURBW	Current Bandwidth.
15:0 (R/NW)	CURCNT	Current Count.

Global Control Register

The SWU global control register (SWU_GCTL) provides SWU reset and enable.

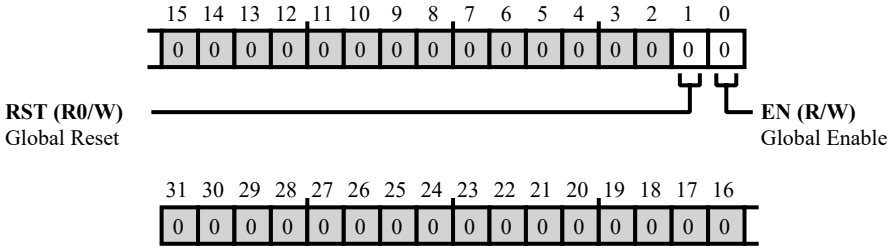


Figure 46-6: SWU_GCTL Register Diagram

Table 46-9: SWU_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R0/W)	RST	Global Reset. The SWU_GCTL.RST) is write-1-action/read zero and controls the SWU operational state. Setting SWU_GCTL.RST resets all SWU registers to their default values and halts all SWU operations.	
		0	No Action
		1	Reset
0 (R/W)	EN	Global Enable. The SWU_GCTL.EN controls the SWU operational state. Clearing SWU_GCTL.EN halts the execution of all watchpoint and bandwidth tracking operations without resetting status registers or associated signals. Setting SWU_GCTL.EN enables the SWU to begin/resume operation with the current configuration and status.	
		0	Disable
		1	Enable

Global Status Register

The SWU global status register (SWU_GSTAT) contains status bits for all four watchpoint groups.

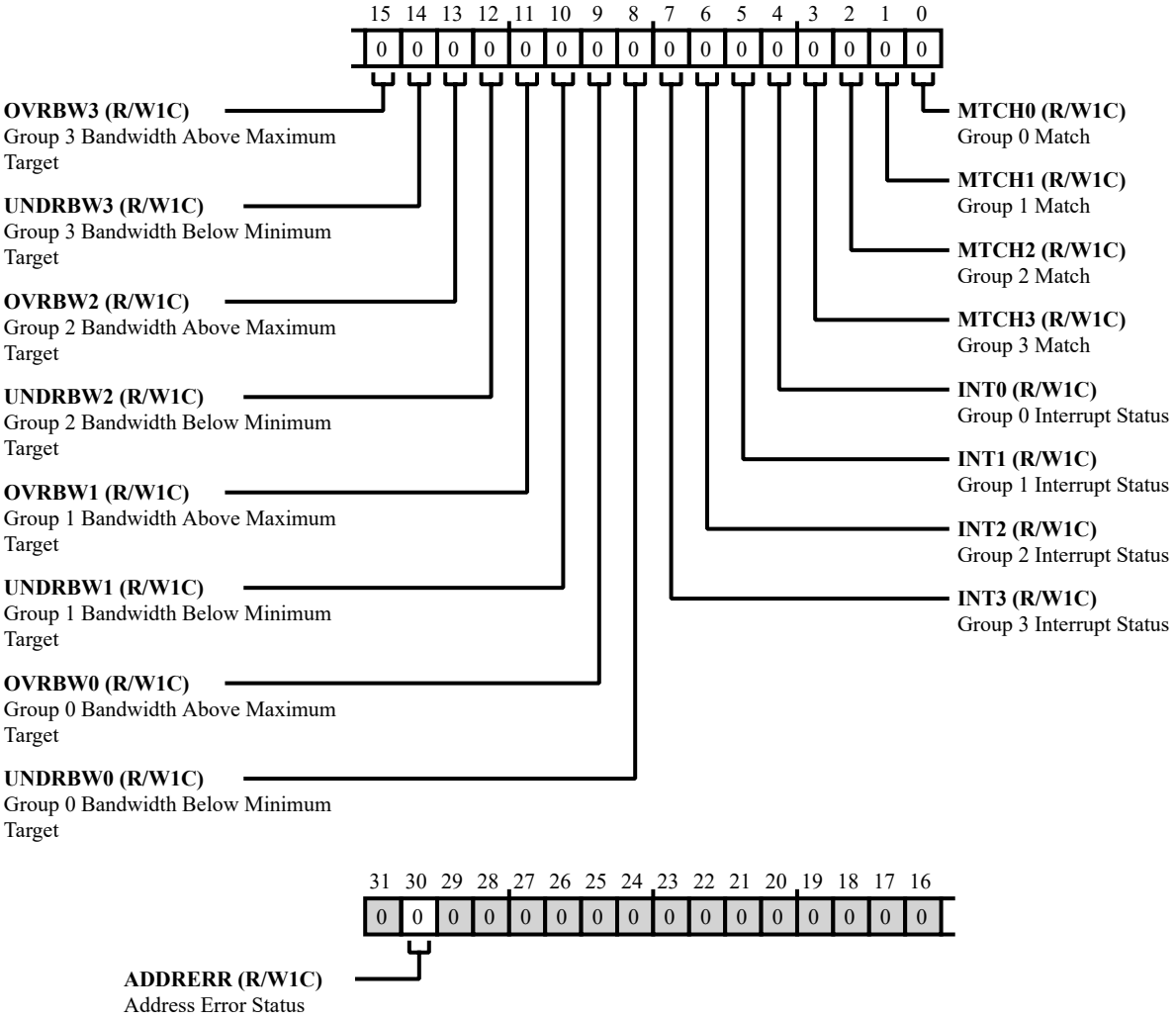


Figure 46-7: SWU_GSTAT Register Diagram

Table 46-10: SWU_GSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W1C)	ADDRERR	Address Error Status. The SWU_GSTAT.ADDRERR indicates that the SWU generated an address error. This status bit is sticky; write-1-to-clear it.	
		0	Inactive
		1	Active

Table 46-10: SWU_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	OVRBW3	Group 3 Bandwidth Above Maximum Target. See SWU_GSTAT.OVRBW0 description.	
		0	Group 3 was not above maximum bandwidth
		1	Group 3 was above maximum bandwidth
14 (R/W1C)	UNDRBW3	Group 3 Bandwidth Below Minimum Target. See SWU_GSTAT.UNDRBW0 description.	
		0	Group 3 was not below minimum bandwitdth
		1	Group 3 was below minimum bandwitdth
13 (R/W1C)	OVRBW2	Group 2 Bandwidth Above Maximum Target. See SWU_GSTAT.OVRBW0 description.	
		0	Group 2 was not above maximum bandwidth
		1	Group 2 was above maximum bandwidth
12 (R/W1C)	UNDRBW2	Group 2 Bandwidth Below Minimum Target. See SWU_GSTAT.UNDRBW0 description.	
		0	Group 2 was not below minimum bandwidth
		1	Group 2 was below minimum bandwidth
11 (R/W1C)	OVRBW1	Group 1 Bandwidth Above Maximum Target. See SWU_GSTAT.OVRBW0 description.	
		0	Group 1 was not above maximum bandwidth
		1	Group 1 was above maximum bandwidth
10 (R/W1C)	UNDRBW1	Group 1 Bandwidth Below Minimum Target. See SWU_GSTAT.UNDRBW0 description.	
		0	Group 1 was not below minimum bandwidth
		1	Group 1 was below minimum bandwidth
9 (R/W1C)	OVRBW0	Group 0 Bandwidth Above Maximum Target. The SWU_GSTAT.OVRBW0 - SWU_GSTAT.OVRBW3 -- Group 0 through 3 watch-point bandwidth over maximum target bits. Each maximum bandwidth bit indicate (for each group)s that the measured bandwidth over the period defined by the SWU_CNT[n] register was over the maximum target. This status bit is sticky; write-1-to-clear it.	
		0	Group 0 was not above maximum bandwidth
		1	Group 0 was above maximum bandwidth

Table 46-10: SWU_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	UNDRBW0	Group 0 Bandwidth Below Minimum Target. The SWU_GSTAT.UNDRBW0 - SWU_GSTAT.UNDRBW3 -- Group 0 through 3 watchpoint bandwidth below minimum target bits. Each minimum bandwidth bit indicates (for each group) that the measured bandwidth over the period defined by the SWU_CNT[n] register was below the minimum target. This status bit is sticky; write-1-to-clear it.	
		0	Group 0 was not below minimum bandwidth
		1	Group 0 was below minimum bandwidth
7 (R/W1C)	INT3	Group 3 Interrupt Status. See SWU_GSTAT.INT0 description.	
		0	No Interrupt
		1	Interrupt Occurred
6 (R/W1C)	INT2	Group 2 Interrupt Status. See SWU_GSTAT.INT0 description.	
		0	No Interrupt
		1	Interrupt Occurred
5 (R/W1C)	INT1	Group 1 Interrupt Status. See SWU_GSTAT.INT0 description.	
		0	No Interrupt
		1	Interrupt Occurred
4 (R/W1C)	INT0	Group 0 Interrupt Status. The SWU_GSTAT.INT0 - SWU_GSTAT.INT3 -- Group 0 through 3 interrupt bits. Each interrupt bit indicates (for each group) whether a watchpoint group is contributing to the SWU's interrupt output. This status bit is sticky; write-1-to-clear it.	
		0	No interrupt
		1	Interrupt Occurred
3 (R/W1C)	MTCH3	Group 3 Match. See SWU_GSTAT.MTCH0 description.	
		0	No Match
		1	Group 3 Watchpoint Match

Table 46-10: SWU_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	MTCH2	Group 2 Match. See SWU_GSTAT.MTCH0 description.	
		0	No match
		1	Group 2 Watchpoint Match
1 (R/W1C)	MTCH1	Group 1 Match. See SWU_GSTAT.MTCH0 description.	
		0	No match
		1	Group 1 Watchpoint Match
0 (R/W1C)	MTCH0	Group 0 Match. The SWU_GSTAT.MTCH0 - SWU_GSTAT.MTCH3 -- Group 0 through 3 match bits. Each match bit indicates (for each group) whether a watchpoint match has occurred in a SWU watchpoint group, as controlled by the group's related watchpoint control register (SWU_CTL[n]). This status bit is sticky; write-1-to-clear it.	
		0	No match
		1	Group 0 Watchpoint Match

Bandwidth History Register n

The SWU bandwidth history registers (SWU_HIST[n]) contain data copied from a watchpoint group's current bandwidth value (SWU_CUR[n] register, SWU_CUR[n].CURBW bits) at the end of the last two watchpoint periods. At the end of each watchpoint period, the SWU copies the previous bandwidth value from the SWU_HIST[n].BWHIST0 field to the SWU_HIST[n].BWHIST1 field and copies the new bandwidth value from the SWU_CUR[n].CURBW field to the SWU_HIST[n].BWHIST0 field.

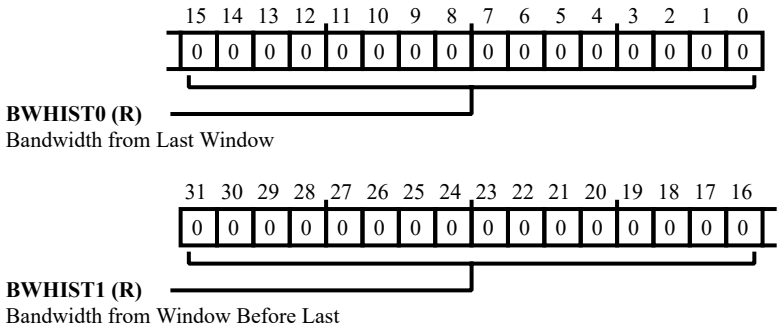


Figure 46-8: SWU_HIST[n] Register Diagram

Table 46-11: SWU_HIST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	BWHIST1	Bandwidth from Window Before Last.
15:0 (R/NW)	BWHIST0	Bandwidth from Last Window.

ID Register n

The SWU ID registers (SWU_ID[n]) contain a 16-bit ID field (SWU_ID[n].ID) and a 16-bit ID mask field (SWU_ID[n].IDMASK) that watchpoint groups use for ID comparison. The ID on the bus is AND'ed with the SWU_ID[n].IDMASK field, then the watchpoint group compares the result against the SWU_ID[n].ID field.

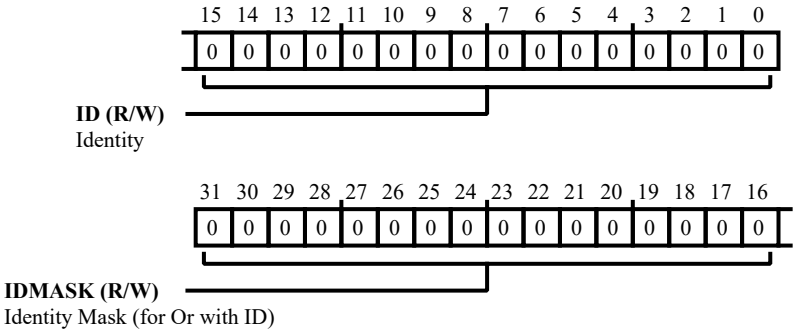


Figure 46-9: SWU_ID[n] Register Diagram

Table 46-12: SWU_ID[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	IDMASK	Identity Mask (for Or with ID).
15:0 (R/W)	ID	Identity.

Lower Address Register n

The SWU lower address registers (SWU_LA[n]) contain each watchpoint group's lower address for address match comparison. In exact match on SWU_LA[n] address mode (SWU_CTL[n].ACMPM bits =01), the watchpoint group uses only this address for match comparison.

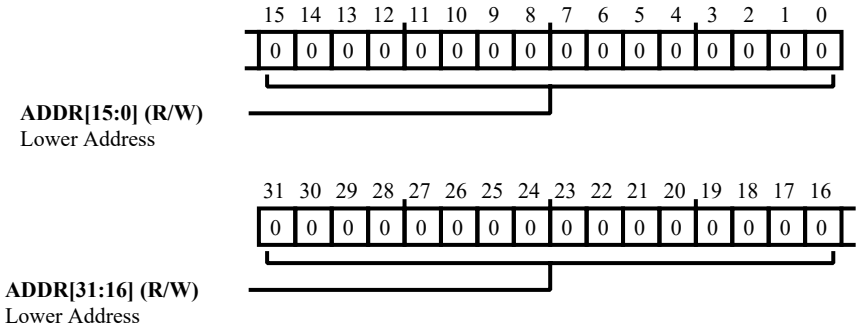


Figure 46-10: SWU_LA[n] Register Diagram

Table 46-13: SWU_LA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR	Lower Address.

Target Register n

The SWU target registers (SWU_TARG[n]) contain a minimum value field (SWU_TARG[n].BWMIN) and maximum value field (SWU_TARG[n].BWMAX) of bandwidth targets used by watchpoint groups in bandwidth mode. When the bandwidth period expires, if the current bandwidth value (SWU_CUR[n] register, SWU_CUR[n].CURBW bits) is below the minimum target or above the maximum target, the watchpoint group takes action as enabled by the SWU_CTL[n] register's SWU_CTL[n].MINACT or SWU_CTL[n].MAXACT bits.

In bandwidth mode, note that the watchpoint group increments its count of either data bus transactions or address bus transactions (bursts) as selected by the SWU_CTL[n].BLENINC bit. Keep this mode selection in mind when programming the bandwidth target values.

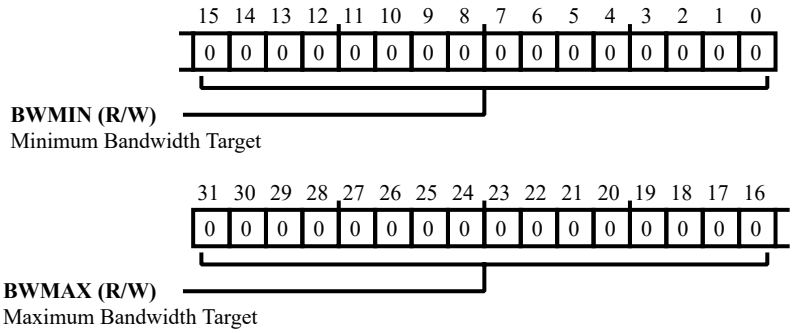


Figure 46-11: SWU_TARG[n] Register Diagram

Table 46-14: SWU_TARG[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	BWMAX	Maximum Bandwidth Target.
15:0 (R/W)	BWMIN	Minimum Bandwidth Target.

Upper Address Register n

The SWU upper address registers (SWU_UA[n]) contain each watchpoint group's upper address for address match comparison. In exact match on SWU_LA[n] address mode (SWU_CTL[n].ACMPM bits =01), the SWU_UA[n] is not used for match comparison.

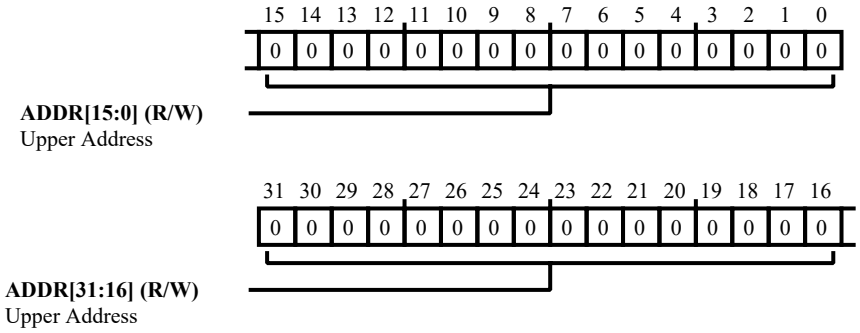


Figure 46-12: SWU_UA[n] Register Diagram

Table 46-15: SWU_UA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR	Upper Address.

47 Memory Error Protection Unit (MEPU)

Memory Error Protection Unit (MEPU) handles single-bit and double-bit memory error detection and correction across the memories in the chip and controls routing of their interrupts/triggers.

It consists of following blocks:

- Memory Error Controller (MEC)
- Parity Controller (PCTL)
- ECC Controller (ECTL)

Memory Error Controller (MEC)

The Memory Error Controller (MEC) manages memory parity, ECC errors and warning inputs from various cores and peripherals and sends out interrupt and trigger outputs. It is a generic N x M multiplexer and interrupt controller for error or warning inputs to interrupt and trigger outputs. It has control features such as enable and disable and interrupt masking, and status signaling for input/outputs. For more information on MEC functionality, see the MEC registers description.

MEC Features

The MEC unit has the following features:

- Interrupt controller for memory parity, ECC errors and warnings
- N x M multiplexer for error inputs to interrupt and trigger outputs
- Peripheral bus completer interface for register read and write
- Control and status registers
    - Control for enabling or disabling each input error status and masking its interrupt
    - Control for enabling or disabling each output error interrupt
    - Input error status bits are sticky and write-1-to-clear
    - Output error interrupt status bits are read only

    - Lock feature for control registers write. Status registers capture lock write error.
- Peripheral and component identification registers
- Capability to retain parity, ECC error and warning status even after any reset for fault source debug
- Fully-configurable design

Parity Controller (PCTL)

The Parity Controller (PCTL) handles parity encoding and decoding to and from memory data for single bit memory error detection.

PCTL Features

The PCTL unit has the following features:

- Generic memory parity controller
    - Generation of write parity bits from write data bits
    - Concatenation of parity bits with data bits for memory write data
    - Generation of write enable mask for parity bits to memory
    - Extraction of read data bits from memory read data
    - Detection of read parity error from memory read data
- Even parity logic for parity generation and error detection
- Provision for parity bit interleaving
- Single parity error output per memory instance
- Memory initialization control logic
- Fully-configurable design

MEC Functional Description

The MEC works with the PCTL to detect and correct memory error across the memories in the chip. It also controls the routing of the generated interrupts and triggers.

ADSP-2184x MEC Register List

The Memory Error Controller (MEC) manages memory parity/ inputs from the core and peripherals and sends interrupt/trigger outputs. It is a generic N x M multiplexer and interrupt controller for error to interrupt/trigger outputs. It has control features such as enable/disable and interrupt masking, and status signaling for inputs/outputs. For more information on MEC functionality, see the MEC register descriptions.

Table 47-1: ADSP-2184x MEC Register List

Name	Description
MEC_A55C0EIRQ_GCTL[n]	A55 Core Memory Errors Interrupt Request Global Control Register
MEC_A55C0EIRQ_GSTAT[n]	A55 Error Interrupt Request Global Status Register
MEC_A55C0ERR_CTL[n]	A55 Error Control Register
MEC_A55C0ERR_IMASK[n]	A55 Error Interrupt Mask Register
MEC_A55C0ERR_STAT[n]	A55 Error Status Register
MEC_A55C2EIRQ_GCTL[n]	A55 Core Memory Errors Interrupt Request Global Control Register
MEC_A55C2EIRQ_GSTAT[n]	A55 Error Interrupt Request Global Status Register
MEC_A55C2ERR_CTL[n]	A55 Error Control Register
MEC_A55C2ERR_IMASK[n]	A55 Error Interrupt Mask Register
MEC_A55C2ERR_STAT[n]	A55 Error Status Register
MEC_CID0	Component ID0 Register
MEC_CID1	Component ID1 Register
MEC_CID2	Component ID2 Register
MEC_CID3	Component ID3 Register
MEC_CLR	Clear Register
MEC_CMEIRQ_GCTL[n]	Core Mem ECC Error Interrupt Request Global Control Register
MEC_CMEIRQ_GSTAT[n]	Core Memory ECC Error Interrupt Request Global Status Register
MEC_CMERR_CTL[n]	Core Memory ECC Error Control Register
MEC_CMERR_IMASK[n]	Core Memory ECC Error Interrupt Mask Register
MEC_CMERR_STAT[n]	Core Memory ECC Error Status Register
MEC_ECCERR_CTL[n]	ECC Error Control Register
MEC_ECCERR_IMASK[n]	ECC Error Interrupt Mask Register
MEC_ECCERR_STAT[n]	ECC Error Status Register
MEC_EEIRQ_GCTL[n]	ECC Error Interrupt Request Global Control Register
MEC_EEIRQ_GSTAT[n]	ECC Error Interrupt Request Global Status Register
MEC_PEIRQ_GCTL[n]	Parity Error Interrupt Request Global Control Register
MEC_PEIRQ_GSTAT[n]	Parity Error Interrupt Request Global Status Register
MEC_PERR_CTL0	Parity Error Control Register
MEC_PERR_CTL1	Parity Error Control Register
MEC_PERR_IMASK0	Parity Error Interrupt Mask Register
MEC_PERR_IMASK1	Parity Error Interrupt Mask Register

Table 47-1: ADSP-2184x MEC Register List (Continued)

Name	Description
MEC_PERR_STAT0	Parity Error Status Register
MEC_PERR_STAT1	Parity Error Status Register
MEC_PID0	Peripheral ID0 Register
MEC_PID1	Peripheral ID1 Register
MEC_PID2	Peripheral ID2 Register
MEC_PID3	Peripheral ID3 Register
MEC_PID4	Peripheral ID4 Register

ADSP-2184x MEC Trigger List

Table 47-2: ADSP-2184x MEC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
85	MEC1_EEIRQ0	MEC1 ECC Error Interrupt Request	Level
86	MEC1_MEIRQ2	MEC1 Memory Error Interrupt Request	Level
87	MEC1_PEIRQ0	MEC1 Parity Error Interrupt Request	Level
88	MEC2_EEIRQ0	MEC2 ECC Error Interrupt/Trigger Request Bus	Level
89	MEC2_MEIRQ2	MEC2 Memory Error Interrupt Request	Level
90	MEC2_PEIRQ0	MEC2 Parity Error Interrupt/Trigger Request Bus	Level
91	MEC1_MEIRQ1	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level
92	MEC2_MEIRQ1	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level
93	MEC1_MEIRQ0	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level
94	MEC2_MEIRQ0	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level
95	MEC0_EEIRQ0	MEC0 ECC Error Interrupt/Trigger Request Bus	Level
96	MEC0_MEIRQ2	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level
97	MEC0_PEIRQ0	MEC0 Parity Error Interrupt Request	Level
98	MEC0_MEIRQ1	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level
99	MEC0_MEIRQ0	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level

Table 47-3: ADSP-2184x MEC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

ADSP-2184x MEC Interrupt List

Table 47-4: ADSP-2184x MEC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
183	MEC1_EEIRQ0	MEC1 ECC Error Interrupt Request	Level	
184	MEC1_EWIRQ0	MEC1 ECC Warning to Interrupt Request (Reserved)	Level	
185	MEC1_MEIRQ1	MEC1 Parity Error Interrupt Request	Level	
186	MEC1_MFIRQ00	MEC1 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
187	MEC1_MEIRQ0	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level	
188	MEC1_MFIRQ20	MEC1 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
189	MEC1_MEIRQ2	MEC1 Core Memory Error Interrupt/Trigger Request Bus	Level	
190	MEC1_PEIRQ0	MEC1 Parity Error Interrupt/Trigger Request Bus	Level	
191	MEC2_EEIRQ0	MEC2 ECC Error Interrupt/Trigger Request Bus	Level	
192	MEC2_EWIRQ0	MEC2 ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
193	MEC2_MWIRQ1	MEC2 Core Memory ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
194	MEC2_MEIRQ1	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level	
195	MEC2_MFIRQ00	MEC2 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
196	MEC2_MEIRQ0	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level	
197	MEC2_MFIRQ20	MEC2 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
198	MEC2_MEIRQ2	MEC2 Core Memory Error Interrupt/Trigger Request Bus	Level	

Table 47-4: ADSP-2184x MEC Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
199	MEC2_PEIRQ0	MEC2 Parity Error Interrupt/Trigger Request Bus	Level	
200	MEC1_MWIRQ1	MEC1 MEC1 Core 1 Mem Warning to Core 1 (Reserved)	Level	
201	MEC0_EEIRQ0	MEC0 ECC Error Interrupt/Trigger Request Bus	Level	
202	MEC0_EWIRQ0	MEC0 ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
203	MEC0_MWIRQ1	MEC0 Core Memory ECC Warning Interrupt/Trigger Request Bus (Reserved)	Level	
204	MEC0_MEIRQ1	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level	
205	MEC0_MFIRQ00	MEC0 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
206	MEC0_MEIRQ0	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level	
207	MEC0_MFIRQ20	MEC0 A55 Core Memory Fault Interrupt/Trigger Request Bus	Level	
208	MEC0_MEIRQ2	MEC0 Core Memory Error Interrupt/Trigger Request Bus	Level	
209	MEC0_PEIRQ0	MEC0 Parity Error Interrupt/Trigger Request Bus	Level	

MEPU Block Diagram

The ***MEPU Block Diagram*** shows the functional blocks within the Memory Error Protection Unit (MEPU) unit.

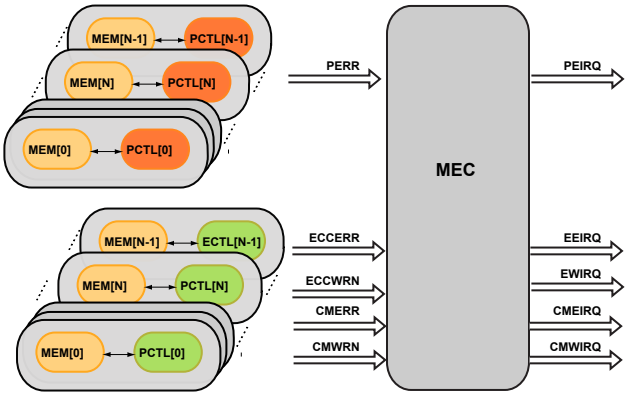


Figure 47-1: MEPU Unit Block Diagram

MEC Block Diagram

The *MEC Interface Block Diagram* shows the functional blocks within the MEC module.

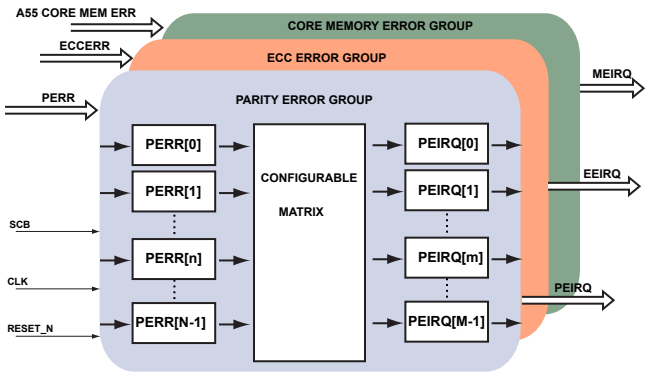


Figure 47-2: MEC Unit Block Diagram

PCTL Block Diagram

The *PCTL Block Diagram* shows the functional blocks within the PCTL module.

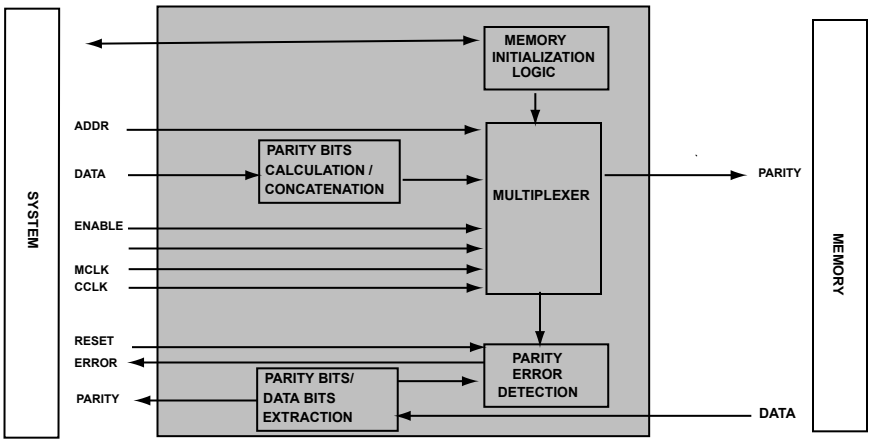


Figure 47-3: PCTL Block Diagram

MECx Input and Output Block Diagram

The *MECx Input and Output Block Diagram* shows the input and output routing to various cores.

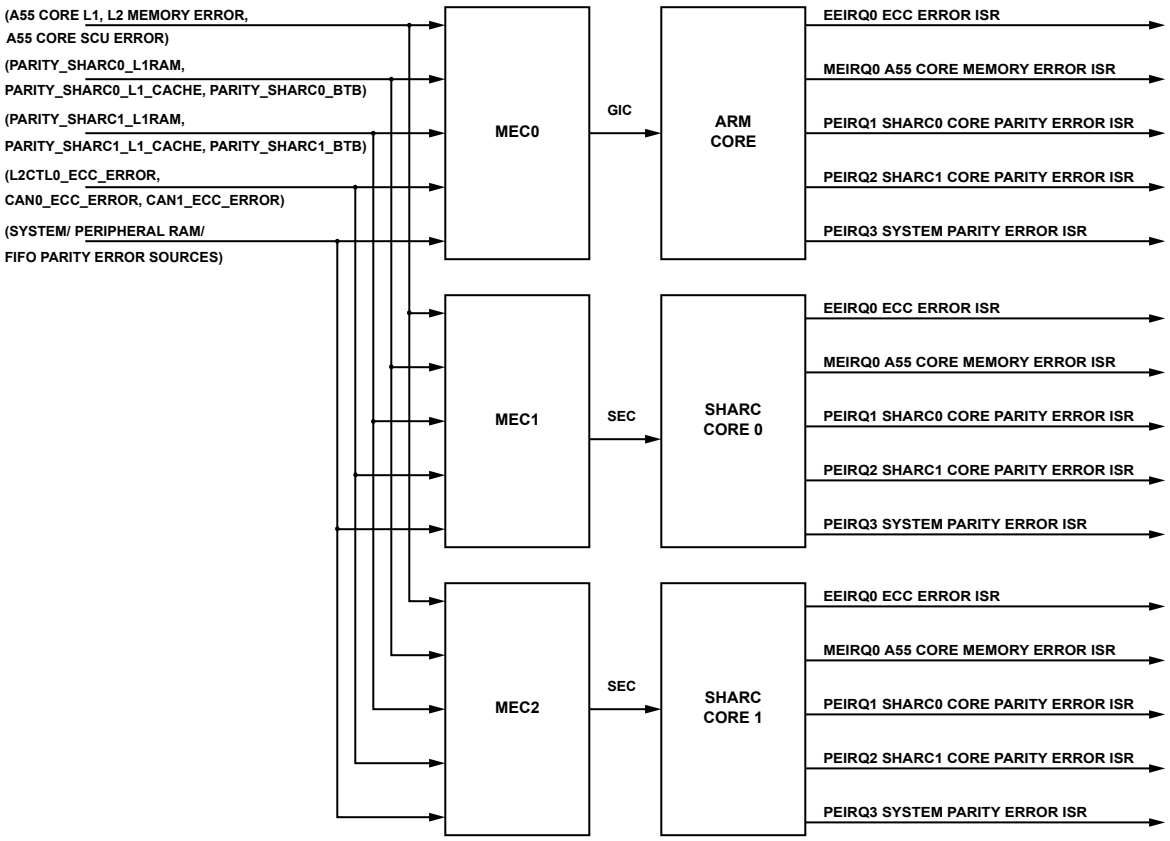


Figure 47-4: MECx Input and Output Block Diagram

MEC Architectural Concepts

The MEC unit communicates with the Cortex-A55 core directly through the subsystem's SYS bus crossbar. It services MMR addresses associated with the MEC function unit and translates the addresses into specific transcendental MEC functions. The results are read from common result registers shared by all the functions in single-precision floating-point format. The block stalls all read accesses until the result is ready.

MEC Configuration

The following tables describe the range of values for configuration and the mapping of parity errors.

Table 47-5: Mapping of Input Parity Errors (PERR)

PERR ID	Name	Description
0	Reserved	Reserved
1	Reserved	Reserved
2	PERR_C1_L1R	Core1 (SHARC0) L1 RAM Parity Error
3	PERR_C1_L1CR	Core1 (SHARC0) L1 Cache RAM Parity Error
4	PERR_C1_BPR	Core1 (SHARC0) Branch Predictor RAM Parity Error

Table 47-5: Mapping of Input Parity Errors (PERR) (Continued)

PERR ID	Name	Description
5	Reserved	Reserved
6	Reserved	Reserved
7	Reserved	Reserved
8	PERR_ASRC0_FR	ASRC0 FIFO RAM Parity Error
9	PERR_ASRC1_FR	ASRC1 FIFO RAM Parity Error
10	PERR_ASRC2_FR	ASRC2 FIFO RAM Parity Error
11	PERR_ASRC3_FR	ASRC3 FIFO RAM Parity Error
12	PERR_ASRC4_FR	ASRC4 FIFO RAM Parity Error
13	PERR_ASRC5_FR	ASRC5 FIFO RAM Parity Error
14	PERR_ASRC6_FR	ASRC6 FIFO RAM Parity Error
15	PERR_ASRC7_FR	ASRC7 FIFO RAM Parity Error
16	PERR_C0_IIR0_R	SH0 IIR0 RAM Parity Error
17	PERR_C0_IIR1_R	SH0 IIR1 RAM Parity Error
18	PERR_C0_IIR2_R	SH0 IIR2 RAM Parity Error
19	PERR_C0_IIR3_R	SH0 IIR3 RAM Parity Error
20	PERR_C0_FIR0_R	FIR0 RAM Parity Error
21	Reserved	Reserved
22	Reserved	Reserved
23	Reserved	Reserved
24	Reserved	Reserved
25	PERR_C0_FIR1_R	FIR1 RAM Parity Error
26	Reserved	Reserved
27	PERR_TRNG0_DBR	TRNG0 Data Buffer RAM Parity Error
28	PERR_PKA0_DR	PKA0 Data RAM Parity Error
29	PERR_SPE0_BR	SPE0 Buffer RAM Parity Error
30	PERR_SPE0_AR	SPE0 ARC4 RAM Parity Error
31	PERR_EMAC0_TFR	EMAC0 Transmit FIFO RAM Parity Error
32	PERR_EMAC0_RFR	EMAC0 Receive FIFO RAM Parity Error
33	PERR_MLB0_DBR	MLB0 Data Buffer RAM Parity Error
34	PERR_MLB0_CTR	MLB0 Channel Table RAM Parity Error
35	PERR_TMC0_TDR	TMC0 Trace Data RAM Parity Error

Table 47-5: Mapping of Input Parity Errors (PERR) (Continued)

PERR ID	Name	Description
36	Reserved	Reserved
37	Reserved	Reserved
38	PERR_EMAC0_TMI	EMAC0 TMI Parity Error
39	PERR_EMAC0_TST	EMAC0 TST Parity Error
40	PERR_MSHC	MSHC Parrity Error
41	PERR_ASRC8_FR	ASRC8 FIFO RAM Parity Error
42	PERR_ASRC9_FR	ASRC9 FIFO RAM Parity Error
43	PERR_ASRC10_FR	ASRC10 FIFO RAM Parity Error
44	PERR_ASRC11_FR	ASRC11 FIFO RAM Parity Error
45	PERR_ASRC12_FR	ASRC12 FIFO RAM Parity Error
47	PERR_ASRC13_FR	ASRC13 FIFO RAM Parity Error
47	PERR_ASRC14_FR	ASRC14 FIFO RAM Parity Error
48	PERR_ASRC15_FR	ASRC15 FIFO RAM Parity Error
49	PERR_XSPI0_FLASH	XSPI0 Flash Ram Parity Error
50	PERR_XSPI0_CTRL	XSPI0 CTRL Ram Parity Error
51	PERR_XSPI1_FLASH	XSPI1 Flash Ram Parity Error
52	PERR_XSPI1_CTRL	XSPI1 CTRL Ram Parity Error
53	Reserved	Reserved
54	Reserved	Reserved
55	PERR_LPDDR_ICCM	LPDDR ICCM Parity Error
56	PERR_LPDDR_DCCM	LPDDR DCCM Parity Error

Table 47-6: Mapping of Output Parity Errors (PERR)

PERR ID	Name	Description
0	PEIRQ_SYS	System Peripheral Parity Error Interrupt

Table 47-7: Mapping of Input ECC Errors (ECCERR)

PERR ID	Name	Description
0	ECCERR_L2CTL0	L2CTL0 ECC Error
1	ECCERR_CAN0	CAN0 ECC Error
2	ECCERR_CAN1	CAN1 ECC Error
3	GIC ERR	GIC Error

Table 47-7: Mapping of Input ECC Errors (ECCERR) (Continued)

PERR ID	Name	Description
4	ECCERR_L2CTL1	L2CTL1 ECC Error

Table 47-8: Mapping of Output ECC Errors (EEIRQ)

PERR ID	Name	Description
0	EWIRQ	ECC Error Interrupt

Table 47-9: Mapping of Input Core Memory ECC Errors (CMERR)

CMERR ID	Name	Description
7	ECCERR_DRAM	Core1 DRAM ECC Error Uncorrectable
6	ECCERR_IRAM	Core1 iRAM ECC Error Uncorrectable
4	ECCERR_DCACHE_DIRTY	Core1 dCache ECC Error Dirty Uncorrectable
3	ECCERR_ICACHE	Core1 iCache ECC Error Uncorrectable
2	ECCERR_DP_CLEAN	Core1 dPref RAM ECC Error Clean Uncorrectable
1	ECCERR_DP_DIRTY	Core1 dPref RAM ECC Error Dirty Uncorrectable
0	ECCERR_IP_CLEAN	Core1 iPref RAM ECC Error Clean Uncorrectable

Table 47-10: Mapping of Input A55 Core n Memory Errors (A55CnERR); n=0, 2

CMERR ID	Name	Description
0	A55 SCU ERR	A55 SCU ERR
1	A55_L1L2_ERR	A55_L1L2_ERR

Table 47-11: Mapping of Output A55 Core n Memory Errors (A55CnEIRQ); n=0, 2

CMERR IRQ ID	Name	Description
0	A55CnEIRQ	A55 Core n Memory Error Interrupt

Error Status and Interrupts

There are two categories of memory errors from the SHARC-FX core:

- Recoverable errors - includes single bit ECC errors and parity and dual bit ECC errors in buffers that can be dropped or refilled from the source.
- Unrecoverable errors - includes the remaining dual bit ECC errors

As the errors occur, this information is relayed over PFaultInfo and sent out to the SEC module via the MPEU module. From the SEC, these signals can be sent out to an external fault pin.

There is an additional signal (PFatalError) that is asserted when the core is stuck in a double/triple exception. The signal directly goes to the SEC module.

The following table lists the recoverable ECC errors in the SHARC-FX core.

Table 47-12: Recoverable Core Memory ECC Errors

SL Number	Fault Category	Type Fault PFaultInfo[50:32]	Description
0		ECC error correctable 32	1-bit ECC error
1		ECC error correctable 34	1-bit ECC error
2		ECC error correctable 36	1-bit ECC error
3		ECC error 37	Clean refill
4		L1S Cache refill 40	Refill from L1V cache
6		ICACHE ECC error correctable 41	1-bit ECC error
7		ECC error refill 42	2-bit ECC error, refill from memory
8	DPREFETCH RAM	DPREF RAM ECC error correctable 44	1-bit ECC error
9	DPREF RAM	ECC error clean drop 45	Clean drop
10	IPREFETCH RAM	IPREF RAM ECC error correctable 48	1-bit ECC error

Table 47-13: Unrecoverable Core Memory ECC Errors

SL Number	Fault Category	Type Fault PFaultInfo[50:32]	Description
1		DRAM ECC error uncorrectable 33	2-bit ECC error
2		IRAM ECC error uncorrectable 35	2-bit ECC error
3		DCACHE ECC error clean uncorrectable 38	2-bit ECC error in unmodified cache line
4		ECC error dirty uncorrectable 39	2-bit ECC error in unmodified cache line
5		ICACHE ECC error uncorrectable 43	2-bit ECC error, cannot be refilled from memory
6	DPREFETCH RAM	DPREF RAM ECC error clean uncorrectable 46	2-bit ECC error in unmodified cache line
7	DPREF RAM	ECC error dirty uncorrectable 47	
8	IPREFETCH RAM	IPREF RAM ECC error uncorrectable 50	

NOTE: In the SH-FX configuration chosen for EHP, the following fault signals are not implemented and tied to 0 in the design and, hence, never occur.

- IPREF RAM ECC error clean drop PFaultInfo[49]
- DCACHE ECC error clean refill PFaultInfo [37]
- DPREF RAM ECC error clean drop PFaultInfo [45]
- DPREF ECC error clean uncorrectable PFaultInfo [46]

Requester Port Bus Interrupts

Bus error responses from the requester ports of the SHARC-FX core are detected and routed to the SEC to raise faults. Although the SHARC-FX core can detect and raise exceptions, when there is a bus error response, there is good chance of data/instruction corruption. Therefore, the core exception handling dependency has been removed. The SEC is directly notified using interrupts.

Instruction Bus Interrupt

Any instruction read error is detected on the instruction requester port; the information is forwarded to the SEC via the core1 instruction read interrupt.

Data Bus Interrupts

Any data read or write error is detected on the data requester port; the information is forwarded to the SEC via the core1 data read interrupt and core1 data write interrupt.

iDMA Bus Interrupts

Any data read or write error that is detected on the iDMA port is forwarded to the SEC via the core1 iDMA read interrupt and core1 iDMA write interrupt.

PCTL Integration

There is one PCTL instance per port per memory instance (for example, one PCTL instance for single port memory and two PCTL instances for dual port memories).

Memory initialization control logic supported by PCTL is used only for instances attached to Arm L1 cache memories.

An additional software trigger requester and initialization trigger completer is provided in TRU for starting memory initialization. A pulse from this trigger completer starts initialization of Arm L1 cache memories. When all the locations of a particular memory instance get initialized, the PCTL generates a memory initialization done signal for the corresponding memory instance. These signals from all PCTL instances corresponding to Arm L1 cache memories are ANDed to generate a single memory initialization done output which is connected as an interrupt to SEC/GIC and a trigger requester to TRU.

ADSP-2184x MEC Register Descriptions

Memory Error Controller (MEC) contains the following registers.

Table 47-14: ADSP-2184x MEC Register List

Name	Description
MEC_A55C0EIRQ_GCTL[n]	A55 Core Memory Errors Interrupt Request Global Control Register
MEC_A55C0EIRQ_GSTAT[n]	A55 Error Interrupt Request Global Status Register
MEC_A55C0ERR_CTL[n]	A55 Error Control Register
MEC_A55C0ERR_IMASK[n]	A55 Error Interrupt Mask Register

Table 47-14: ADSP-2184x MEC Register List (Continued)

Name	Description
MEC_A55C0ERR_STAT[n]	A55 Error Status Register
MEC_A55C2EIRQ_GCTL[n]	A55 Core Memory Errors Interrupt Request Global Control Register
MEC_A55C2EIRQ_GSTAT[n]	A55 Error Interrupt Request Global Status Register
MEC_A55C2ERR_CTL[n]	A55 Error Control Register
MEC_A55C2ERR_IMASK[n]	A55 Error Interrupt Mask Register
MEC_A55C2ERR_STAT[n]	A55 Error Status Register
MEC_CID0	Component ID0 Register
MEC_CID1	Component ID1 Register
MEC_CID2	Component ID2 Register
MEC_CID3	Component ID3 Register
MEC_CLR	Clear Register
MEC_CMEIRQ_GCTL[n]	Core Mem ECC Error Interrupt Request Global Control Register
MEC_CMEIRQ_GSTAT[n]	Core Memory ECC Error Interrupt Request Global Status Register
MEC_CMERR_CTL[n]	Core Memory ECC Error Control Register
MEC_CMERR_IMASK[n]	Core Memory ECC Error Interrupt Mask Register
MEC_CMERR_STAT[n]	Core Memory ECC Error Status Register
MEC_ECCERR_CTL[n]	ECC Error Control Register
MEC_ECCERR_IMASK[n]	ECC Error Interrupt Mask Register
MEC_ECCERR_STAT[n]	ECC Error Status Register
MEC_EEIRQ_GCTL[n]	ECC Error Interrupt Request Global Control Register
MEC_EEIRQ_GSTAT[n]	ECC Error Interrupt Request Global Status Register
MEC_PEIRQ_GCTL[n]	Parity Error Interrupt Request Global Control Register
MEC_PEIRQ_GSTAT[n]	Parity Error Interrupt Request Global Status Register
MEC_PERR_CTL0	Parity Error Control Register
MEC_PERR_CTL1	Parity Error Control Register
MEC_PERR_IMASK0	Parity Error Interrupt Mask Register
MEC_PERR_IMASK1	Parity Error Interrupt Mask Register
MEC_PERR_STAT0	Parity Error Status Register
MEC_PERR_STAT1	Parity Error Status Register
MEC_PID0	Peripheral ID0 Register
MEC_PID1	Peripheral ID1 Register

Table 47-14: ADSP-2184x MEC Register List (Continued)

Name	Description
MEC_PID2	Peripheral ID2 Register
MEC_PID3	Peripheral ID3 Register
MEC_PID4	Peripheral ID4 Register

A55 Core Memory Errors Interrupt Request Global Control Register

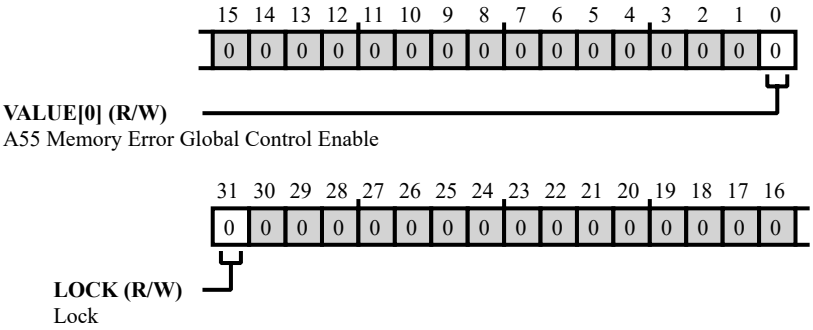


Figure 47-5: MEC_A55C0EIRQ_GCTL[n] Register Diagram

Table 47-15: MEC_A55C0EIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the `MEC_A55C0EIRQ_GCTL[n].LOCK` bit is enabled, the `MEC_A55C0EIRQ_GCTL[n]` register is read only.	
		0	Unlock
		1	Lock
0 (R/W)	VALUE	A55 Memory Error Global Control Enable. The `MEC_A55C0EIRQ_GCTL[n].VALUE` bit enables the A55 memory error global control.	
		0	Disable
		1	Enable

A55 Error Interrupt Request Global Status Register

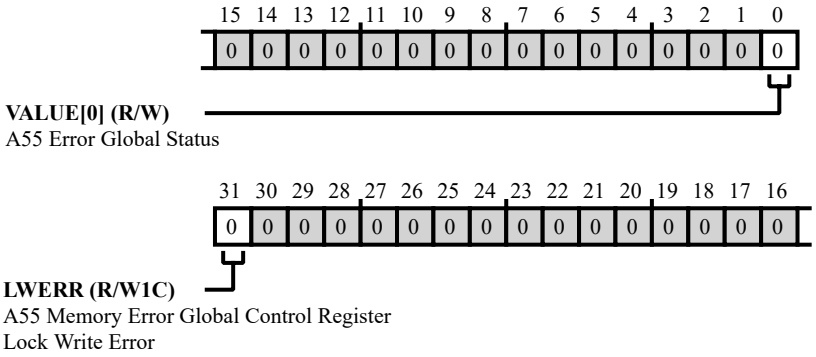


Figure 47-6: MEC_A55C0EIRQ_GSTAT[n] Register Diagram

Table 47-16: MEC_A55C0EIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	A55 Memory Error Global Control Register Lock Write Error. When set (=1), the `MEC_A55C0EIRQ_GSTAT[n].LWERR` bit indicates that there was an attempted write to an MEC register while the `MEC_EEIRQ_GCTL[n].LOCK` bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
0 (R/W)	VALUE	A55 Error Global Status. When set (=1), the `MEC_A55C0EIRQ_GSTAT[n].VALUE` bit indicates the assertion of an interrupt/trigger. When cleared (=0), there is no interrupt or trigger.	
		0	No Interrupt/Trigger
		1	Interrupt/Trigger Asserted

A55 Error Control Register

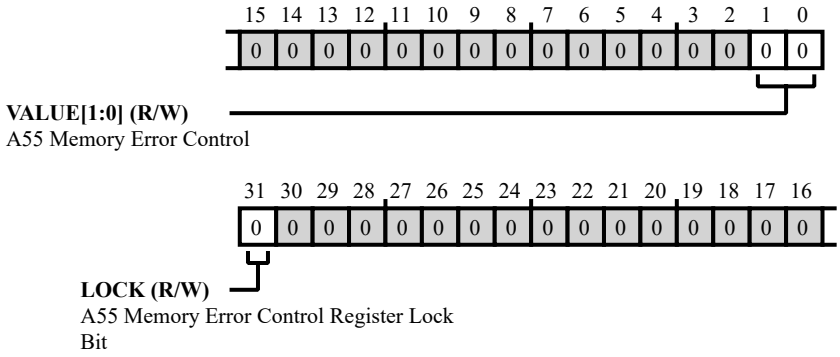


Figure 47-7: MEC_A55C0ERR_CTL[n] Register Diagram

Table 47-17: MEC_A55C0ERR_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	A55 Memory Error Control Register Lock Bit. If the global lock is enabled (SPU_CTL.GLCK =1) and the `MEC_A55C0ERR_CTL[n].LOCK` bit is enabled, the `MEC_A55C0ERR_CTL[n]` register is read only.	
		0	Unlock
		1	Lock
1:0 (R/W)	VALUE	A55 Memory Error Control. The `MEC_A55C0ERR_CTL[n].VALUE` field enables A55 memory error control. When set (=1), control is enabled. When cleared (=0), control is disabled.	
		1	A55 SCU Err
		2	A55 L1,L2 Err

A55 Error Interrupt Mask Register

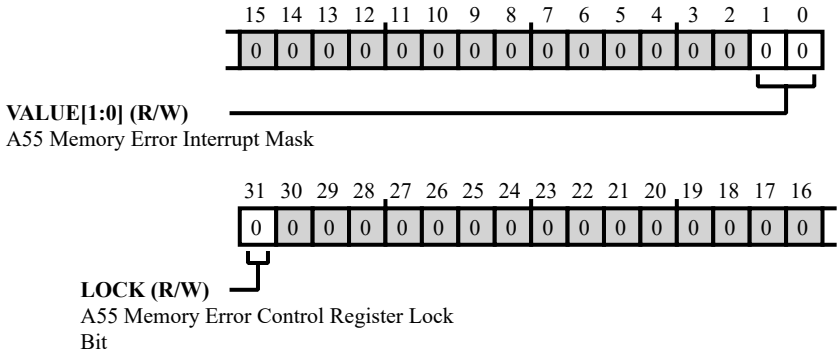


Figure 47-8: MEC_A55C0ERR_IMASK[n] Register Diagram

Table 47-18: MEC_A55C0ERR_IMASK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	A55 Memory Error Control Register Lock Bit. If the global lock is enabled (SPU_CTL.GLCK =1) and the `MEC_A55C0ERR_IMASK[n].LOCK` bit is enabled, the `MEC_A55C0ERR_IMASK[n]` register is read only.	
		0	Unlock
		1	Lock
1:0 (R/W)	VALUE	A55 Memory Error Interrupt Mask. The `MEC_A55C0ERR_IMASK[n].VALUE` field indicates whether an interrupt is masked (when set (=1)) or unmasked (=0).	
		1	A55 SCU Err
		2	A55 L1,L2 Err

A55 Error Status Register

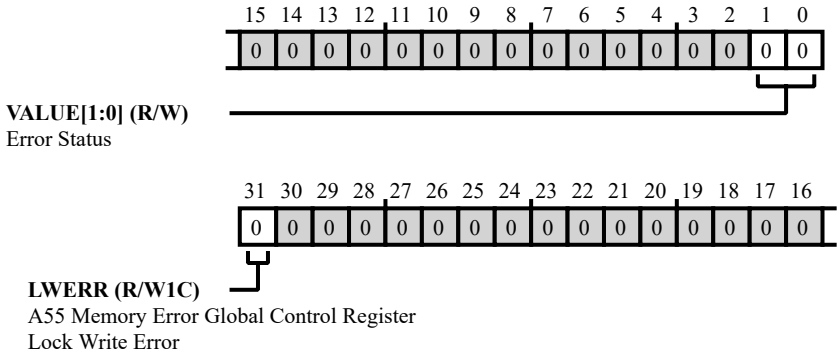


Figure 47-9: MEC_A55C0ERR_STAT[n] Register Diagram

Table 47-19: MEC_A55C0ERR_STAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	A55 Memory Error Global Control Register Lock Write Error. The `MEC_A55C0ERR_STAT[n].LWERR` bit indicates (when set) there was an attempted write to an MEC register while the `MEC_EEIRQ_GCTL[n].LOCK` bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
1:0 (R/W)	VALUE	Error Status. The `MEC_A55C0ERR_STAT[n].VALUE` field indicates the status of an interrupt/trigger. When set (=1), an interrupt or trigger is asserted. When cleared (=0), there is no interrupt or trigger.	
		1	A55 SCU Error
		2	A55 L1,L2 Error

A55 Core Memory Errors Interrupt Request Global Control Register

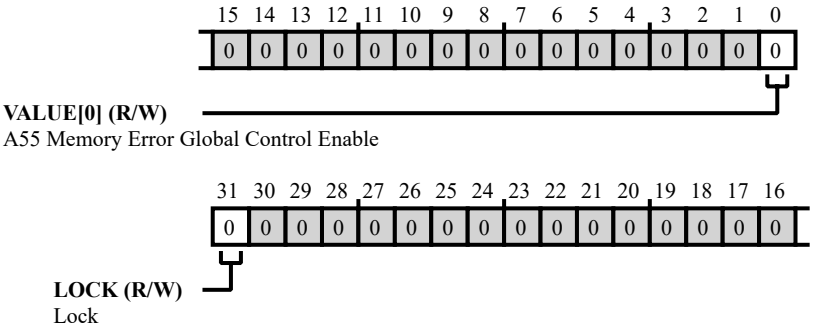


Figure 47-10: MEC_A55C2EIRQ_GCTL[n] Register Diagram

Table 47-20: MEC_A55C2EIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the `MEC_A55C2EIRQ_GCTL[n].LOCK` bit is enabled, the `MEC_A55C2EIRQ_GCTL[n]` register is read only.	
		0	Unlock
		1	Lock
0 (R/W)	VALUE	A55 Memory Error Global Control Enable. The `MEC_A55C2EIRQ_GCTL[n].VALUE` bit enables the A55 memory error global control.	
		0	Disable
		1	Enable

A55 Error Interrupt Request Global Status Register

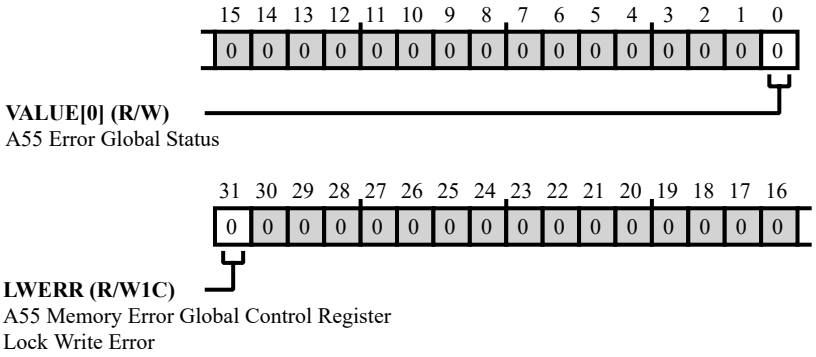


Figure 47-11: MEC_A55C2EIRQ_GSTAT[n] Register Diagram

Table 47-21: MEC_A55C2EIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	A55 Memory Error Global Control Register Lock Write Error. When set (=1), the `MEC_A55C2EIRQ_GSTAT[n].LWERR` bit indicates that there was an attempted write to an MEC register while the `MEC_EEIRQ_GCTL[n].LOCK` bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
0 (R/W)	VALUE	A55 Error Global Status. When set (=1), the `MEC_A55C2EIRQ_GSTAT[n].VALUE` bit indicates the assertion of an interrupt/trigger. When cleared (=0), there is no interrupt or trigger.	
		0	No Interrupt/Trigger
		1	Interrupt/Trigger Asserted

A55 Error Control Register

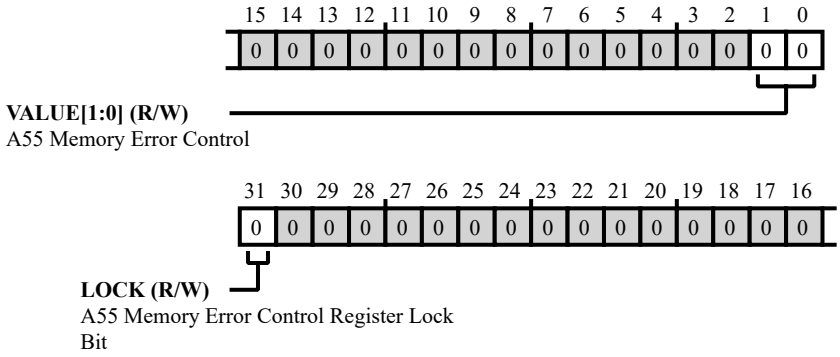


Figure 47-12: MEC_A55C2ERR_CTL[n] Register Diagram

Table 47-22: MEC_A55C2ERR_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	A55 Memory Error Control Register Lock Bit. If the global lock is enabled (SPU_CTL.GLCK =1) and the `MEC_A55C2ERR_CTL[n].LOCK` bit is enabled, the `MEC_A55C2ERR_CTL[n]` register is read only.	
		0	Unlock
		1	Lock
1:0 (R/W)	VALUE	A55 Memory Error Control. The `MEC_A55C2ERR_CTL[n].VALUE` field enables A55 memory error control. When set (=1), control is enabled. When cleared (=0), control is disabled.	
		1	A55 SCU Err
		2	A55 L1,L2 Err

A55 Error Interrupt Mask Register

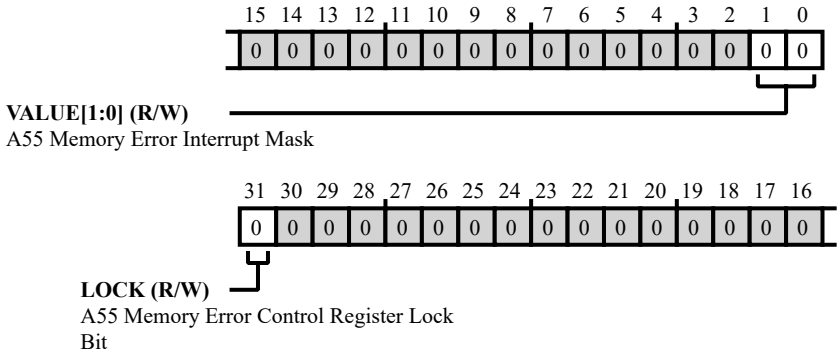


Figure 47-13: MEC_A55C2ERR_IMASK[n] Register Diagram

Table 47-23: MEC_A55C2ERR_IMASK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	A55 Memory Error Control Register Lock Bit. If the global lock is enabled (SPU_CTL.GLCK =1) and the `MEC_A55C2ERR_IMASK[n].LOCK` bit is enabled, the `MEC_A55C2ERR_IMASK[n]` register is read only.	
		0	Unlock
		1	Lock
1:0 (R/W)	VALUE	A55 Memory Error Interrupt Mask. The `MEC_A55C2ERR_IMASK[n].VALUE` field indicates whether an interrupt is masked (when set (=1)) or unmasked (=0).	
		1	A55 SCU Err
		2	A55 L1,L2 Err

A55 Error Status Register

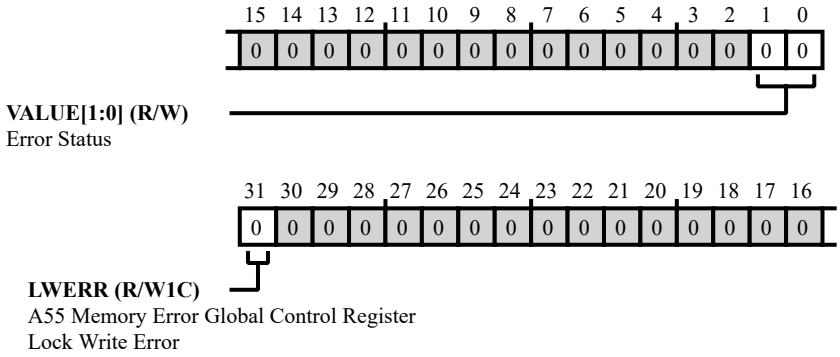


Figure 47-14: MEC_A55C2ERR_STAT[n] Register Diagram

Table 47-24: MEC_A55C2ERR_STAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	A55 Memory Error Global Control Register Lock Write Error. The `MEC_A55C2ERR_STAT[n].LWERR` bit indicates (when set) there was an attempted write to an MEC register while the `MEC_EEIRQ_GCTL[n].LOCK` bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
1:0 (R/W)	VALUE	Error Status. The `MEC_A55C2ERR_STAT[n].VALUE` field indicates the status of an interrupt/trigger. When set (=1), an interrupt or trigger is asserted. When cleared (=0), there is no interrupt or trigger.	
		1	A55 SCU Error
		2	A55 L1,L2 Error

Component ID0 Register

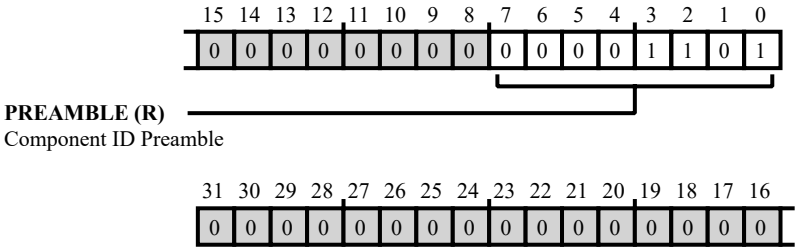


Figure 47-15: MEC_CID0 Register Diagram

Table 47-25: MEC_CID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PREAMBLE	Component ID Preamble. The `MEC_CID0.PREAMBLE` field indicates the component ID preamble.

Component ID1 Register

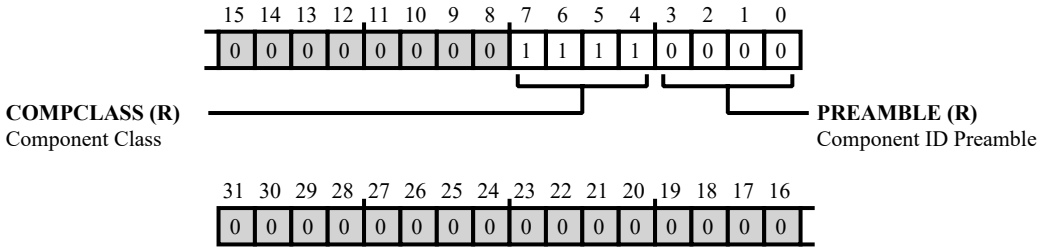


Figure 47-16: MEC_CID1 Register Diagram

Table 47-26: MEC_CID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	COMPCLASS	Component Class. The `MEC_CID1.COMPCLASS` field indicates the component class. Dedicated debug blocks (core debug access port, Program Trace, etc.) should identify as CoreSight (i.e. 0x9) and implement the full compliment of CoreSight registers including DEVTYPE. All other ADI components should identify as System (i.e. 0xF) components. See CoreSight Architecture Specification for details.	
		9	CoreSight
		15	System
3:0 (R/NW)	PREAMBLE	Component ID Preamble. The `MEC_CID1.PREAMBLE` field indicates the component ID preamble.	

Component ID2 Register

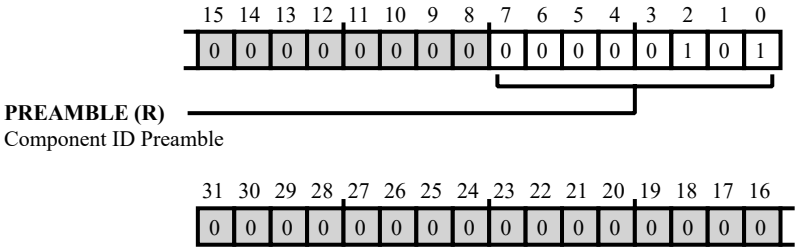


Figure 47-17: MEC_CID2 Register Diagram

Table 47-27: MEC_CID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PREAMBLE	Component ID Preamble. The `MEC_CID2.PREAMBLE` field indicates the component ID preamble.

Component ID3 Register

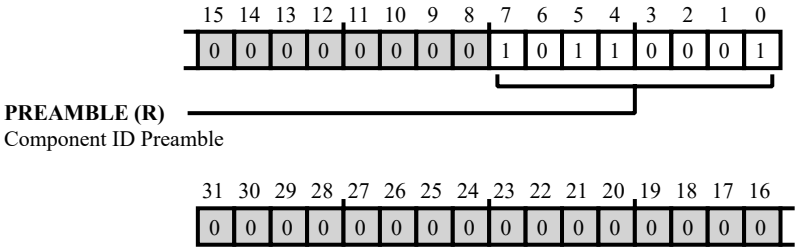


Figure 47-18: MEC_CID3 Register Diagram

Table 47-28: MEC_CID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PREAMBLE	Component ID Preamble. The `MEC_CID3.PREAMBLE` field indicates the component ID preamble.

Clear Register

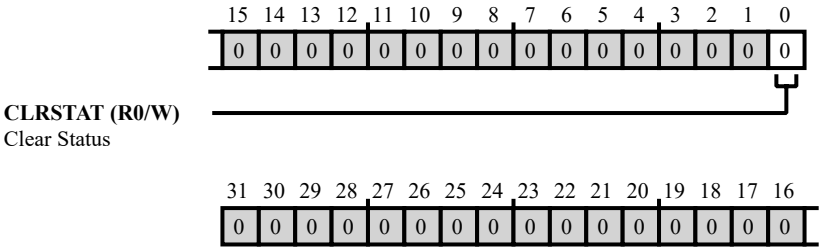


Figure 47-19: MEC_CLR Register Diagram

Table 47-29: MEC_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R0/W)	CLRSTAT	Clear Status. Writing 1 to the `MEC_CLR.CLRSTAT` bit clears all status registers of MEC.

Core Mem ECC Error Interrupt Request Global Control Register

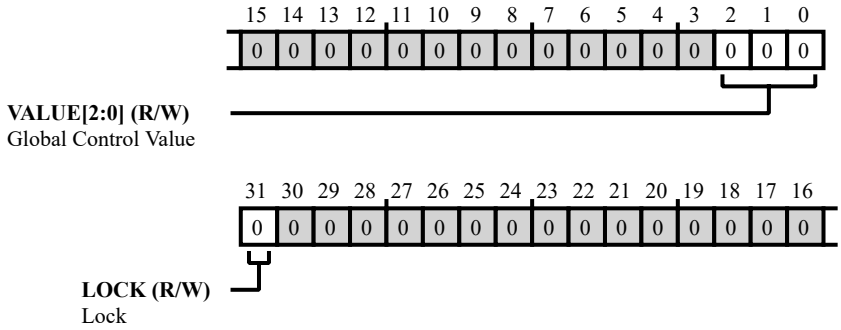


Figure 47-20: MEC_CMEIRQ_GCTL[n] Register Diagram

Table 47-30: MEC_CMEIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock.	
		0	Unlock
		1	Lock
2:0 (R/W)	VALUE	Global Control Value.	

Core Memory ECC Error Interrupt Request Global Status Register

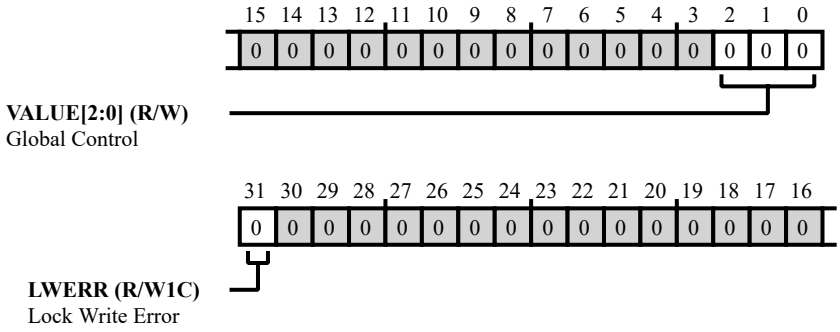


Figure 47-21: MEC_CMEIRQ_GSTAT[n] Register Diagram

Table 47-31: MEC_CMEIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Lock Write Error. The `MEC_CMEIRQ_GSTAT[n].LWERR` bit indicates (when set) there was an attempted write to an MEC register while the `MEC_EEIRQ_GCTL[n].LOCK` bit was set and while the global lock bit was enabled (SPU_CTL.GLCK = 1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
2:0 (R/W)	VALUE	Global Control. '1' indicates enable and '0' indicates disable.	

Core Memory ECC Error Control Register

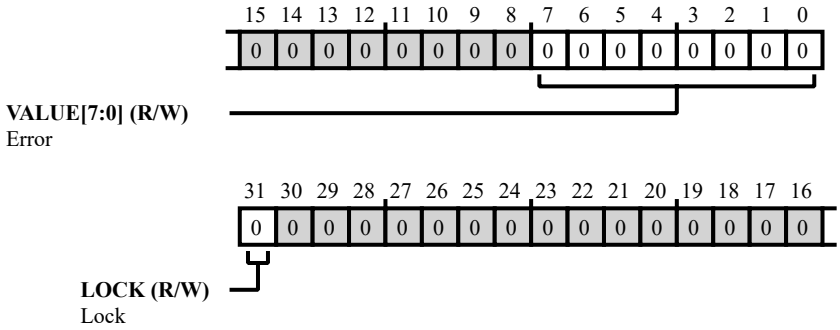


Figure 47-22: MEC_CMERR_CTL[n] Register Diagram

Table 47-32: MEC_CMERR_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. The bit indicates the core memory error control register lock status.	
		0	Unlock
		1	Lock
7:0 (R/W)	VALUE	Error. '1' indicates enable and '0' indicates disable	
		1	iPref RAM ECC error clean-uncorrectable
		2	dPref RAM ECC error dirty-uncorrectable
		4	Reserved
		8	iCache ECC error uncorrectable
		16	dCache ECC error dirty-uncorrectable
		32	dCache ECC error clean-uncorrectable
		64	iRAM ECC error uncorrectable
		128	dRAM ECC error uncorrectable

Core Memory ECC Error Interrupt Mask Register

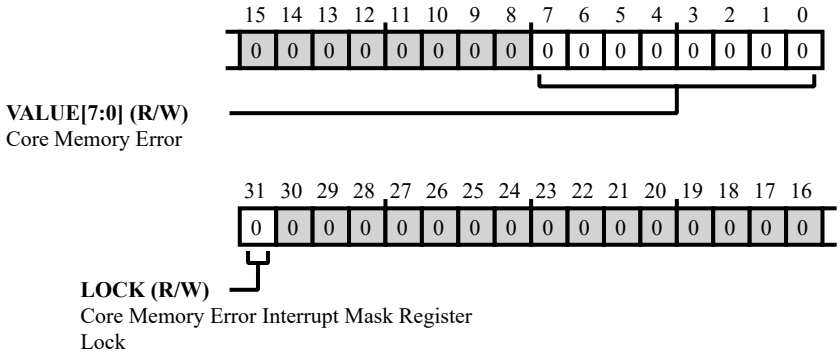


Figure 47-23: MEC_CMERR_IMASK[n] Register Diagram

Table 47-33: MEC_CMERR_IMASK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Core Memory Error Interrupt Mask Register Lock.	
		0	Unlock
		1	Lock
7:0 (R/W)	VALUE	Core Memory Error. '1' indicates enable and '0' indicates disable.	
		1	iPref RAM ECC error clean-uncorrectable
		2	dPref RAM ECC error dirty-uncorrectable
		4	Reserved
		8	iCache ECC error uncorrectable
		16	dCache ECC error dirty-uncorrectable
		32	dCache ECC error clean-uncorrectable
		64	iRAM ECC error uncorrectable
		128	dRAM ECC error uncorrectable

Core Memory ECC Error Status Register

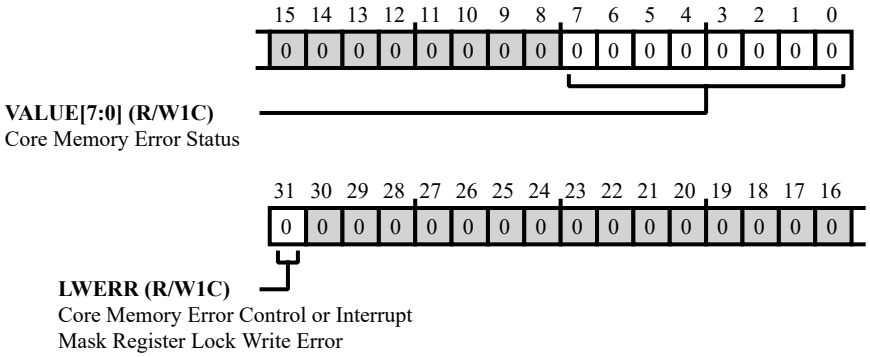


Figure 47-24: MEC_CMERR_STAT[n] Register Diagram

Table 47-34: MEC_CMERR_STAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Core Memory Error Control or Interrupt Mask Register Lock Write Error. The `MEC_CMERR_STAT[n].LWERR` bit indicates (when set) there was an attempted write to an MEC register while the `MEC_EEIRQ_GCTL[n].LOCK` bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
7:0 (R/W1C)	VALUE	Core Memory Error Status. '1' indicates assertion of interrupt/trigger and '0' indicates no interrupt/trigger.	
		1	iPref RAM ECC error clean-uncorrectable
		2	dPref RAM ECC error dirty-uncorrectable
		4	Reserved
		8	iCache ECC error uncorrectable
		16	dCache ECC error dirty-uncorrectable
		32	dCache ECC error clean-uncorrectable
		64	iRAM ECC error uncorrectable
		128	dRAM ECC error uncorrectable

ECC Error Control Register

The MEC_ECCERR_CTL[n] register bits control enable/disable for ECC error inputs from various cores/peripherals and decide whether their status will be reflected in ECC error status register bits.

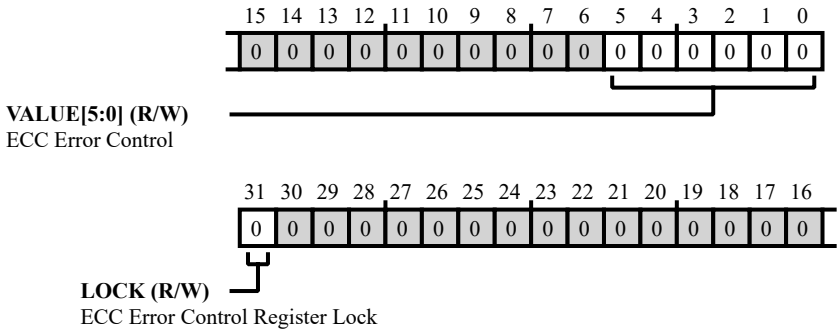


Figure 47-25: MEC_ECCERR_CTL[n] Register Diagram

Table 47-35: MEC_ECCERR_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	ECC Error Control Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_ECCERR_CTL[n].LOCK bit is enabled, the MEC_ECCERR_CTL[n] register is read only.	
		0	Unlock
		1	Lock
5:0 (R/W)	VALUE	ECC Error Control. The MEC_ECCERR_CTL[n].VALUE bit indicates whether ECC control is enabled (=1) or disabled (=0).	
		1	L2 CTL ECC Error
		2	CAN0 ECC Error
		4	CAN1 ECC Error
		8	Reserved
		16	L2 CTL 1 ECC Error

ECC Error Interrupt Mask Register

The MEC_ECCERR_IMASK[n] register bits control interrupt masks for ECC error inputs from various cores/peripherals.

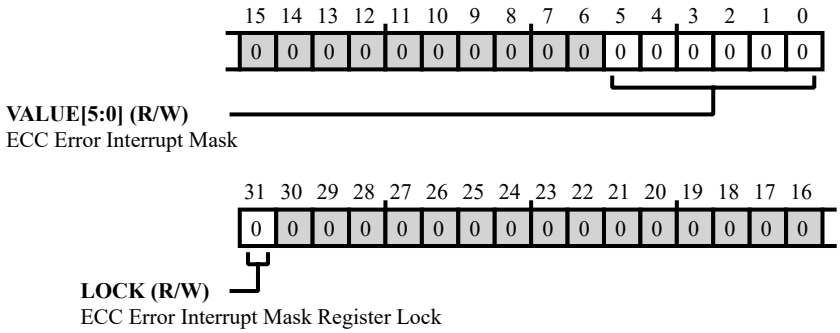


Figure 47-26: MEC_ECCERR_IMASK[n] Register Diagram

Table 47-36: MEC_ECCERR_IMASK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	ECC Error Interrupt Mask Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_ECCERR_IMASK[n].LOCK bit is enabled, the MEC_ECCERR_IMASK[n] register is read only.	
		0	Unlock
		1	Lock
5:0 (R/W)	VALUE	ECC Error Interrupt Mask. The MEC_ECCERR_IMASK[n].VALUE bit indicates when the ECC error interrupt is masked (=1) or unmasked (=0).	
		1	L2 CTL ECC Error
		2	CAN0 ECC Error
		4	CAN1 ECC Error
		8	Reserved
		16	L2 CTL 1 ECC Error

ECC Error Status Register

The MEC_ECCERR_STAT[n] register bits reflect status for ECC error inputs from various cores/peripherals. Writing '1' to these bits clear corresponding ECC error status.

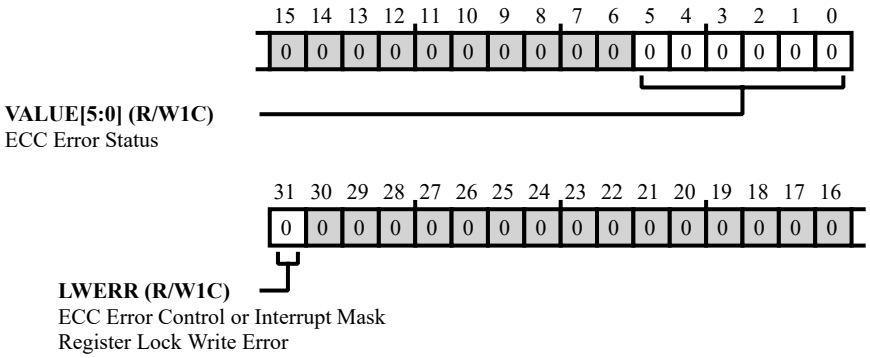


Figure 47-27: MEC_ECCERR_STAT[n] Register Diagram

Table 47-37: MEC_ECCERR_STAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	ECC Error Control or Interrupt Mask Register Lock Write Error. The MEC_ECCERR_STAT[n].LWERR bit indicates (when set) there was an attempted write to an MEC register while the bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write 1 to clear it.	
		0	No Error
		1	Error Occurred
5:0 (R/W1C)	VALUE	ECC Error Status. '1' indicates error and '0' indicates no error. Sticky bit, write '1' to clear.	
		1	L2 CTL ECC Error
		2	CAN0 ECC Error
		4	CAN1 ECC Error
		8	Reserved
		16	L2 CTL 1 ECC Error

ECC Error Interrupt Request Global Control Register

The MEC_EEIRQ_GCTL[n] register bits control enable/disable of ECC error interrupt/trigger outputs.

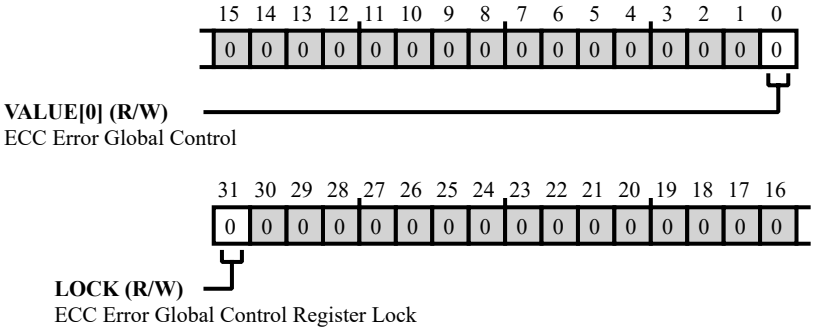


Figure 47-28: MEC_EEIRQ_GCTL[n] Register Diagram

Table 47-38: MEC_EEIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	ECC Error Global Control Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_EEIRQ_GCTL[n].LOCK bit is enabled, the MEC_EEIRQ_GCTL[n] register is read only.	
		0	Unlock
		1	Lock
0 (R/W)	VALUE	ECC Error Global Control. '1' indicates enable and '0' indicates disable.	

ECC Error Interrupt Request Global Status Register

The `MEC_EEIRQ_GSTAT[n]` register bits reflect the status of the ECC error interrupt/trigger outputs.

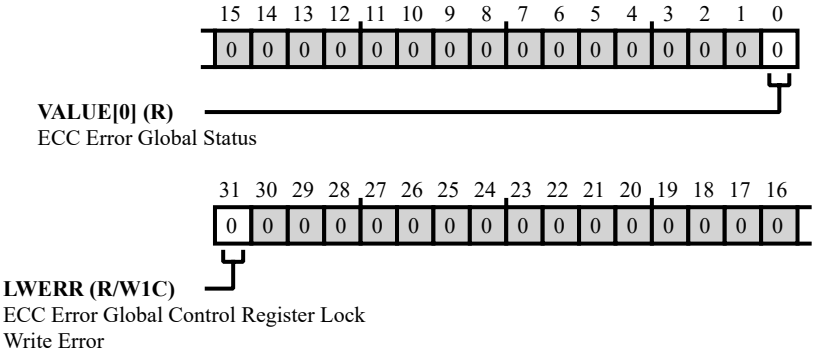


Figure 47-29: MEC_EEIRQ_GSTAT[n] Register Diagram

Table 47-39: MEC_EEIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	ECC Error Global Control Register Lock Write Error. The `MEC_EEIRQ_GSTAT[n].LWERR` bit indicates (when set) there was an attempted write to an MEC register while the bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write 1 to clear it.	
		0	No Error
		1	Error Occurred
0 (R/NW)	VALUE	ECC Error Global Status. '1' indicates assertion of interrupt/trigger and '0' indicates no interrupt/trigger.	

Parity Error Interrupt Request Global Control Register

The MEC_PEIRQ_GCTL[n] register bits control enable/disable of parity error interrupt/trigger outputs.

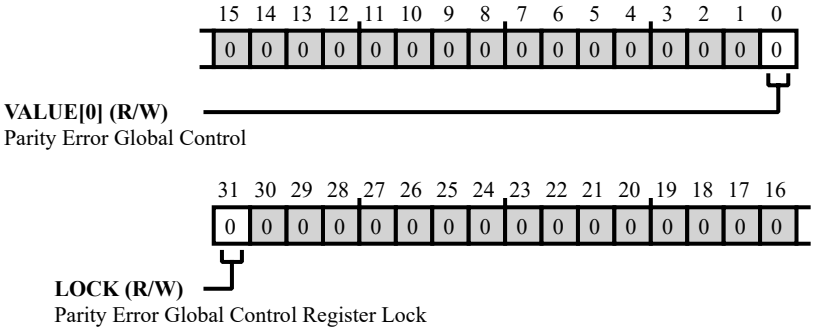


Figure 47-30: MEC_PEIRQ_GCTL[n] Register Diagram

Table 47-40: MEC_PEIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Parity Error Global Control Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PEIRQ_GCTL[n].LOCK bit is enabled, the MEC_PEIRQ_GCTL[n] register is read only.	
		0	Unlock
		1	Lock
0 (R/W)	VALUE	Parity Error Global Control. The MEC_PEIRQ_GCTL[n].VALUE bit field indicates whether parity error control is enabled (=1) or disabled (=0).	

Parity Error Interrupt Request Global Status Register

The MEC_PEIRQ_GSTAT[n] register bits reflect status of parity error interrupt/trigger outputs.

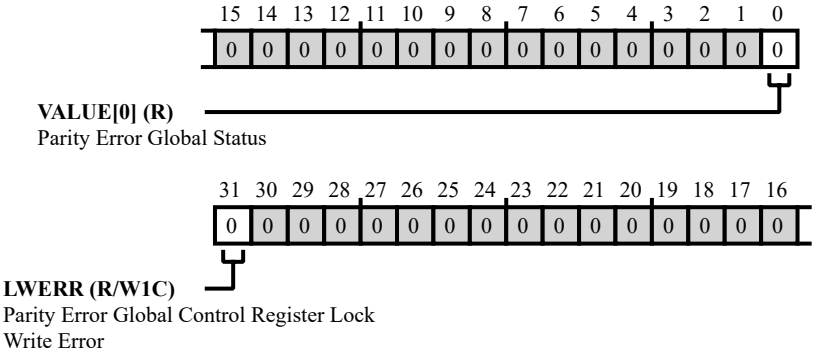


Figure 47-31: MEC_PEIRQ_GSTAT[n] Register Diagram

Table 47-41: MEC_PEIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Parity Error Global Control Register Lock Write Error. When set (=1), the MEC_PEIRQ_GSTAT[n].LWERR bit indicates an attempted write to an MEC register while the MEC_PEIRQ_GCTL[n].LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write 1 to clear it.	
		0	No Error
		1	Error Occurred
0 (R/NW)	VALUE	Parity Error Global Status. The MEC_PEIRQ_GSTAT[n].VALUE field indicates the parity error global status. 1 indicates the assertion of the interrupt/trigger; 0 indicates no interrupt/trigger.	

Parity Error Control Register

The MEC_PERR_CTL0 register bits control enable/disable for parity error inputs from various cores/peripherals and decide whether their status will be reflected in parity error status register bits.

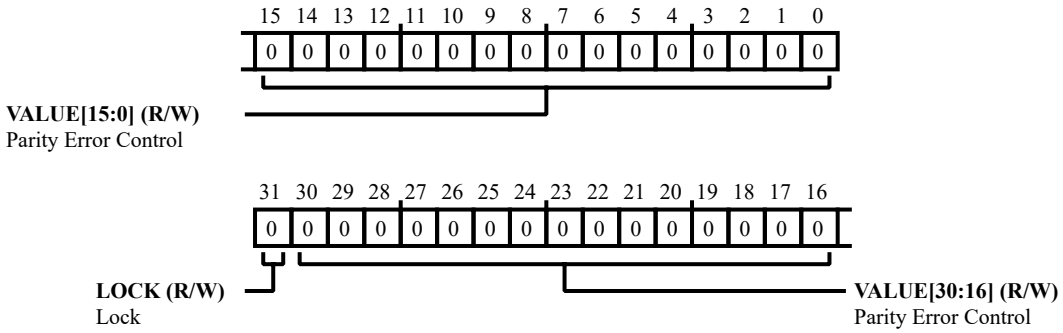


Figure 47-32: MEC_PERR_CTL0 Register Diagram

Table 47-42: MEC_PERR_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_CTL0.LOCK bit is enabled, the MEC_PERR_CTL0 register is read only.	
		0	Unlock
		1	Lock
30:0 (R/W)	VALUE	Parity Error Control. The MEC_PERR_CTL0.VALUE bit indicates whether parity error control is enabled (=1) or disabled (=0).	
		4	Core 1 L1 RAM Parity Error
		8	Core 1 L1 Cache Parity Error
		16	Core 1 Branch Predictor Parity Error
		256	ASRC0 FIFO Parity Error
		512	ASRC1 FIFO Parity Error
		1024	ASRC2 FIFO Parity Error
		2048	ASRC3 FIFO Parity Error
		4096	ASRC4 FIFO Parity Error
		8192	ASRC5 FIFO Parity Error
		16384	ASRC6 FIFO Parity Error

Table 47-42: MEC_PERR_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		32768	ASRC7 FIFO Parity Error
		65536	SHARC 0 IIR0 RAM Parity Error
		131072	SHARC 0 IIR1 RAM Parity Error
		262144	SHARC 0 IIR2 RAM Parity Error
		524288	SHARC 0 IIR3 RAM Parity Error
		1048576	SHARC 0 FIR0 RAM Parity Error
		33554432	SHARC0 FIR1 RAM Parity Error
		134217728	TRNG0 Data Buffer Parity Error
		268435456	PKA0 Data RAM Parity Error
		536870912	SPE0 Buffer RAM Parity Error
		1073741824	SPE0 ARC4 RAM Parity Error

Parity Error Control Register

MEC_PERR_CTL1 register bits control enable/disable for parity error inputs from various cores/peripherals and decide whether their status will be reflected in parity error status register bits.

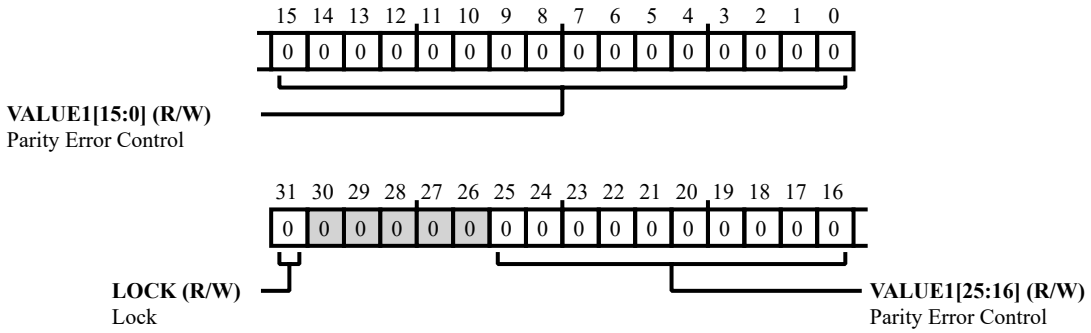


Figure 47-33: MEC_PERR_CTL1 Register Diagram

Table 47-43: MEC_PERR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_CTL1.LOCK bit is enabled, the MEC_PERR_CTL1 register is read only.	
		0	Unlock
		1	Lock
25:0 (R/W)	VALUE1	Parity Error Control. The MEC_PERR_CTL1.VALUE1 field indicates the parity error inputs. 1 indicates enable and 0 indicates disable.	
		1	EMAC0 Transmit FIFO RAM Parity Error
		2	EMAC0 Receive FIFO RAM Parity Error
		4	MLB0 Data Buffer RAM Parity Error
		8	MLB0 Channel Table RAM Parity Error
		16	TMC0 Trace Data RAM Parity Error
		128	EMAC0 TMI FIFO RAM Parity Error
		256	EMAC0 EST FIFO RAM Parity Error
		512	EMSI0 FIFO RAM Parity Error
		1024	ASRC8 FIFO RAM Parity Error
		2048	ASRC9 FIFO RAM Parity Error

Table 47-43: MEC_PERR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4096	ASRC10 FIFO RAM Parity Error
		8192	ASRC11 FIFO RAM Parity Error
		16384	ASRC12 FIFO RAM Parity Error
		32768	ASRC13 FIFO RAM Parity Error
		65536	ASRC14 FIFO RAM Parity Error
		131072	ASRC15 FIFO RAM Parity Error
		262144	XSPI Flash RAM Parity Error
		524288	XSPI CTRL RAM Parity Error
		1048576	XSPI 1 Flash RAM Parity Error
		2097152	XSPI 1 CTRL RAM Parity Error
		16777216	LPDDR ICCM Parity Error
		33554432	LPDDR DCCM Parity Error

Parity Error Interrupt Mask Register

The MEC_PERR_IMASK0 register bits control interrupt masks for parity error inputs from various cores/peripherals.

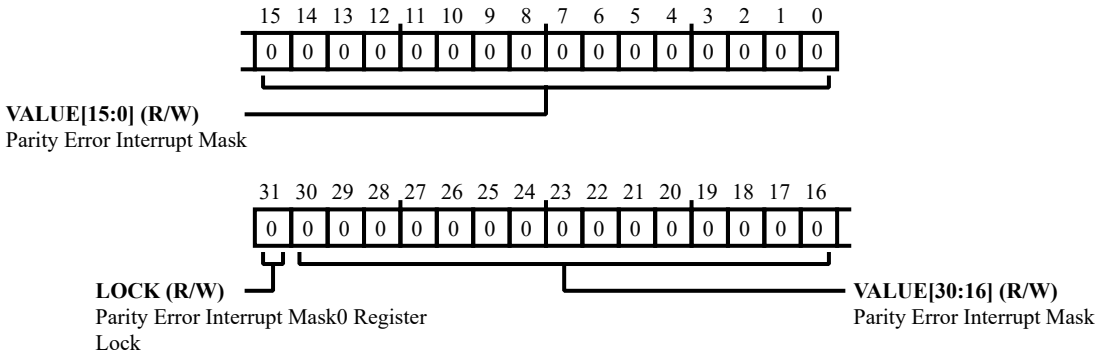


Figure 47-34: MEC_PERR_IMASK0 Register Diagram

Table 47-44: MEC_PERR_IMASK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Parity Error Interrupt Mask0 Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_IMASK0.LOCK bit is enabled, the MEC_PERR_IMASK0 register is read only.	
		0	Unlock
		1	Lock
30:0 (R/W)	VALUE	Parity Error Interrupt Mask. The MEC_PERR_IMASK0.VALUE bit field, when set (=1), indicates an interrupt masked and, when cleared (=0), indicates an interrupt unmasked.	
		4	Core 1 L1 RAM Parity Error
		8	Core 1 L1 Cache Parity Error
		16	Core 1 Branch Predictor Parity Error
		256	ASRC0 FIFO Parity Error
		512	ASRC1 FIFO Parity Error
		1024	ASRC2 FIFO Parity Error
		2048	ASRC3 FIFO Parity Error
		4096	ASRC4 FIFO Parity Error
		8192	ASRC5 FIFO Parity Error
		16384	ASRC6 FIFO Parity Error

Table 47-44: MEC_PERR_IMASK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		32768	ASRC7 FIFO Parity Error
		65536	SHARC 0 IIR0 RAM Parity Error
		131072	SHARC 0 IIR1 RAM Parity Error
		262144	SHARC 0 IIR2 RAM Parity Error
		524288	SHARC 0 IIR3 RAM Parity Error
		1048576	SHARC 0 FIR0 RAM Parity Error
		33554432	SHARC0 FIR1 RAM Parity Error
		134217728	TRNG0 Data Buffer Parity Error
		268435456	PKA0 Data RAM Parity Error
		536870912	SPE0 Buffer RAM Parity Error
		1073741824	SPE0 ARC4 RAM Parity Error

Parity Error Interrupt Mask Register

MEC_PERR_IMASK1 register bits control interrupt masks for parity error inputs from various cores/peripherals.

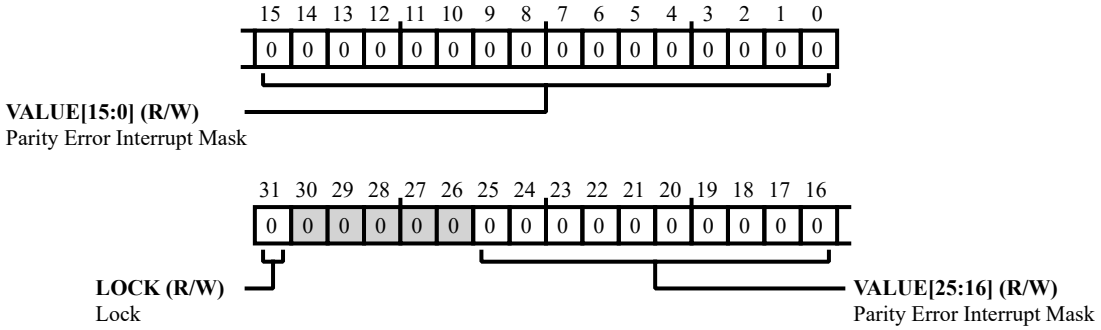


Figure 47-35: MEC_PERR_IMASK1 Register Diagram

Table 47-45: MEC_PERR_IMASK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_IMASK1.LOCK bit is enabled, the MEC_PERR_IMASK1 register is read only.	
		0	Unlock
		1	Lock
25:0 (R/W)	VALUE	Parity Error Interrupt Mask. '1' indicates interrupt masked and '0' indicates interrupt unmasked.	
		1	EMAC0 Transmit FIFO RAM Parity Error
		2	EMAC0 Receive FIFO RAM Parity Error
		4	MLB0 Data Buffer RAM Parity Error
		8	MLB0 Channel Table RAM Parity Error
		16	TMC0 Trace Data RAM Parity Error
		128	EMAC0 TMI FIFO RAM Parity Error
		256	EMAC0 EST FIFO RAM Parity Error
		512	EMSI0 FIFO RAM Parity Error
		1024	ASRC8 FIFO RAM Parity Error
		2048	ASRC9 FIFO RAM Parity Error
		4096	ASRC10 FIFO RAM Parity Error
		8192	ASRC11 FIFO RAM Parity Error

Table 47-45: MEC_PERR_IMASK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		16384	ASRC12 FIFO RAM Parity Error
		32768	ASRC13 FIFO RAM Parity Error
		65536	ASRC14 FIFO RAM Parity Error
		131072	ASRC15 FIFO RAM Parity Error
		262144	XSPI Flash RAM Parity Error
		524288	XSPI CTRL RAM Parity Error
		1048576	XSPI 1 Flash RAM Parity Error
		2097152	XSPI 1 CTRL RAM Parity Error
		16777216	LPDDR ICCM Parity Error
		33554432	LPDDR DCCM Parity Error

Parity Error Status Register

The MEC_PERR_STAT0 register bits reflect status for parity error inputs from various cores/peripherals. Writing '1' to these bits clear corresponding parity error status.

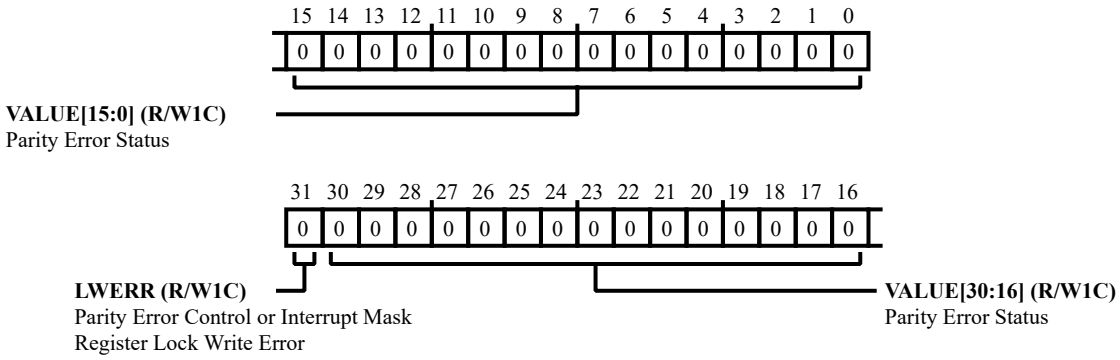


Figure 47-36: MEC_PERR_STAT0 Register Diagram

Table 47-46: MEC_PERR_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Parity Error Control or Interrupt Mask Register Lock Write Error. The MEC_PERR_STAT0.LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_PERR_CTL0.LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
30:0 (R/W1C)	VALUE	Parity Error Status. When set (=1), the MEC_PERR_STAT0.VALUE bit indicates the parity error status. When cleared (=0), the bit indicates no error. This is a sticky bit; write 1 to clear.	
		4	Core 1 L1 RAM Parity Error
		8	Core 1 L1 Cache Parity Error
		16	Core 1 Branch Predictor Parity Error
		256	ASRC0 FIFO Parity Error
		512	ASRC1 FIFO Parity Error
		1024	ASRC2 FIFO Parity Error
		2048	ASRC3 FIFO Parity Error
		4096	ASRC4 FIFO Parity Error
		8192	ASRC5 FIFO Parity Error

Table 47-46: MEC_PERR_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		16384	ASRC6 FIFO Parity Error
		32768	ASRC7 FIFO Parity Error
		65536	SHARC 0 IIR0 RAM Parity Error
		131072	SHARC 0 IIR1 RAM Parity Error
		262144	SHARC 0 IIR2 RAM Parity Error
		524288	SHARC 0 IIR3 RAM Parity Error
		1048576	SHARC 0 FIR0 RAM Parity Error
		33554432	SHARC 0 FIR1 RAM Parity Error
		134217728	TRNG0 Data Buffer Parity Error
		268435456	PKA0 Data RAM Parity Error
		536870912	SPE0 Buffer RAM Parity Error
		1073741824	SPE0 ARC4 RAM Parity Error

Parity Error Status Register

MEC_PERR_STAT1 register bits reflect status for parity error inputs from various cores/peripherals. Writing '1' to these bits clear corresponding parity error status.

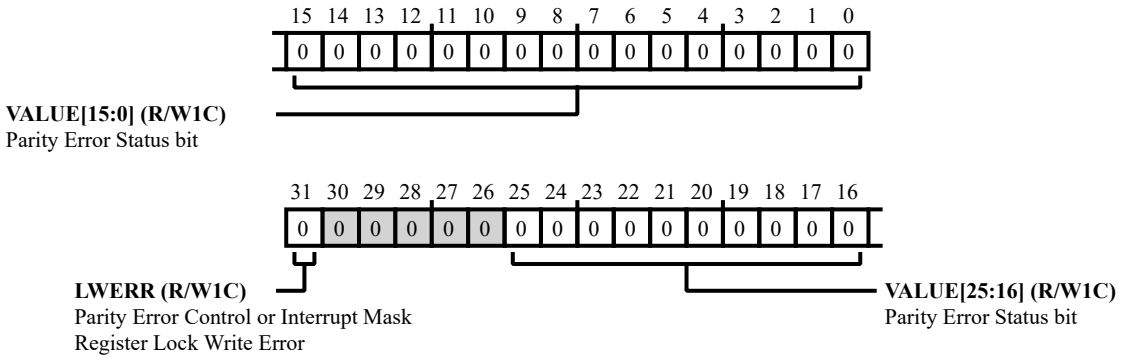


Figure 47-37: MEC_PERR_STAT1 Register Diagram

Table 47-47: MEC_PERR_STAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Parity Error Control or Interrupt Mask Register Lock Write Error. The MEC_PERR_STAT1.LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_PERR_CTL1.LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
25:0 (R/W1C)	VALUE	Parity Error Status bit. 1 indicates error and 0 indicates no error. Sticky bit, write 1 to clear.	
		1	EMAC0 Transmit FIFO RAM Parity Error
		2	EMAC0 Receive FIFO RAM Parity Error
		4	MLB0 Data Buffer RAM Parity Error
		8	MLB0 Channel Table RAM Parity Error
		16	TMC0 Trace Data RAM Parity Error
		128	EMAC0 TMI FIFO RAM Parity Error
		256	EMAC0 EST FIFO RAM Parity Error
		512	EMSI0 FIFO RAM Parity Error
		1024	ASRC8 FIFO RAM Parity Error
		2048	ASRC9 FIFO RAM Parity Error

Table 47-47: MEC_PERR_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4096	ASRC10 FIFO RAM Parity Error
		8192	ASRC11 FIFO RAM Parity Error
		16384	ASRC12 FIFO RAM Parity Error
		32768	ASRC13 FIFO RAM Parity Error
		65536	ASRC14 FIFO RAM Parity Error
		131072	ASRC15 FIFO RAM Parity Error
		262144	XSPI Flash RAM Parity Error
		524288	XSPI CTRL RAM Parity Error
		1048576	XSPI 1 Flash RAM Parity Error
		2097152	XSPI 1 CTRL RAM Parity Error
		16777216	LPDDR ICCM Parity Error
		33554432	LPDDR DCCM Parity Error

Peripheral ID0 Register

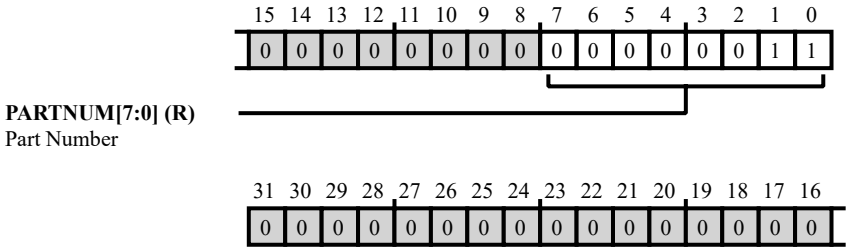


Figure 47-38: MEC_PID0 Register Diagram

Table 47-48: MEC_PID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PARTNUM	Part Number. The `MEC_PID0.PARTNUM` field indicates the part number for component identification

Peripheral ID1 Register

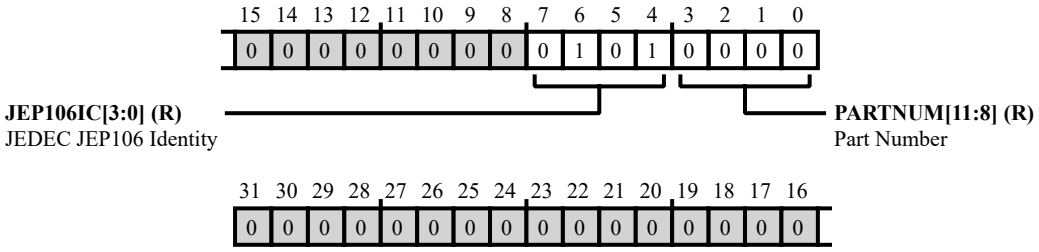


Figure 47-39: MEC_PID1 Register Diagram

Table 47-49: MEC_PID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	JEP106IC	JEDEC JEP106 Identity. The `MEC_PID1.JEP106IC` field indicates JEDEC JEP106 identity (manufacturer ID) code.
3:0 (R/NW)	PARTNUM	Part Number. The `MEC_PID1.PARTNUM` field indicates the part number for component identification

Peripheral ID2 Register

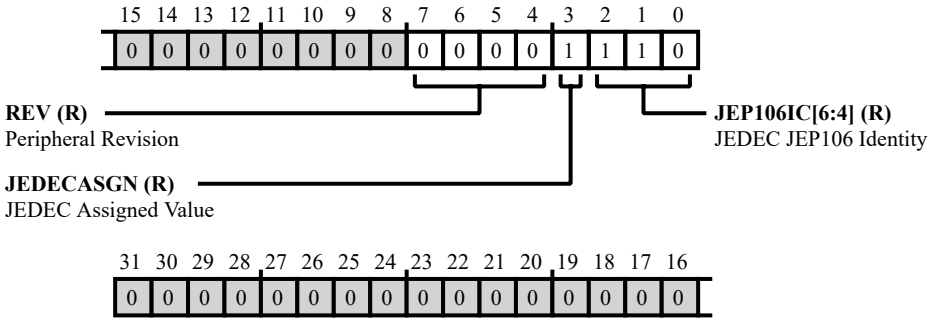


Figure 47-40: MEC_PID2 Register Diagram

Table 47-50: MEC_PID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV	Peripheral Revision.
3 (R/NW)	JEDECASGN	JEDEC Assigned Value.
2:0 (R/NW)	JEP106IC	JEDEC JEP106 Identity. The `MEC_PID2.JEP106IC` field indicates JEDEC JEP106 identity (manufacturer ID) code.

Peripheral ID3 Register

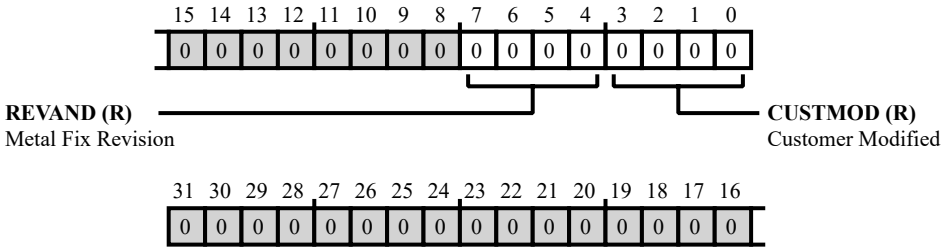


Figure 47-41: MEC_PID3 Register Diagram

Table 47-51: MEC_PID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REVAND	Metal Fix Revision.
3:0 (R/NW)	CUSTMOD	Customer Modified.

Peripheral ID4 Register

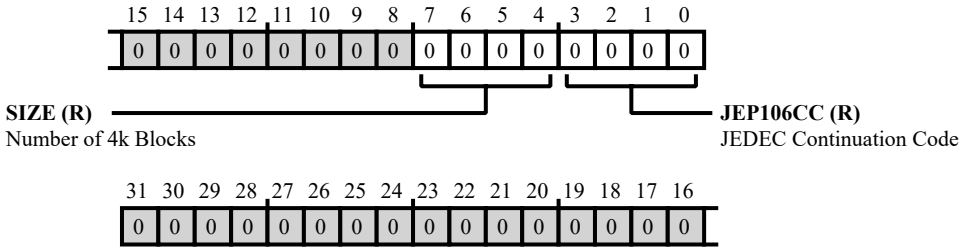


Figure 47-42: MEC_PID4 Register Diagram

Table 47-52: MEC_PID4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	SIZE	Number of 4k Blocks. The `MEC_PID4.SIZE` field indicates the size of component 4k chunks minus 1 (for example, 0=4k).
3:0 (R/NW)	JEP106CC	JEDEC Continuation Code. The `MEC_PID4.JEP106CC` field indicates the JEDEC JEP106 continuation code (number of leading 0x7Fs).

48 System Debug and Trace Unit (DBG)

The system debug and trace unit is based on Arm Core Sight technology. CoreSight is a set of architecture specifications defining debug and trace architecture. The processor uses CoreSight infrastructure to provide industry standard debug and trace capabilities.

http://infocenter.arm.com/help/

The applicable documentation for more details about the ArmCoreSight feature includes:

- CoreSight PFT Architecture Specification, ARM IHI 0035B (PFT)
- System Trace Macrocell, Programmers' Model Architecture Specification, ARM IHI 0054A (STM)
- CoreSight Trace Memory Controller, ARM DDI0461B (TMC)
- CoreSight Components Technical Reference Manual, ARM DDI 0314H (TPIU)
- Embedded Cross Trigger Technical Reference Manual, ARM DDI 0291A
- ETM Architecture Specification, ARM IHI0014Q

DBG Features

The debug unit has the following features.

- System JTAG TAP controller for system debug features, boundary scan, and public JTAG features
- A debug interface to core, and other system resources
- Direct and run-time access to the memory system and system MMRs
- Direct control over system reset
- Support for debug immediately after reset (boot debug)
- Group halt (debug event immediately halts all specified endpoints)
- Real-time on-chip visibility is made available to all developers, including software developers

Debug Functional Description

The following sections provide functional descriptions of the DBG unit.

DBG Block Diagram

The DBG block diagram is shown below.

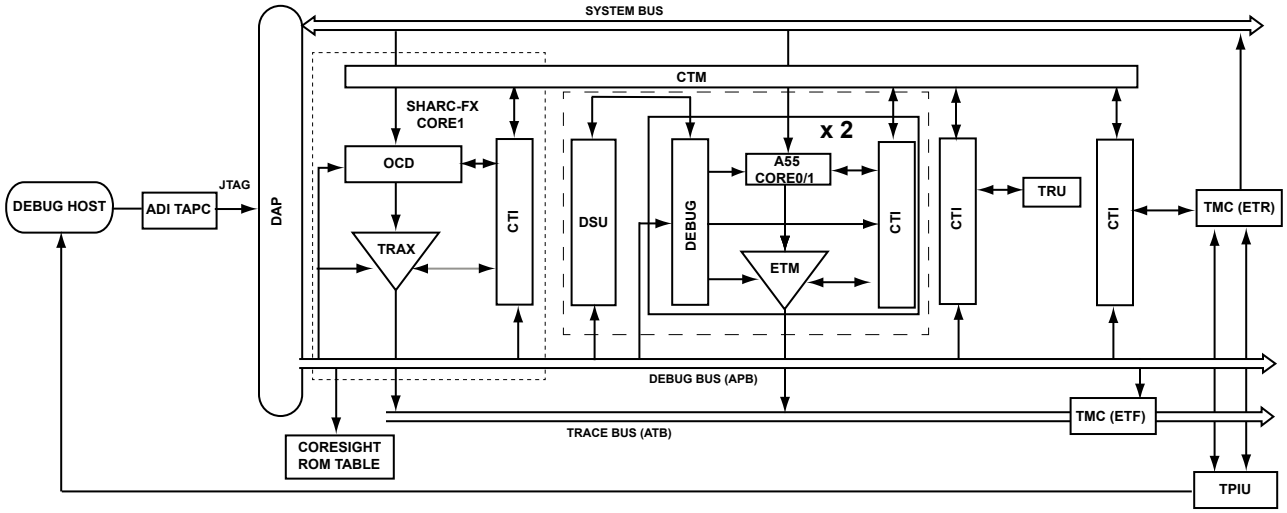


Figure 48-1: Coresight Block Diagram

DBG Definitions

The following terms are useful when working with the debug features of the processor and programming tools.

Test Access Port Controller (TAPC)

Provides IDCODE and SDBGKEY features.

Debug Access Port (DAP)

Core Sight Interface providing a single port for two debug options: JTAG-DP (JTAG Debug Port), SW-DP (Serial Wire Debug Port)

Embedded Trace Macrocell (ETM)

Provides Arm Core Trace.

Program Trace Macrocell (PTM)

Provides DSP (SHARC +) Core Trace.

Standard Trace Macrocell (STM)

Provides capability to trace up to 32 hardware events and supports 32 software stimulus for data transfer between the user and emulator.

Embedded Cross Trigger (ECT)

The ECTs are responsible managing events and triggers as follows.

- CTI (Cross Trigger Interface) is a CoreSight component for enabling cross triggering of events across a system. From the view of the ECT, it is responsible for combining and mapping trigger requests.
- CTM (Cross Trigger Matrix) is a CoreSight component for connecting multiple CTIs. From the view of the ECT, it is responsible for connecting CTIs and distribution of events.

NOTE: An embedded cross trigger is not the same as the requester and completer trigger in the trigger routing unit.

Trace Capture Devices

The trace capture devices capture and format the trace data. There are two trace capture devices in the system:

- ETF - Embedded Trace FIFO - Provides a buffer for burst trace data.
- ETR - Embedded Trace Router - Provides interface for trace data to be stored in system memories.

Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the on-chip trace data, with separate IDs, to a data stream. It encapsulates IDs, when required, that the Trace Port Analyzer (TPA) captures.

Serial Wire Output (SWO)

SWO is a trace data drain that acts as a bridge between the on-chip trace data and a data stream that the Trace Port Analyzer (TPA) captures.

Test Access Port Controller (TAPC)

TAPC is an independent component daisy-chained to the DAP in the JTAG scan path. The TAPC component provides the product IDCODE (Chip ID) and a security feature.

The TAPC component provides security features to the chip using a debug key match features. This feature permits only those components which have the SDBGKEY to connect and debug the chip. Initially, a 128-bits user key is programmed in the SDBGKEY registers (`TAPC_SDBGKEY0`, `TAPC_SDBGKEY1`, `TAPC_SDBGKEY2`, `TAPC_SDBGKEY3`) in TAPC by a secure requester. The `TAPC_SDBGKEY_CTL.VALID` bit is set.

Then, a user key is entered through the emulator for a match to the initially programmed-entered user key. On a successful match, the chip connects to the emulator. It can be debugged on a failed user key match. Further attempts at keys matching are disabled. JTAG reset or system reset is required to re-enable the user key match logic.

Embedded Trace Macrocell (ETM)

ETM is the standard trace support provided for the Arm processor. For details of programming and functionality, refer to the Arm documentation.

Debug Access Ports

The DAP provides access to all debug and trace capabilities through a single external interface, a JTAG Debug Port (JTAG-DP). The JTAG-DP is based on the IEEE 1149.1 Test Access Port (TAP) and Boundary Scan Architecture. DAP has some additions to support BSCA and IDCODE features that Core Sight DAP does not support.

Trace Unit

The trace module provides instruction, data tracing, and system activity tracing for the processor. The program trace module on the processor is similar to the embedded trace macrocell module provided by the Arm processor. System trace is provided using the system trace macrocell as part of the CoreSight debug and trace interface. The trace module uses an interface based on the AMBA Trace Bus (ATB) standard to output its trace data. The trace data can be either exported to an off-chip trace port analyzer or captured on an on-chip buffer. The PFT and STM modules capture information on the processor both before and after a specific event. The modules add no burden to the processor performance when it runs at full speed.

- Programmable Flow Trace (CSPFT)
- System Trace Module (STM)

Programmable Flow Trace (CSPFT)

When tracing processor execution, trace information can be generated for every instruction the processor executes. This information would be easy to interpret, but would require a prohibitively high trace bandwidth to get the trace data off the chip. With program flow tracing, only branch points are traced. The debugger uses the source code to infer the rest of the executed code.

Certain instructions in the program and events are identified as waypoints. A waypoint is a point where instruction execution involves a change of program flow. The CSPFT only traces those waypoints. These waypoints are:

- All indirect branches
- All direct branches
- Exceptions or interrupts
- Emulator debug entry and exit

When a waypoint occurs, trace data is generated to describe it. From this data and the source code, a trace decompressor can determine what instructions were executed and recreate the instruction flow. To allow the

decompressor to calculate where it is in the source code, conditional instructions are marked as waypoints, regardless of whether they pass or fail their condition test. Events like interrupts or debug entry and exit can be promoted from non-waypoint instructions to waypoints to trace the interrupted program flow.

Tracing a waypoint implies the execution of all instructions from the target address of the previous waypoint up to the current waypoint. Non-waypoint instructions are not explicitly traced but the debugger must infer them using the source code. The concept of an instruction block is used throughout this manual and refers to the contiguous block of instructions between two waypoints.

The programming model and function is a subset of the PTM (Programmable Trace Module) of Arm.

System Trace Module (STM)

The STM is a trace source that is integrated into a CoreSight system, and is designed primarily for high-bandwidth trace of instrumentation embedded into software. The STM enables tracing of system activity from various sources:

- Instrumented software, using memory-mapped stimulus ports
- Hardware events

The STM supports the following features:

- Multiple software requesters writing software instrumentation independently. Each requester can use multiple stimulus ports.
- Time stamping of the system activity. The time stamp is a global time stamp which can be shared with other trace sources in the system to enable correlation of activity from multiple trace sources.
- Indication that specific events have occurred, such as a particular hardware event or a piece of software instrumentation. These events are known as triggers and can be indicated in the trace stream, or through signals to other system components.

Embedded Cross Trigger (ECT)

ECT provides an interface to the CoreSight debug system enabling the subsystems to interact (cross trigger) with each other. ECT provides a mechanism to forward debug events from one connected subsystem to another connected subsystem. The different subsystems connected to the ECT depend on the processor design. For example, in a multiprocessor system, the interface can be connected to each of the cores and one to the trace subsystem. For a uniprocessor system, the interface can include just the core and trace subsystem connection.

- CTI cross trigger interface. A CoreSight component for enabling cross triggering of events across a system.
- CTM cross trigger matrix. A CoreSight component for connecting multiple cross trigger interfaces.

The main function of the ECT (CTI and CTM) is to pass debug events from one connected subsystem to another connected subsystem.

For example, the ECT can communicate debug state information from the core to trace subsystem for a single processor system or to another core in a multiprocessor-based system. Program execution on both the subsystem can be stopped at the same time.

The ***Trigger Flow*** figure shows a simple debug trigger flow sequence. On each CTI, there are four channel, eight input, and eight output debug triggers. All the eight inputs and outputs can be mapped to a single channel or different channels based on the debug trigger to channel mapping. When a trigger input occurs, it creates a channel event. The channel event causes all the output debug triggers to be triggered. The embedded cross trigger depends on the debug trigger it connects to.

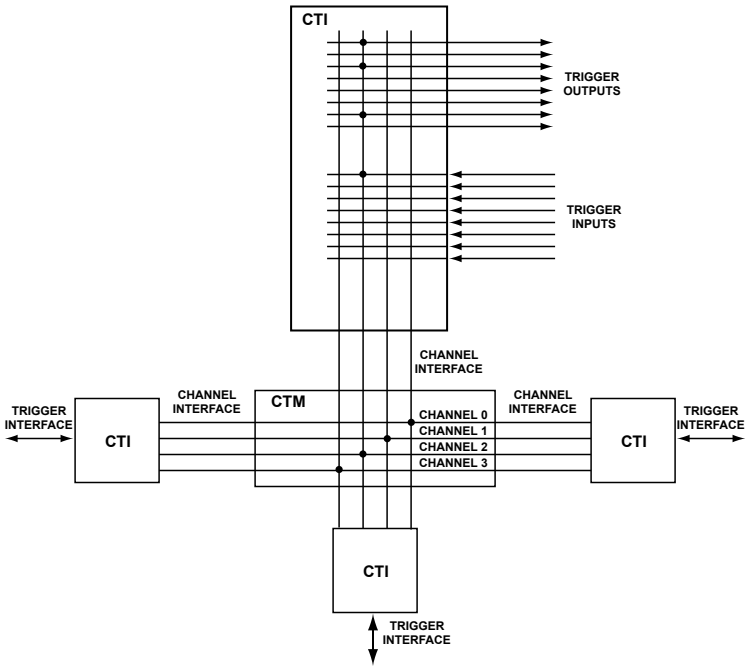


Figure 48-2: Trigger Flow

The ***ECT Integration*** figure shows 3 CTIs in the system that connect to the core, trace, and system module, respectively. The CTIs all interconnect through the CTM. This configuration allows the core to trigger debug events on the trace, on the system and on the core itself.

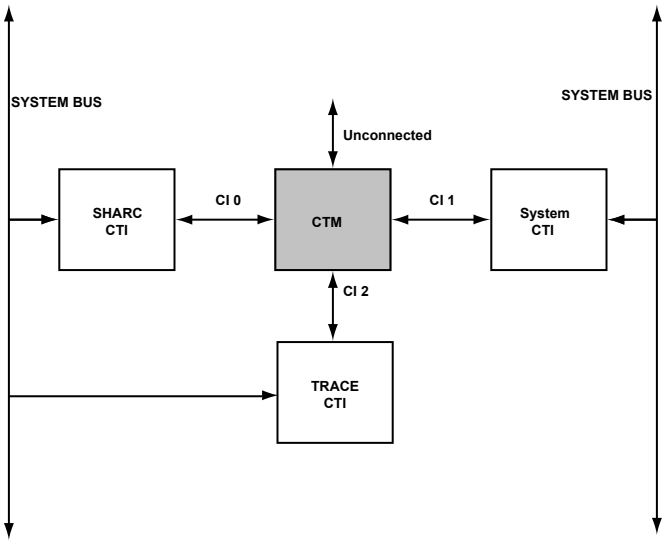


Figure 48-3: ECT Integration

ADSP-2184x A55_CLUSTER Register Descriptions

Your module description, here. (A55_CLUSTER) contains the following registers.

Table 48-1: ADSP-2184x A55_CLUSTER Register List

Name	Description
A55_CLUSTER_ASICCTL	CTI External Multiplexer Control Register
A55_CLUSTER_CTIAPPCLEAR	CTI Application Trigger Clear Register
A55_CLUSTER_CTIAPPPULSE	CTI Application Pulse Register
A55_CLUSTER_CTIAPPSET	CTI Application Trigger Set Register
A55_CLUSTER_CTIAUTHSTATUS	CTI Authentication Status Register
A55_CLUSTER_CTICHINSTATUS	CTI Channel In Status Register
A55_CLUSTER_CTICHOUTSTATUS	CTI Channel Out Status Register
A55_CLUSTER_CTICIDR0	CTI Component Identification 0 Register
A55_CLUSTER_CTICIDR1	CTI Component Identification 1 Register
A55_CLUSTER_CTICIDR2	CTI Component Identification 2 Register
A55_CLUSTER_CTICIDR3	CTI Component Identification 3 Register
A55_CLUSTER_CTICLAIMCLR	CTI Claim Tag Clear Register
A55_CLUSTER_CTICLAIMSET	CTI Claim Tag Set Register
A55_CLUSTER_CTICONTROL	CTI Control Register
A55_CLUSTER_CTIDEVAFF0	CTI Device Affinity 0 Register
A55_CLUSTER_CTIDEVAFF1	CTI Device Affinity 1 Register
A55_CLUSTER_CTIDEVARCH	CTI Device Architecture Register
A55_CLUSTER_CTIDEVID	CTI Device ID 0 Register
A55_CLUSTER_CTIDEVID1	CTI Device ID 1 Register
A55_CLUSTER_CTIDEVID2	CTI Device ID 2 Register
A55_CLUSTER_CTIDEVTYPE	CTI Device Type Register
A55_CLUSTER_CTIGATE	CTI Channel Gate Enable Register
A55_CLUSTER_CTIINEN0	CTI Input Trigger to Output Channel Enable 0 Register
A55_CLUSTER_CTIINEN1	CTI Input Trigger to Output Channel Enable 1 Register
A55_CLUSTER_CTIINEN2	CTI Input Trigger to Output Channel Enable 2 Register
A55_CLUSTER_CTIINEN3	CTI Input Trigger to Output Channel Enable 3 Register
A55_CLUSTER_CTIINEN4	CTI Input Trigger to Output Channel Enable 4 Register
A55_CLUSTER_CTIINEN5	CTI Input Trigger to Output Channel Enable 5 Register
A55_CLUSTER_CTIINEN6	CTI Input Trigger to Output Channel Enable 6 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_CTIINEN7	CTI Input Trigger to Output Channel Enable 7 Register
A55_CLUSTER_CTIINTACK	CTI Output Trigger Acknowledge Register
A55_CLUSTER_CTIITCTRL	CTI Integration mode Control Register
A55_CLUSTER_CTILAR	CTI Lock Access Register
A55_CLUSTER_CTILSR	CTI Lock Status Register
A55_CLUSTER_CTIOUTEN0	CTI Input Channel to Output Trigger Enable 0 Register
A55_CLUSTER_CTIOUTEN1	CTI Input Channel to Output Trigger Enable 1 Register
A55_CLUSTER_CTIOUTEN2	CTI Input Channel to Output Trigger Enable 2 Register
A55_CLUSTER_CTIOUTEN3	CTI Input Channel to Output Trigger Enable 3 Register
A55_CLUSTER_CTIOUTEN4	CTI Input Channel to Output Trigger Enable 4 Register
A55_CLUSTER_CTIOUTEN5	CTI Input Channel to Output Trigger Enable 5 Register
A55_CLUSTER_CTIOUTEN6	CTI Input Channel to Output Trigger Enable 6 Register
A55_CLUSTER_CTIOUTEN7	CTI Input Channel to Output Trigger Enable 7 Register
A55_CLUSTER_CTIPIDR0	CTI Peripheral Identification 0 Register
A55_CLUSTER_CTIPIDR1	CTI Peripheral Identification 1 Register
A55_CLUSTER_CTIPIDR2	CTI Peripheral Identification 2 Register
A55_CLUSTER_CTIPIDR3	CTI Peripheral Identification 3 Register
A55_CLUSTER_CTIPIDR4	CTI Peripheral Identification 4 Register
A55_CLUSTER_CTIPIDR5	CTI Peripheral Identification 5 Register
A55_CLUSTER_CTIPIDR6	CTI Peripheral Identification 6 Register
A55_CLUSTER_CTIPIDR7	CTI Peripheral Identification 7 Register
A55_CLUSTER_CTITRIGINSTATUS	CTI Trigger In Status Register
A55_CLUSTER_CTITRIGOUTSTATUS	CTI Trigger Out Status Register
A55_CLUSTER_DBGAUTHSTATUS_EL1	Debug Authentication Status Register
A55_CLUSTER_EDCIDR0	External Debug Component Identification 0 Register
A55_CLUSTER_EDCIDR1	External Debug Component Identification 1 Register
A55_CLUSTER_EDCIDR2	External Debug Component Identification 2 Register
A55_CLUSTER_EDCIDR3	External Debug Component Identification 3 Register
A55_CLUSTER_EDDEVAFF0	External Debug Device Affinity 0 Register
A55_CLUSTER_EDDEVAFF1	External Debug Device Affinity 1 Register
A55_CLUSTER_EDDEVARCH	External Debug Device Architecture Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_EDDEVID	External Debug Device ID 0 Register
A55_CLUSTER_EDDEVID1	External Debug Device ID 1 Register
A55_CLUSTER_EDDEVID2	External Debug Device ID 2 Register
A55_CLUSTER_EDDEVTYPE	External Debug Device Type Register
A55_CLUSTER_EDDFR_31_0	External Debug Feature Register
A55_CLUSTER_EDDFR_63_32	External Debug Feature Register
A55_CLUSTER_EDECR	External Debug Execution Control Register
A55_CLUSTER_EDLAR	External Debug Lock Access Register
A55_CLUSTER_EDLSR	External Debug Lock Status Register
A55_CLUSTER_EDPFR_31_0	External Debug Processor Feature Register
A55_CLUSTER_EDPFR_63_32	External Debug Processor Feature Register
A55_CLUSTER_EDPIDR0	External Debug Peripheral Identification 0 Register
A55_CLUSTER_EDPIDR1	External Debug Peripheral Identification 1 Register
A55_CLUSTER_EDPIDR2	External Debug Peripheral Identification 2 Register
A55_CLUSTER_EDPIDR3	External Debug Peripheral Identification 3 Register
A55_CLUSTER_EDPIDR4	External Debug Peripheral Identification 4 Register
A55_CLUSTER_EDPIDR5	External Debug Peripheral Identification 5 Register
A55_CLUSTER_EDPIDR6	External Debug Peripheral Identification 6 Register
A55_CLUSTER_EDPIDR7	External Debug Peripheral Identification 7 Register
A55_CLUSTER_MIDR_EL1	Main ID Register
A55_CLUSTER_TRCAUTHSTATUS	Trace Authentication Status Register
A55_CLUSTER_TRCCIDR0	Trace Component ID 0 Register
A55_CLUSTER_TRCCIDR1	Trace Component ID 1 Register
A55_CLUSTER_TRCCIDR2	Trace Component ID 2 Register
A55_CLUSTER_TRCCIDR3	Trace Component ID 3 Register
A55_CLUSTER_TRCDEVAFF0	Trace Device Affinity 0 Register
A55_CLUSTER_TRCDEVAFF1	Trace Device Affinity 1 Register
A55_CLUSTER_TRCDEVARCH	Trace Device Architecture Register
A55_CLUSTER_TRCDEVID	Trace Device ID Register
A55_CLUSTER_TRCDEVTYPE	Trace Device Type Register
A55_CLUSTER_TRCLAR	Trace Software Lock Access Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_TRCLSR	Trace Software Lock Status Register
A55_CLUSTER_TRCPDCR	Trace PowerDown Control Register
A55_CLUSTER_TRCPIDR0	Trace Peripheral ID 0 Register
A55_CLUSTER_TRCPIDR1	Trace Peripheral ID 1 Register
A55_CLUSTER_TRCPIDR2	Trace Peripheral ID 2 Register
A55_CLUSTER_TRCPIDR3	Trace Peripheral ID 3 Register
A55_CLUSTER_TRCPIDR4	Trace Peripheral ID 4 Register
A55_CLUSTER_PMAUTHSTATUS	Performance Monitors Authentication Status Register
A55_CLUSTER_PMCIDR0	Performance Monitors Component Identification 0 Register
A55_CLUSTER_PMCIDR1	Performance Monitors Component Identification 1 Register
A55_CLUSTER_PMCIDR2	Performance Monitors Component Identification 2 Register
A55_CLUSTER_PMCIDR3	Performance Monitors Component Identification 3 Register
A55_CLUSTER_PMDEVAFF0	Performance Monitors Device Affinity 0 Register
A55_CLUSTER_PMDEVAFF1	Performance Monitors Device Affinity 1 Register
A55_CLUSTER_PMDEVARCH	Performance Monitors Device Architecture Register
A55_CLUSTER_PMDEVTYPE	Performance Monitors Device Type Register
A55_CLUSTER_PMLAR	Performance Monitors Lock Access Register
A55_CLUSTER_PMLSR	Performance Monitors Lock Status Register
A55_CLUSTER_PMPIDR0	Performance Monitors Peripheral Identification 0 Register
A55_CLUSTER_PMPIDR1	Performance Monitors Peripheral Identification 1 Register
A55_CLUSTER_PMPIDR2	Performance Monitors Peripheral Identification 2 Register
A55_CLUSTER_PMPIDR3	Performance Monitors Peripheral Identification 3 Register
A55_CLUSTER_PMPIDR4	Performance Monitors Peripheral Identification 4 Register
A55_CLUSTER_PMPIDR5	Performance Monitors Peripheral Identification 5 Register
A55_CLUSTER_PMPIDR6	Performance Monitors Peripheral Identification 6 Register
A55_CLUSTER_PMPIDR7	Performance Monitors Peripheral Identification 7 Register
A55_CLUSTER_ASICCTL	CTI External Multiplexer Control Register
A55_CLUSTER_CTIAPPCLEAR	CTI Application Trigger Clear Register
A55_CLUSTER_CTIAPPPULSE	CTI Application Pulse Register
A55_CLUSTER_CTIAPPSET	CTI Application Trigger Set Register
A55_CLUSTER_CTIAUTHSTATUS	CTI Authentication Status Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_CTICHINSTATUS	CTI Channel In Status Register
A55_CLUSTER_CTICHOUTSTATUS	CTI Channel Out Status Register
A55_CLUSTER_CTICIDR0	CTI Component Identification 0 Register
A55_CLUSTER_CTICIDR1	CTI Component Identification 1 Register
A55_CLUSTER_CTICIDR2	CTI Component Identification 2 Register
A55_CLUSTER_CTICIDR3	CTI Component Identification 3 Register
A55_CLUSTER_CTICLAIMCLR	CTI Claim Tag Clear Register
A55_CLUSTER_CTICLAIMSET	CTI Claim Tag Set Register
A55_CLUSTER_CTICONTROL	CTI Control Register
A55_CLUSTER_CTIDEVAFF0	CTI Device Affinity 0 Register
A55_CLUSTER_CTIDEVAFF1	CTI Device Affinity 1 Register
A55_CLUSTER_CTIDEVARCH	CTI Device Architecture Register
A55_CLUSTER_CTIDEVID	CTI Device ID 0 Register
A55_CLUSTER_CTIDEVID1	CTI Device ID 1 Register
A55_CLUSTER_CTIDEVID2	CTI Device ID 2 Register
A55_CLUSTER_CTIDEVTYPE	CTI Device Type Register
A55_CLUSTER_CTIGATE	CTI Channel Gate Enable Register
A55_CLUSTER_CTIINEN0	CTI Input Trigger to Output Channel Enable 0 Register
A55_CLUSTER_CTIINEN1	CTI Input Trigger to Output Channel Enable 1 Register
A55_CLUSTER_CTIINEN2	CTI Input Trigger to Output Channel Enable 2 Register
A55_CLUSTER_CTIINEN3	CTI Input Trigger to Output Channel Enable 3 Register
A55_CLUSTER_CTIINEN4	CTI Input Trigger to Output Channel Enable 4 Register
A55_CLUSTER_CTIINEN5	CTI Input Trigger to Output Channel Enable 5 Register
A55_CLUSTER_CTIINEN6	CTI Input Trigger to Output Channel Enable 6 Register
A55_CLUSTER_CTIINEN7	CTI Input Trigger to Output Channel Enable 7 Register
A55_CLUSTER_CTIINTACK	CTI Output Trigger Acknowledge Register
A55_CLUSTER_CTIITCTRL	CTI Integration mode Control Register
A55_CLUSTER_CTILAR	CTI Lock Access Register
A55_CLUSTER_CTILSR	CTI Lock Status Register
A55_CLUSTER_CTIOUTEN0	CTI Input Channel to Output Trigger Enable 0 Register
A55_CLUSTER_CTIOUTEN1	CTI Input Channel to Output Trigger Enable 1 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_CTIOUTEN2	CTI Input Channel to Output Trigger Enable 2 Register
A55_CLUSTER_CTIOUTEN3	CTI Input Channel to Output Trigger Enable 3 Register
A55_CLUSTER_CTIOUTEN4	CTI Input Channel to Output Trigger Enable 4 Register
A55_CLUSTER_CTIOUTEN5	CTI Input Channel to Output Trigger Enable 5 Register
A55_CLUSTER_CTIOUTEN6	CTI Input Channel to Output Trigger Enable 6 Register
A55_CLUSTER_CTIOUTEN7	CTI Input Channel to Output Trigger Enable 7 Register
A55_CLUSTER_CTIPIDR0	CTI Peripheral Identification 0 Register
A55_CLUSTER_CTIPIDR1	CTI Peripheral Identification 1 Register
A55_CLUSTER_CTIPIDR2	CTI Peripheral Identification 2 Register
A55_CLUSTER_CTIPIDR3	CTI Peripheral Identification 3 Register
A55_CLUSTER_CTIPIDR4	CTI Peripheral Identification 4 Register
A55_CLUSTER_CTIPIDR5	CTI Peripheral Identification 5 Register
A55_CLUSTER_CTIPIDR6	CTI Peripheral Identification 6 Register
A55_CLUSTER_CTIPIDR7	CTI Peripheral Identification 7 Register
A55_CLUSTER_CTITRIGINSTATUS	CTI Trigger In Status Register
A55_CLUSTER_CTITRIGOUTSTATUS	CTI Trigger Out Status Register
A55_CLUSTER_DBGAUTHSTATUS_EL1	Debug Authentication Status Register
A55_CLUSTER_EDCIDR0	External Debug Component Identification 0 Register
A55_CLUSTER_EDCIDR1	External Debug Component Identification 1 Register
A55_CLUSTER_EDCIDR2	External Debug Component Identification 2 Register
A55_CLUSTER_EDCIDR3	External Debug Component Identification 3 Register
A55_CLUSTER_EDDEVAFF0	External Debug Device Affinity 0 Register
A55_CLUSTER_EDDEVAFF1	External Debug Device Affinity 1 Register
A55_CLUSTER_EDDEVARCH	External Debug Device Architecture Register
A55_CLUSTER_EDDEVID	External Debug Device ID 0 Register
A55_CLUSTER_EDDEVID1	External Debug Device ID 1 Register
A55_CLUSTER_EDDEVID2	External Debug Device ID 2 Register
A55_CLUSTER_EDDEVTYPE	External Debug Device Type Register
A55_CLUSTER_EDDFR_31_0	External Debug Feature Register
A55_CLUSTER_EDDFR_63_32	External Debug Feature Register
A55_CLUSTER_EDECR	External Debug Execution Control Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_EDLAR	External Debug Lock Access Register
A55_CLUSTER_EDLSR	External Debug Lock Status Register
A55_CLUSTER_EDPFR_31_0	External Debug Processor Feature Register
A55_CLUSTER_EDPFR_63_32	External Debug Processor Feature Register
A55_CLUSTER_EDPIDR0	External Debug Peripheral Identification 0 Register
A55_CLUSTER_EDPIDR1	External Debug Peripheral Identification 1 Register
A55_CLUSTER_EDPIDR2	External Debug Peripheral Identification 2 Register
A55_CLUSTER_EDPIDR3	External Debug Peripheral Identification 3 Register
A55_CLUSTER_EDPIDR4	External Debug Peripheral Identification 4 Register
A55_CLUSTER_EDPIDR5	External Debug Peripheral Identification 5 Register
A55_CLUSTER_EDPIDR6	External Debug Peripheral Identification 6 Register
A55_CLUSTER_EDPIDR7	External Debug Peripheral Identification 7 Register
A55_CLUSTER_MIDR_EL1	Main ID Register
A55_CLUSTER_TRCAUTHSTATUS	Trace Authentication Status Register
A55_CLUSTER_TRCCIDR0	Trace Component ID 0 Register
A55_CLUSTER_TRCCIDR1	Trace Component ID 1 Register
A55_CLUSTER_TRCCIDR2	Trace Component ID 2 Register
A55_CLUSTER_TRCCIDR3	Trace Component ID 3 Register
A55_CLUSTER_TRCDEVAFF0	Trace Device Affinity 0 Register
A55_CLUSTER_TRCDEVAFF1	Trace Device Affinity 1 Register
A55_CLUSTER_TRCDEVARCH	Trace Device Architecture Register
A55_CLUSTER_TRCDEVID	Trace Device ID Register
A55_CLUSTER_TRCDEVTYPE	Trace Device Type Register
A55_CLUSTER_TRCLAR	Trace Software Lock Access Register
A55_CLUSTER_TRCLSR	Trace Software Lock Status Register
A55_CLUSTER_TRCPDCR	Trace PowerDown Control Register
A55_CLUSTER_TRCPIDR0	Trace Peripheral ID 0 Register
A55_CLUSTER_TRCPIDR1	Trace Peripheral ID 1 Register
A55_CLUSTER_TRCPIDR2	Trace Peripheral ID 2 Register
A55_CLUSTER_TRCPIDR3	Trace Peripheral ID 3 Register
A55_CLUSTER_TRCPIDR4	Trace Peripheral ID 4 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_PMAUTHSTATUS	Performance Monitors Authentication Status Register
A55_CLUSTER_PMCIDR0	Performance Monitors Component Identification 0 Register
A55_CLUSTER_PMCIDR1	Performance Monitors Component Identification 1 Register
A55_CLUSTER_PMCIDR2	Performance Monitors Component Identification 2 Register
A55_CLUSTER_PMCIDR3	Performance Monitors Component Identification 3 Register
A55_CLUSTER_PMDEVAFF0	Performance Monitors Device Affinity 0 Register
A55_CLUSTER_PMDEVAFF1	Performance Monitors Device Affinity 1 Register
A55_CLUSTER_PMDEVARCH	Performance Monitors Device Architecture Register
A55_CLUSTER_PMDEVTYPE	Performance Monitors Device Type Register
A55_CLUSTER_PMLAR	Performance Monitors Lock Access Register
A55_CLUSTER_PMLSR	Performance Monitors Lock Status Register
A55_CLUSTER_PMPIDR0	Performance Monitors Peripheral Identification 0 Register
A55_CLUSTER_PMPIDR1	Performance Monitors Peripheral Identification 1 Register
A55_CLUSTER_PMPIDR2	Performance Monitors Peripheral Identification 2 Register
A55_CLUSTER_PMPIDR3	Performance Monitors Peripheral Identification 3 Register
A55_CLUSTER_PMPIDR4	Performance Monitors Peripheral Identification 4 Register
A55_CLUSTER_PMPIDR5	Performance Monitors Peripheral Identification 5 Register
A55_CLUSTER_PMPIDR6	Performance Monitors Peripheral Identification 6 Register
A55_CLUSTER_PMPIDR7	Performance Monitors Peripheral Identification 7 Register
A55_CLUSTER_AUTHSTATUS	Authentication Status Register
A55_CLUSTER_CIDR0	Component Identification 0 Register
A55_CLUSTER_CIDR1	Component Identification 1 Register
A55_CLUSTER_CIDR2	Component Identification 2 Register
A55_CLUSTER_CIDR3	Component Identification 3 Register
A55_CLUSTER_DBGPCR0	Debug Power Control 0 Register
A55_CLUSTER_DBGPCR1	Debug Power Control 1 Register
A55_CLUSTER_DBGPCR2	Debug Power Control 2 Register
A55_CLUSTER_DBGPCR3	Debug Power Control 3 Register
A55_CLUSTER_DBGPCR4	Debug Power Control 4 Register
A55_CLUSTER_DBGPCR5	Debug Power Control 5 Register
A55_CLUSTER_DBGPCR6	Debug Power Control 6 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_DBGPCR7	Debug Power Control 7 Register
A55_CLUSTER_DBGPCR8	Debug Power Control 8 Register
A55_CLUSTER_DBGPSR0	Debug Power Status 0 Register
A55_CLUSTER_DBGPSR1	Debug Power Status 1 Register
A55_CLUSTER_DBGPSR2	Debug Power Status 2 Register
A55_CLUSTER_DBGPSR3	Debug Power Status 3 Register
A55_CLUSTER_DBGPSR4	Debug Power Status 4 Register
A55_CLUSTER_DBGPSR5	Debug Power Status 5 Register
A55_CLUSTER_DBGPSR6	Debug Power Status 6 Register
A55_CLUSTER_DBGPSR7	Debug Power Status 7 Register
A55_CLUSTER_DBGPSR8	Debug Power Status 8 Register
A55_CLUSTER_DEVARCH	Device Architecture Register
A55_CLUSTER_DEVID	Device ID Register
A55_CLUSTER_DEVTYPE	Device Type Register
A55_CLUSTER_PIDR0	Peripheral Identification 0 Register
A55_CLUSTER_PIDR1	Peripheral Identification 1 Register
A55_CLUSTER_PIDR2	Peripheral Identification 2 Register
A55_CLUSTER_PIDR3	Peripheral Identification 3 Register
A55_CLUSTER_PIDR4	Peripheral Identification 4 Register
A55_CLUSTER_PIDR5	Peripheral Identification 5 Register
A55_CLUSTER_PIDR6	Peripheral Identification 6 Register
A55_CLUSTER_PIDR7	Peripheral Identification 7 Register
A55_CLUSTER_PRIDR0	Power Reset Identification 0 Register
A55_CLUSTER_ROMENTRY0	ROM Table Entry 0 Register
A55_CLUSTER_ROMENTRY1	ROM Table Entry 1 Register
A55_CLUSTER_ROMENTRY10	ROM Table Entry 10 Register
A55_CLUSTER_ROMENTRY11	ROM Table Entry 11 Register
A55_CLUSTER_ROMENTRY12	ROM Table Entry 12 Register
A55_CLUSTER_ROMENTRY13	ROM Table Entry 13 Register
A55_CLUSTER_ROMENTRY14	ROM Table Entry 14 Register
A55_CLUSTER_ROMENTRY15	ROM Table Entry 15 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_ROMENTRY16	ROM Table Entry 16 Register
A55_CLUSTER_ROMENTRY17	ROM Table Entry 17 Register
A55_CLUSTER_ROMENTRY18	ROM Table Entry 18 Register
A55_CLUSTER_ROMENTRY19	ROM Table Entry 19 Register
A55_CLUSTER_ROMENTRY2	ROM Table Entry 2 Register
A55_CLUSTER_ROMENTRY20	ROM Table Entry 20 Register
A55_CLUSTER_ROMENTRY21	ROM Table Entry 21 Register
A55_CLUSTER_ROMENTRY22	ROM Table Entry 22 Register
A55_CLUSTER_ROMENTRY23	ROM Table Entry 23 Register
A55_CLUSTER_ROMENTRY24	ROM Table Entry 24 Register
A55_CLUSTER_ROMENTRY25	ROM Table Entry 25 Register
A55_CLUSTER_ROMENTRY26	ROM Table Entry 26 Register
A55_CLUSTER_ROMENTRY27	ROM Table Entry 27 Register
A55_CLUSTER_ROMENTRY28	ROM Table Entry 28 Register
A55_CLUSTER_ROMENTRY29	ROM Table Entry 29 Register
A55_CLUSTER_ROMENTRY3	ROM Table Entry 3 Register
A55_CLUSTER_ROMENTRY30	ROM Table Entry 30 Register
A55_CLUSTER_ROMENTRY31	ROM Table Entry 31 Register
A55_CLUSTER_ROMENTRY32	ROM Table Entry 32 Register
A55_CLUSTER_ROMENTRY33	ROM Table Entry 33 Register
A55_CLUSTER_ROMENTRY34	ROM Table Entry 34 Register
A55_CLUSTER_ROMENTRY35	ROM Table Entry 35 Register
A55_CLUSTER_ROMENTRY36	ROM Table Entry 36 Register
A55_CLUSTER_ROMENTRY37	ROM Table Entry 37 Register
A55_CLUSTER_ROMENTRY38	ROM Table Entry 38 Register
A55_CLUSTER_ROMENTRY39	ROM Table Entry 39 Register
A55_CLUSTER_ROMENTRY4	ROM Table Entry 4 Register
A55_CLUSTER_ROMENTRY40	ROM Table Entry 40 Register
A55_CLUSTER_ROMENTRY41	ROM Table Entry 41 Register
A55_CLUSTER_ROMENTRY42	ROM Table Entry 42 Register
A55_CLUSTER_ROMENTRY43	ROM Table Entry 43 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_ROMENTRY44	ROM Table Entry 44 Register
A55_CLUSTER_ROMENTRY45	ROM Table Entry 45 Register
A55_CLUSTER_ROMENTRY46	ROM Table Entry 46 Register
A55_CLUSTER_ROMENTRY47	ROM Table Entry 47 Register
A55_CLUSTER_ROMENTRY48	ROM Table Entry 48 Register
A55_CLUSTER_ROMENTRY49	ROM Table Entry 49 Register
A55_CLUSTER_ROMENTRY5	ROM Table Entry 5 Register
A55_CLUSTER_ROMENTRY50	ROM Table Entry 50 Register
A55_CLUSTER_ROMENTRY51	ROM Table Entry 51 Register
A55_CLUSTER_ROMENTRY52	ROM Table Entry 52 Register
A55_CLUSTER_ROMENTRY53	ROM Table Entry 53 Register
A55_CLUSTER_ROMENTRY54	ROM Table Entry 54 Register
A55_CLUSTER_ROMENTRY55	ROM Table Entry 55 Register
A55_CLUSTER_ROMENTRY56	ROM Table Entry 56 Register
A55_CLUSTER_ROMENTRY57	ROM Table Entry 57 Register
A55_CLUSTER_ROMENTRY58	ROM Table Entry 58 Register
A55_CLUSTER_ROMENTRY59	ROM Table Entry 59 Register
A55_CLUSTER_ROMENTRY6	ROM Table Entry 6 Register
A55_CLUSTER_ROMENTRY60	ROM Table Entry 60 Register
A55_CLUSTER_ROMENTRY61	ROM Table Entry 61 Register
A55_CLUSTER_ROMENTRY62	ROM Table Entry 62 Register
A55_CLUSTER_ROMENTRY63	ROM Table Entry 63 Register
A55_CLUSTER_ROMENTRY64	ROM Table Entry 64 Register
A55_CLUSTER_ROMENTRY65	ROM Table Entry 65 Register
A55_CLUSTER_ROMENTRY66	ROM Table Entry 66 Register
A55_CLUSTER_ROMENTRY67	ROM Table Entry 67 Register
A55_CLUSTER_ROMENTRY68	ROM Table Entry 68 Register
A55_CLUSTER_ROMENTRY69	ROM Table Entry 69 Register
A55_CLUSTER_ROMENTRY7	ROM Table Entry 7 Register
A55_CLUSTER_ROMENTRY70	ROM Table Entry 70 Register
A55_CLUSTER_ROMENTRY71	ROM Table Entry 71 Register

Table 48-1: ADSP-2184x A55_CLUSTER Register List (Continued)

Name	Description
A55_CLUSTER_ROMENTRY72	ROM Table Entry 72 Register
A55_CLUSTER_ROMENTRY73	ROM Table Entry 73 Register
A55_CLUSTER_ROMENTRY74	ROM Table Entry 74 Register
A55_CLUSTER_ROMENTRY75	ROM Table Entry 75 Register
A55_CLUSTER_ROMENTRY76	ROM Table Entry 76 Register
A55_CLUSTER_ROMENTRY77	ROM Table Entry 77 Register
A55_CLUSTER_ROMENTRY78	ROM Table Entry 78 Register
A55_CLUSTER_ROMENTRY79	ROM Table Entry 79 Register
A55_CLUSTER_ROMENTRY8	ROM Table Entry 8 Register
A55_CLUSTER_ROMENTRY80	ROM Table Entry 80 Register
A55_CLUSTER_ROMENTRY81	ROM Table Entry 81 Register
A55_CLUSTER_ROMENTRY9	ROM Table Entry 9 Register

CTI External Multiplexer Control Register

Can be used to provide IMPLEMENTATION DEFINED controls for the CTI. For example, the register might be used to control multiplexors for additional IMPLEMENTATION DEFINED triggers. The IMPLEMENTATION DEFINED controls provided by this register might modify the architecturally defined behavior of the CTI. The architecturally-defined triggers must not be multiplexed.

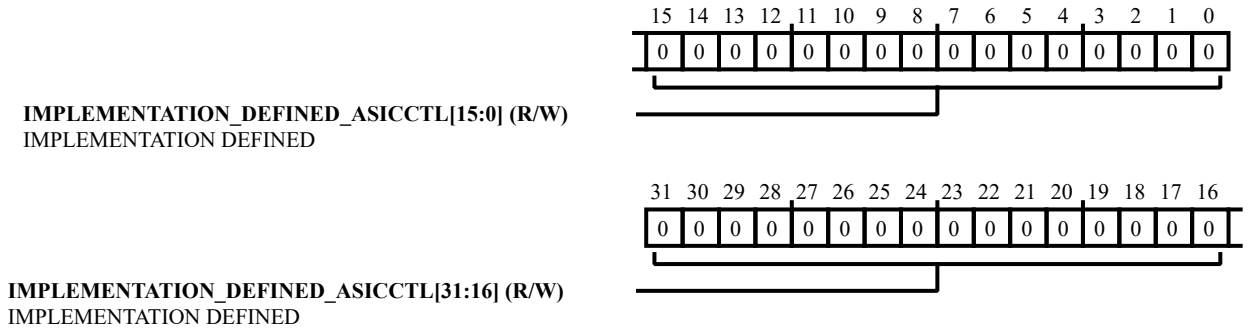


Figure 48-4: A55_CLUSTER_ASICCTL Register Diagram

Table 48-2: A55_CLUSTER_ASICCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	IMPLEMENTATION_DEFINED_ASICCTL	IMPLEMENTATION DEFINED. IMPLEMENTATION DEFINED.

CTI Application Trigger Clear Register

Clears bits of the Application Trigger register.

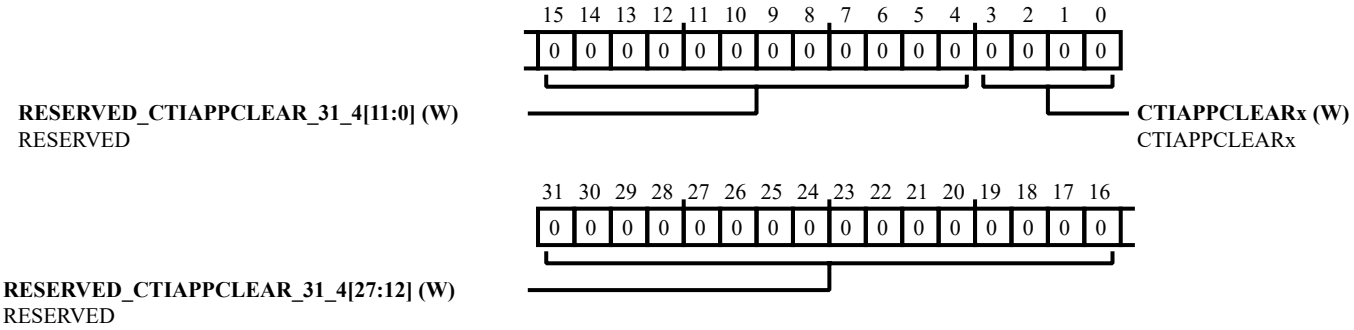


Figure 48-5: A55_CLUSTER_CTIAPPCLEAR Register Diagram

Table 48-3: A55_CLUSTER_CTIAPPCLEAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (RX/W)	RESERVED_CTIAPP-CLEAR_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (RX/W)	CTIAPPCLEARX	CTIAPPCLEARx. Application trigger <x> disable.	
		0	No effect.
		1	Clear corresponding bit in CTIAPPTRIG to 0 and clear the corresponding channel event.

CTI Application Pulse Register

Causes event pulses to be generated on ECT channels.

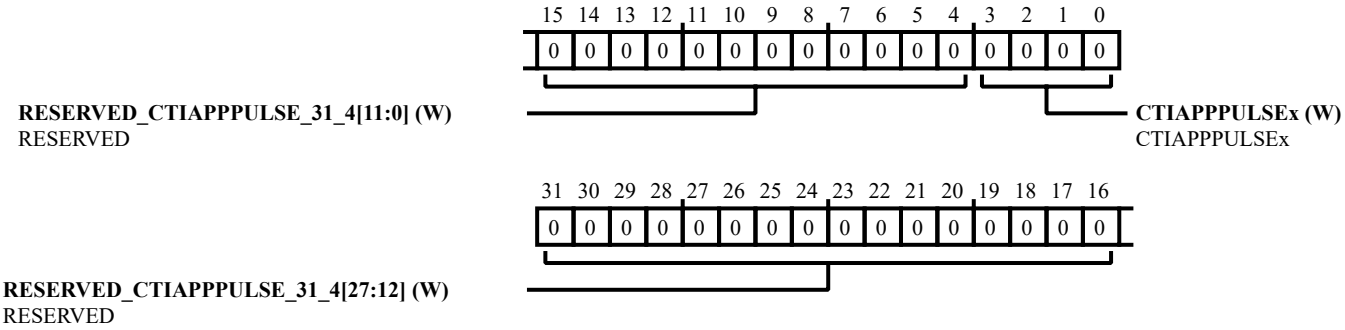


Figure 48-6: A55_CLUSTER_CTIAPPPULSE Register Diagram

Table 48-4: A55_CLUSTER_CTIAPPPULSE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (RX/W)	RESERVED_CTIAPP-PULSE_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (RX/W)	CTIAPPPULSEX	CTIAPPPULSEx. Generate event pulse on ECT channel <x>.	
		0	No effect.
		1	Channel <x> event pulse generated for one clock period.

CTI Application Trigger Set Register

Sets bits of the Application Trigger register.

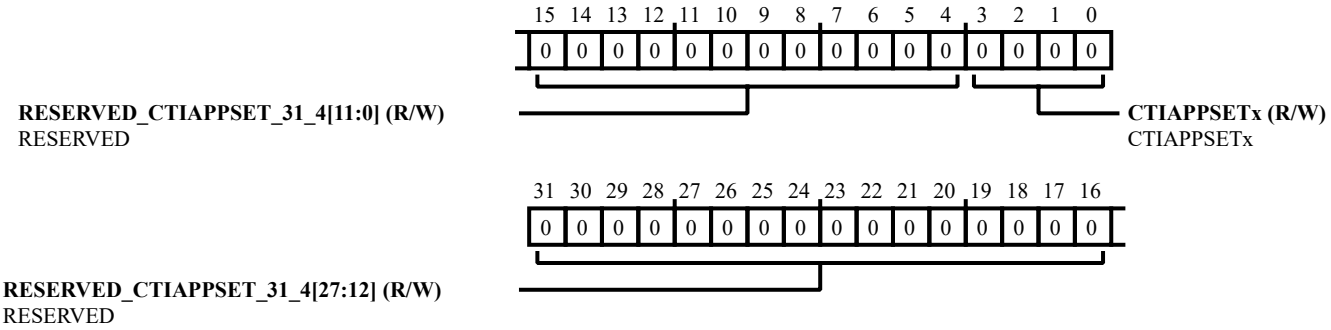


Figure 48-7: A55_CLUSTER_CTIAPPSET Register Diagram

Table 48-5: A55_CLUSTER_CTIAPPSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIAP-PSET_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	CTIAPPSETX	CTIAPPSETx. Application trigger <x> enable.	
		0	Reading this means the application trigger is inactive. Writing this has no effect.
		1	Reading this means the application trigger is active. Writing this sets the corresponding bit in CTIAPP-TRIG to 1 and generates a channel event.

CTI Authentication Status Register

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for CTI.

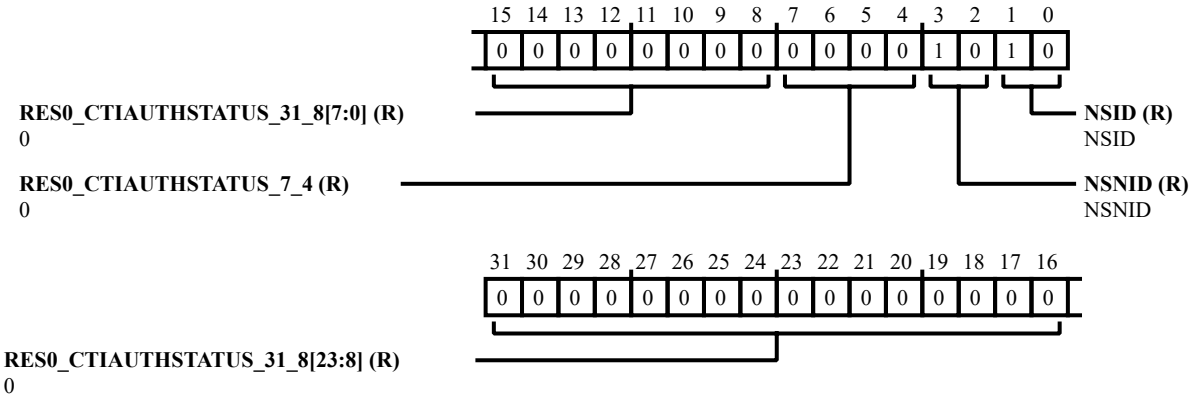


Figure 48-8: A55_CLUSTER_CTIAUTHSTATUS Register Diagram

Table 48-6: A55_CLUSTER_CTIAUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIAUTHSTATUS_31_8	0. Reserved, RES0.
7:4 (R/NW)	RES0_CTIAUTHSTATUS_7_4	0. Reserved, RAZ.
3:2 (R/NW)	NSNID	NSNID. If EL3 is not implemented and the implemented Security state is Secure state, holds the same value as DBGAUTHSTATUS_EL1.SNID. Otherwise, holds the same value as DBGAUTHSTATUS_EL1.NSNID.
1:0 (R/NW)	NSID	NSID. If EL3 is not implemented and the implemented Security state is Secure state, holds the same value as DBGAUTHSTATUS_EL1.SID. Otherwise, holds the same value as DBGAUTHSTATUS_EL1.NSID.

CTI Channel In Status Register

Provides the raw status of the ECT channel inputs to the CTI.

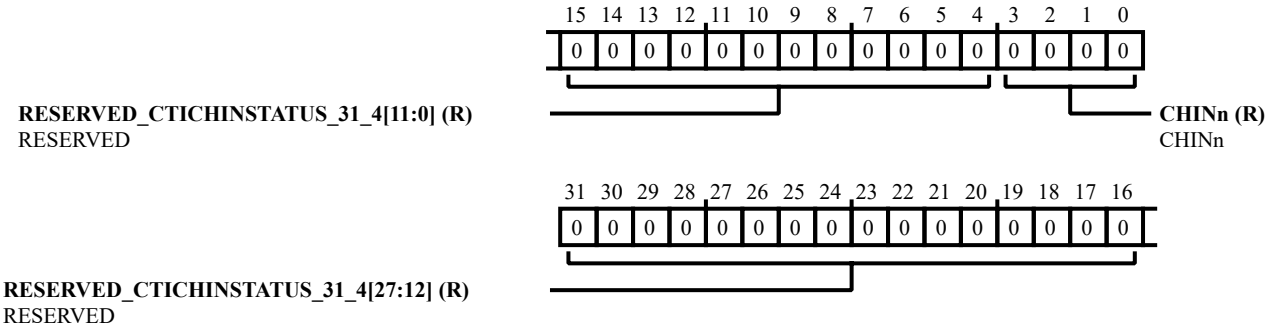


Figure 48-9: A55_CLUSTER_CTICHINSTATUS Register Diagram

Table 48-7: A55_CLUSTER_CTICHINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RESERVED_CTICHIN-STATUS_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	CHINN	CHINn. Provides the raw status of the ECT channel inputs to the CTI.	
		0	Input channel <n> is inactive.
		1	Input channel <n> is active.

CTI Channel Out Status Register

Provides the status of the ECT channel outputs from the CTI.

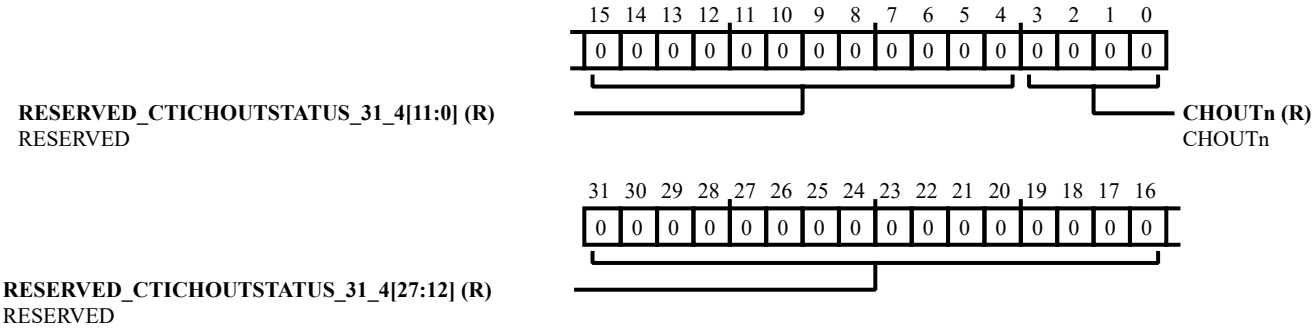


Figure 48-10: A55_CLUSTER_CTICHOUTSTATUS Register Diagram

Table 48-8: A55_CLUSTER_CTICHOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RESERVED_CTICHOUT-STATUS_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	CHOUTN	CHOUTn. Provides the status of the ECT channel outputs from the CTI.	
		0	Output channel <n> is inactive.
		1	Output channel <n> is active.

CTI Component Identification 0 Register

Provides information to identify a CTI component.

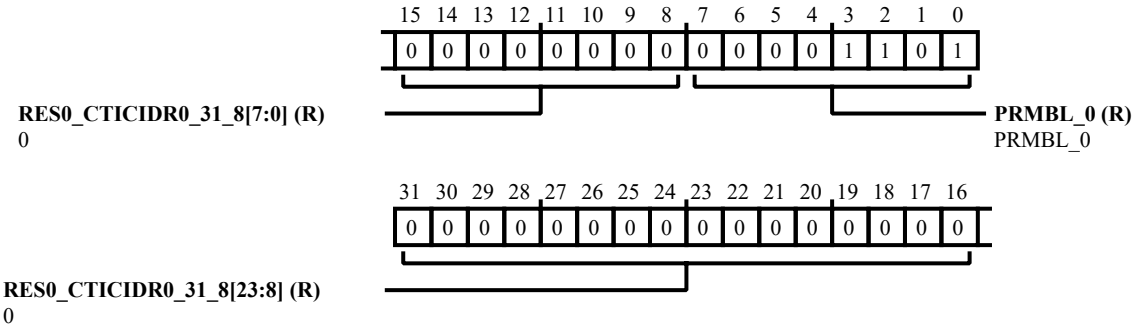


Figure 48-11: A55_CLUSTER_CTICIDR0 Register Diagram

Table 48-9: A55_CLUSTER_CTICIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

CTI Component Identification 1 Register

Provides information to identify a CTI component.

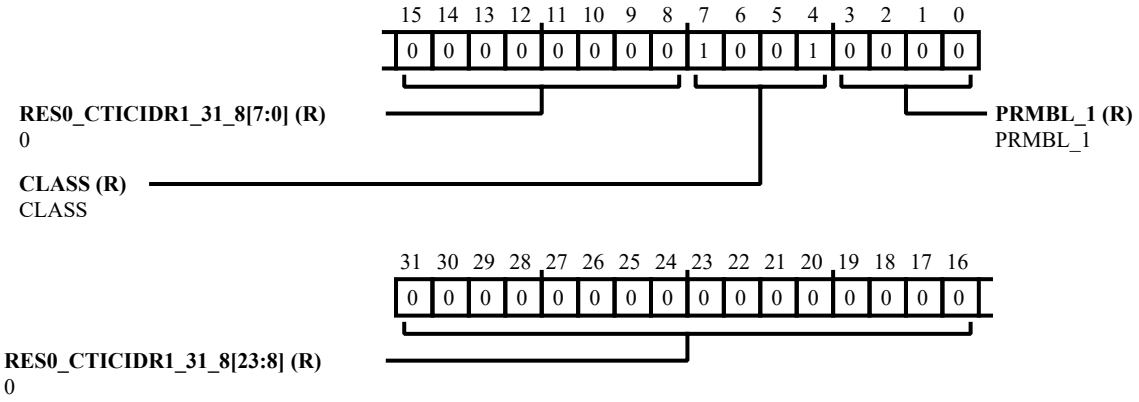


Figure 48-12: A55_CLUSTER_CTICIDR1 Register Diagram

Table 48-10: A55_CLUSTER_CTICIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, debug component.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. RAZ.

CTI Component Identification 2 Register

Provides information to identify a CTI component.

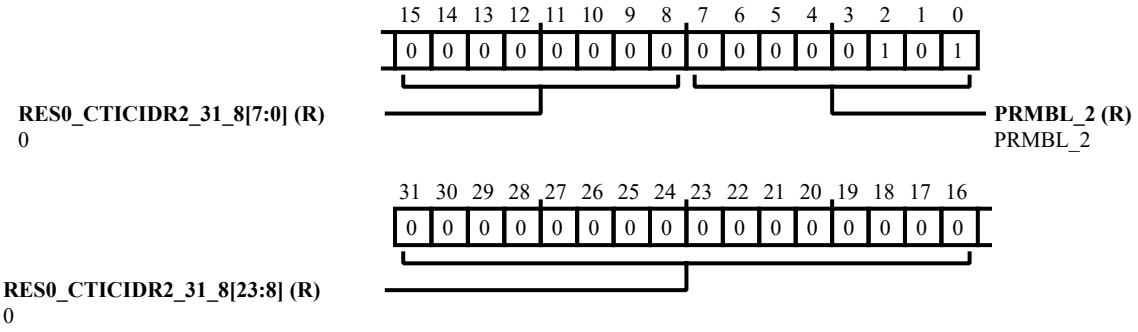


Figure 48-13: A55_CLUSTER_CTICIDR2 Register Diagram

Table 48-11: A55_CLUSTER_CTICIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

CTI Component Identification 3 Register

Provides information to identify a CTI component.

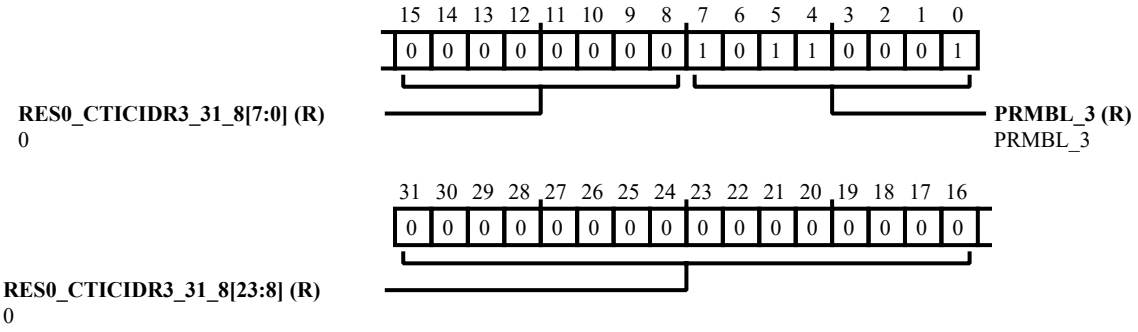


Figure 48-14: A55_CLUSTER_CTICIDR3 Register Diagram

Table 48-12: A55_CLUSTER_CTICIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

CTI Claim Tag Clear Register

Used by software to read the values of the CLAIM bits, and to clear these bits to 0.

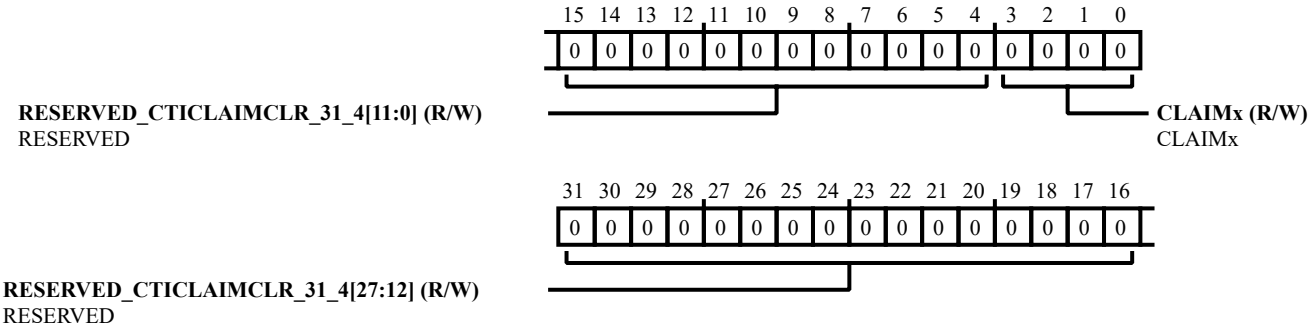


Figure 48-15: A55_CLUSTER_CTICLAIMCLR Register Diagram

Table 48-13: A55_CLUSTER_CTICLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTI-CLAIMCLR_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	CLAIMX	CLAIMx. Clear CLAIM tag.	
		0	No action.
		1	Indirectly clear CLAIM[x] to 0.

CTI Claim Tag Set Register

Used by software to set CLAIM bits to 1.

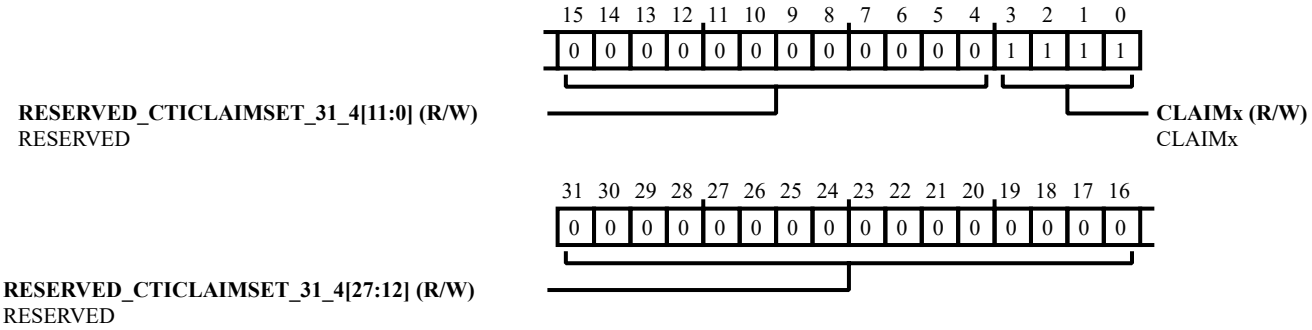


Figure 48-16: A55_CLUSTER_CTICLAIMSET Register Diagram

Table 48-14: A55_CLUSTER_CTICLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTICLAIM-SET_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	CLAIMX	CLAIMx. CLAIM tag set bit.	
		0	No action.
		1	Indirectly set CLAIM[x] tag to 1.

CTI Control Register

Controls whether the CTI is enabled.

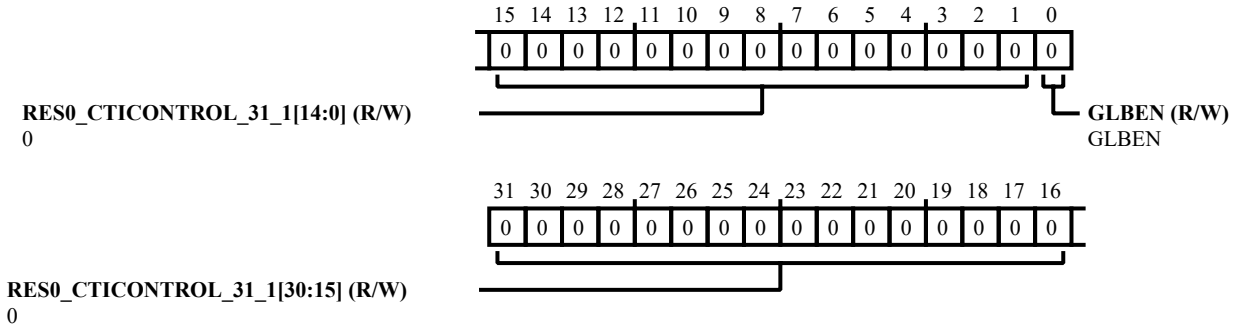


Figure 48-17: A55_CLUSTER_CTICONTROL Register Diagram

Table 48-15: A55_CLUSTER_CTICONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_CTICON-TROL_31_1	0. Reserved, RES0.	
0 (R/W)	GLBEN	GLBEN. Enables or disables the CTI mapping functions. Possible values of this field are: 0 CTI mapping functions disabled. 1 CTI mapping functions enabled. When the mapping functions are disabled, no new events are signaled on either output triggers or output channels. If a previously asserted output trigger has not been acknowledged, it remains asserted after the mapping functions are disabled. All output triggers are disabled by CTI reset. If the ECT supports multicycle channel events any existing output channel events will be terminated.	
		0	CTI mapping functions disabled.
		1	CTI mapping functions enabled.

CTI Device Affinity 0 Register

Copy of the low half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

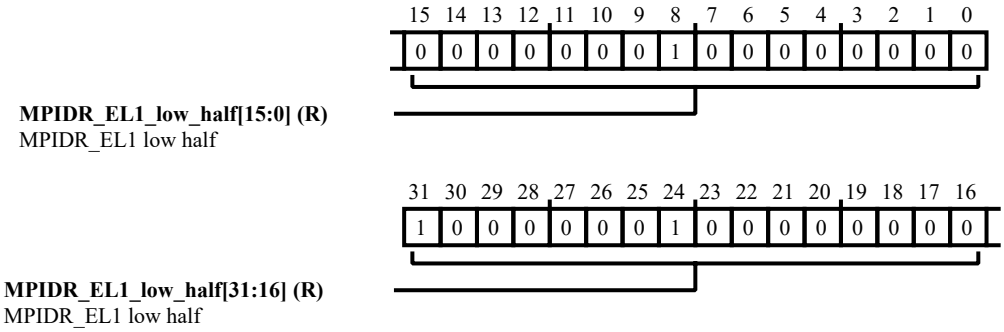


Figure 48-18: A55_CLUSTER_CTIDEVAFF0 Register Diagram

Table 48-16: A55_CLUSTER_CTIDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_LOW_HALF	MPIDR_EL1 low half. MPIDR_EL1 low half. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented Exception level.

CTI Device Affinity 1 Register

Copy of the high half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

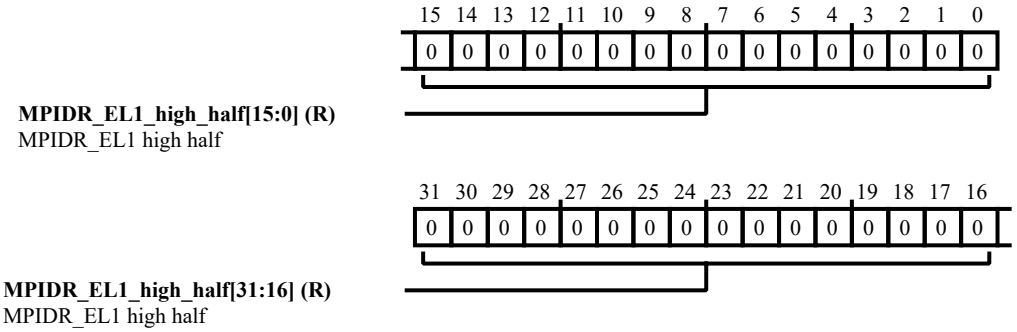


Figure 48-19: A55_CLUSTER_CTIDEVAFF1 Register Diagram

Table 48-17: A55_CLUSTER_CTIDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_HIGH_HALF	MPIDR_EL1 high half. MPIDR_EL1 high half. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented Exception level.

CTI Device Architecture Register

Identifies the programmers' model architecture of the CTI component.

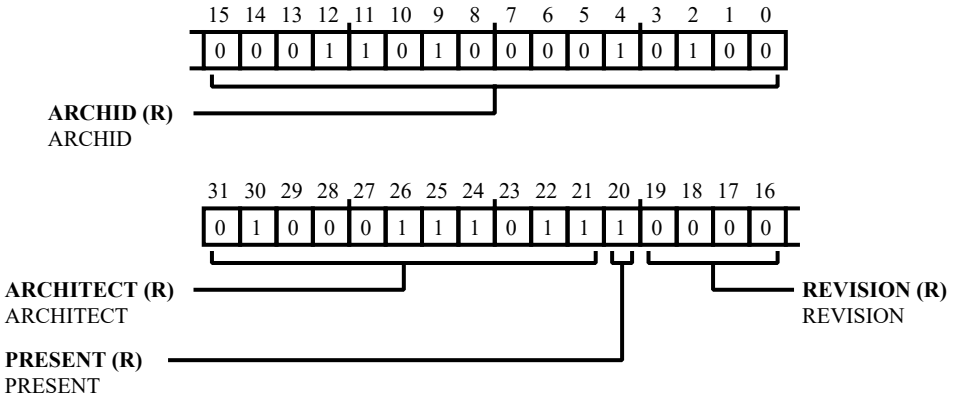


Figure 48-20: A55_CLUSTER_CTIDEVARCH Register Diagram

Table 48-18: A55_CLUSTER_CTIDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For CTI, this is ARM Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present. This field is 1 in ARMv8.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision. For CTI, the revision defined by ARMv8 is 0x0. All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided. For CTI: Bits [15:12] are the architecture version, 0x1. Bits [11:0] are the architecture part number, 0xA14. This corresponds to CTI architecture version CTIv2.

CTI Device ID 0 Register

Describes the CTI component to the debugger.

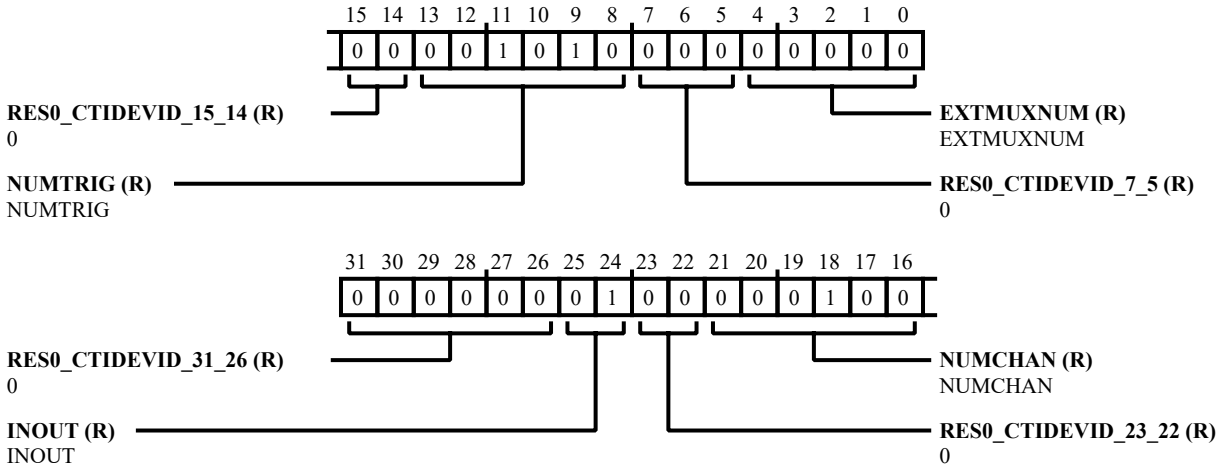


Figure 48-21: A55_CLUSTER_CTIDEVID Register Diagram

Table 48-19: A55_CLUSTER_CTIDEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:26 (R/NW)	RES0_CTIDEVID_31_26	0. Reserved, RES0.	
25:24 (R/NW)	INOUT	INOUT. Input/output options. Indicates presence of the input gate. If the CTM is not implemented, this field is RAZ. 00 CTIGATE does not mask propagation of input events from external channels. 01 CTIGATE masks propagation of input events from external channels. All other values are reserved.	
		0	CTIGATE does not mask propagation of input events from external channels.
		1	CTIGATE masks propagation of input events from external channels.
23:22 (R/NW)	RES0_CTIDEVID_23_22	0. Reserved, RES0.	

Table 48-19: A55_CLUSTER_CTIDEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/NW)	NUMCHAN	NUMCHAN. Number of ECT channels implemented. IMPLEMENTATION DEFINED. For ARMv8, valid values are: 000011 3 channels (0..2) implemented. 000100 4 channels (0..3) implemented. 000101 5 channels (0..4) implemented. 000110 6 channels (0..5) implemented. and so on up to 100000, 32 channels (0..31) implemented. All other values are reserved.	
		3	3 channels (0..2) implemented.
		4	4 channels (0..3) implemented.
		5	5 channels (0..4) implemented.
		6	6 channels (0..5) implemented.
15:14 (R/NW)	RES0_CTIDEVID_15_14	0. Reserved, RES0.	
13:8 (R/NW)	NUMTRIG	NUMTRIG. Number of triggers implemented. IMPLEMENTATION DEFINED. This is one more than the index of the largest trigger, rather than the actual number of triggers. For ARMv8, valid values are: 000011 Up to 3 triggers (0..2) implemented. 001000 Up to 8 triggers (0..7) implemented. 001001 Up to 9 triggers (0..8) implemented. 001010 Up to 10 triggers (0..9) implemented. and so on up to 100000, 32 triggers (0..31) implemented. All other values are reserved. If the contains a Trace extension, this field must be at least 0b001000. There is no guarantee that any of the implemented triggers, including the highest numbered, are connected to any components.	
		3	Up to 3 triggers (0..2) implemented.
		8	Up to 8 triggers (0..7) implemented.
		9	Up to 9 triggers (0..8) implemented.
		10	Up to 10 triggers (0..9) implemented.
7:5 (R/NW)	RES0_CTIDEVID_7_5	0. Reserved, RES0.	
4:0 (R/NW)	EXTMUXNUM	EXTMUXNUM. Number of multiplexors available on triggers. This value is used in conjunction with External Control register, ASICCTL.	

CTI Device ID 1 Register

Reserved for future information about the CTI component to the debugger.

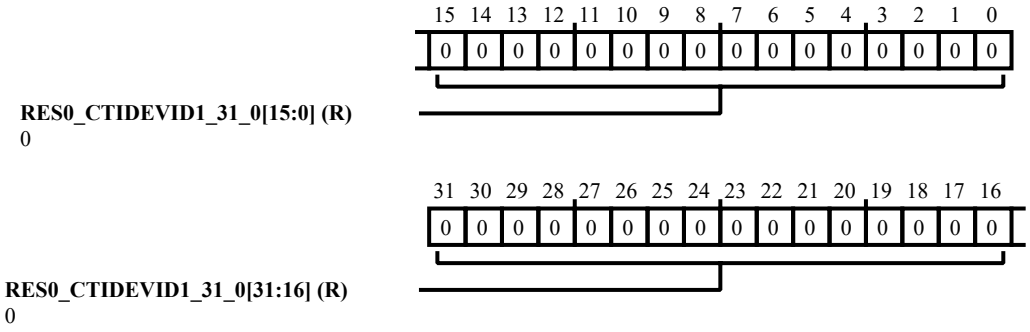


Figure 48-22: A55_CLUSTER_CTIDEVID1 Register Diagram

Table 48-20: A55_CLUSTER_CTIDEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_CTIDEVID1_31_0	0. Reserved, RES0.

CTI Device ID 2 Register

Reserved for future information about the CTI component to the debugger.

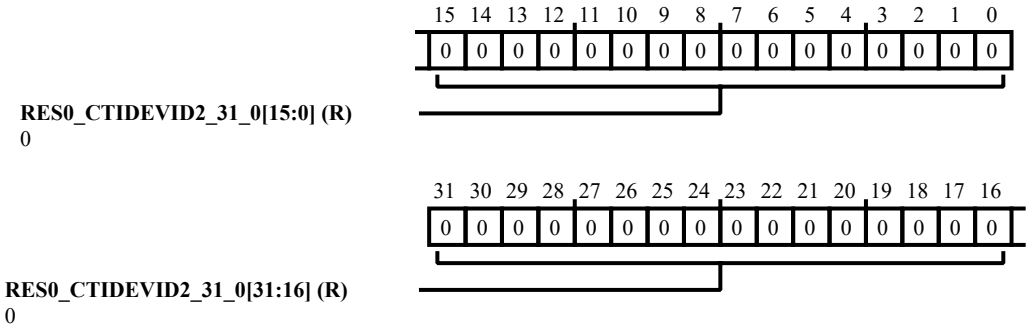


Figure 48-23: A55_CLUSTER_CTIDEVID2 Register Diagram

Table 48-21: A55_CLUSTER_CTIDEVID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_CTIDEVID2_31_0	0. Reserved, RES0.

CTI Device Type Register

Indicates to a debugger that this component is part of a PEs cross-trigger interface.

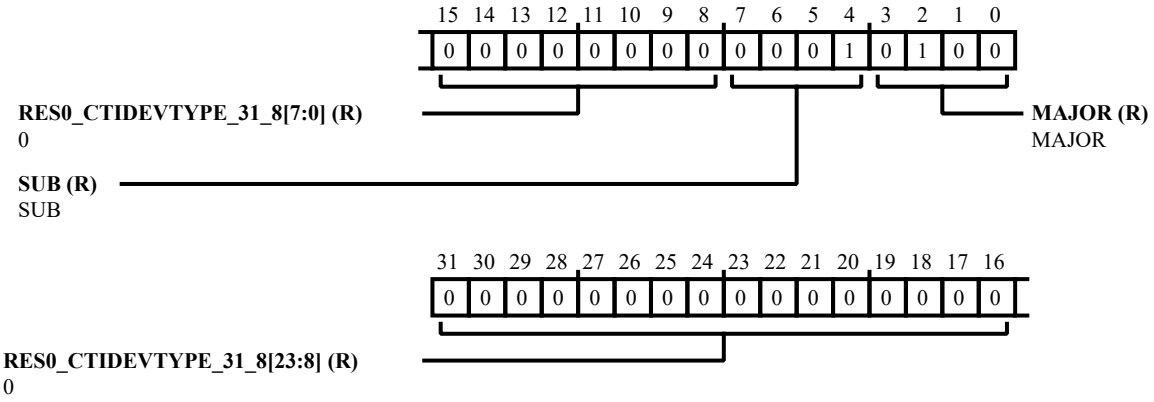


Figure 48-24: A55_CLUSTER_CTIDEVTYPE Register Diagram

Table 48-22: A55_CLUSTER_CTIDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIDEV-TYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Subtype. Must read as 0x1 to indicate this is a component within a PE.
3:0 (R/NW)	MAJOR	MAJOR. Major type. Must read as 0x4 to indicate this is a cross-trigger component.

CTI Channel Gate Enable Register

Determines whether events on channels propagate through the CTM to other ECT components, or from the CTM into the CTI.

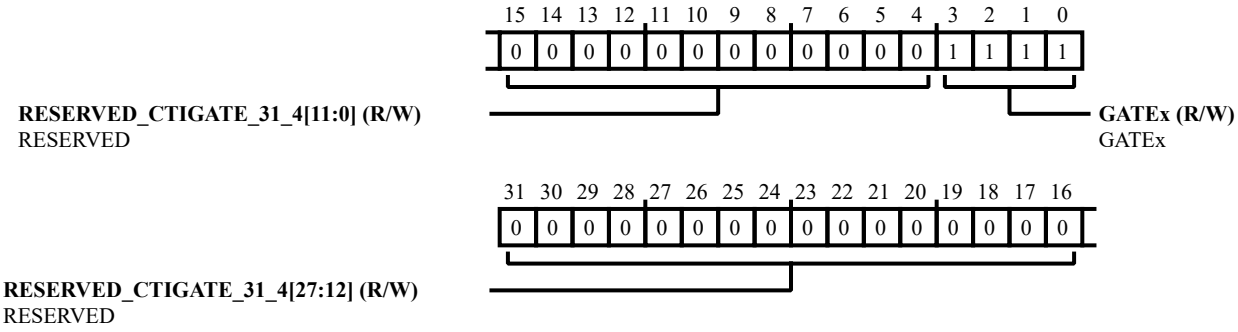


Figure 48-25: A55_CLUSTER_CTIGATE Register Diagram

Table 48-23: A55_CLUSTER_CTIGATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTI-GATE_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	GATEX	GATEx. Determines whether events on channels propagate through the CTM to other ECT components or from the CTM into the CTI.	
		0	Disable output and if CTIDEVID.INOUT == 0b01 input channel <x> propagation.
		1	Enable output and if CTIDEVID.INOUT == 0b01 input channel <x> propagation.

CTI Input Trigger to Output Channel Enable 0 Register

Enables the signaling of an event on output channels when input trigger event 0 is received by the CTI.

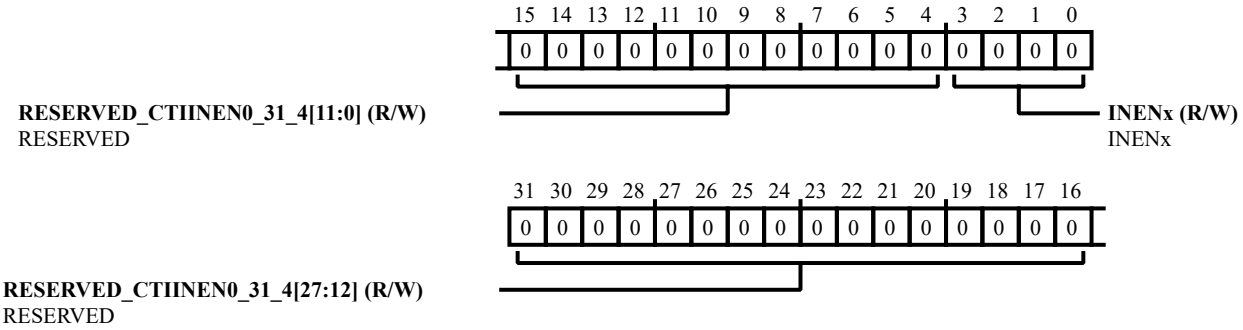


Figure 48-26: A55_CLUSTER_CTIINEN0 Register Diagram

Table 48-24: A55_CLUSTER_CTIINEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN0_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 0 to output channel <x> enable.	
		0	Input trigger 0 will not generate an event on output channel <x>.
		1	Input trigger 0 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 1 Register

Enables the signaling of an event on output channels when input trigger event 1 is received by the CTI.

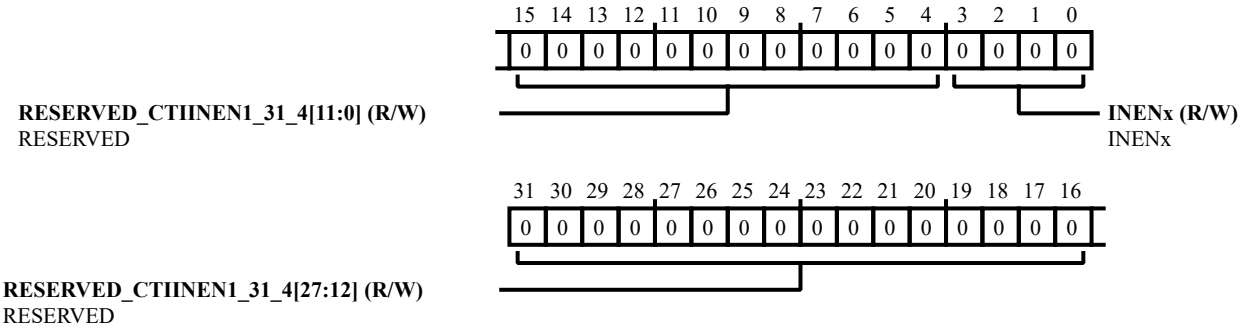


Figure 48-27: A55_CLUSTER_CTIINEN1 Register Diagram

Table 48-25: A55_CLUSTER_CTIINEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN1_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 1 to output channel <x> enable.	
		0	Input trigger 1 will not generate an event on output channel <x>.
		1	Input trigger 1 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 2 Register

Enables the signaling of an event on output channels when input trigger event 2 is received by the CTI.

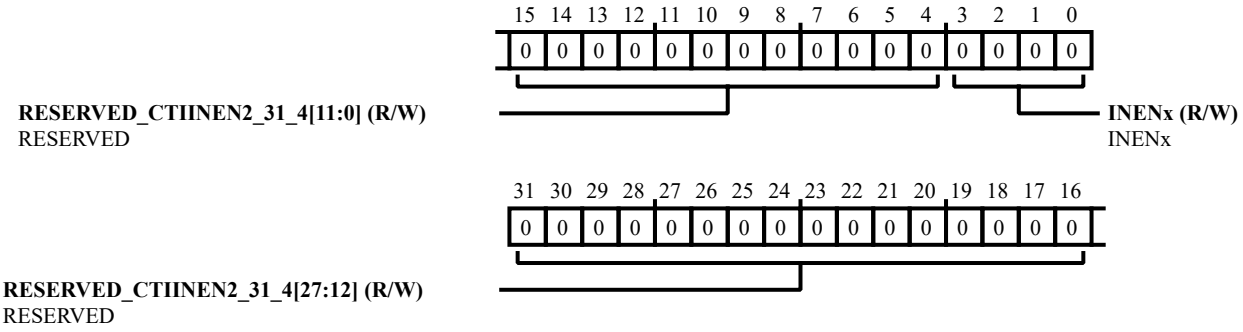


Figure 48-28: A55_CLUSTER_CTIINEN2 Register Diagram

Table 48-26: A55_CLUSTER_CTIINEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN2_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 2 to output channel <x> enable.	
		0	Input trigger 2 will not generate an event on output channel <x>.
		1	Input trigger 2 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 3 Register

Enables the signaling of an event on output channels when input trigger event 3 is received by the CTI.

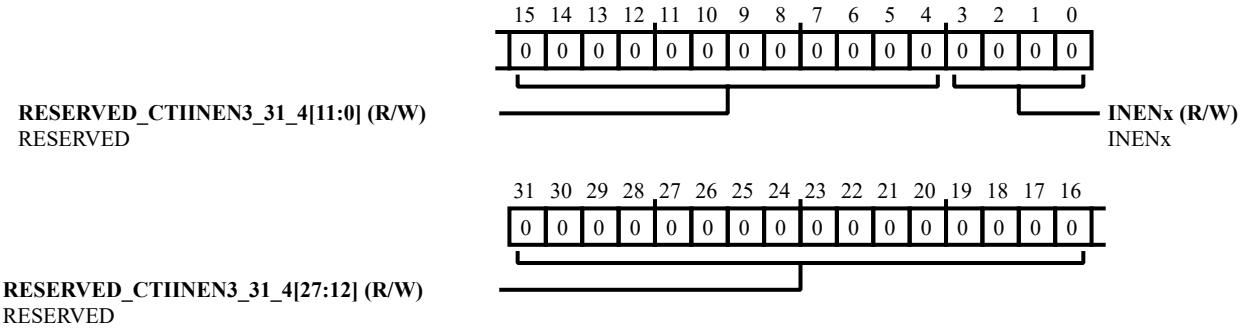


Figure 48-29: A55_CLUSTER_CTIINEN3 Register Diagram

Table 48-27: A55_CLUSTER_CTIINEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN3_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 3 to output channel <x> enable.	
		0	Input trigger 3 will not generate an event on output channel <x>.
		1	Input trigger 3 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 4 Register

Enables the signaling of an event on output channels when input trigger event 4 is received by the CTI.

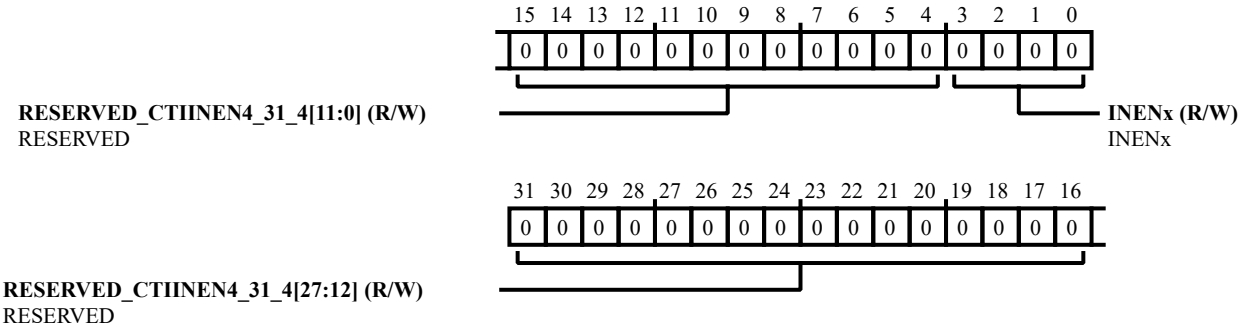


Figure 48-30: A55_CLUSTER_CTIINEN4 Register Diagram

Table 48-28: A55_CLUSTER_CTIINEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN4_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 4 to output channel <x> enable.	
		0	Input trigger 4 will not generate an event on output channel <x>.
		1	Input trigger 4 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 5 Register

Enables the signaling of an event on output channels when input trigger event 5 is received by the CTI.

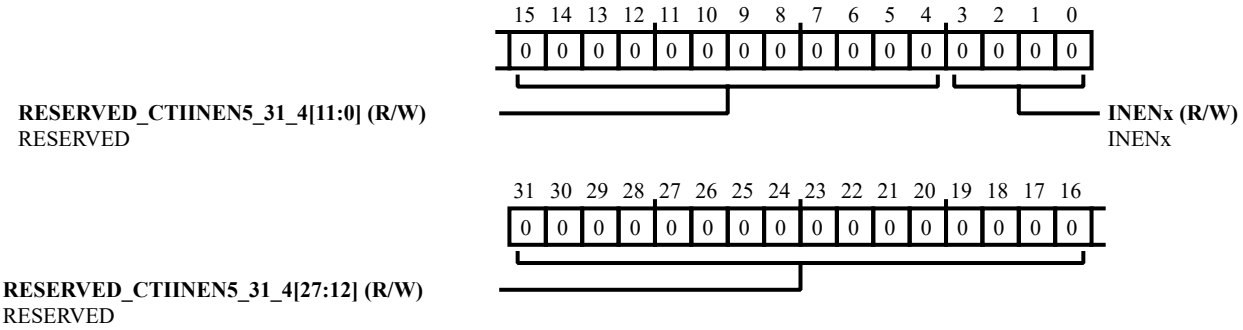


Figure 48-31: A55_CLUSTER_CTIINEN5 Register Diagram

Table 48-29: A55_CLUSTER_CTIINEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN5_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 5 to output channel <x> enable.	
		0	Input trigger 5 will not generate an event on output channel <x>.
		1	Input trigger 5 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 6 Register

Enables the signaling of an event on output channels when input trigger event 6 is received by the CTI.

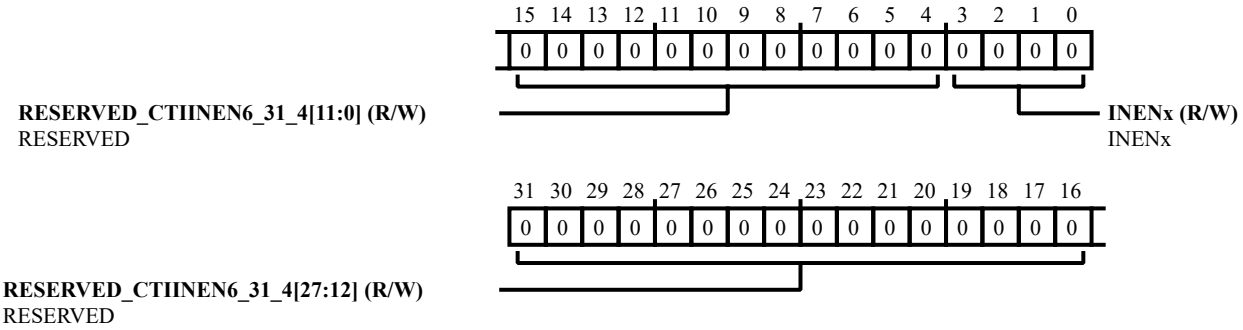


Figure 48-32: A55_CLUSTER_CTIINEN6 Register Diagram

Table 48-30: A55_CLUSTER_CTIINEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN6_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 6 to output channel <x> enable.	
		0	Input trigger 6 will not generate an event on output channel <x>.
		1	Input trigger 6 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 7 Register

Enables the signaling of an event on output channels when input trigger event 7 is received by the CTI.

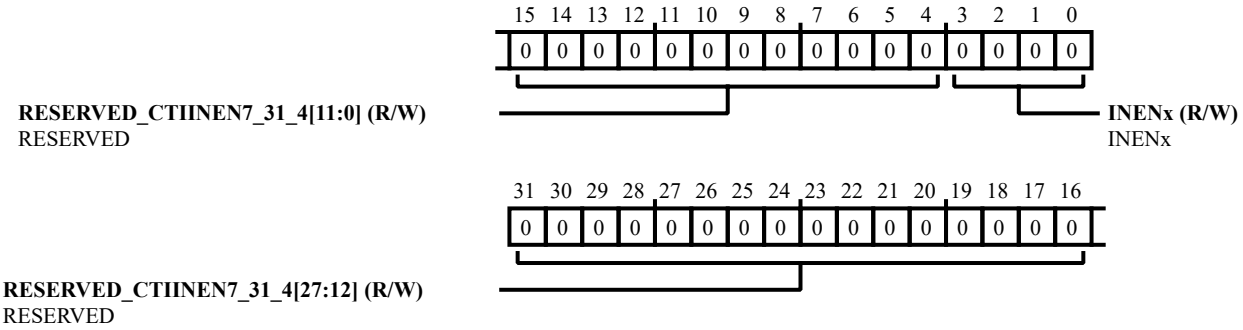


Figure 48-33: A55_CLUSTER_CTIINEN7 Register Diagram

Table 48-31: A55_CLUSTER_CTIINEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN7_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 7 to output channel <x> enable.	
		0	Input trigger 7 will not generate an event on output channel <x>.
		1	Input trigger 7 will generate an event on output channel <x>.

CTI Output Trigger Acknowledge Register

Can be used to deactivate the output triggers.

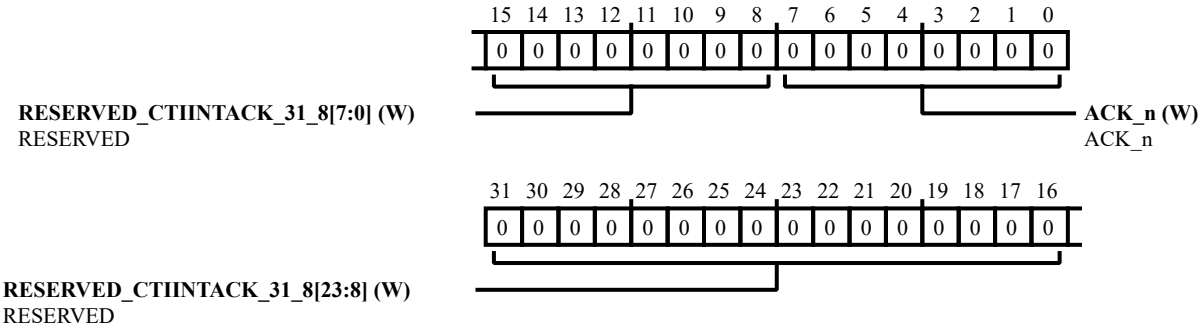


Figure 48-34: A55_CLUSTER_CTIINTACK Register Diagram

Table 48-32: A55_CLUSTER_CTIINTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (RX/W)	RESERVED_CTIIN-TACK_31_8	RESERVED. Reserved RES0.	
		0	Reserved.
7:0 (RX/W)	ACK_N	ACK_n. Can be used to create soft acknowledges for output triggers.	
		0	No effect.
		1	Deactivate the trigger.

CTI Integration mode Control Register

Enables the CTI to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

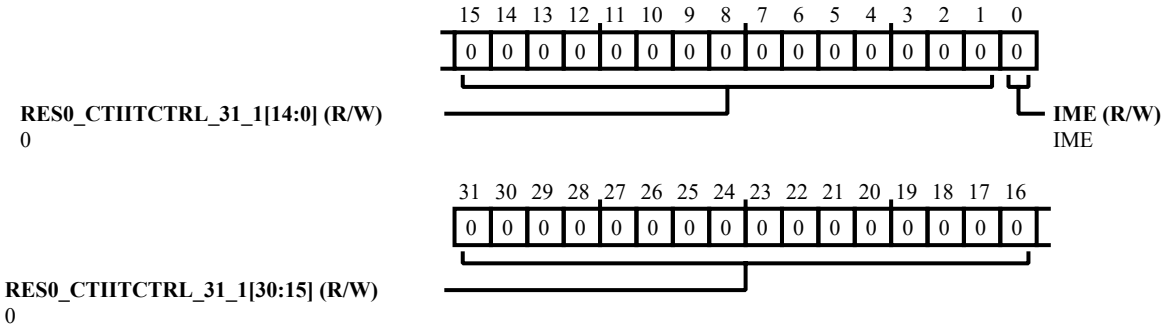


Figure 48-35: A55_CLUSTER_CTIITCTRL Register Diagram

Table 48-33: A55_CLUSTER_CTIITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_CTIITCTRL_31_1	0. Reserved, RES0.	
0 (R/W)	IME	IME. Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED. 0 Normal operation. 1 Integration mode enabled.	
		0	Normal operation.
		1	Integration mode enabled.

CTI Lock Access Register

Allows or disallows access to the CTI registers through a memory-mapped interface.

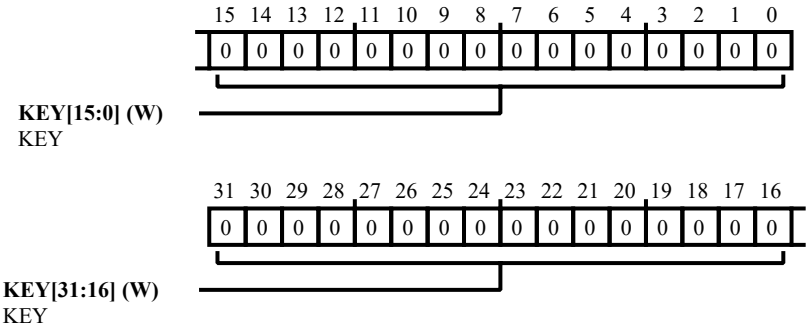


Figure 48-36: A55_CLUSTER_CTILAR Register Diagram

Table 48-34: A55_CLUSTER_CTILAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

CTI Lock Status Register

Indicates the current status of the Software Lock for CTI registers.

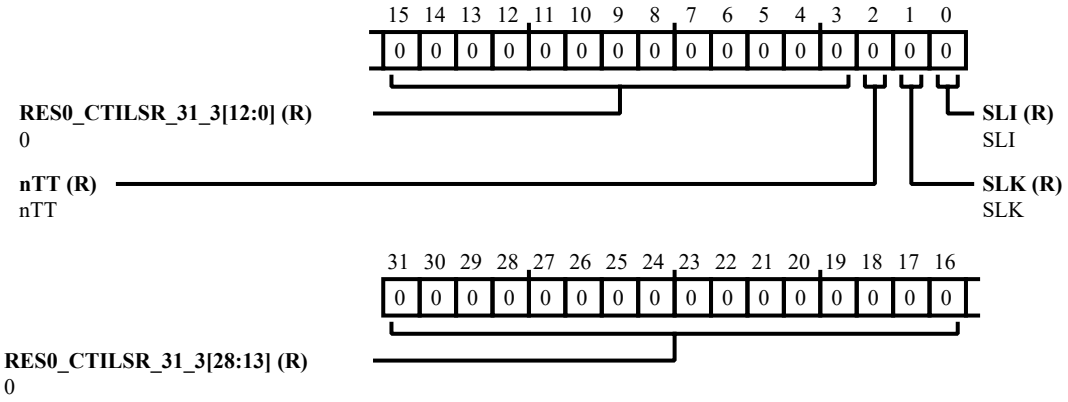


Figure 48-37: A55_CLUSTER_CTILSR Register Diagram

Table 48-35: A55_CLUSTER_CTILSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_CTILSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0. For memory-mapped accesses when the Software Lock is implemented, possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

Table 48-35: A55_CLUSTER_CTILSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SLI	SLI. Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are: 0 Software Lock not implemented or not memory-mapped access. 1 Software Lock implemented and memory-mapped access.	
		0	Software Lock not implemented or not memory-mapped access.
		1	Software Lock implemented and memory-mapped access.

CTI Input Channel to Output Trigger Enable 0 Register

Defines which input channels generate output trigger 0.

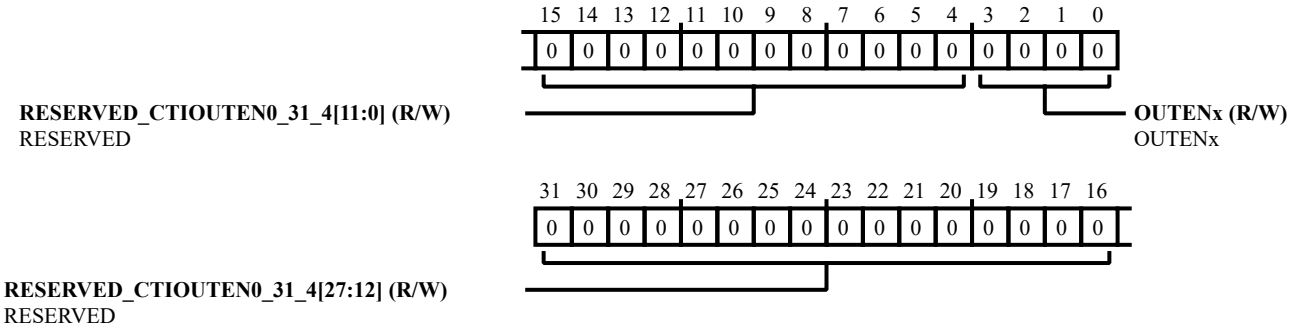


Figure 48-38: A55_CLUSTER_CTIOUTEN0 Register Diagram

Table 48-36: A55_CLUSTER_CTIOUTEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN0_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 0 enable.	
		0	An event on input channel <x> will not cause output trigger 0 to be asserted.
		1	An event on input channel <x> will cause output trigger 0 to be asserted.

CTI Input Channel to Output Trigger Enable 1 Register

Defines which input channels generate output trigger 1.

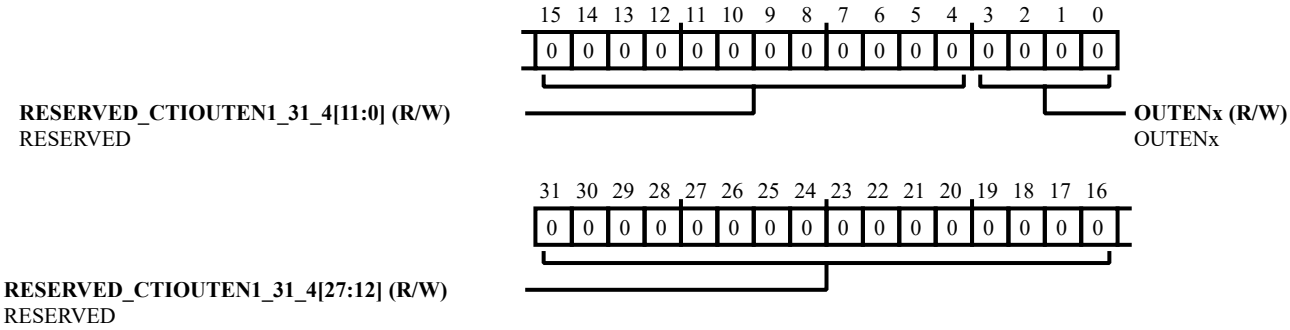


Figure 48-39: A55_CLUSTER_CTIOUTEN1 Register Diagram

Table 48-37: A55_CLUSTER_CTIOUTEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN1_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 1 enable.	
		0	An event on input channel <x> will not cause output trigger 1 to be asserted.
		1	An event on input channel <x> will cause output trigger 1 to be asserted.

CTI Input Channel to Output Trigger Enable 2 Register

Defines which input channels generate output trigger 2.

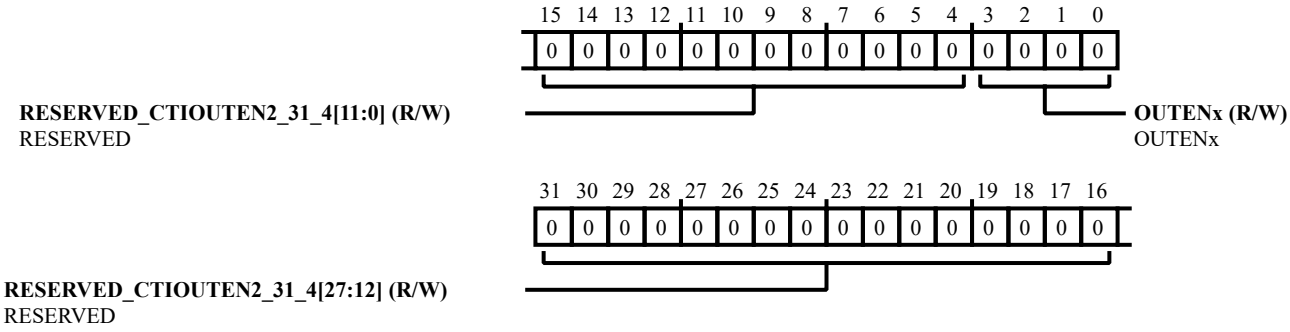


Figure 48-40: A55_CLUSTER_CTIOUTEN2 Register Diagram

Table 48-38: A55_CLUSTER_CTIOUTEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN2_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 2 enable.	
		0	An event on input channel <x> will not cause output trigger 2 to be asserted.
		1	An event on input channel <x> will cause output trigger 2 to be asserted.

CTI Input Channel to Output Trigger Enable 3 Register

Defines which input channels generate output trigger 3.

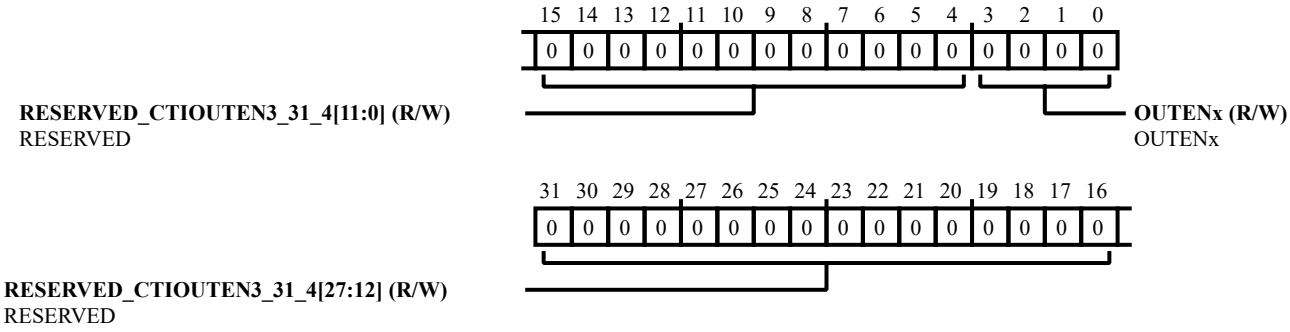


Figure 48-41: A55_CLUSTER_CTIOUTEN3 Register Diagram

Table 48-39: A55_CLUSTER_CTIOUTEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN3_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 3 enable.	
		0	An event on input channel <x> will not cause output trigger 3 to be asserted.
		1	An event on input channel <x> will cause output trigger 3 to be asserted.

CTI Input Channel to Output Trigger Enable 4 Register

Defines which input channels generate output trigger 4.

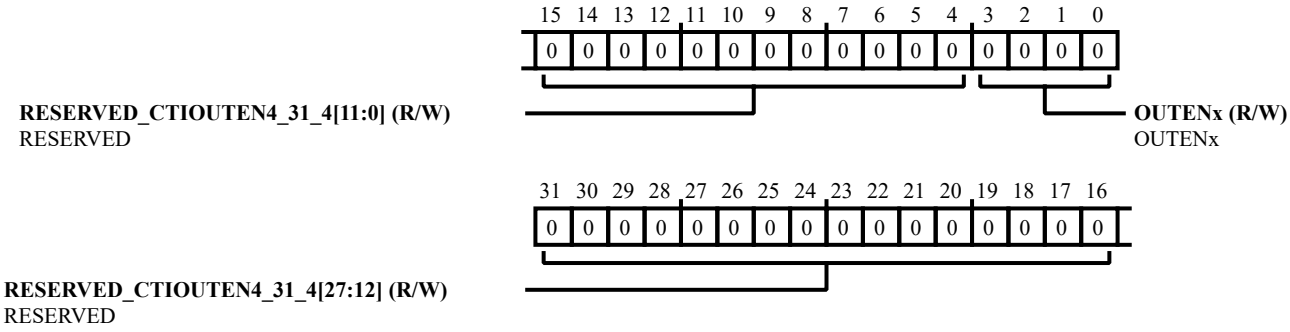


Figure 48-42: A55_CLUSTER_CTIOUTEN4 Register Diagram

Table 48-40: A55_CLUSTER_CTIOUTEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN4_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 4 enable.	
		0	An event on input channel <x> will not cause output trigger 4 to be asserted.
		1	An event on input channel <x> will cause output trigger 4 to be asserted.

CTI Input Channel to Output Trigger Enable 5 Register

Defines which input channels generate output trigger 5.

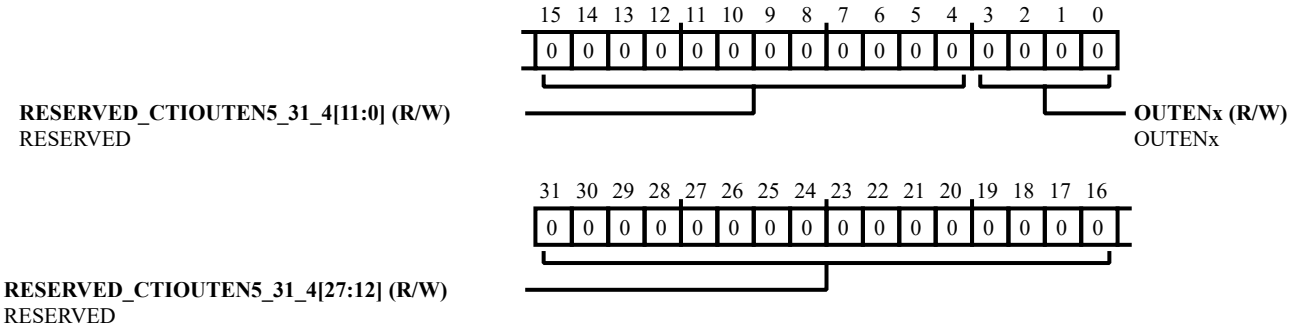


Figure 48-43: A55_CLUSTER_CTIOUTEN5 Register Diagram

Table 48-41: A55_CLUSTER_CTIOUTEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN5_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 5 enable.	
		0	An event on input channel <x> will not cause output trigger 5 to be asserted.
		1	An event on input channel <x> will cause output trigger 5 to be asserted.

CTI Input Channel to Output Trigger Enable 6 Register

Defines which input channels generate output trigger 6.

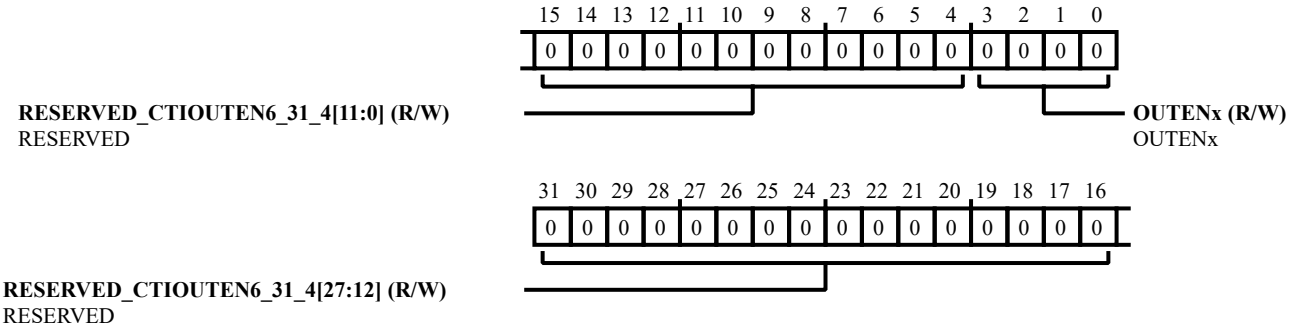


Figure 48-44: A55_CLUSTER_CTIOUTEN6 Register Diagram

Table 48-42: A55_CLUSTER_CTIOUTEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN6_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 6 enable.	
		0	An event on input channel <x> will not cause output trigger 6 to be asserted.
		1	An event on input channel <x> will cause output trigger 6 to be asserted.

CTI Input Channel to Output Trigger Enable 7 Register

Defines which input channels generate output trigger 7.

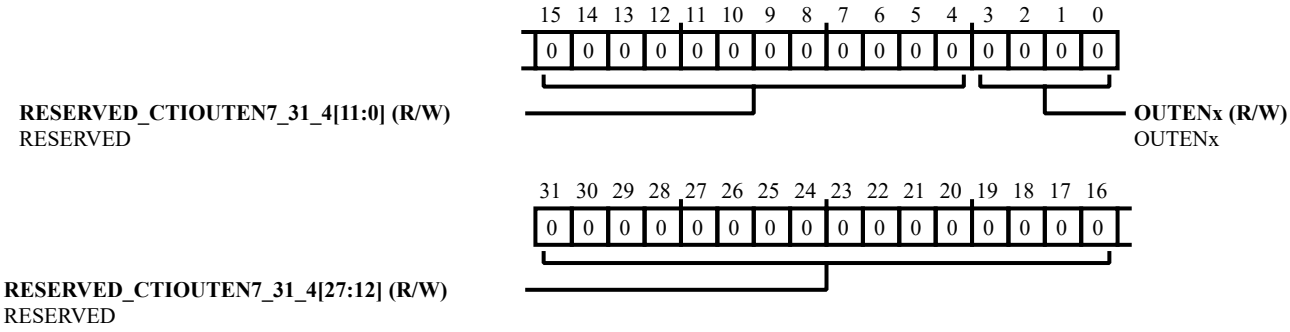


Figure 48-45: A55_CLUSTER_CTIOUTEN7 Register Diagram

Table 48-43: A55_CLUSTER_CTIOUTEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN7_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 7 enable.	
		0	An event on input channel <x> will not cause output trigger 7 to be asserted.
		1	An event on input channel <x> will cause output trigger 7 to be asserted.

CTI Peripheral Identification 0 Register

Provides information to identify a CTI component.

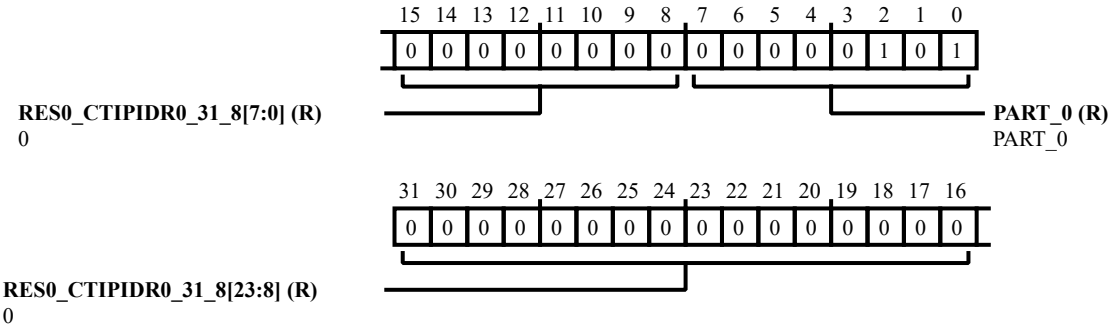


Figure 48-46: A55_CLUSTER_CTIPIDR0 Register Diagram

Table 48-44: A55_CLUSTER_CTIPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, least significant byte.

CTI Peripheral Identification 1 Register

Provides information to identify a CTI component.

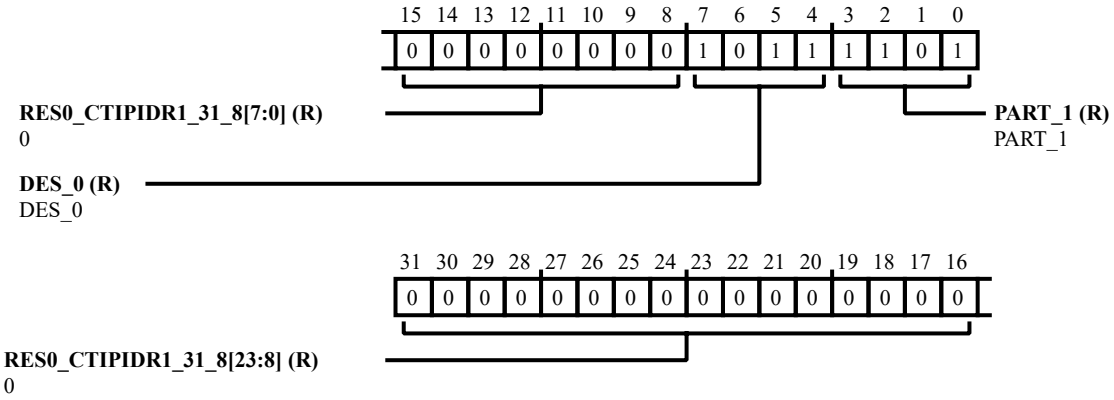


Figure 48-47: A55_CLUSTER_CTIPIDR1 Register Diagram

Table 48-45: A55_CLUSTER_CTIPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, most significant nibble.

CTI Peripheral Identification 2 Register

Provides information to identify a CTI component.

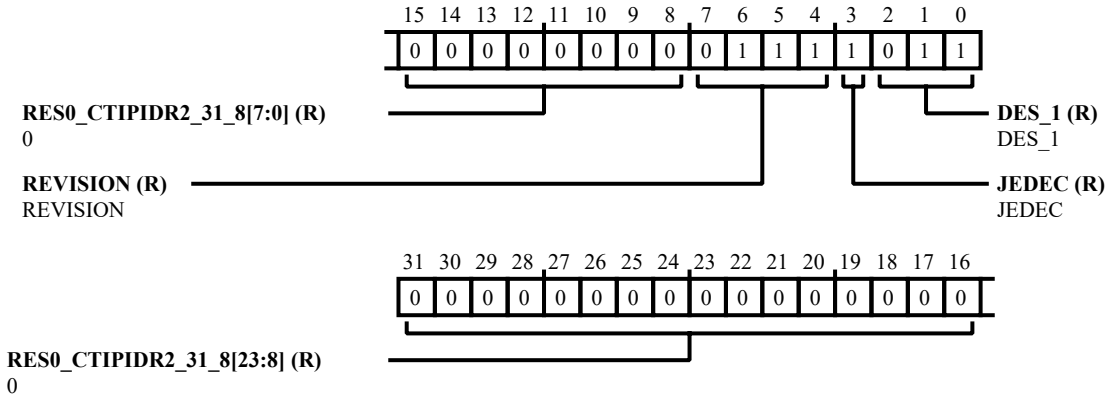


Figure 48-48: A55_CLUSTER_CTIPIDR2 Register Diagram

Table 48-46: A55_CLUSTER_CTIPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. Part major revision. Parts can also use this field to extend Part number to 16-bits.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

CTI Peripheral Identification 3 Register

Provides information to identify a CTI component.

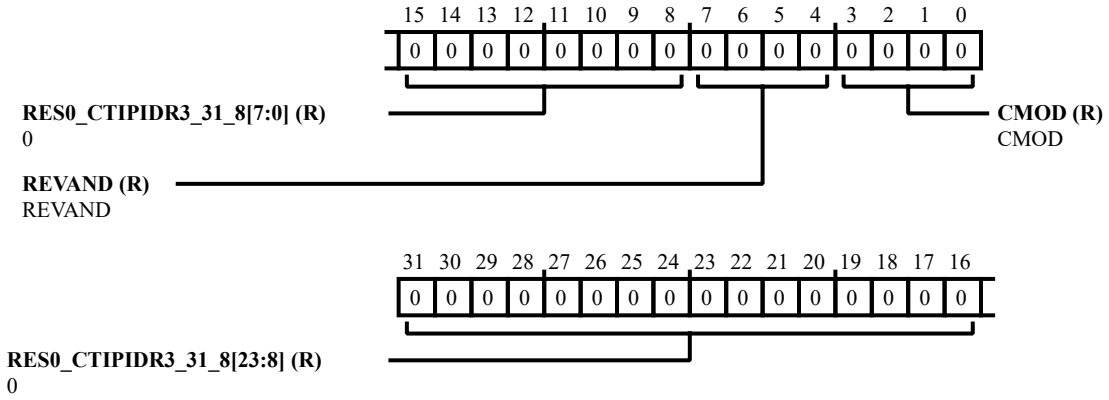


Figure 48-49: A55_CLUSTER_CTIPIDR3 Register Diagram

Table 48-47: A55_CLUSTER_CTIPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. Part minor revision. Parts using CTIPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

CTI Peripheral Identification 4 Register

Provides information to identify a CTI component.

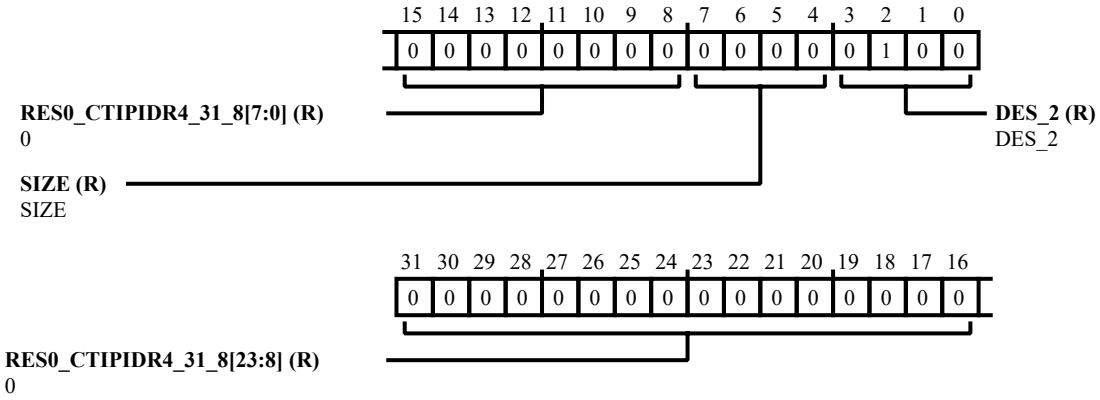


Figure 48-50: A55_CLUSTER_CTIPIDR4 Register Diagram

Table 48-48: A55_CLUSTER_CTIPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

CTI Peripheral Identification 5 Register

Provides information to identify a CTI component.

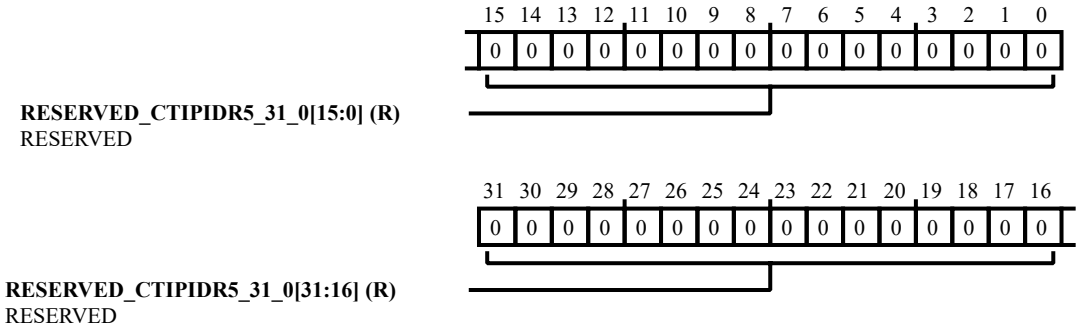


Figure 48-51: A55_CLUSTER_CTIPIDR5 Register Diagram

Table 48-49: A55_CLUSTER_CTIPIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_CTI-PIDR5_31_0	RESERVED. Reserved RES0.

CTI Peripheral Identification 6 Register

Provides information to identify a CTI component.

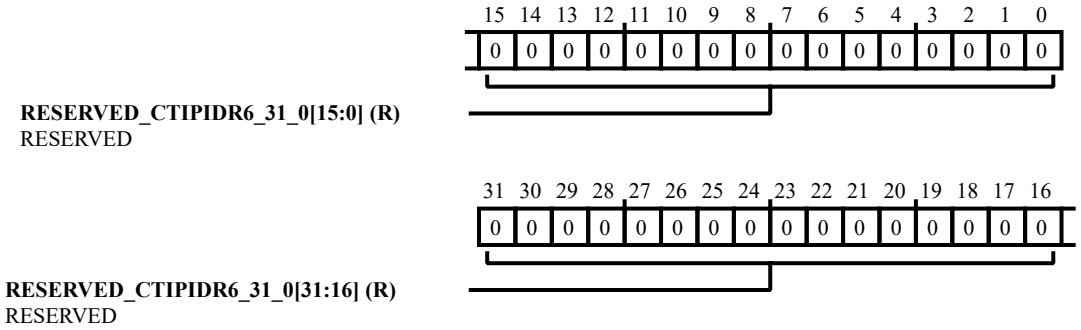


Figure 48-52: A55_CLUSTER_CTIPIDR6 Register Diagram

Table 48-50: A55_CLUSTER_CTIPIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_CTI-PIDR6_31_0	RESERVED. Reserved RES0.

CTI Peripheral Identification 7 Register

Provides information to identify a CTI component.

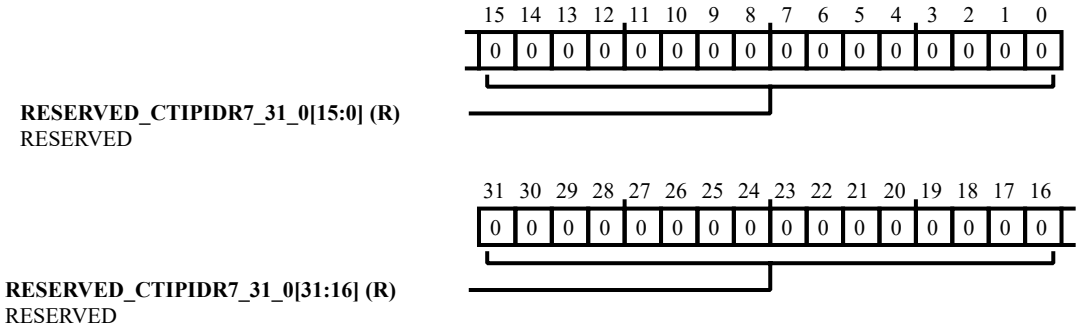


Figure 48-53: A55_CLUSTER_CTIPIDR7 Register Diagram

Table 48-51: A55_CLUSTER_CTIPIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_CTI-PIDR7_31_0	RESERVED. Reserved RES0.

CTI Trigger In Status Register

Provides the status of the trigger inputs.

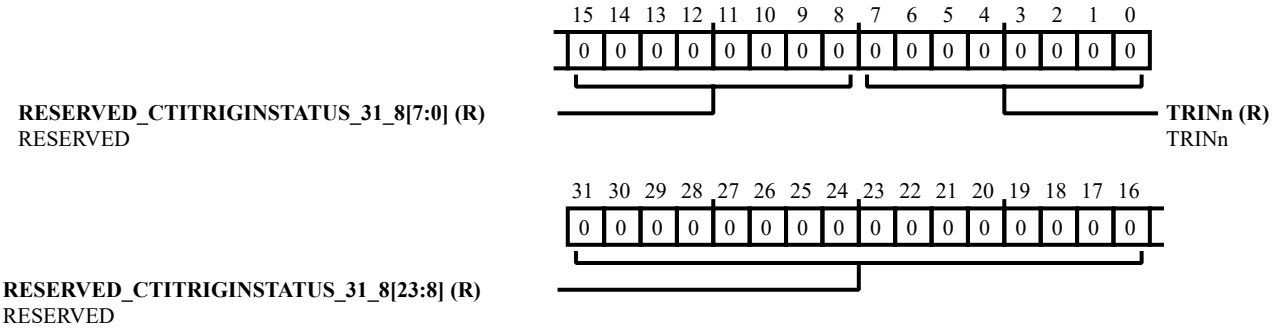


Figure 48-54: A55_CLUSTER_CTITRIGINSTATUS Register Diagram

Table 48-52: A55_CLUSTER_CTITRIGINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RESERVED_CTITRIGIN-STATUS_31_8	RESERVED. Reserved RES0.	
		0	Reserved.
7:0 (R/NW)	TRINN	TRINn. Provides the status of the trigger inputs.	
		0	Input trigger n is inactive.
		1	Input trigger n is active.

CTI Trigger Out Status Register

Provides the raw status of the trigger outputs, after processing by any IMPLEMENTATION DEFINED trigger interface logic. For output triggers that are self-acknowledging, this is only meaningful if the CTI implements multicycle channel events.

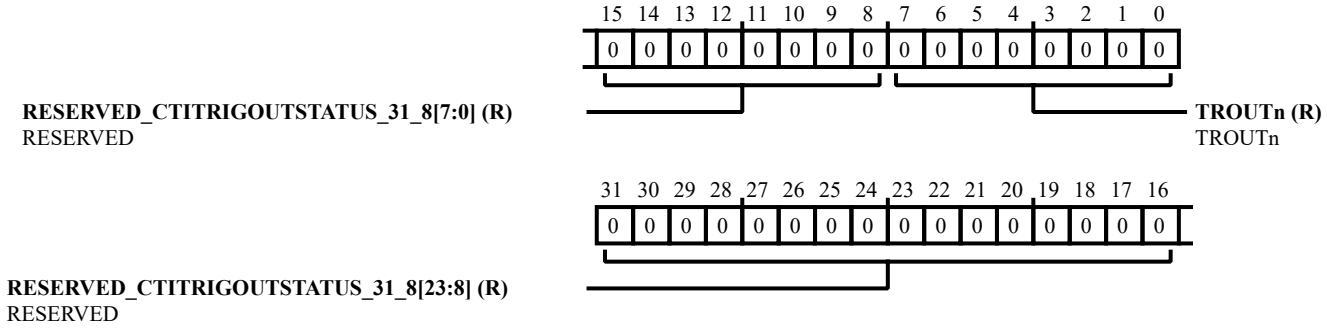


Figure 48-55: A55_CLUSTER_CTITRIGOUTSTATUS Register Diagram

Table 48-53: A55_CLUSTER_CTITRIGOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RESERVED_CTITRI-GOUTSTATUS_31_8	RESERVED. Reserved RES0.	
		0	Reserved.
7:0 (R/NW)	TROUTN	TROUTn. Provides the status of the trigger outputs.	
		0	Output trigger n is inactive.
		1	Output trigger n is active.

Debug Authentication Status Register

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

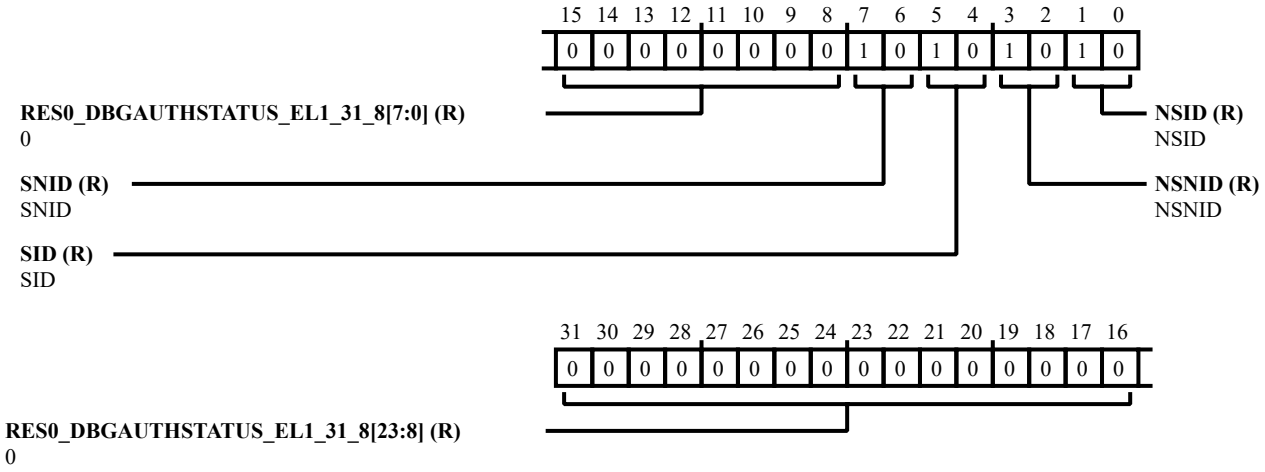


Figure 48-56: A55_CLUSTER_DBGAUTHSTATUS_EL1 Register Diagram

Table 48-54: A55_CLUSTER_DBGAUTHSTATUS_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_DBGAUTHSTATUS_EL1_31_8	0. Reserved, RES0.	
7:6 (R/NW)	SNID	SNID. Secure non-invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state. 10 Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
		2	Implemented and disabled. ExternalSecureNoninvasive-DebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalSecureNoninvasive-DebugEnabled() == TRUE.

Table 48-54: A55_CLUSTER_DBGAUTHSTATUS_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	SID	SID. Secure invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state. 10 Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
		2	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.
3:2 (R/NW)	NSNID	NSNID. Non-secure non-invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Secure state. 10 Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
		2	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.
1:0 (R/NW)	NSID	NSID. Non-secure invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Secure state. 10 Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
		2	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

External Debug Component Identification 0 Register

Provides information to identify an external debug component.

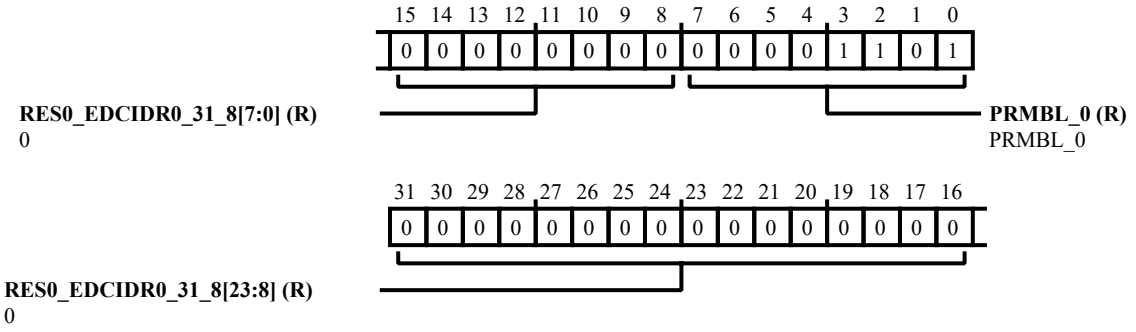


Figure 48-57: A55_CLUSTER_EDCIDR0 Register Diagram

Table 48-55: A55_CLUSTER_EDCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

External Debug Component Identification 1 Register

Provides information to identify an external debug component.

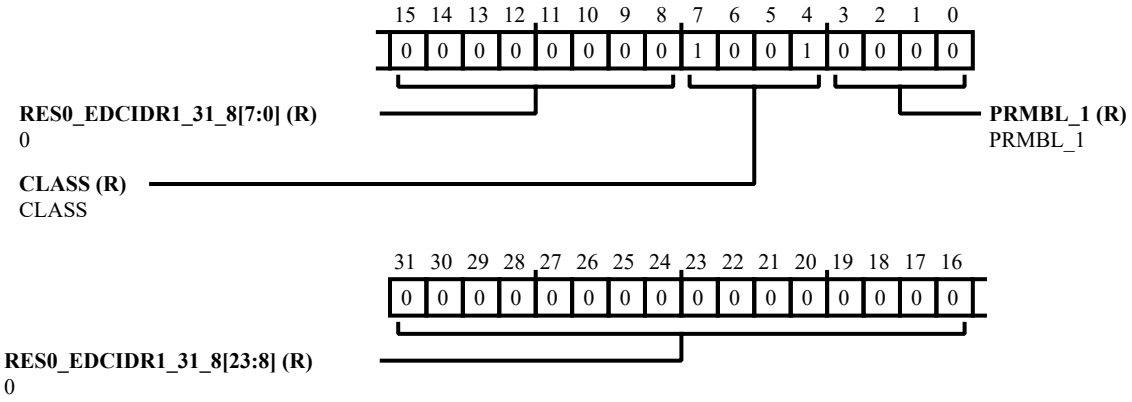


Figure 48-58: A55_CLUSTER_EDCIDR1 Register Diagram

Table 48-56: A55_CLUSTER_EDCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, debug component.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. RAZ.

External Debug Component Identification 2 Register

Provides information to identify an external debug component.

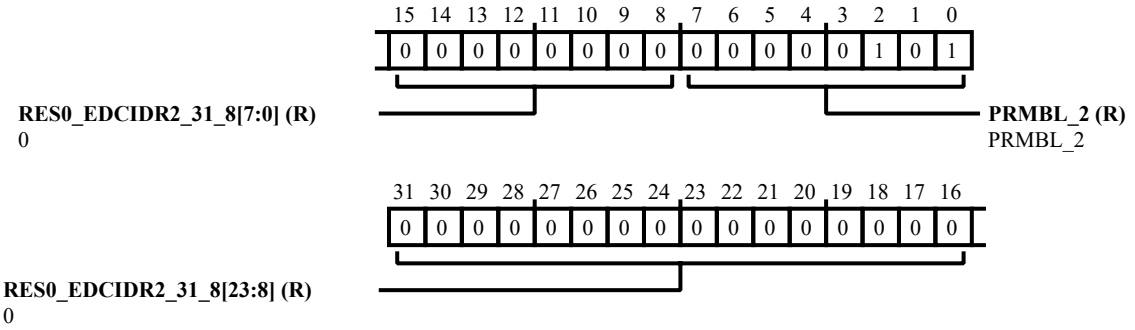


Figure 48-59: A55_CLUSTER_EDCIDR2 Register Diagram

Table 48-57: A55_CLUSTER_EDCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

External Debug Component Identification 3 Register

Provides information to identify an external debug component.

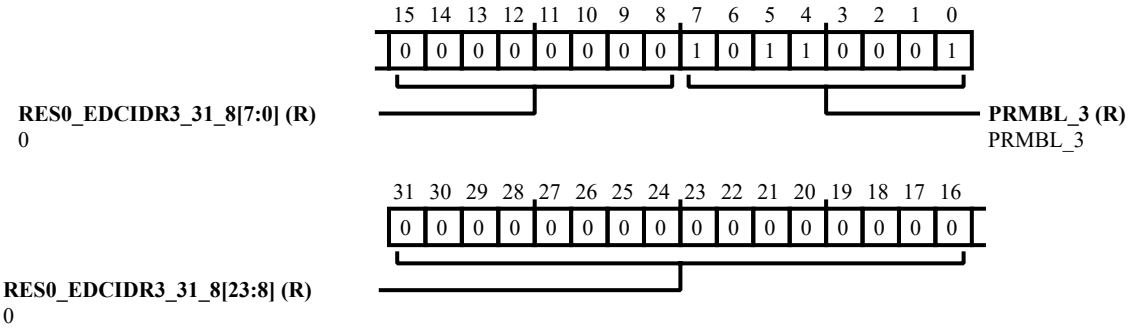


Figure 48-60: A55_CLUSTER_EDCIDR3 Register Diagram

Table 48-58: A55_CLUSTER_EDCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

External Debug Device Affinity 0 Register

Copy of the low half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

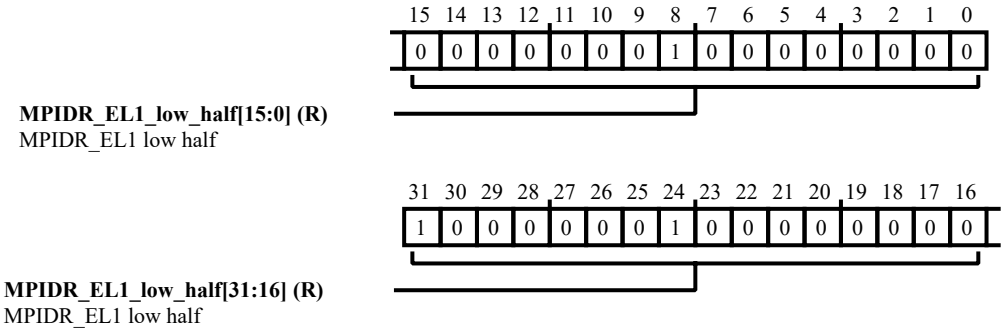


Figure 48-61: A55_CLUSTER_EDDEVAFF0 Register Diagram

Table 48-59: A55_CLUSTER_EDDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_LOW_HALF	MPIDR_EL1 low half. MPIDR_EL1 low half. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented Exception level.

External Debug Device Affinity 1 Register

Copy of the high half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

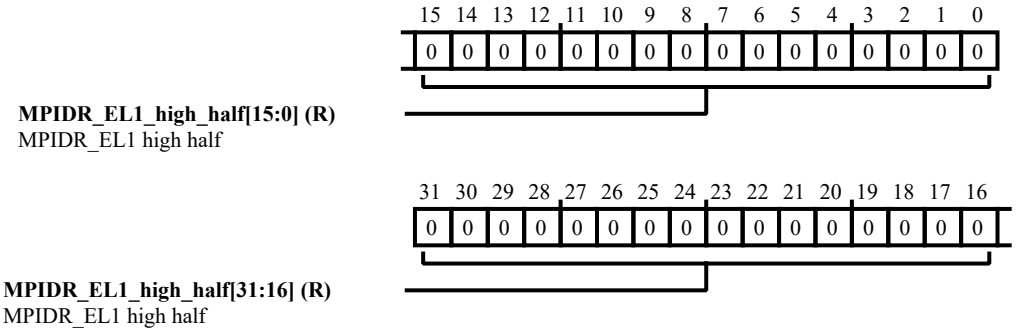


Figure 48-62: A55_CLUSTER_EDDEVAFF1 Register Diagram

Table 48-60: A55_CLUSTER_EDDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_HIGH_HALF	MPIDR_EL1 high half. MPIDR_EL1 high half. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented Exception level.

External Debug Device Architecture Register

Identifies the programmers' model architecture of the external debug component.

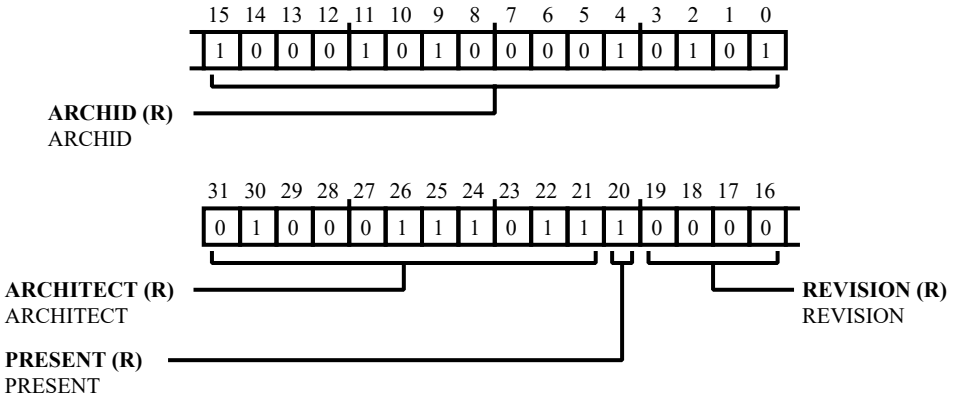


Figure 48-63: A55_CLUSTER_EDDEVARCH Register Diagram

Table 48-61: A55_CLUSTER_EDDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For debug, this is ARM Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present. This field is 1 in ARMv8.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision. For debug, the revision defined by ARMv8 is 0x0. All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided. For debug: Bits [15:12] are the architecture version:In ARMv8-A, this is 0x6.In ARMv8.1, this is 0x7.In ARMv8.2, this is0x8. Bits [11:0] are the architecture part number, 0xA15. This corresponds to the ARMv8 debug architecture version.

External Debug Device ID 0 Register

Provides extra information for external debuggers about features of the debug implementation.

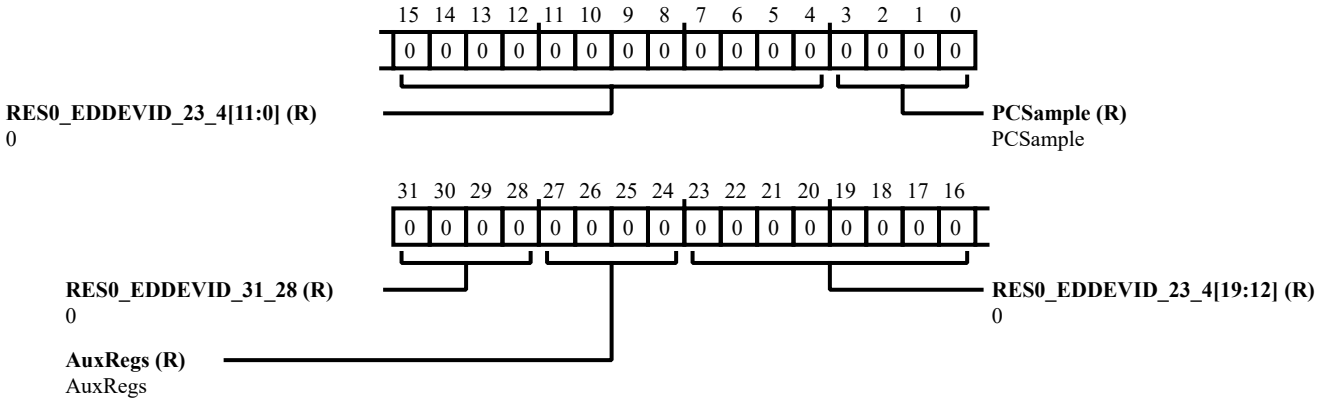


Figure 48-64: A55_CLUSTER_EDDEVID Register Diagram

Table 48-62: A55_CLUSTER_EDDEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_EDDEVID_31_28	0. Reserved, RES0.	
27:24 (R/NW)	AUXREGS	AuxRegs. Indicates support for Auxiliary registers. Permitted values for this field are: 0000 None supported. 0001 Support for External Debug Auxiliary Control Register, EDACR. All other values are reserved.	
		0	None supported.
		1	Support for External Debug Auxiliary Control Register, EDACR.
23:4 (R/NW)	RES0_EDDEVID_23_4	0. Reserved, RES0.	

Table 48-62: A55_CLUSTER_EDDEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/NW)	PCSAMPLE	PCSample. Indicates the level of PC Sample-based profiling support using external debug registers 40 through 43. Permitted values of this field in ARMv8 are: 0000 Architecture-defined form of PC Sample-based profiling not implemented. 0010 EDPCSR and EDCIDSR are implemented (only permitted if EL3 and EL2 are not implemented). 0011 EDPCSR, EDCIDSR, and EDVIDSR are implemented. All other values are reserved. In ARMv8.2 implementations, this field reads as 0b0000.	
		0	Architecture-defined form of PC Sample-based profiling not implemented.
		2	EDPCSR and EDCIDSR are implemented (only permitted if EL3 and EL2 are not implemented).
		3	EDPCSR, EDCIDSR, and EDVIDSR are implemented.

External Debug Device ID 1 Register

Provides extra information for external debuggers about features of the debug implementation.

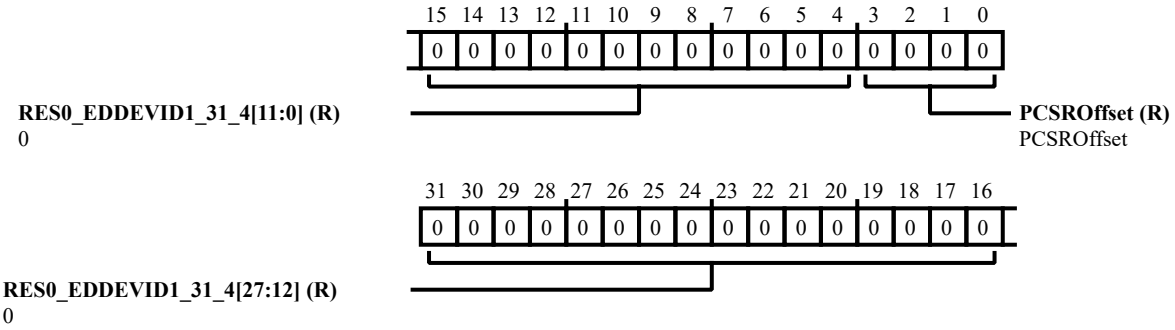


Figure 48-65: A55_CLUSTER_EDDEVID1 Register Diagram

Table 48-63: A55_CLUSTER_EDDEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_EDDEVID1_31_4	0. Reserved, RES0.	
3:0 (R/NW)	PCSROFFSET	PCSROffset. This field indicates the offset applied to PC samples returned by reads of EDPCSR. Permitted values of this field in ARMv8 are: 0000 EDPCSR not implemented. 0010 EDPCSR implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state. In ARMv8.2 implementations, this field reads as 0b0000.	
		0	EDPCSR not implemented.
		2	EDPCSR implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state.

External Debug Device ID 2 Register

Reserved for future descriptions of features of the debug implementation.

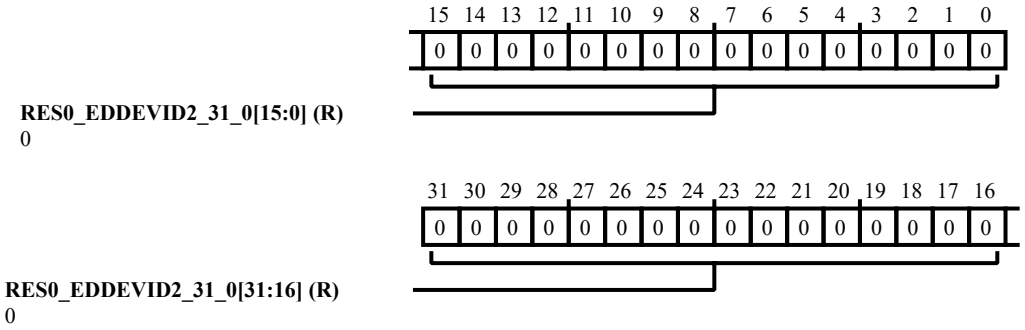


Figure 48-66: A55_CLUSTER_EDDEVID2 Register Diagram

Table 48-64: A55_CLUSTER_EDDEVID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_EDDEVID2_31_0	0. Reserved, RES0.

External Debug Device Type Register

Indicates to a debugger that this component is part of a PEs debug logic.

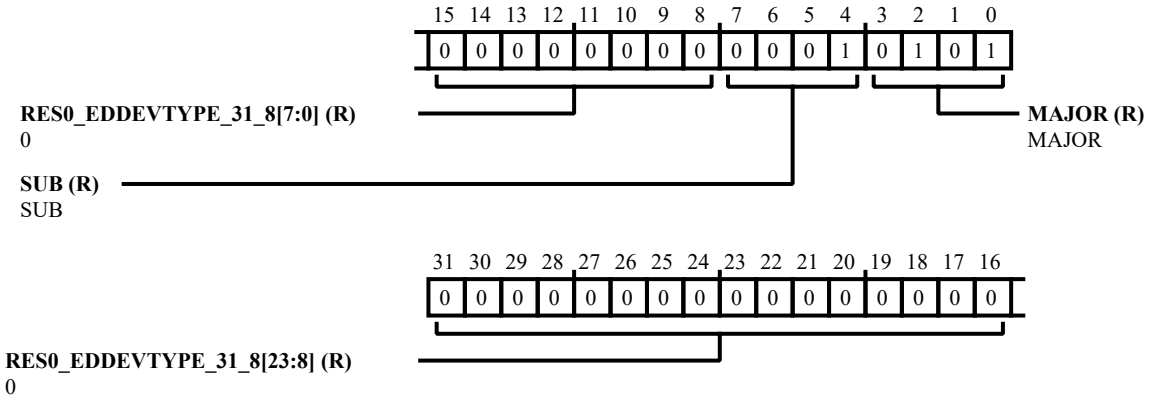


Figure 48-67: A55_CLUSTER_EDDEVTYPE Register Diagram

Table 48-65: A55_CLUSTER_EDDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDDEVTYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Subtype. Must read as 0x1 to indicate this is a component within a PE.
3:0 (R/NW)	MAJOR	MAJOR. Major type. Must read as 0x5 to indicate this is a debug logic component.

External Debug Feature Register

Provides top level information about the debug system. Debuggers must use EDDEVARCH to determine the Debug architecture version. For general information about the interpretation of the ID registers see .

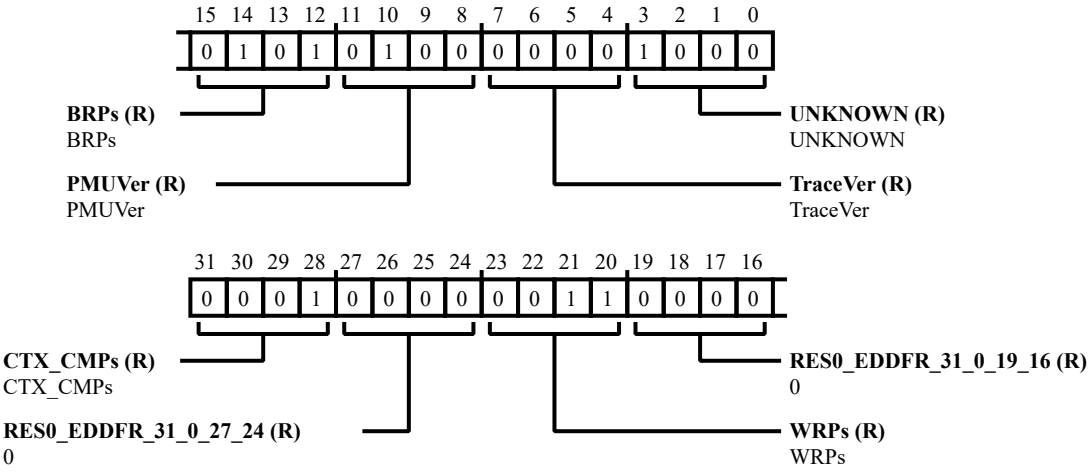


Figure 48-68: A55_CLUSTER_EDDFR_31_0 Register Diagram

Table 48-66: A55_CLUSTER_EDDFR_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/NW)	CTX_CMPS	CTX_CMPs. Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.CTX_CMPs.
27:24 (R/NW)	RES0_EDDFR_31_0_27_24	0. Reserved, RES0.
23:20 (R/NW)	WRPS	WRPs. Number of watchpoints, minus 1. The value of 0b0000 is reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.WRPs.
19:16 (R/NW)	RES0_EDDFR_31_0_19_16	0. Reserved, RES0.
15:12 (R/NW)	BRPS	BRPs. Number of breakpoints, minus 1. The value of 0b0000 is reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.BRPs.

Table 48-66: A55_CLUSTER_EDDFR_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/NW)	PMUVER	PMUVer. Performance Monitors Extension version. Indicates whether System register interface to Performance Monitors extension is implemented. Defined values are: 0000 Performance Monitors Extension System registers not implemented. 0001 Performance Monitors Extension System registers implemented, PMUv3. 0100 Performance Monitors Extension System registers implemented, PMUv3, with a 16-bit evtCount field, and if EL2 is implemented, the addition of the MDCR_EL2.HPMD bit. 1111 IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported. All other values are reserved. In ARMv8-A the permitted values are 0b0000, 0b0001 and 0b1111. In ARMv8.1 the permitted values are 0b0000, 0b0100 and 0b1111. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.PMUVer.	
		0	Performance Monitors Extension System registers not implemented.
		1	Performance Monitors Extension System registers implemented, PMUv3.
		4	Performance Monitors Extension System registers implemented, PMUv3, with a 16-bit evtCount field, and if EL2 is implemented, the addition of the MDCR_EL2.HPMD bit.
		15	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported.
7:4 (R/NW)	TRACEVER	TraceVer. Trace support. Indicates whether System register interface to a trace macrocell is implemented. Defined values are: 0000 Trace macrocell System registers not implemented. 0001 Trace macrocell System registers implemented. All other values are reserved. A value of 0b0000 only indicates that no System register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a System register interface. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.TraceVer.	
		0	Trace macrocell System registers not implemented.
		1	Trace macrocell System registers implemented.
3:0 (R/NW)	UNKNOWN	UNKNOWN. Reserved, UNKNOWN.	

External Debug Feature Register

Provides top level information about the debug system. Debuggers must use EDDEVARCH to determine the Debug architecture version. For general information about the interpretation of the ID registers see .

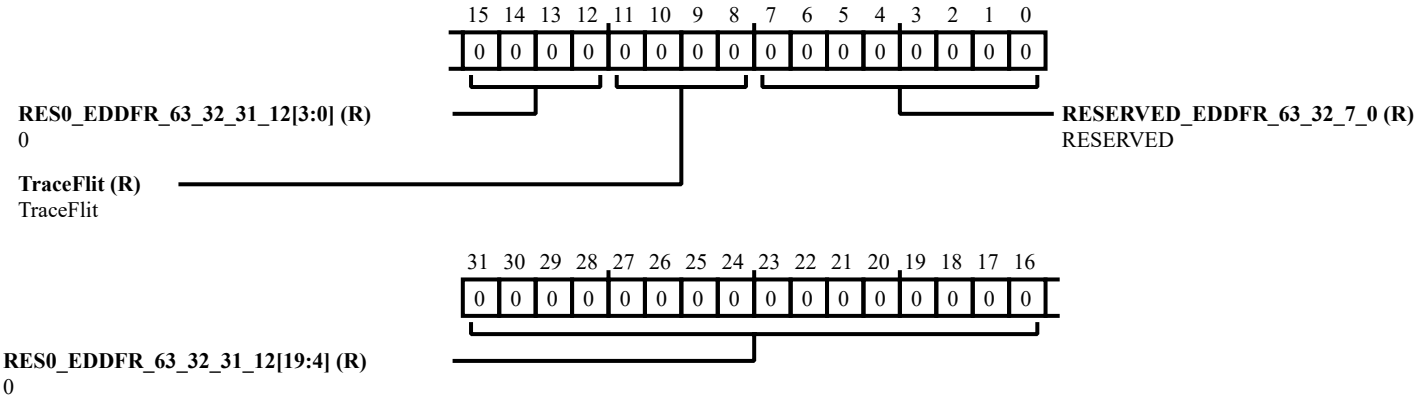


Figure 48-69: A55_CLUSTER_EDDFR_63_32 Register Diagram

Table 48-67: A55_CLUSTER_EDDFR_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_EDDFR_63_32_31_12	0. Reserved, RES0.
11:8 (R/NW)	TRACEFLIT	TraceFlit. ARMv8.4 Self-hosted Trace Extension implemented.
7:0 (R/NW)	RESERVED_EDDFR_63_32_7_0	RESERVED. Reserved. This field reads as an UNKNOWN value.

External Debug Execution Control Register

Controls Halting debug events.

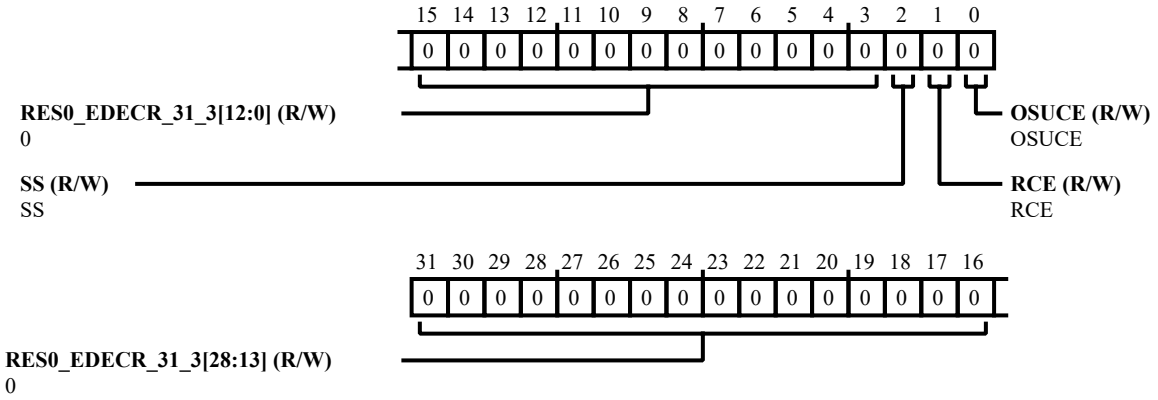


Figure 48-70: A55_CLUSTER_EDECR Register Diagram

Table 48-68: A55_CLUSTER_EDECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/W)	RES0_EDECR_31_3	0. Reserved, RES0.	
2 (R/W)	SS	SS. Halting step enable. Possible values of this field are: 0 Halting step debug event disabled. 1 Halting step debug event enabled. If the value of EDECR.SS is changed when the PE is in Non-debug state, behavior is CONSTRAINED UNPREDICTABLE as described in .	
		0	Halting step debug event disabled.
		1	Halting step debug event enabled.
1 (R/W)	RCE	RCE. Reset Catch enable. Possible values of this field are: 0 Reset Catch debug event disabled. 1 Reset Catch debug event enabled.	
		0	Reset Catch debug event disabled.
		1	Reset Catch debug event enabled.
0 (R/W)	OSUCE	OSUCE. OS Unlock Catch enabled. Possible values of this field are: 0 OS Unlock Catch debug event disabled. 1 OS Unlock Catch debug event enabled.	
		0	OS Unlock Catch debug event disabled.
		1	OS Unlock Catch debug event enabled.

External Debug Lock Access Register

Allows or disallows access to the external debug registers through a memory-mapped interface.

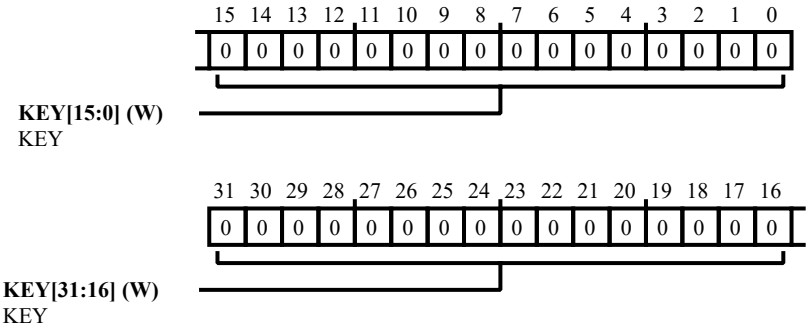


Figure 48-71: A55_CLUSTER_EDLAR Register Diagram

Table 48-69: A55_CLUSTER_EDLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

External Debug Lock Status Register

Indicates the current status of the software lock for external debug registers.

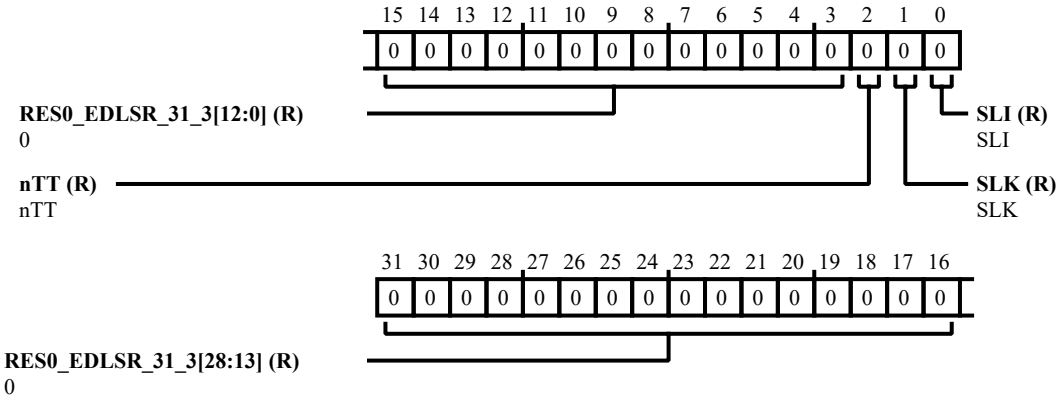


Figure 48-72: A55_CLUSTER_EDLSR Register Diagram

Table 48-70: A55_CLUSTER_EDLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_EDLSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0. For memory-mapped accesses when the Software Lock is implemented, possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

Table 48-70: A55_CLUSTER_EDLSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SLI	SLI. Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are: 0 Software Lock not implemented or not memory-mapped access. 1 Software Lock implemented and memory-mapped access.	
		0	Software Lock not implemented or not memory-mapped access.
		1	Software Lock implemented and memory-mapped access.

External Debug Processor Feature Register

Provides information about implemented PE features. For general information about the interpretation of the ID registers see .

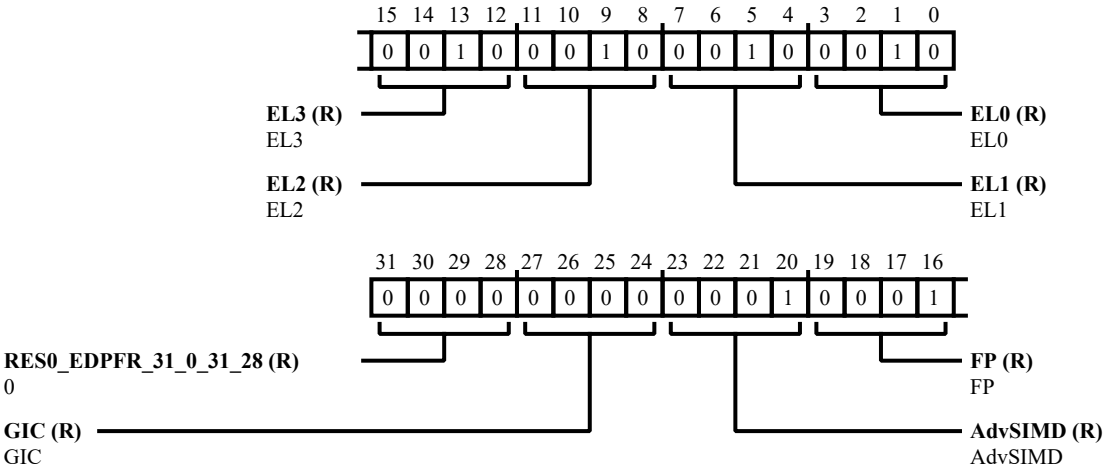


Figure 48-73: A55_CLUSTER_EDPFR_31_0 Register Diagram

Table 48-71: A55_CLUSTER_EDPFR_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_EDPFR_31_0_31_28	0. Reserved, RES0.	
27:24 (R/NW)	GIC	GIC. System register GIC interface support. Defined values are: 0000 No System register interface to the GIC is supported. 0001 System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.GIC.	
		0	No System register interface to the GIC is supported.
		1	System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported.
23:20 (R/NW)	ADVSIMD	AdvSIMD. Advanced SIMD. Defined values are: 0000 Advanced SIMD is implemented. 1111 Advanced SIMD is not implemented. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.AdvSIMD.	
		0	Advanced SIMD is implemented.
		15	Advanced SIMD is not implemented.

Table 48-71: A55_CLUSTER_EDPFR_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/NW)	FP	FP. Floating-point. Defined values are: 0000 Floating-point is implemented. 1111 Floating-point is not implemented. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.FP.	
		0	Floating-point is implemented.
		15	Floating-point is not implemented.
15:12 (R/NW)	EL3	EL3. AArch64 EL3 Exception level handling. Defined values are: 0000 EL3 is not implemented or cannot be executed in AArch64 state. 0001 EL3 can be executed in AArch64 state only. 0010 EL3 can be executed in either AArch64 or AArch32 state. When the value of EDAA32PFR.EL3 is non-zero, this field must be 0000. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL3.	
		0	EL3 is not implemented or cannot be executed in AArch64 state.
		1	EL3 can be executed in AArch64 state only.
		2	EL3 can be executed in either AArch64 or AArch32 state.
11:8 (R/NW)	EL2	EL2. AArch64 EL2 Exception level handling. Defined values are: 0000 EL2 is not implemented or cannot be executed in AArch64 state. 0001 EL2 can be executed in AArch64 state only. 0010 EL2 can be executed in either AArch64 or AArch32 state. When the value of EDAA32PFR.EL2 is non-zero, this field must be 0000. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL2.	
		0	EL2 is not implemented or cannot be executed in AArch64 state.
		1	EL2 can be executed in AArch64 state only.
		2	EL2 can be executed in either AArch64 or AArch32 state.

Table 48-71: A55_CLUSTER_EDPFR_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	EL1	EL1. AArch64 EL1 Exception level handling. Defined values are: 0000 EL1 can be executed in AArch32 state only. 0001 EL1 can be executed in AArch64 state only. 0010 EL1 can be executed in either AArch64 or AArch32 state. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL1.	
		0	EL1 can be executed in AArch32 state only.
		1	EL1 can be executed in AArch64 state only.
		2	EL1 can be executed in either AArch64 or AArch32 state.
3:0 (R/NW)	EL0	EL0. AArch64 EL0 Exception level handling. Defined values are: 0000 EL0 can be executed in AArch32 state only. 0001 EL0 can be executed in AArch64 state only. 0010 EL0 can be executed in either AArch64 or AArch32 state. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL0.	
		0	EL0 can be executed in AArch32 state only.
		1	EL0 can be executed in AArch64 state only.
		2	EL0 can be executed in either AArch64 or AArch32 state.

External Debug Processor Feature Register

Provides information about implemented PE features. For general information about the interpretation of the ID registers see .

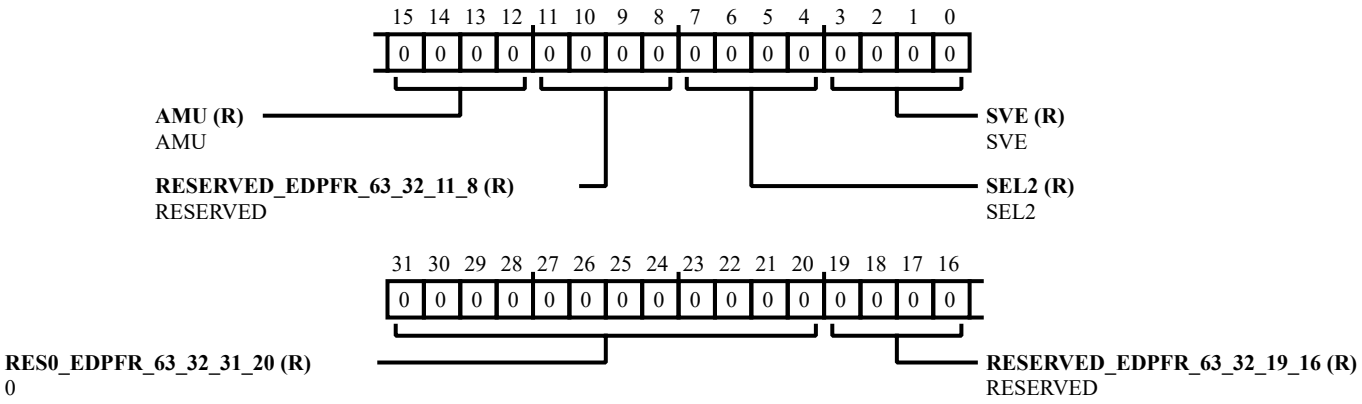


Figure 48-74: A55_CLUSTER_EDPFR_63_32 Register Diagram

Table 48-72: A55_CLUSTER_EDPFR_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:20 (R/NW)	RES0_EDPFR_63_32_31_20	0. Reserved, RES0.
19:16 (R/NW)	RESERVED_EDPFR_63_32_19_16	RESERVED. Reserved. This field reads as an UNKNOWN value.
15:12 (R/NW)	AMU	AMU. AMU.
11:8 (R/NW)	RESERVED_EDPFR_63_32_11_8	RESERVED. Reserved. This field reads as an UNKNOWN value.
7:4 (R/NW)	SEL2	SEL2. Secure EL2 implemented.
3:0 (R/NW)	SVE	SVE. SVE.

External Debug Peripheral Identification 0 Register

Provides information to identify an external debug component.

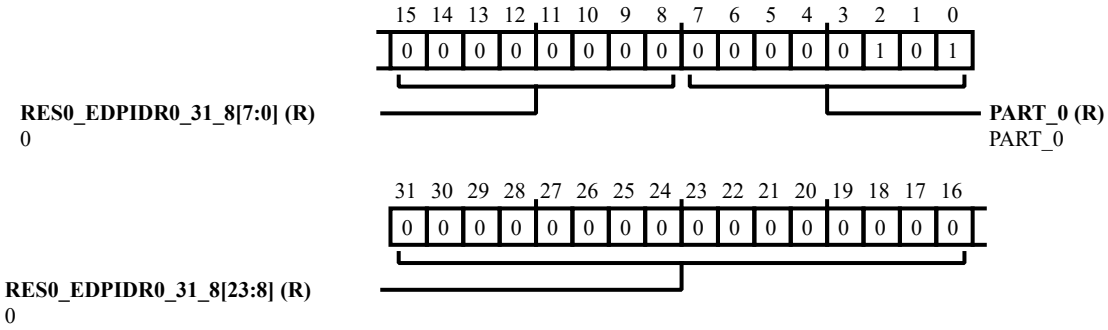


Figure 48-75: A55_CLUSTER_EDPIDR0 Register Diagram

Table 48-73: A55_CLUSTER_EDPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, least significant byte.

External Debug Peripheral Identification 1 Register

Provides information to identify an external debug component.

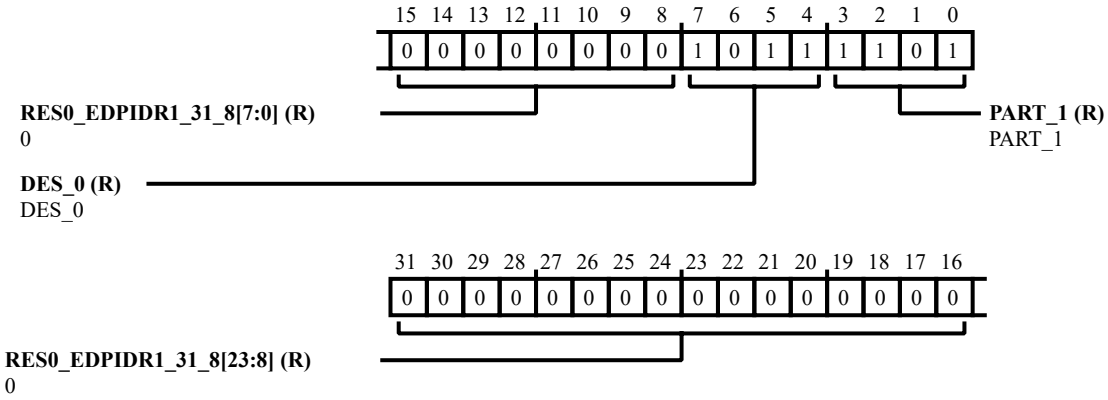


Figure 48-76: A55_CLUSTER_EDPIDR1 Register Diagram

Table 48-74: A55_CLUSTER_EDPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, most significant nibble.

External Debug Peripheral Identification 2 Register

Provides information to identify an external debug component.

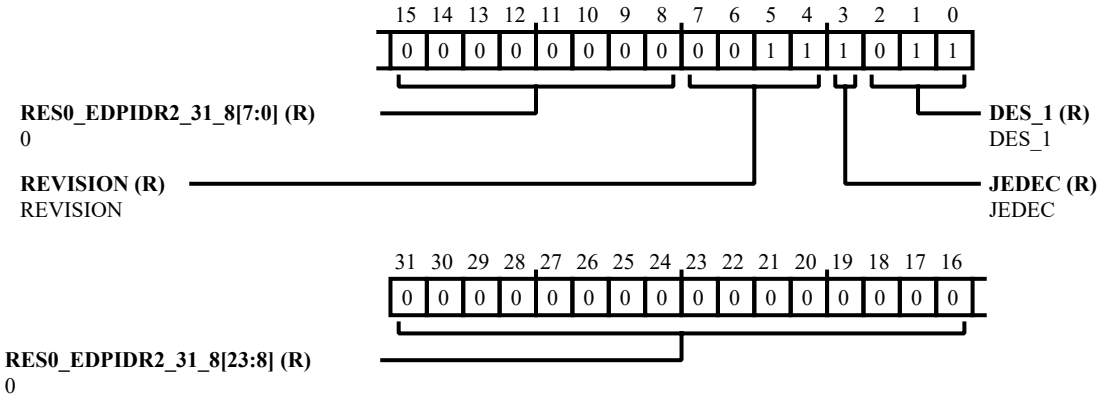


Figure 48-77: A55_CLUSTER_EDPIDR2 Register Diagram

Table 48-75: A55_CLUSTER_EDPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. Part major revision. Parts can also use this field to extend Part number to 16-bits.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

External Debug Peripheral Identification 3 Register

Provides information to identify an external debug component.

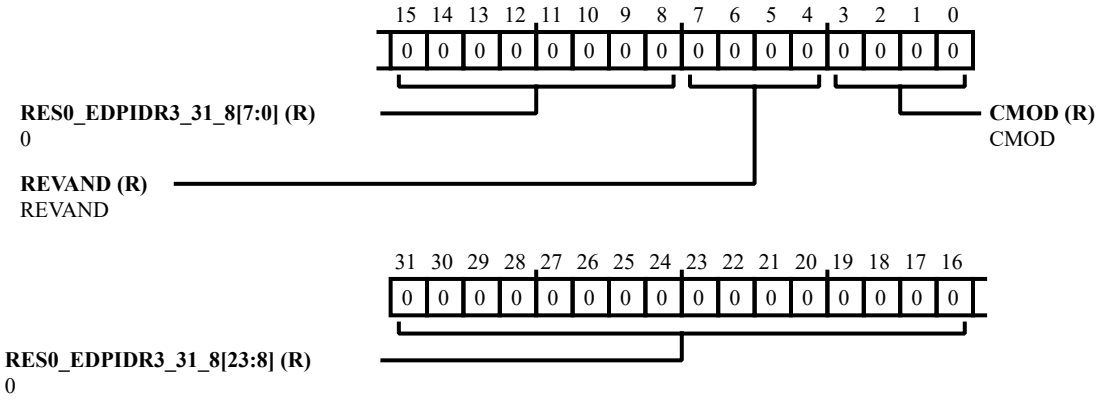


Figure 48-78: A55_CLUSTER_EDPIDR3 Register Diagram

Table 48-76: A55_CLUSTER_EDPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. Part minor revision. Parts using EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

External Debug Peripheral Identification 4 Register

Provides information to identify an external debug component.

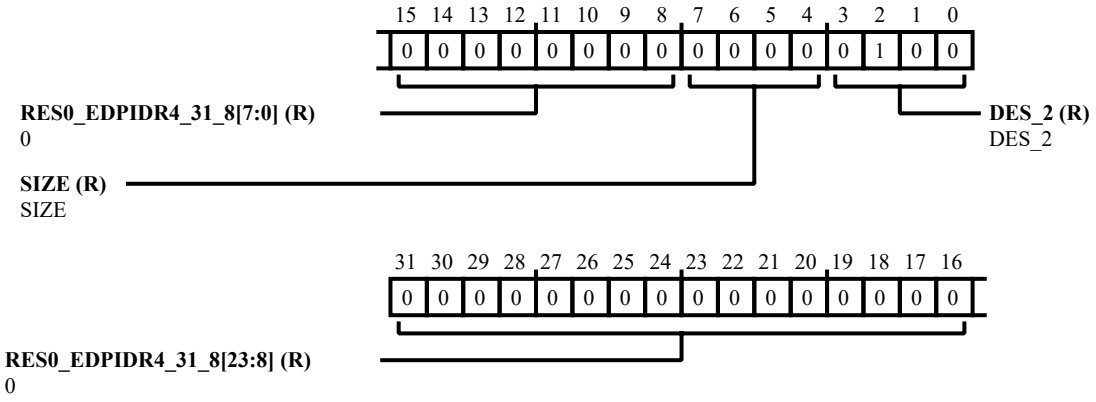


Figure 48-79: A55_CLUSTER_EDPIDR4 Register Diagram

Table 48-77: A55_CLUSTER_EDPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

External Debug Peripheral Identification 5 Register

Provides information to identify an external debug component.

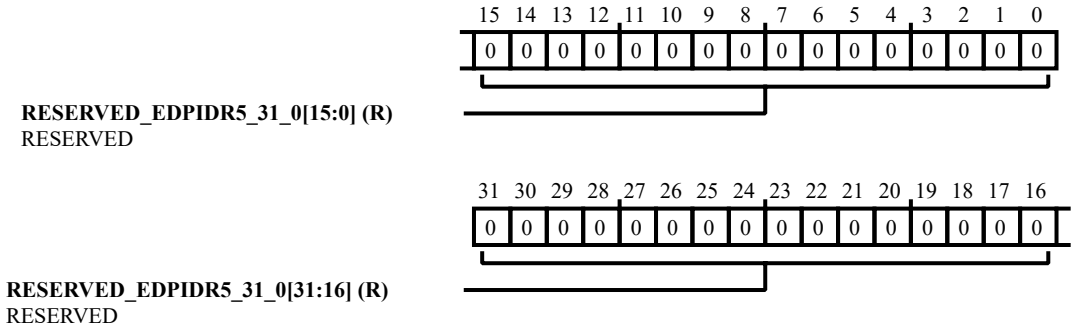


Figure 48-80: A55_CLUSTER_EDPIDR5 Register Diagram

Table 48-78: A55_CLUSTER_EDPIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_ED-PIDR5_31_0	RESERVED. Reserved RES0.

External Debug Peripheral Identification 6 Register

Provides information to identify an external debug component.

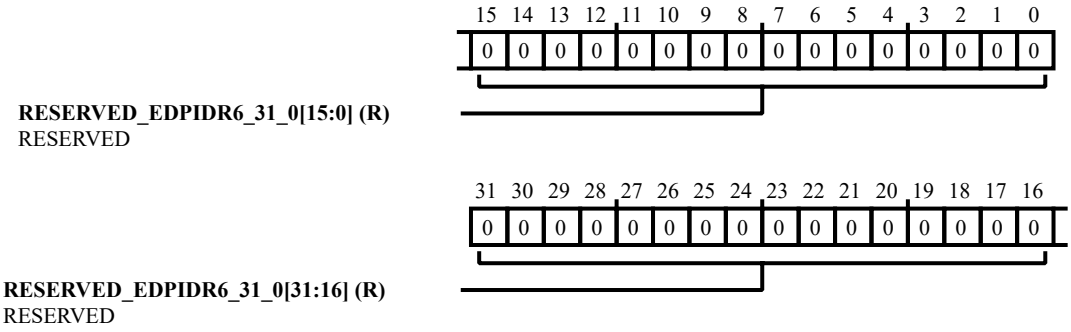


Figure 48-81: A55_CLUSTER_EDPIDR6 Register Diagram

Table 48-79: A55_CLUSTER_EDPIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_ED-PIDR6_31_0	RESERVED. Reserved RES0.

External Debug Peripheral Identification 7 Register

Provides information to identify an external debug component.

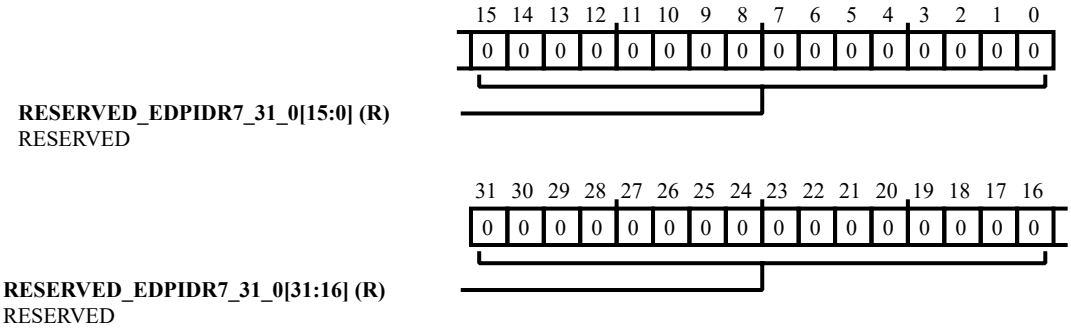


Figure 48-82: A55_CLUSTER_EDPIDR7 Register Diagram

Table 48-80: A55_CLUSTER_EDPIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_ED-PIDR7_31_0	RESERVED. Reserved RES0.

Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

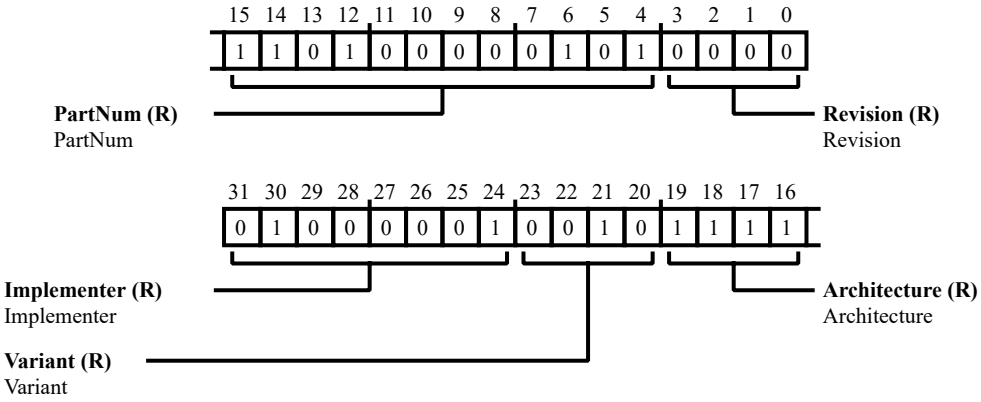


Figure 48-83: A55_CLUSTER_MIDR_EL1 Register Diagram

Table 48-81: A55_CLUSTER_MIDR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/NW)	IMPLEMENTER	Implementer. The Implementer code. This field must hold an implementer code that has been assigned by ARM.	
23:20 (R/NW)	VARIANT	Variant. An IMPLEMENTATION DEFINED variant number. Typically this field is used to distinguish between different product variants or major revisions of a product.	
19:16 (R/NW)	ARCHITECTURE	Architecture. The Architecture used.	
		1	ARMv4.
		2	ARMv4T.
		3	ARMv5 (obsolete).
		4	ARMv5T.
		5	ARMv5TE.
		6	ARMv5TEJ.
		7	ARMv6.
		15	Defined by CPUID scheme.

Table 48-81: A55_CLUSTER_MIDR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:4 (R/NW)	PARTNUM	PartNum. An IMPLEMENTATION DEFINED primary part number for the device. On processors implemented by ARM if the top four bits of the primary part number are 0x0 or 0x7 the variant and architecture are encoded differently.
3:0 (R/NW)	REVISION	Revision. An IMPLEMENTATION DEFINED revision number for the device.

Trace Authentication Status Register

Returns the level of tracing that the trace unit can support.

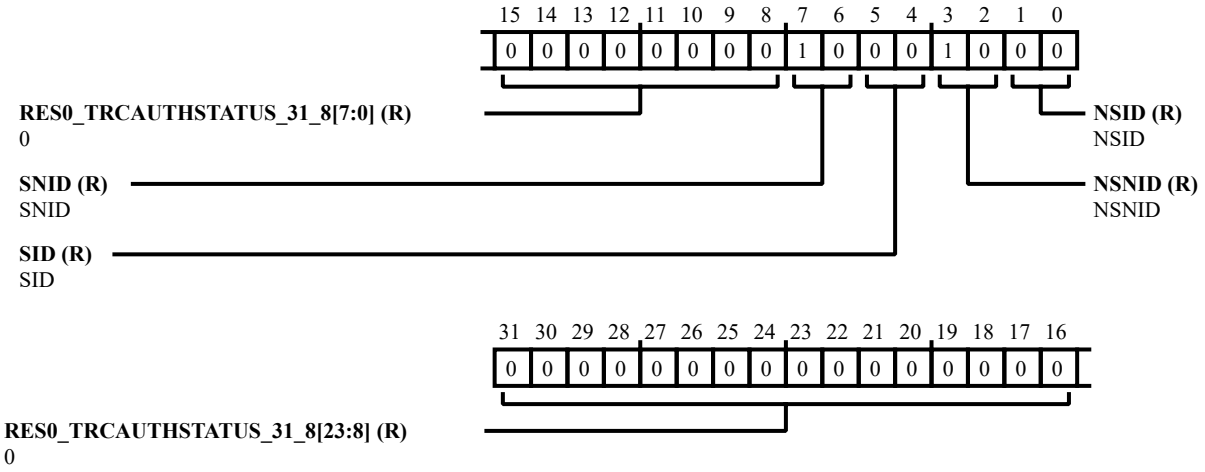


Figure 48-84: A55_CLUSTER_TRCAUTHSTATUS Register Diagram

Table 48-82: A55_CLUSTER_TRCAUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_TRCAUTHSTA-TUS_31_8	0. Reserved, RES0.	
7:6 (R/NW)	SNID	SNID. Indicates whether the system enables the trace unit to support Secure non-invasive debug: 00 The trace unit does not implement support for Secure non-invasive debug. 01 Reserved. 10 Secure non-invasive debug is disabled. 11 Secure non-invasive debug is enabled.	
		0	The trace unit does not implement support for Secure non-invasive debug.
		1	Reserved.
		2	Secure non-invasive debug is disabled.
		3	Secure non-invasive debug is enabled.
5:4 (R/NW)	SID	SID. Indicates whether the trace unit supports Secure invasive debug: 00 The trace unit does not support Secure invasive debug. All other values are reserved.	
		0	The trace unit does not support Secure invasive debug.

Table 48-82: A55_CLUSTER_TRCAUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/NW)	NSNID	NSNID. Indicates whether the system enables the trace unit to support Non-secure non-invasive debug: 00 The trace unit does not implement support for Non-secure non-invasive debug. 01 Reserved. 10 Non-secure non-invasive debug is disabled. 11 Non-secure non-invasive debug is enabled.	
		0	The trace unit does not implement support for Non-secure non-invasive debug.
		1	Reserved.
		2	Non-secure non-invasive debug is disabled.
		3	Non-secure non-invasive debug is enabled.
1:0 (R/NW)	NSID	NSID. Indicates whether the trace unit supports Non-secure invasive debug: 00 The trace unit does not support Non-secure invasive debug. All other values are reserved.	
		0	The trace unit does not support Non-secure invasive debug.

Trace Component ID 0 Register

Return byte 0 of the CoreSight preamble information.

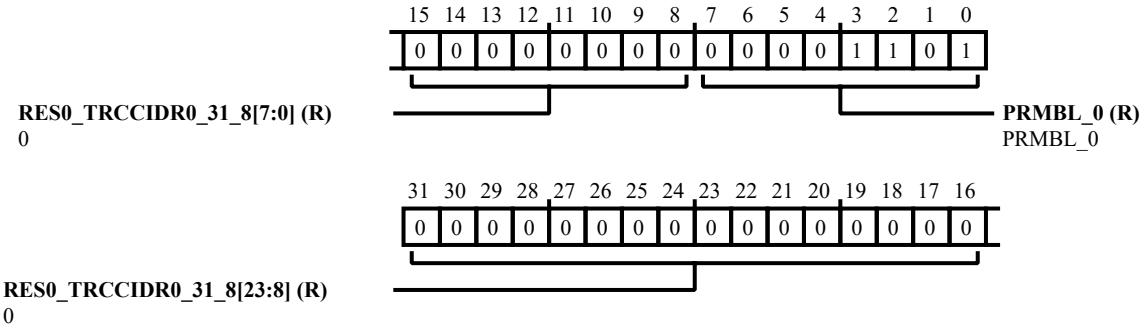


Figure 48-85: A55_CLUSTER_TRCCIDR0 Register Diagram

Table 48-83: A55_CLUSTER_TRCCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

Trace Component ID 1 Register

Return byte 1 of the CoreSight preamble information.

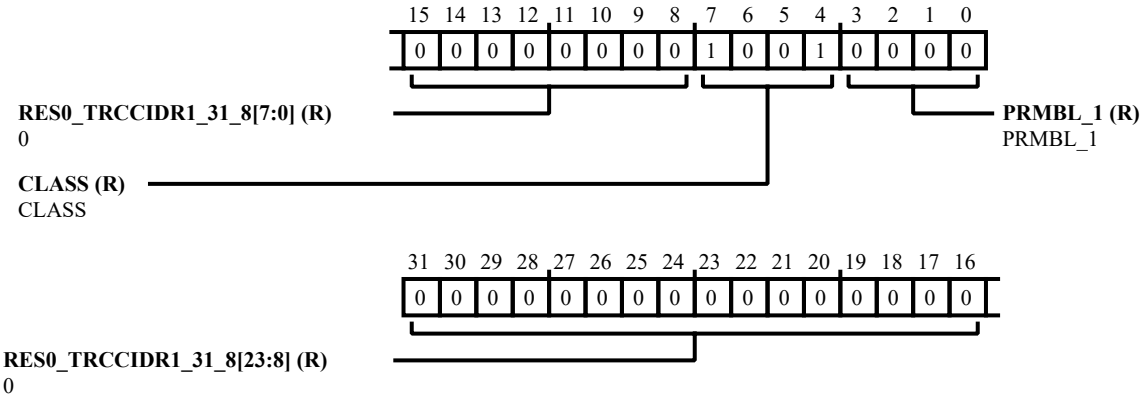


Figure 48-86: A55_CLUSTER_TRCCIDR1 Register Diagram

Table 48-84: A55_CLUSTER_TRCCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, to indicate that the ETM is a debug component, with CoreSight architecture compliant management registers.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. Must read as 0x0.

Trace Component ID 2 Register

Return byte 2 of the CoreSight preamble information.

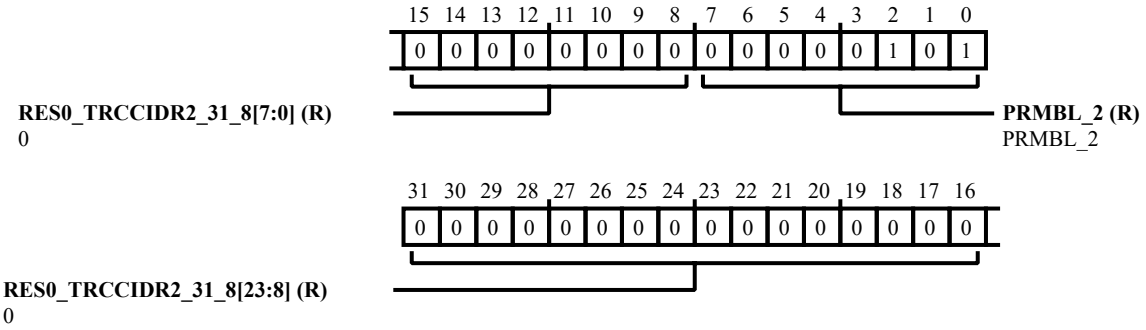


Figure 48-87: A55_CLUSTER_TRCCIDR2 Register Diagram

Table 48-85: A55_CLUSTER_TRCCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

Trace Component ID 3 Register

Return byte 3 of the CoreSight preamble information.

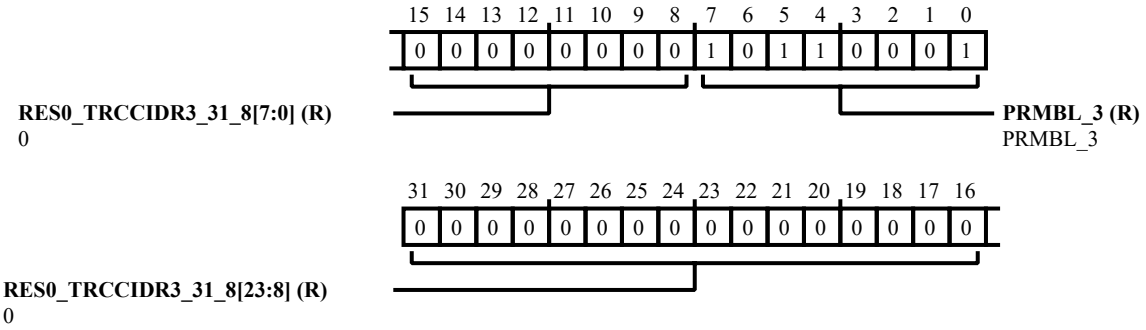


Figure 48-88: A55_CLUSTER_TRCCIDR3 Register Diagram

Table 48-86: A55_CLUSTER_TRCCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

Trace Device Affinity 0 Register

Returns the value of the lower 32 bits of MPIDR_EL1 as seen from the highest implemented exception level, unaffected by VMPIDR_EL2. If the trace unit is shared between multiple processors, the value in this register indicates the MPIDR value of the processor that TRCPROCSELR selects.

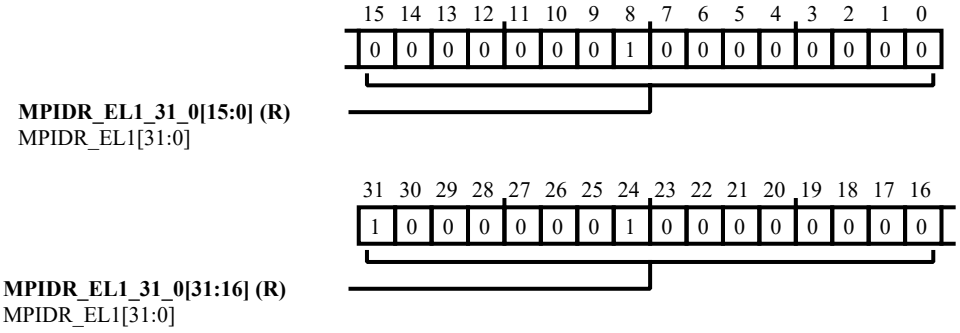


Figure 48-89: A55_CLUSTER_TRCDEVAFF0 Register Diagram

Table 48-87: A55_CLUSTER_TRCDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_31_0	MPIDR_EL1[31:0]. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented exception level.

Trace Device Affinity 1 Register

Returns the value of the upper 32 bits of MPIDR_EL1 as seen from the highest implemented exception level, unaffected by VMPIDR_EL2. If the trace unit is shared between multiple processors, the value in this register indicates the MPIDR value of the processor that TRCPROCSELR selects.

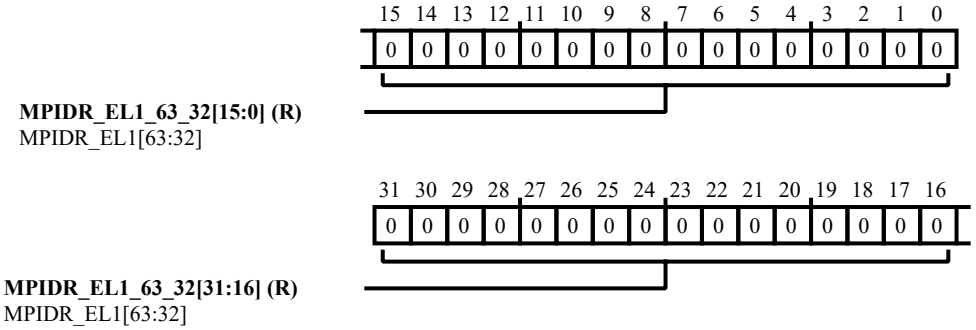


Figure 48-90: A55_CLUSTER_TRCDEVAFF1 Register Diagram

Table 48-88: A55_CLUSTER_TRCDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_63_32	MPIDR_EL1[63:32]. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented exception level.

Trace Device Architecture Register

Identifies the trace unit as an ETMv4 component. This enables debuggers to make some use of a component that conforms to the architecture definition but whose part number is not necessarily recognized.

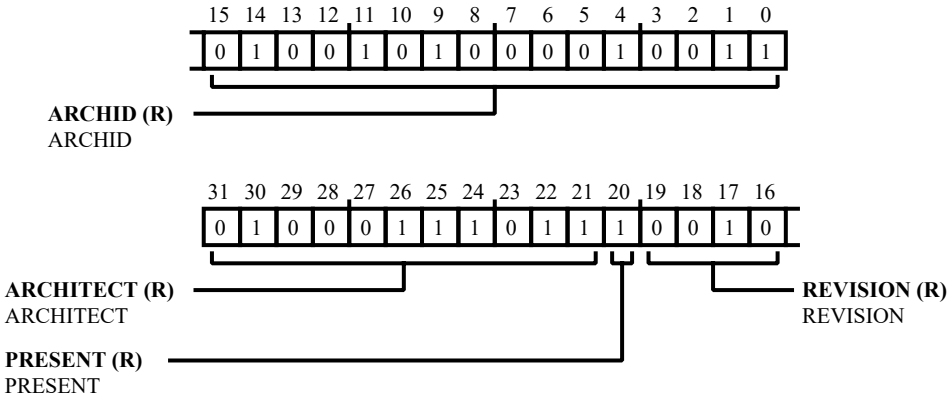


Figure 48-91: A55_CLUSTER_TRCDEVARCH Register Diagram

Table 48-89: A55_CLUSTER_TRCDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For trace, this is ARM Limited.Bits [31:28] are the JEP 106 continuation code, 0x4.Bits [27:21] are the JEP 106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present.This field is RAO.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision.For trace, the revision defined by ETMv4 is 0x0.All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be a v8-A debug component. For architectures defined by ARM this is further subdivided.For trace, bits [15:12] are the architecture version, 0x4; bits [11:0] are the architecture part number, 0xA13.This corresponds to trace architecture version ETMv4.

Trace Device ID Register

Returns an IMPLEMENTATION DEFINED CoreSight component capabilities field.

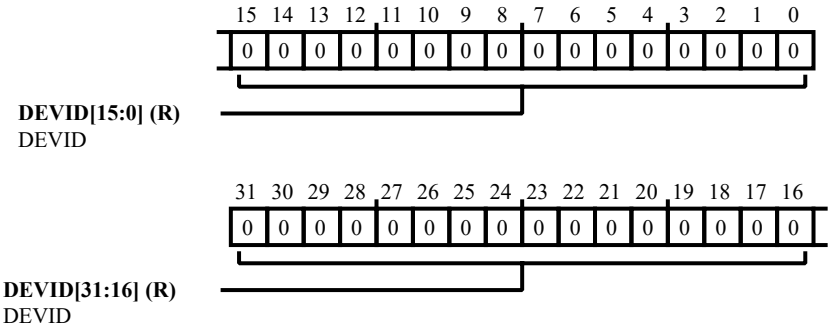


Figure 48-92: A55_CLUSTER_TRCDEVID Register Diagram

Table 48-90: A55_CLUSTER_TRCDEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVID	DEVID. Indicates the capabilities of the trace unit. The implemented width of this field and its bit assignments are IMPLEMENTATION DEFINED. Unimplemented bits are RAZ/WI.If a component is configurable then ARM recommends that this field can also indicate which configuration options are implemented that differ from the standard configuration.

Trace Device Type Register

Indicates what type of CoreSight device the trace unit is.

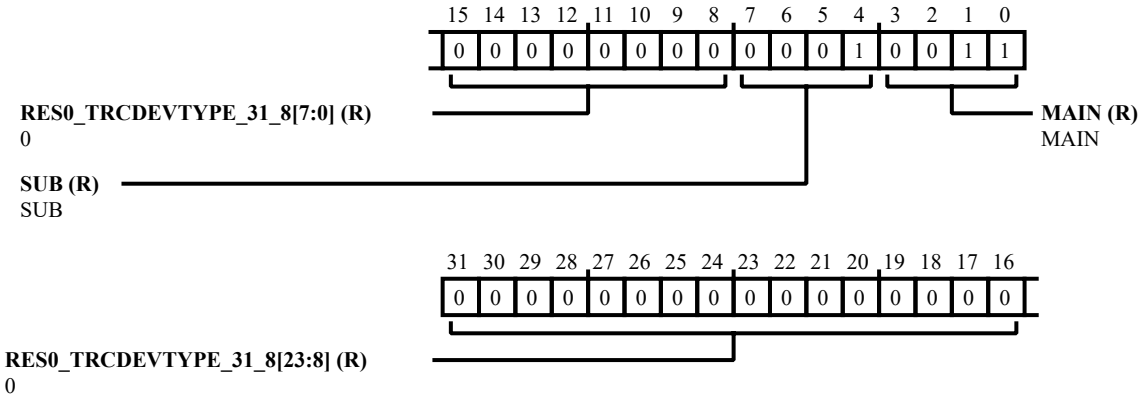


Figure 48-93: A55_CLUSTER_TRCDEVTYPE Register Diagram

Table 48-91: A55_CLUSTER_TRCDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCDEV-TYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Returns 0x1, to indicate that the ETM generates processor trace.All other values are reserved.
3:0 (R/NW)	MAIN	MAIN. Returns 0x3, to indicate that the ETM is a trace source.All other values are reserved.

Trace Software Lock Access Register

Controls whether the Software Lock is locked. When the Software Lock is locked, the trace unit ignores write accesses from the memory-mapped interface to all ETM registers other than the TRCLAR, and does not change the TRCPDSR.STICKYPD bit if a read access occurs to the TRCPDSR on the memory-mapped interface.

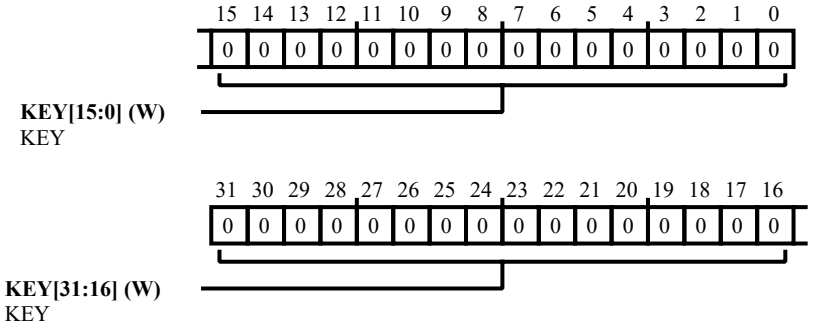


Figure 48-94: A55_CLUSTER_TRCLAR Register Diagram

Table 48-92: A55_CLUSTER_TRCLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Writing the key value 0xC5ACCE55 to this field clears the lock, enabling write accesses to this component's registers through a memory-mapped interface.Writing any other value to this register sets the lock, disabling write accesses to this component's registers through a memory mapped interface.Software can use the Software Lock to prevent accidental modification of the trace unit registers by software being debugged. For example, software that accidentally initializes an incorrect region of memory might disable the trace unit and make it impossible to trace the software.

Trace Software Lock Status Register

Trace Software Lock Status Register.

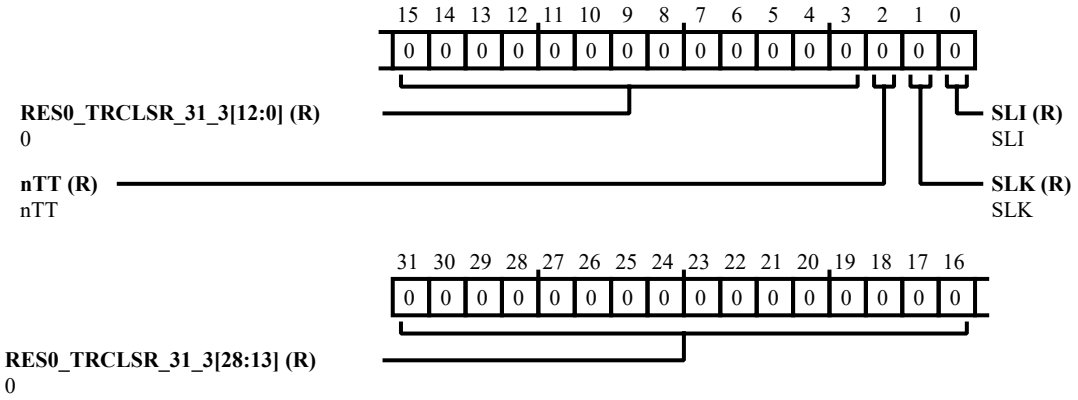


Figure 48-95: A55_CLUSTER_TRCLSR Register Diagram

Table 48-93: A55_CLUSTER_TRCLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_TRCLSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software lock status for this component. Possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.
0 (R/NW)	SLI	SLI. Software lock implemented. RAO.	

Trace PowerDown Control Register

Requests the system to provide power to the trace unit.

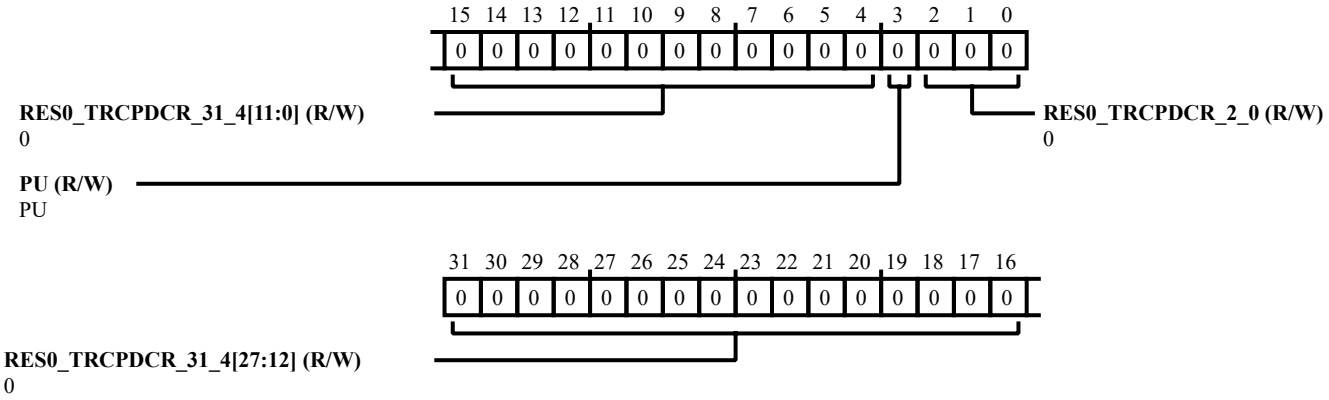


Figure 48-96: A55_CLUSTER_TRCPDCR Register Diagram

Table 48-94: A55_CLUSTER_TRCPDCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RES0_TRCPDCR_31_4	0. Reserved, RES0.	
3 (R/W)	PU	PU. Powerup request bit: 0 The system can remove power from the trace unit. The TRCPDSR indicates if the trace unit is powered down. 1 The system must provide power to the trace unit. Typically, a trace unit drives a signal representing the value of this bit to a power controller to request that the trace unit core power domain is powered up. However, if the trace unit and the processor are in the same power domain then the implementation might combine the PU status with a signal from the processor.	
		0	The system can remove power from the trace unit. The TRCPDSR indicates if the trace unit is powered down.
		1	The system must provide power to the trace unit.
2:0 (R/W)	RES0_TRCPDCR_2_0	0. Reserved, RES0.	

Trace Peripheral ID 0 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

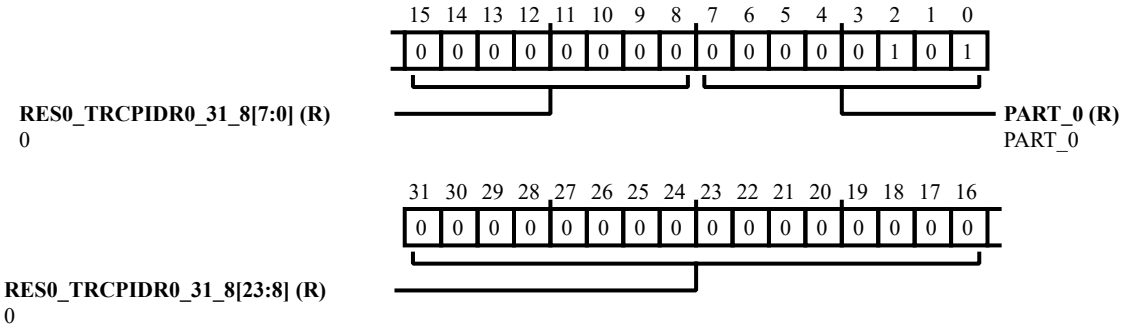


Figure 48-97: A55_CLUSTER_TRCPIDR0 Register Diagram

Table 48-95: A55_CLUSTER_TRCPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, bits[7:0].

Trace Peripheral ID 1 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

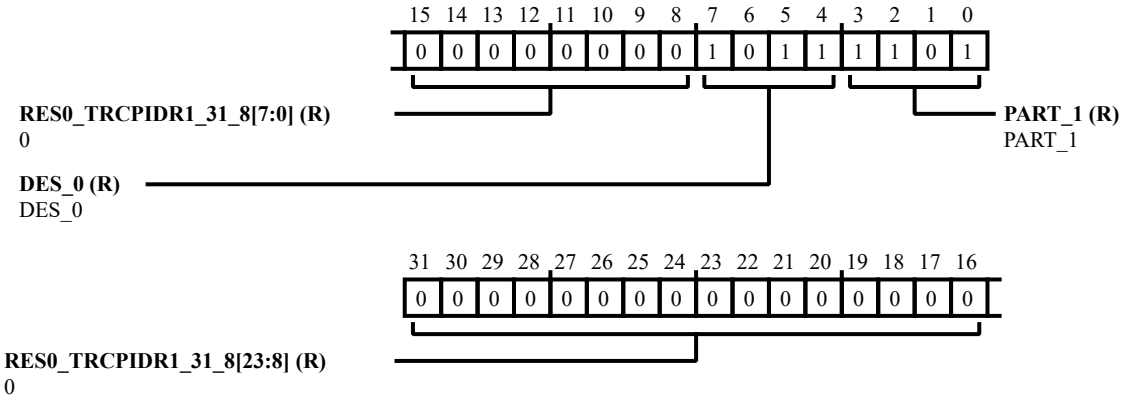


Figure 48-98: A55_CLUSTER_TRCPIDR1 Register Diagram

Table 48-96: A55_CLUSTER_TRCPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, bits[3:0] of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, bits[11:8].

Trace Peripheral ID 2 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

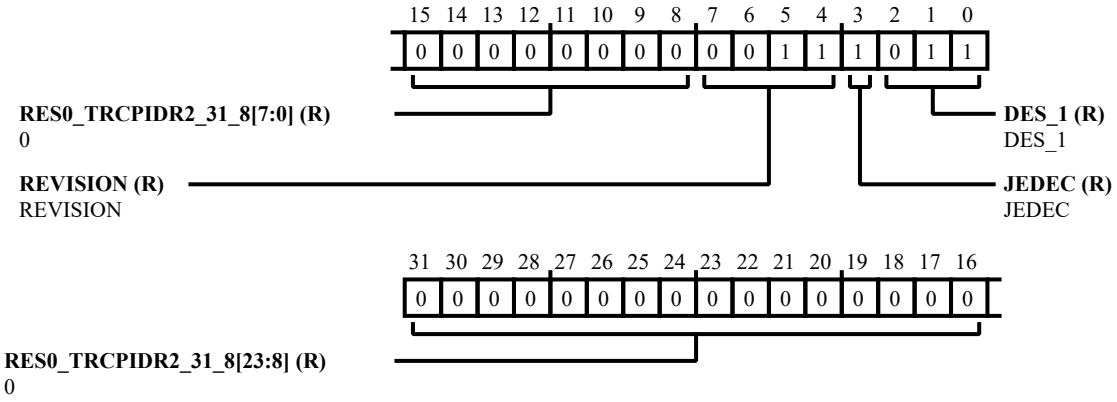


Figure 48-99: A55_CLUSTER_TRCPIDR2 Register Diagram

Table 48-97: A55_CLUSTER_TRCPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. The IMPLEMENTATION DEFINED revision number for the ETM implementation. See also TRCIDR1.REVISION.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

Trace Peripheral ID 3 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

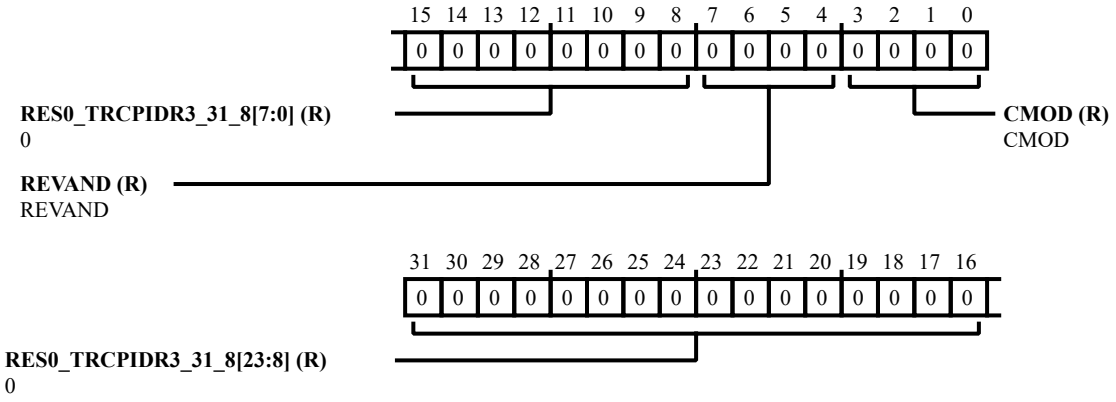


Figure 48-100: A55_CLUSTER_TRCPIDR3 Register Diagram

Table 48-98: A55_CLUSTER_TRCPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. The IMPLEMENTATION DEFINED manufacturing revision number for the implementation. After silicon is available, if metal fixes are necessary, the manufacturer can alter the top metal layer so that this field can indicate any post-fab silicon changes.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

Trace Peripheral ID 4 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

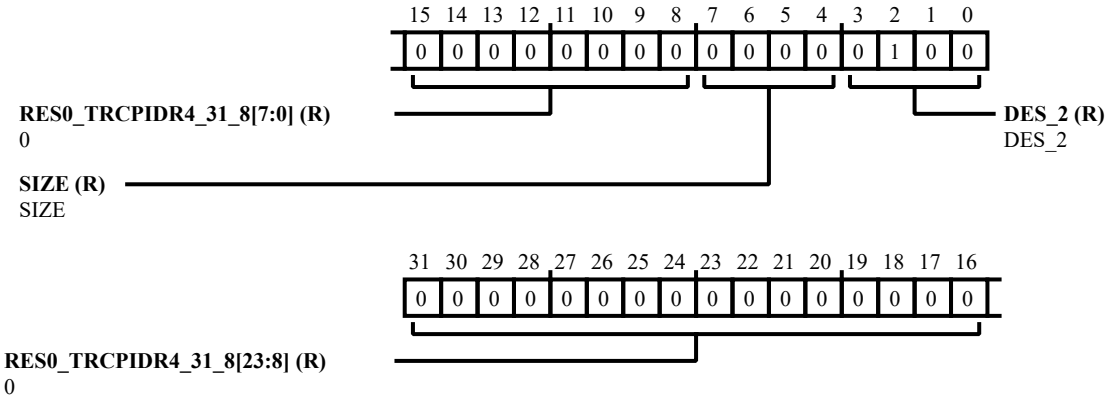


Figure 48-101: A55_CLUSTER_TRCPIDR4 Register Diagram

Table 48-99: A55_CLUSTER_TRCPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RES0. This indicates that the ETM memory map occupies 4KB.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code. For ARM Limited, this field is 0b0100.

Performance Monitors Authentication Status Register

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for Performance Monitors.

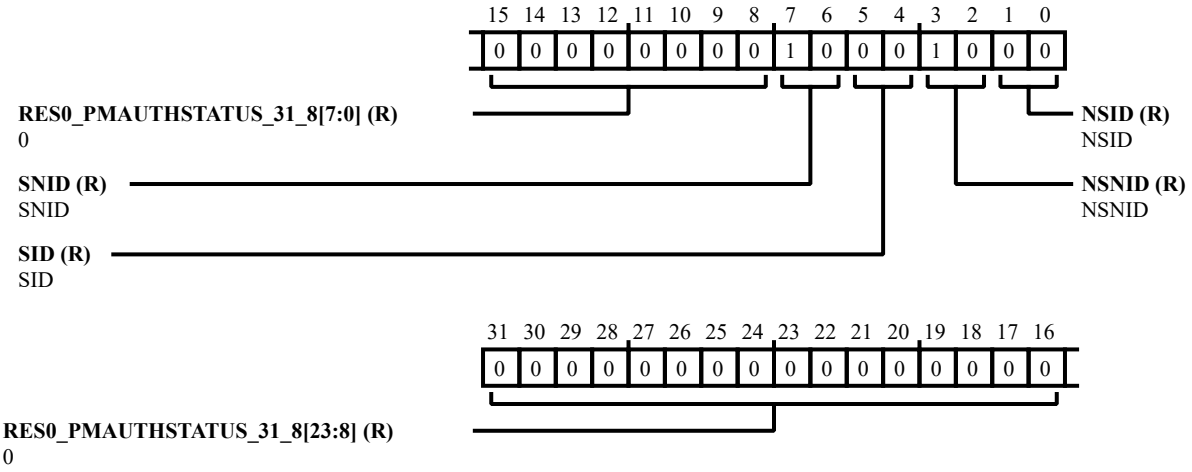


Figure 48-102: A55_CLUSTER_PMAUTHSTATUS Register Diagram

Table 48-100: A55_CLUSTER_PMAUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_PMAUTHSTATUS_31_8	0. Reserved, RES0.	
7:6 (R/NW)	SNID	SNID. Holds the same value as DBGAUTHSTATUS_EL1.SNID.	
5:4 (R/NW)	SID	SID. Secure invasive debug. Possible values of this field are: 00 Not implemented. All other values are reserved.	
		0	Not implemented.
3:2 (R/NW)	NSNID	NSNID. Holds the same value as DBGAUTHSTATUS_EL1.NSNID.	
1:0 (R/NW)	NSID	NSID. Non-secure invasive debug. Possible values of this field are: 00 Not implemented. All other values are reserved.	
		0	Not implemented.

Performance Monitors Component Identification 0 Register

Provides information to identify a Performance Monitor component.

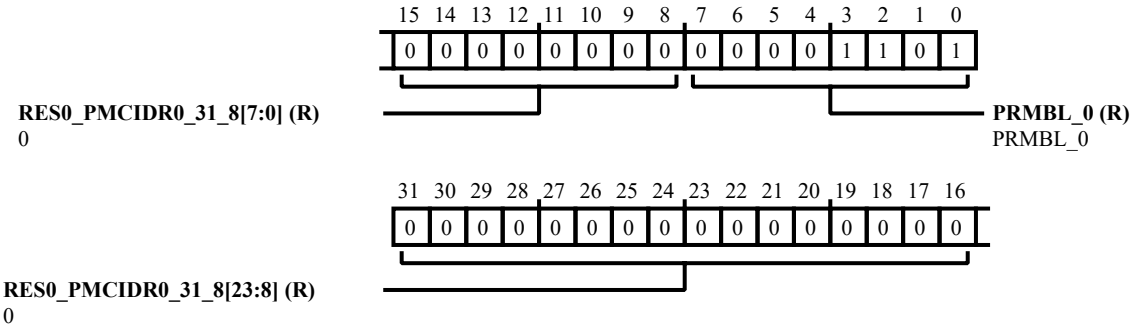


Figure 48-103: A55_CLUSTER_PMCIDR0 Register Diagram

Table 48-101: A55_CLUSTER_PMCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

Performance Monitors Component Identification 1 Register

Provides information to identify a Performance Monitor component.

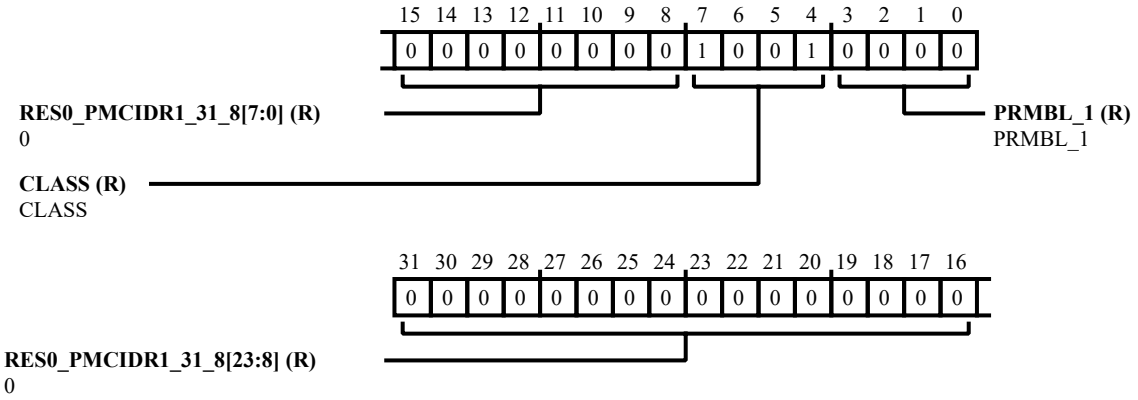


Figure 48-104: A55_CLUSTER_PMCIDR1 Register Diagram

Table 48-102: A55_CLUSTER_PMCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, debug component.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. RAZ.

Performance Monitors Component Identification 2 Register

Provides information to identify a Performance Monitor component.

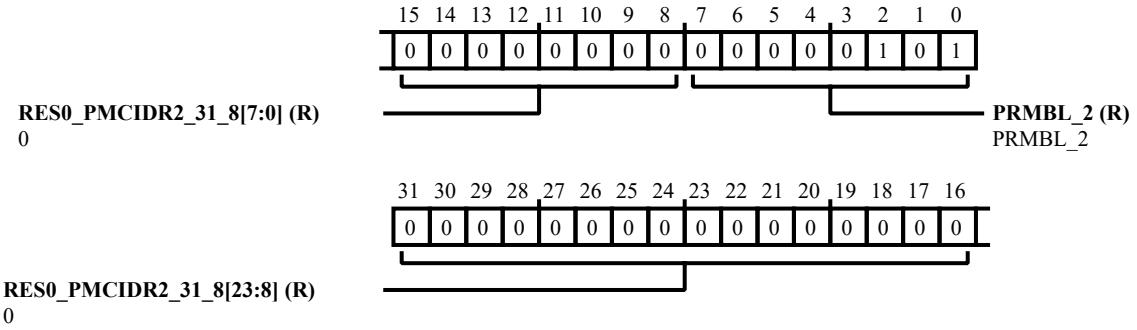


Figure 48-105: A55_CLUSTER_PMCIDR2 Register Diagram

Table 48-103: A55_CLUSTER_PMCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

Performance Monitors Component Identification 3 Register

Provides information to identify a Performance Monitor component.

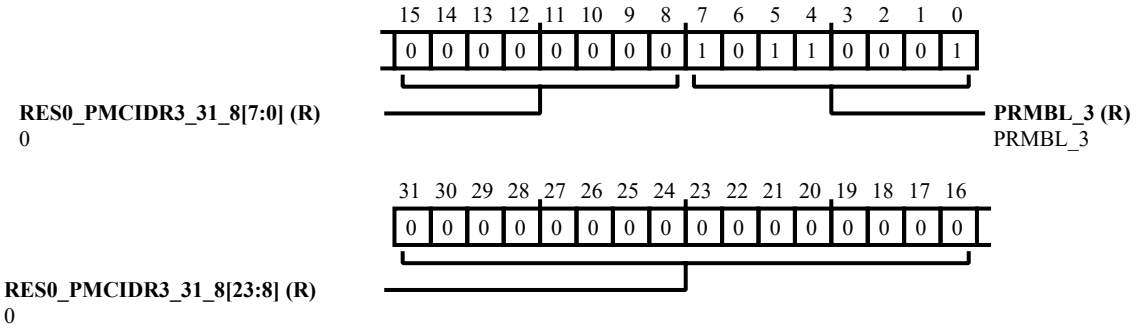


Figure 48-106: A55_CLUSTER_PMCIDR3 Register Diagram

Table 48-104: A55_CLUSTER_PMCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

Performance Monitors Device Affinity 0 Register

Copy of the low half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

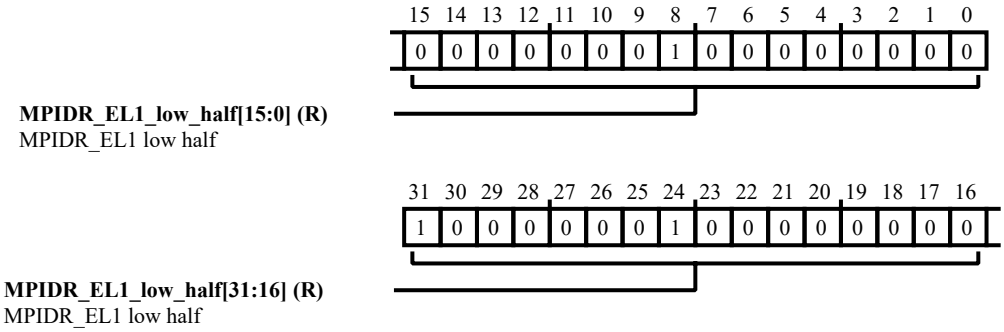


Figure 48-107: A55_CLUSTER_PMDEVAFF0 Register Diagram

Table 48-105: A55_CLUSTER_PMDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_LOW_HALF	MPIDR_EL1 low half. MPIDR_EL1 low half. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented Exception level.

Performance Monitors Device Affinity 1 Register

Copy of the high half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

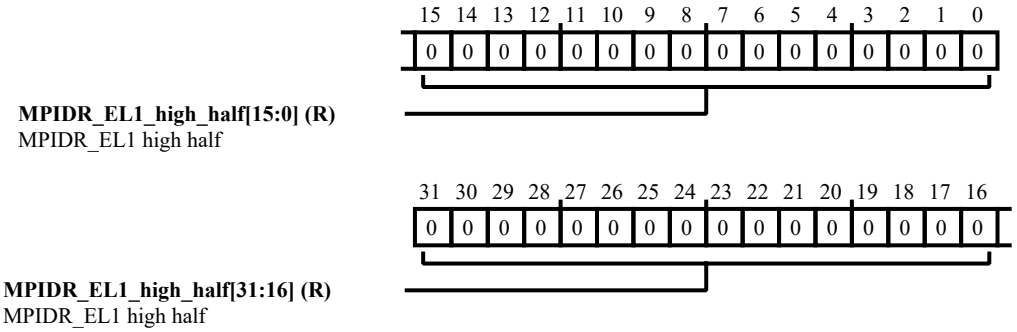


Figure 48-108: A55_CLUSTER_PMDEVAFF1 Register Diagram

Table 48-106: A55_CLUSTER_PMDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_HIGH_HALF	MPIDR_EL1 high half. MPIDR_EL1 high half. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented Exception level.

Performance Monitors Device Architecture Register

Identifies the programmers' model architecture of the Performance Monitor component.

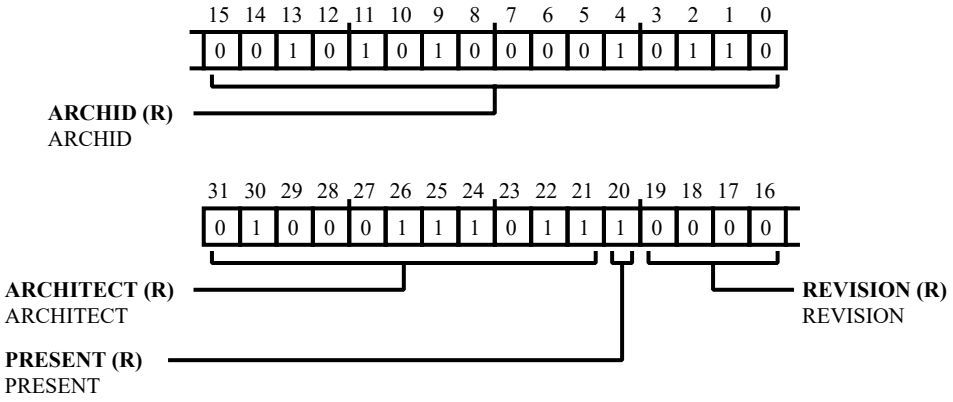


Figure 48-109: A55_CLUSTER_PMDEVARCH Register Diagram

Table 48-107: A55_CLUSTER_PMDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For Performance Monitors, this is ARM Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present. This field is 1 in ARMv8.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision. For Performance Monitors, the revision defined by ARMv8 is 0x0. All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided. For Performance Monitors: Bits [15:12] are the architecture version, 0x2. Bits [11:0] are the architecture part number, 0xA16. This corresponds to Performance Monitors architecture version PMUv3. The PMUv3 memory-mapped programmers' model can be used by devices other than ARMv8 processors. Software must determine whether the PMU is attached to an ARMv8 processor by using the PMDEVAFF0 and PMDEVAFF1 registers to discover the affinity of the PMU to any ARMv8 processors.

Performance Monitors Device Type Register

Indicates to a debugger that this component is part of a PEs performance monitor interface.

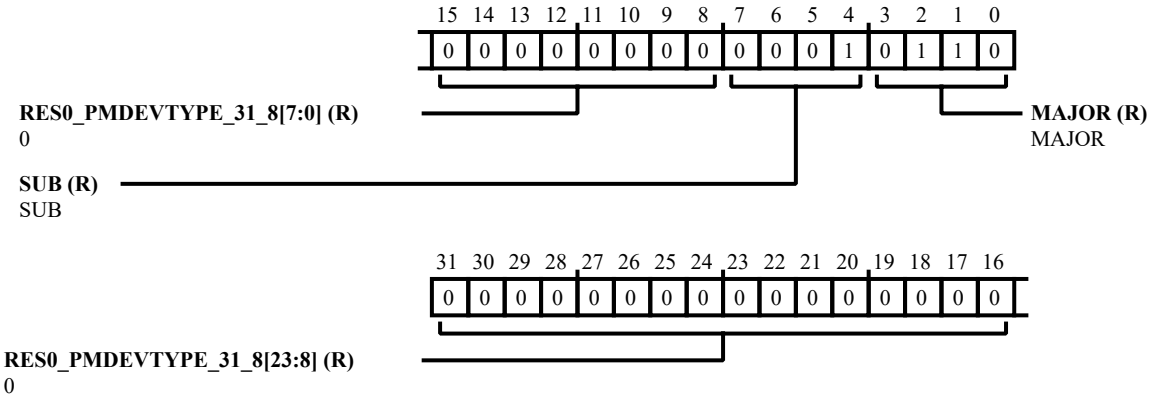


Figure 48-110: A55_CLUSTER_PMDEVTYPE Register Diagram

Table 48-108: A55_CLUSTER_PMDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMDEVTYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Subtype. Must read as 0x1 to indicate this is a component within a PE.
3:0 (R/NW)	MAJOR	MAJOR. Major type. Must read as 0x6 to indicate this is a performance monitor component.

Performance Monitors Lock Access Register

Allows or disallows access to the Performance Monitors registers through a memory-mapped interface.

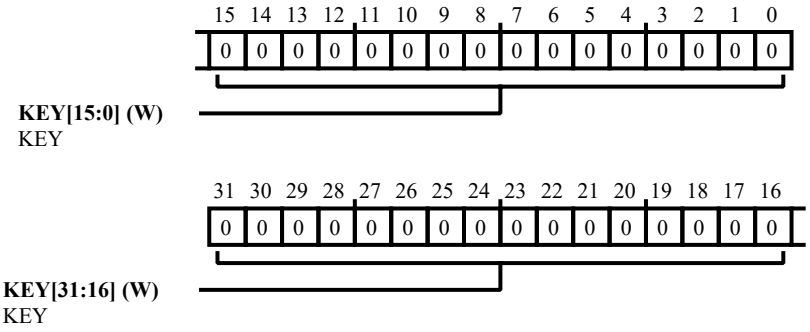


Figure 48-111: A55_CLUSTER_PMLAR Register Diagram

Table 48-109: A55_CLUSTER_PMLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

Performance Monitors Lock Status Register

Indicates the current status of the software lock for Performance Monitors registers.

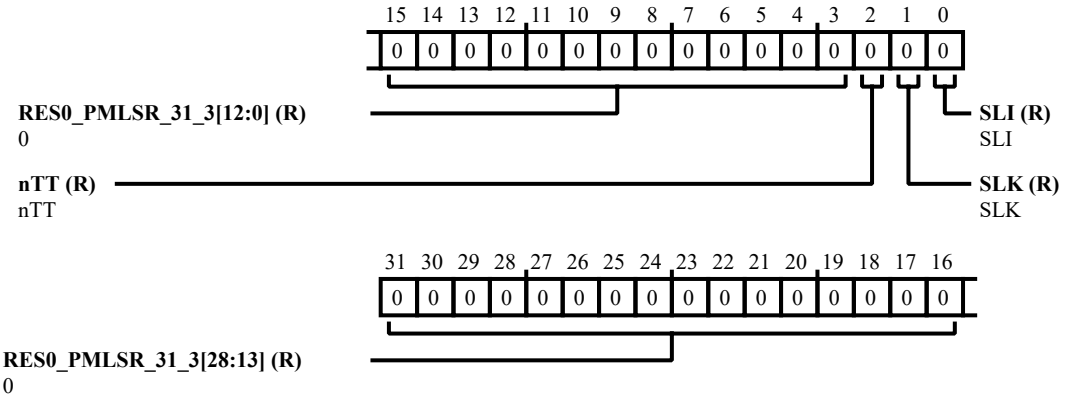


Figure 48-112: A55_CLUSTER_PMLSR Register Diagram

Table 48-110: A55_CLUSTER_PMLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_PMLSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0. For memory-mapped accesses when the software lock is implemented, possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

Table 48-110: A55_CLUSTER_PMLSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SLI	SLI. Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are: 0 Software Lock not implemented or not memory-mapped access. 1 Software Lock implemented and memory-mapped access.	
		0	Software Lock not implemented or not memory-mapped access.
		1	Software Lock implemented and memory-mapped access.

Performance Monitors Peripheral Identification 0 Register

Provides information to identify a Performance Monitor component.

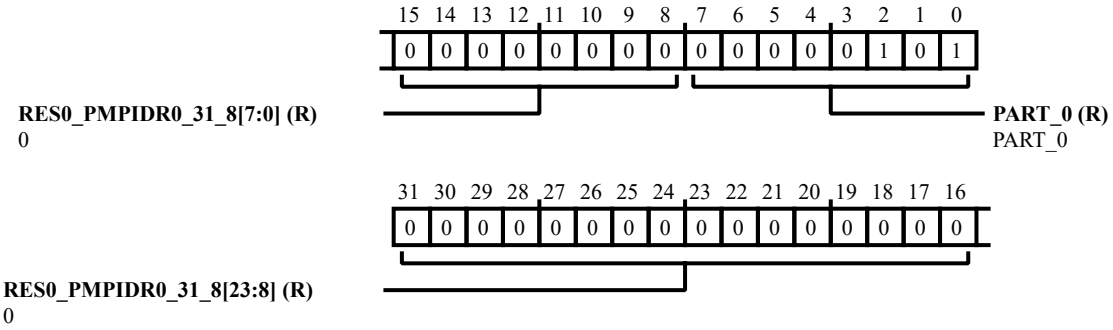


Figure 48-113: A55_CLUSTER_PMPIDR0 Register Diagram

Table 48-111: A55_CLUSTER_PMPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, least significant byte.

Performance Monitors Peripheral Identification 1 Register

Provides information to identify a Performance Monitor component.

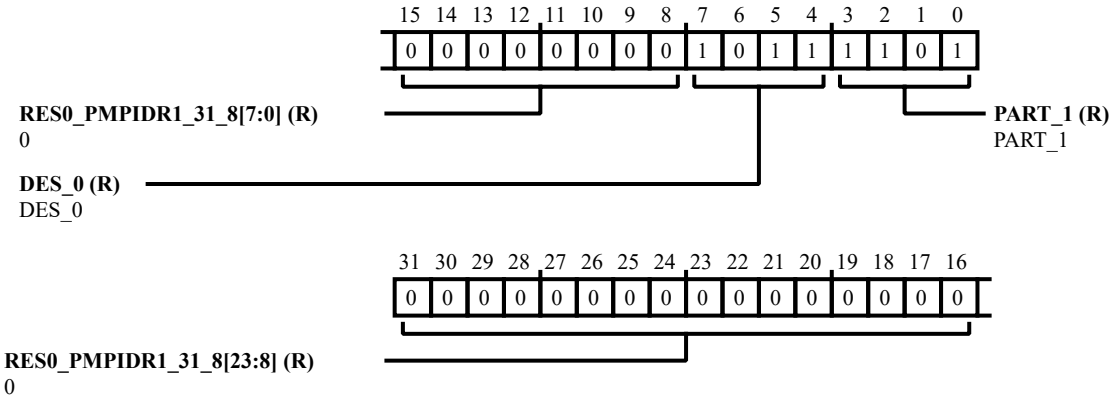


Figure 48-114: A55_CLUSTER_PMPIDR1 Register Diagram

Table 48-112: A55_CLUSTER_PMPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, most significant nibble.

Performance Monitors Peripheral Identification 2 Register

Provides information to identify a Performance Monitor component.

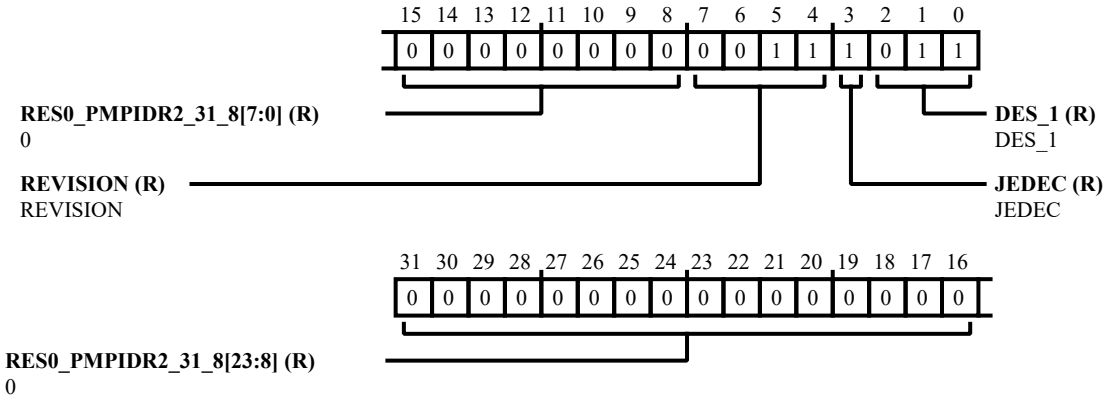


Figure 48-115: A55_CLUSTER_PMPIDR2 Register Diagram

Table 48-113: A55_CLUSTER_PMPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. Part major revision. Parts can also use this field to extend Part number to 16-bits.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

Performance Monitors Peripheral Identification 3 Register

Provides information to identify a Performance Monitor component.

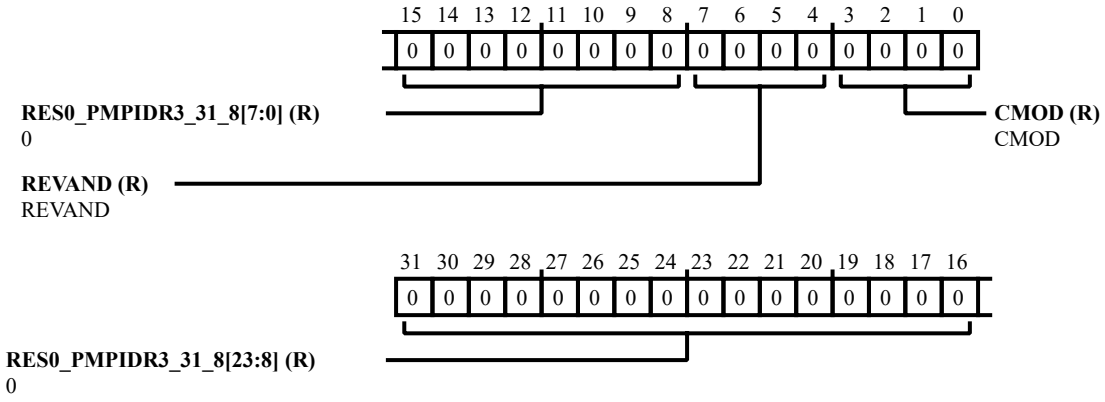


Figure 48-116: A55_CLUSTER_PMPIDR3 Register Diagram

Table 48-114: A55_CLUSTER_PMPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. Part minor revision. Parts using PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

Performance Monitors Peripheral Identification 4 Register

Provides information to identify a Performance Monitor component.

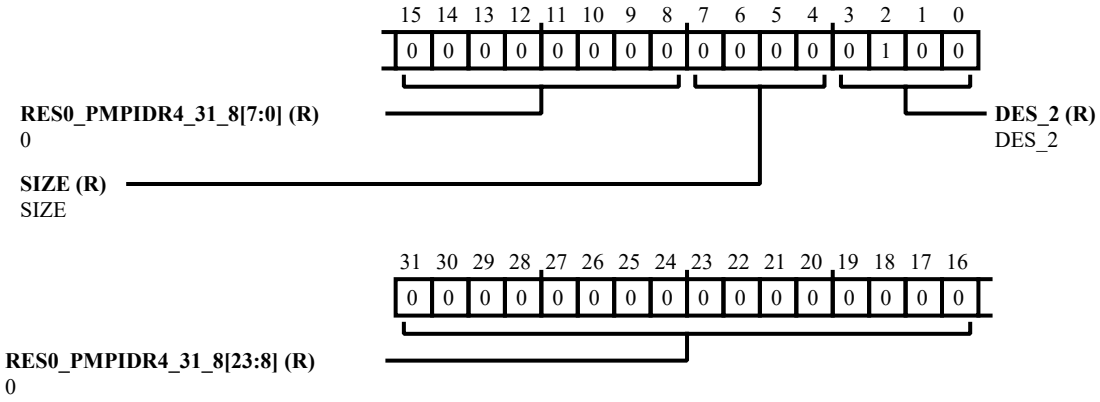


Figure 48-117: A55_CLUSTER_PMPIDR4 Register Diagram

Table 48-115: A55_CLUSTER_PMPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

Performance Monitors Peripheral Identification 5 Register

Provides information to identify a Performance Monitor component.

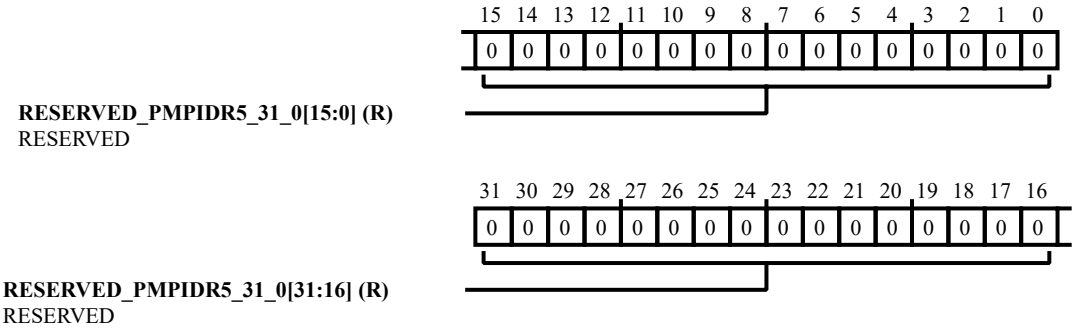


Figure 48-118: A55_CLUSTER_PMPIDR5 Register Diagram

Table 48-116: A55_CLUSTER_PMPIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_PMPIDR5_31_0	RESERVED. Reserved RES0.

Performance Monitors Peripheral Identification 6 Register

Provides information to identify a Performance Monitor component.

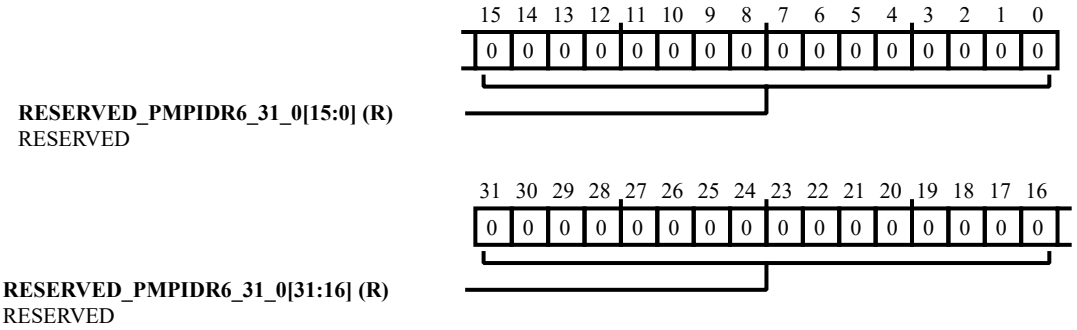


Figure 48-119: A55_CLUSTER_PMPIDR6 Register Diagram

Table 48-117: A55_CLUSTER_PMPIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_PMPIDR6_31_0	RESERVED. Reserved RES0.

Performance Monitors Peripheral Identification 7 Register

Provides information to identify a Performance Monitor component.

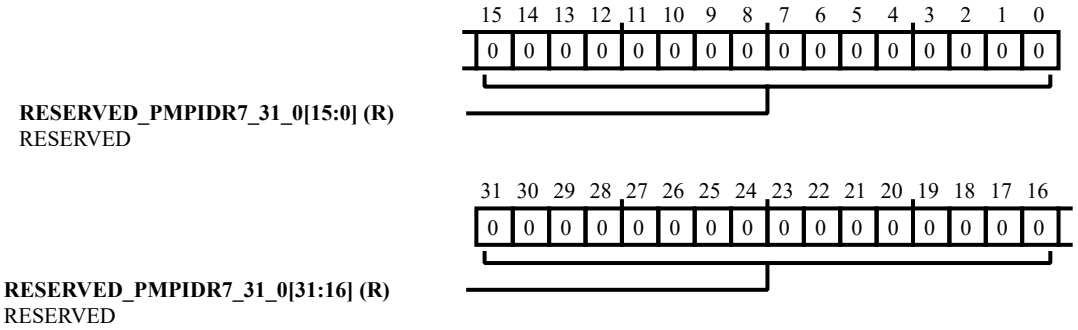


Figure 48-120: A55_CLUSTER_PMPIDR7 Register Diagram

Table 48-118: A55_CLUSTER_PMPIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_PMPIDR7_31_0	RESERVED. Reserved RES0.

CTI External Multiplexer Control Register

Can be used to provide IMPLEMENTATION DEFINED controls for the CTI. For example, the register might be used to control multiplexors for additional IMPLEMENTATION DEFINED triggers. The IMPLEMENTATION DEFINED controls provided by this register might modify the architecturally defined behavior of the CTI. The architecturally-defined triggers must not be multiplexed.

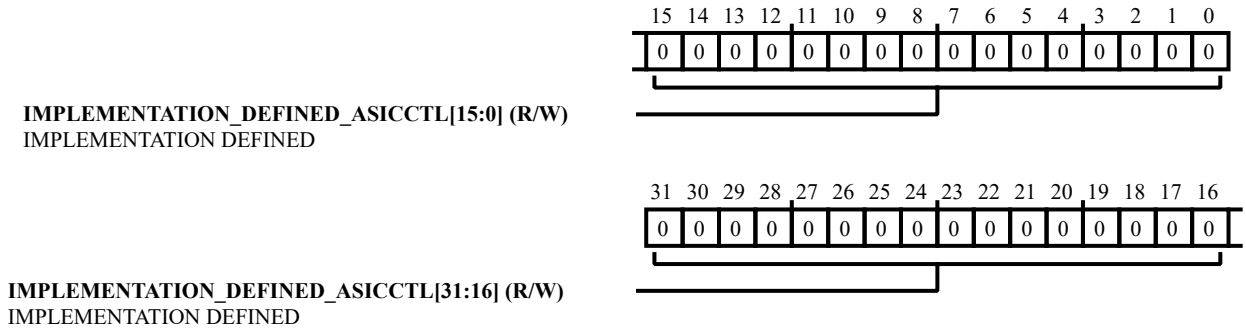


Figure 48-121: A55_CLUSTER_ASICCTL Register Diagram

Table 48-119: A55_CLUSTER_ASICCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	IMPLEMENTATION_DEFINED_ASICCTL	IMPLEMENTATION DEFINED. IMPLEMENTATION DEFINED.

CTI Application Trigger Clear Register

Clears bits of the Application Trigger register.

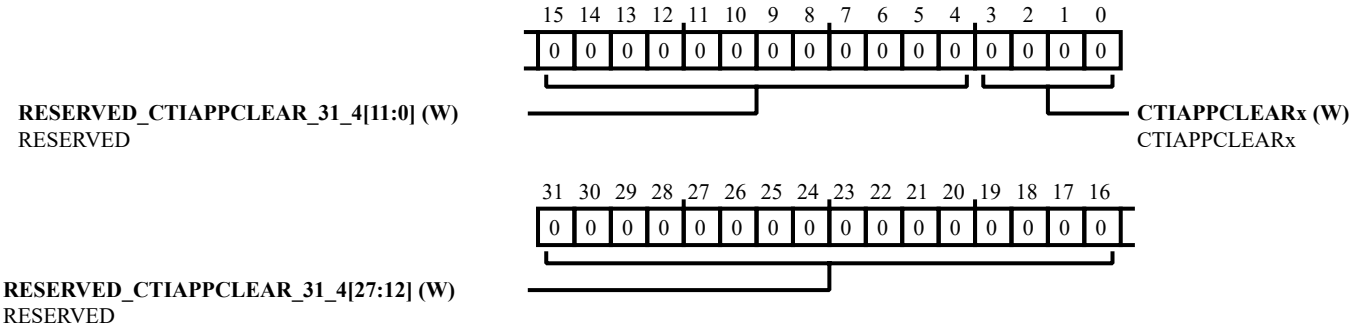


Figure 48-122: A55_CLUSTER_CTIAPPCLEAR Register Diagram

Table 48-120: A55_CLUSTER_CTIAPPCLEAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (RX/W)	RESERVED_CTIAPP-CLEAR_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (RX/W)	CTIAPPCLEARX	CTIAPPCLEARx. Application trigger <x> disable.	
		0	No effect.
		1	Clear corresponding bit in CTIAPPTRIG to 0 and clear the corresponding channel event.

CTI Application Pulse Register

Causes event pulses to be generated on ECT channels.

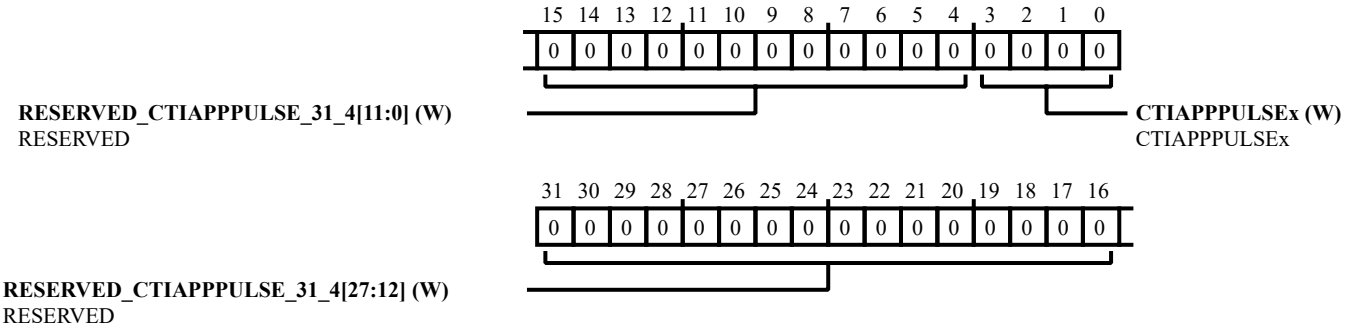


Figure 48-123: A55_CLUSTER_CTIAPPPULSE Register Diagram

Table 48-121: A55_CLUSTER_CTIAPPPULSE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (RX/W)	RESERVED_CTIAPP-PULSE_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (RX/W)	CTIAPPPULSEX	CTIAPPPULSEx. Generate event pulse on ECT channel <x>.	
		0	No effect.
		1	Channel <x> event pulse generated for one clock period.

CTI Application Trigger Set Register

Sets bits of the Application Trigger register.

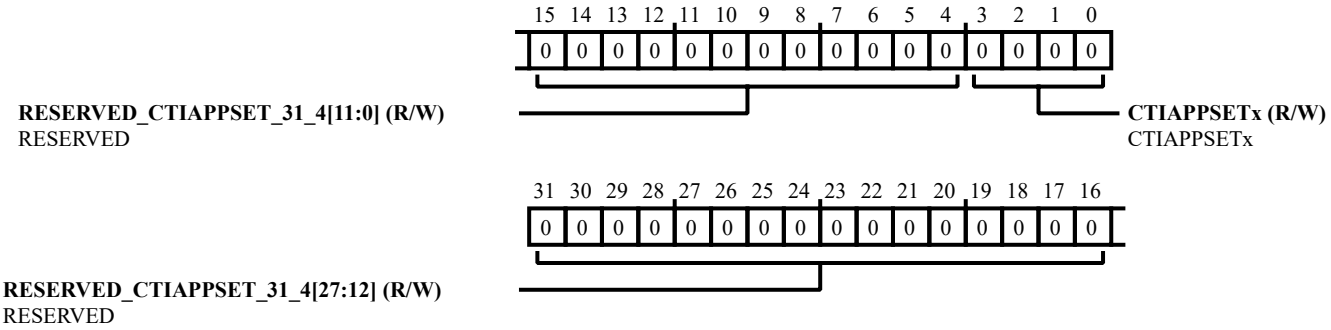


Figure 48-124: A55_CLUSTER_CTIAPPSET Register Diagram

Table 48-122: A55_CLUSTER_CTIAPPSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIAP-PSET_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	CTIAPPSETX	CTIAPPSETx. Application trigger <x> enable.	
		0	Reading this means the application trigger is inactive. Writing this has no effect.
		1	Reading this means the application trigger is active. Writing this sets the corresponding bit in CTIAPP-TRIG to 1 and generates a channel event.

CTI Authentication Status Register

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for CTI.

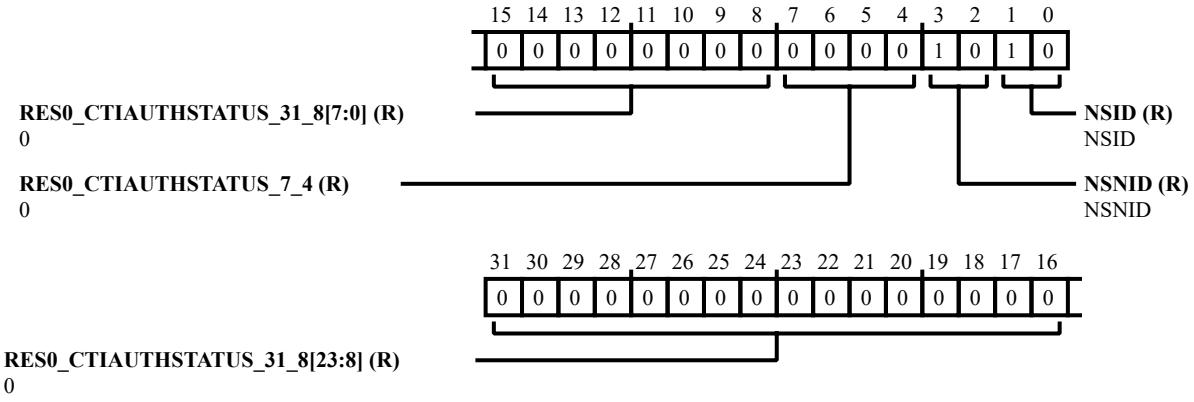


Figure 48-125: A55_CLUSTER_CTIAUTHSTATUS Register Diagram

Table 48-123: A55_CLUSTER_CTIAUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIAUTHSTATUS_31_8	0. Reserved, RES0.
7:4 (R/NW)	RES0_CTIAUTHSTATUS_7_4	0. Reserved, RAZ.
3:2 (R/NW)	NSNID	NSNID. If EL3 is not implemented and the implemented Security state is Secure state, holds the same value as DBGAUTHSTATUS_EL1.SNID. Otherwise, holds the same value as DBGAUTHSTATUS_EL1.NSNID.
1:0 (R/NW)	NSID	NSID. If EL3 is not implemented and the implemented Security state is Secure state, holds the same value as DBGAUTHSTATUS_EL1.SID. Otherwise, holds the same value as DBGAUTHSTATUS_EL1.NSID.

CTI Channel In Status Register

Provides the raw status of the ECT channel inputs to the CTI.

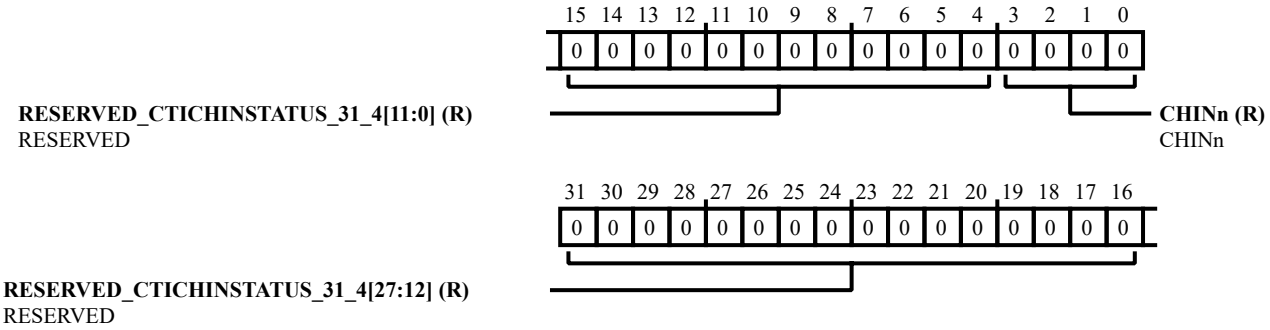


Figure 48-126: A55_CLUSTER_CTICHINSTATUS Register Diagram

Table 48-124: A55_CLUSTER_CTICHINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RESERVED_CTICHIN-STATUS_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	CHINN	CHINn. Provides the raw status of the ECT channel inputs to the CTI.	
		0	Input channel <n> is inactive.
		1	Input channel <n> is active.

CTI Channel Out Status Register

Provides the status of the ECT channel outputs from the CTI.

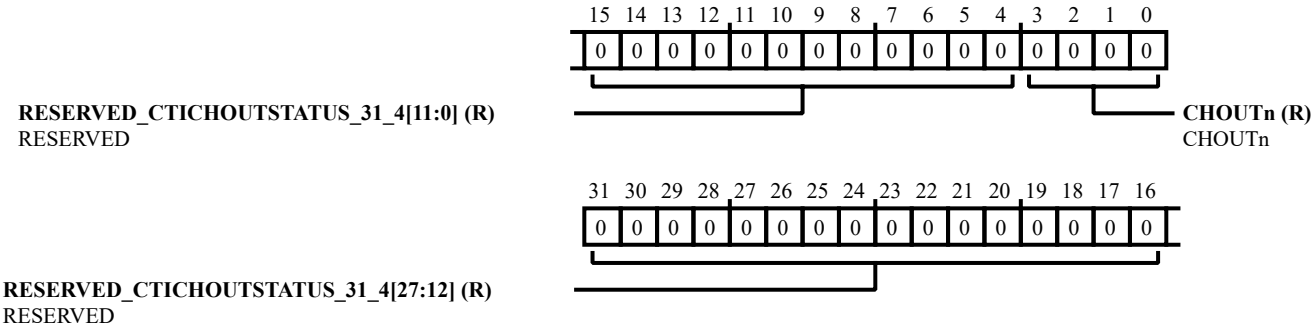


Figure 48-127: A55_CLUSTER_CTICHOUTSTATUS Register Diagram

Table 48-125: A55_CLUSTER_CTICHOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RESERVED_CTICHOUT-STATUS_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	CHOUTN	CHOUTn. Provides the status of the ECT channel outputs from the CTI.	
		0	Output channel <n> is inactive.
		1	Output channel <n> is active.

CTI Component Identification 0 Register

Provides information to identify a CTI component.

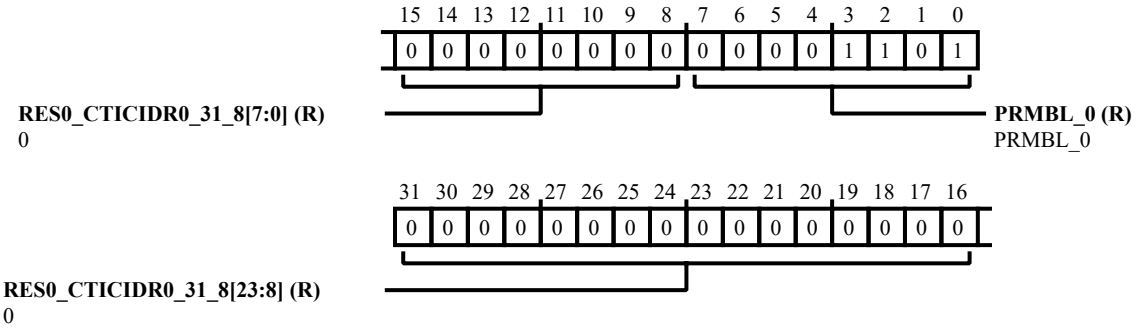


Figure 48-128: A55_CLUSTER_CTICIDR0 Register Diagram

Table 48-126: A55_CLUSTER_CTICIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

CTI Component Identification 1 Register

Provides information to identify a CTI component.

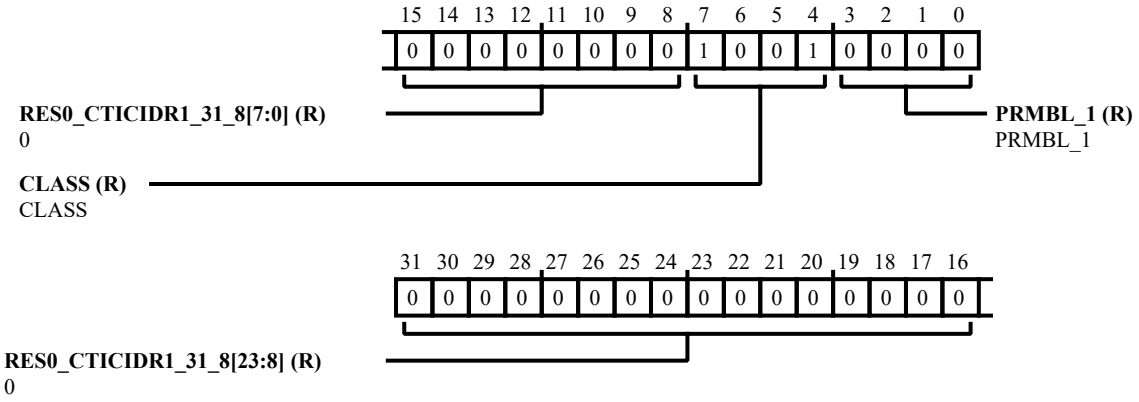


Figure 48-129: A55_CLUSTER_CTICIDR1 Register Diagram

Table 48-127: A55_CLUSTER_CTICIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, debug component.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. RAZ.

CTI Component Identification 2 Register

Provides information to identify a CTI component.

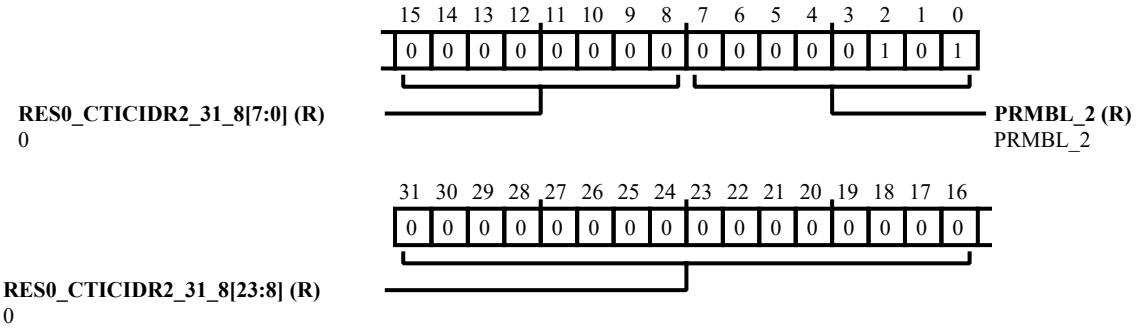


Figure 48-130: A55_CLUSTER_CTICIDR2 Register Diagram

Table 48-128: A55_CLUSTER_CTICIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

CTI Component Identification 3 Register

Provides information to identify a CTI component.

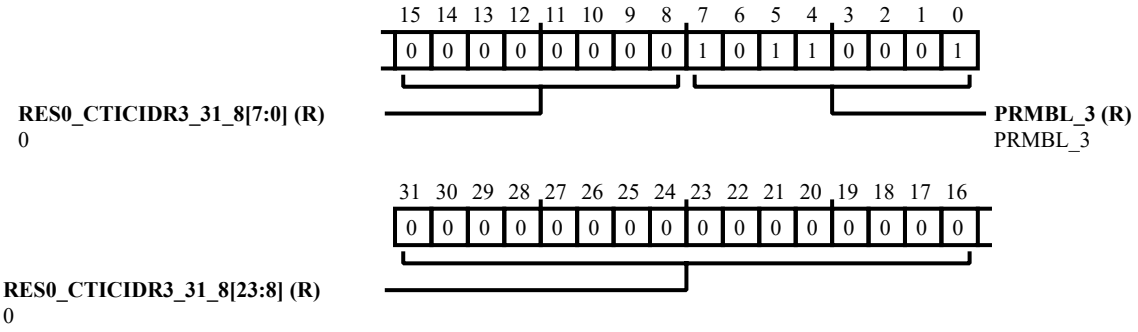


Figure 48-131: A55_CLUSTER_CTICIDR3 Register Diagram

Table 48-129: A55_CLUSTER_CTICIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTICIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

CTI Claim Tag Clear Register

Used by software to read the values of the CLAIM bits, and to clear these bits to 0.

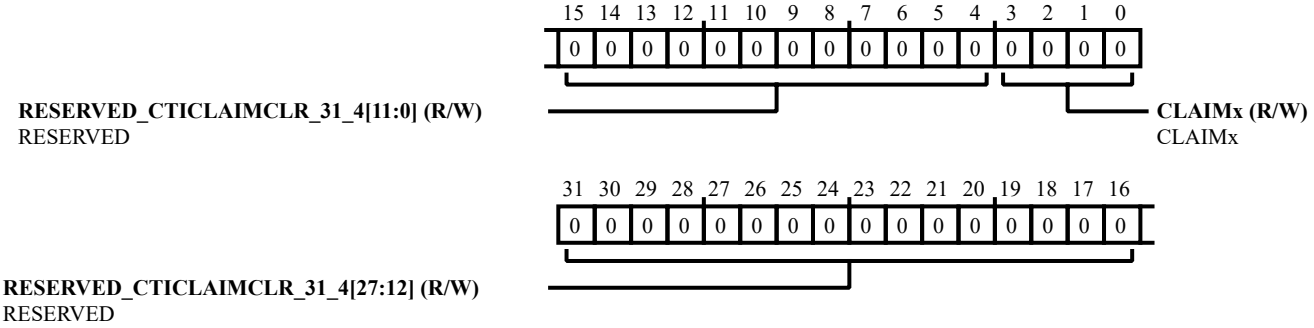


Figure 48-132: A55_CLUSTER_CTICLAIMCLR Register Diagram

Table 48-130: A55_CLUSTER_CTICLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTI-CLAIMCLR_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	CLAIMX	CLAIMx. Clear CLAIM tag.	
		0	No action.
		1	Indirectly clear CLAIM[x] to 0.

CTI Claim Tag Set Register

Used by software to set CLAIM bits to 1.

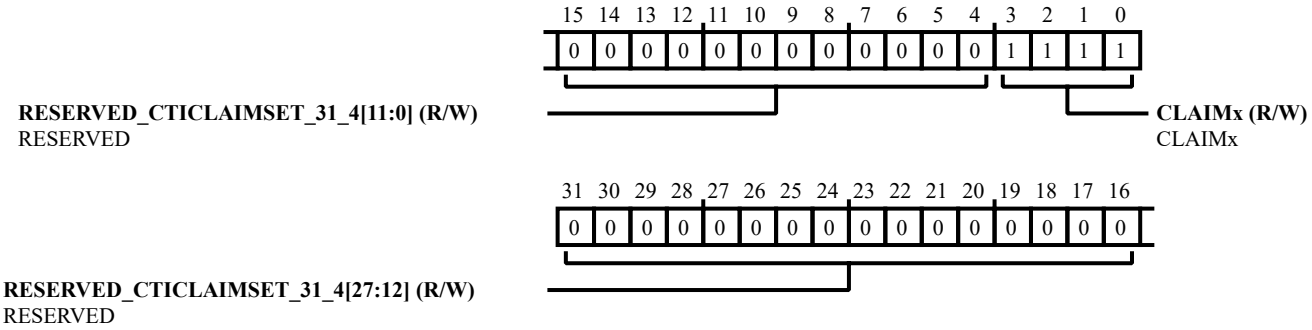


Figure 48-133: A55_CLUSTER_CTICLAIMSET Register Diagram

Table 48-131: A55_CLUSTER_CTICLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTICLAIM-SET_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	CLAIMX	CLAIMx. CLAIM tag set bit.	
		0	No action.
		1	Indirectly set CLAIM[x] tag to 1.

CTI Control Register

Controls whether the CTI is enabled.

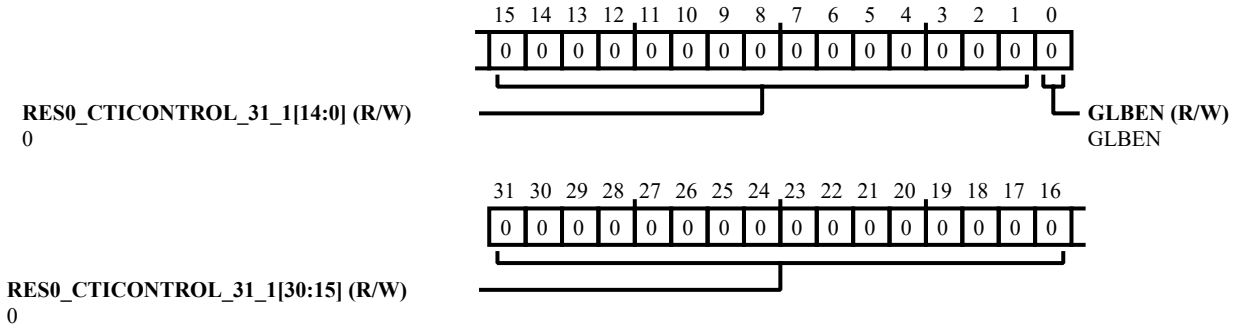


Figure 48-134: A55_CLUSTER_CTICONTROL Register Diagram

Table 48-132: A55_CLUSTER_CTICONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_CTICON-TROL_31_1	0. Reserved, RES0.	
0 (R/W)	GLBEN	GLBEN. Enables or disables the CTI mapping functions. Possible values of this field are: 0 CTI mapping functions disabled. 1 CTI mapping functions enabled. When the mapping functions are disabled, no new events are signaled on either output triggers or output channels. If a previously asserted output trigger has not been acknowledged, it remains asserted after the mapping functions are disabled. All output triggers are disabled by CTI reset. If the ECT supports multicycle channel events any existing output channel events will be terminated.	
		0	CTI mapping functions disabled.
		1	CTI mapping functions enabled.

CTI Device Affinity 0 Register

Copy of the low half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

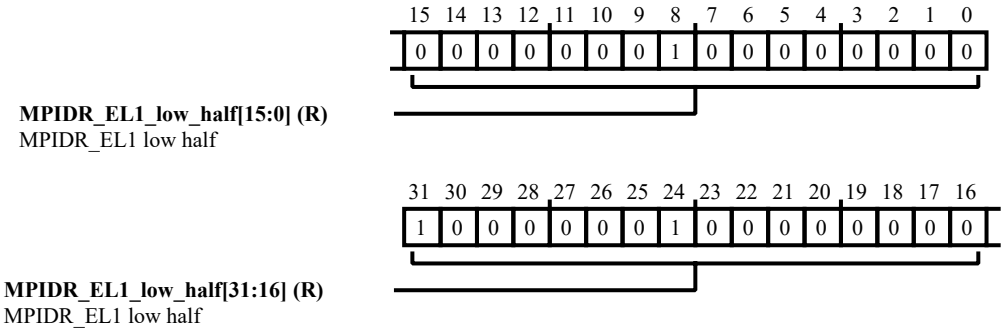


Figure 48-135: A55_CLUSTER_CTIDEVAFF0 Register Diagram

Table 48-133: A55_CLUSTER_CTIDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_LOW_HALF	MPIDR_EL1 low half. MPIDR_EL1 low half. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented Exception level.

CTI Device Affinity 1 Register

Copy of the high half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

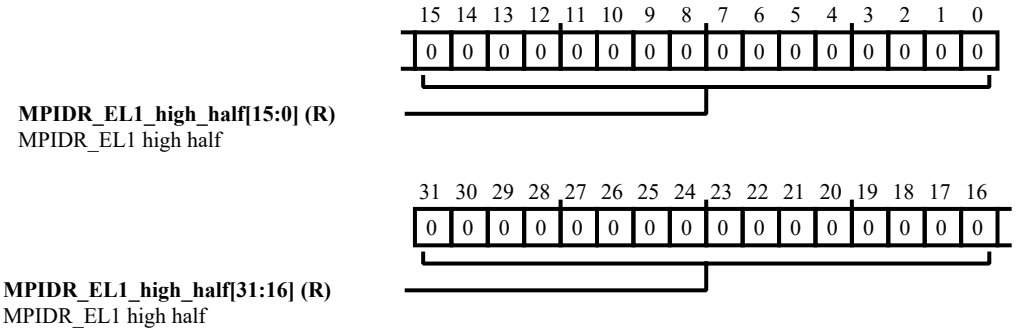


Figure 48-136: A55_CLUSTER_CTIDEVAFF1 Register Diagram

Table 48-134: A55_CLUSTER_CTIDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_HIGH_HALF	MPIDR_EL1 high half. MPIDR_EL1 high half. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented Exception level.

CTI Device Architecture Register

Identifies the programmers' model architecture of the CTI component.

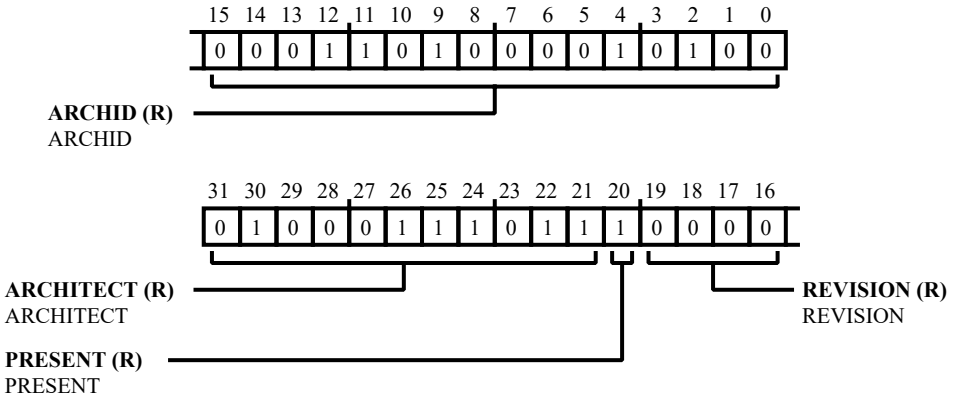


Figure 48-137: A55_CLUSTER_CTIDEVARCH Register Diagram

Table 48-135: A55_CLUSTER_CTIDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For CTI, this is ARM Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present. This field is 1 in ARMv8.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision. For CTI, the revision defined by ARMv8 is 0x0. All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided. For CTI: Bits [15:12] are the architecture version, 0x1. Bits [11:0] are the architecture part number, 0xA14. This corresponds to CTI architecture version CTIv2.

CTI Device ID 0 Register

Describes the CTI component to the debugger.

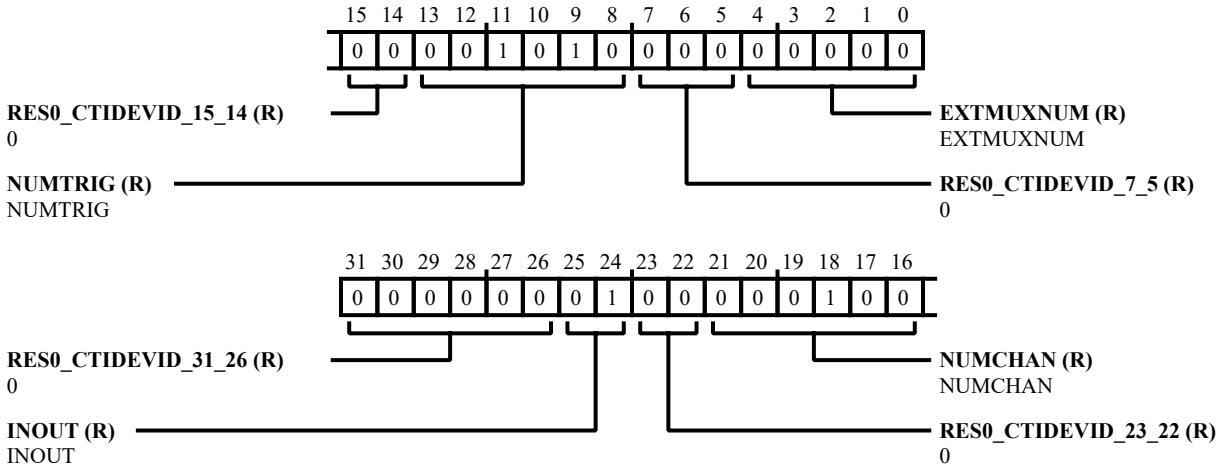


Figure 48-138: A55_CLUSTER_CTIDEVID Register Diagram

Table 48-136: A55_CLUSTER_CTIDEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:26 (R/NW)	RES0_CTIDEVID_31_26	0. Reserved, RES0.	
25:24 (R/NW)	INOUT	INOUT. Input/output options. Indicates presence of the input gate. If the CTM is not implemented, this field is RAZ. 00 CTIGATE does not mask propagation of input events from external channels. 01 CTIGATE masks propagation of input events from external channels. All other values are reserved.	
		0	CTIGATE does not mask propagation of input events from external channels.
		1	CTIGATE masks propagation of input events from external channels.
23:22 (R/NW)	RES0_CTIDEVID_23_22	0. Reserved, RES0.	

Table 48-136: A55_CLUSTER_CTIDEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/NW)	NUMCHAN	NUMCHAN. Number of ECT channels implemented. IMPLEMENTATION DEFINED. For ARMv8, valid values are: 000011 3 channels (0..2) implemented. 000100 4 channels (0..3) implemented. 000101 5 channels (0..4) implemented. 000110 6 channels (0..5) implemented. and so on up to 100000, 32 channels (0..31) implemented. All other values are reserved.	
		3	3 channels (0..2) implemented.
		4	4 channels (0..3) implemented.
		5	5 channels (0..4) implemented.
		6	6 channels (0..5) implemented.
15:14 (R/NW)	RES0_CTIDEVID_15_14	0. Reserved, RES0.	
13:8 (R/NW)	NUMTRIG	NUMTRIG. Number of triggers implemented. IMPLEMENTATION DEFINED. This is one more than the index of the largest trigger, rather than the actual number of triggers. For ARMv8, valid values are: 000011 Up to 3 triggers (0..2) implemented. 001000 Up to 8 triggers (0..7) implemented. 001001 Up to 9 triggers (0..8) implemented. 001010 Up to 10 triggers (0..9) implemented. and so on up to 100000, 32 triggers (0..31) implemented. All other values are reserved. If the contains a Trace extension, this field must be at least 0b001000. There is no guarantee that any of the implemented triggers, including the highest numbered, are connected to any components.	
		3	Up to 3 triggers (0..2) implemented.
		8	Up to 8 triggers (0..7) implemented.
		9	Up to 9 triggers (0..8) implemented.
		10	Up to 10 triggers (0..9) implemented.
7:5 (R/NW)	RES0_CTIDEVID_7_5	0. Reserved, RES0.	
4:0 (R/NW)	EXTMUXNUM	EXTMUXNUM. Number of multiplexors available on triggers. This value is used in conjunction with External Control register, ASICCTL.	

CTI Device ID 1 Register

Reserved for future information about the CTI component to the debugger.

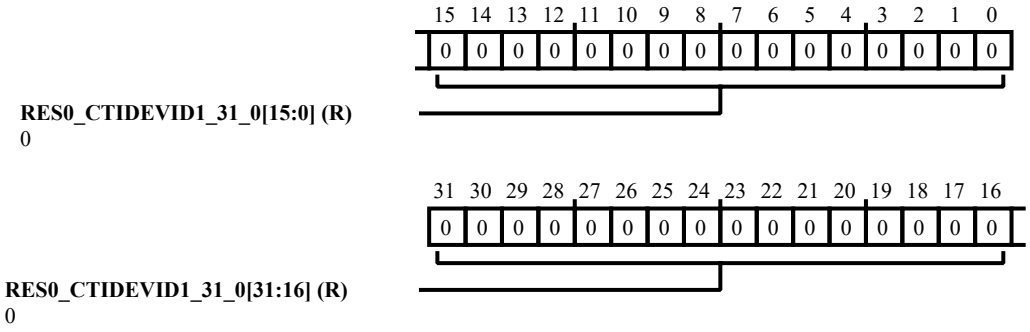


Figure 48-139: A55_CLUSTER_CTIDEVID1 Register Diagram

Table 48-137: A55_CLUSTER_CTIDEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_CTIDEVID1_31_0	0. Reserved, RES0.

CTI Device ID 2 Register

Reserved for future information about the CTI component to the debugger.

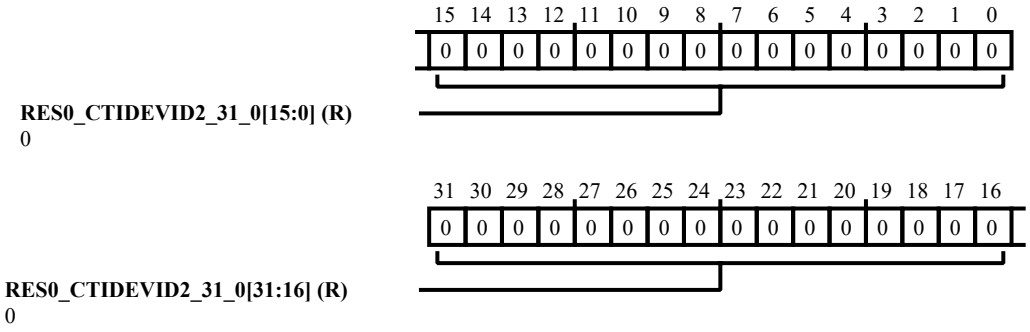


Figure 48-140: A55_CLUSTER_CTIDEVID2 Register Diagram

Table 48-138: A55_CLUSTER_CTIDEVID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_CTIDEVID2_31_0	0. Reserved, RES0.

CTI Device Type Register

Indicates to a debugger that this component is part of a PEs cross-trigger interface.

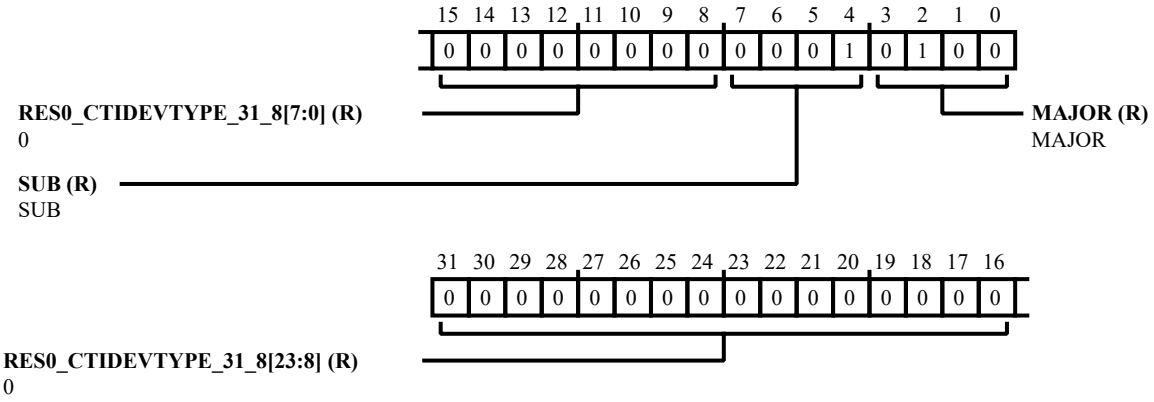


Figure 48-141: A55_CLUSTER_CTIDEVTYPE Register Diagram

Table 48-139: A55_CLUSTER_CTIDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIDEV-TYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Subtype. Must read as 0x1 to indicate this is a component within a PE.
3:0 (R/NW)	MAJOR	MAJOR. Major type. Must read as 0x4 to indicate this is a cross-trigger component.

CTI Channel Gate Enable Register

Determines whether events on channels propagate through the CTM to other ECT components, or from the CTM into the CTI.

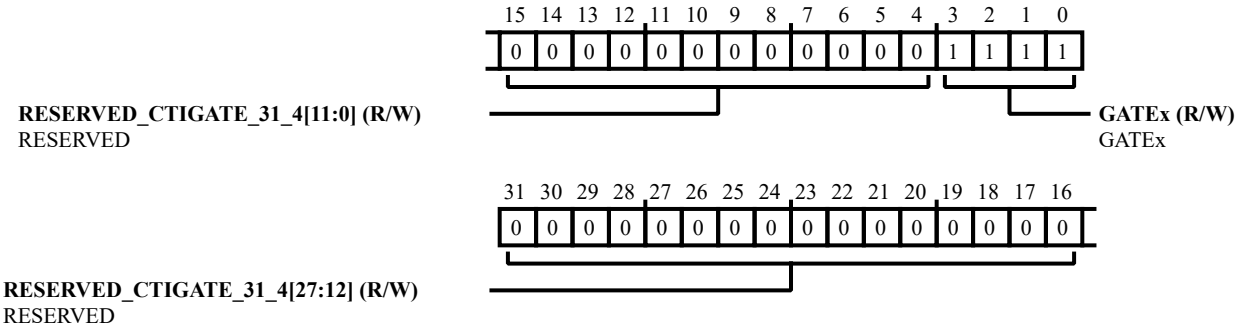


Figure 48-142: A55_CLUSTER_CTIGATE Register Diagram

Table 48-140: A55_CLUSTER_CTIGATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTI-GATE_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	GATEX	GATEx. Determines whether events on channels propagate through the CTM to other ECT components or from the CTM into the CTI.	
		0	Disable output and if CTIDEVID.INOUT == 0b01 input channel <x> propagation.
		1	Enable output and if CTIDEVID.INOUT == 0b01 input channel <x> propagation.

CTI Input Trigger to Output Channel Enable 0 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

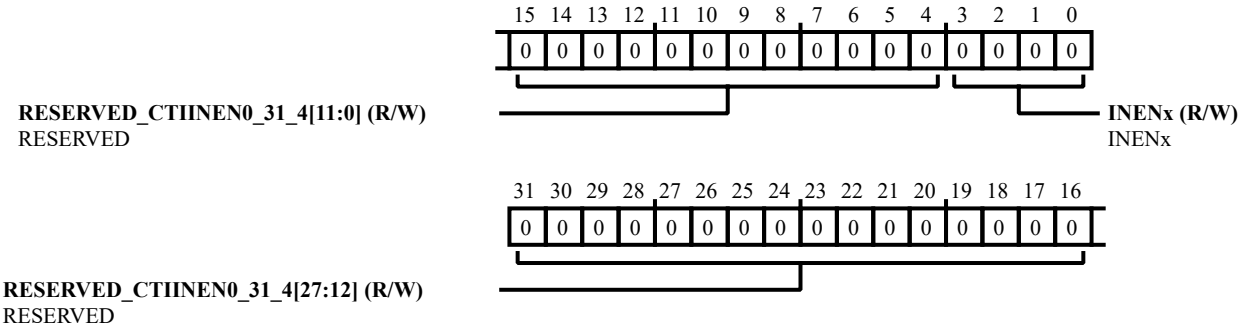


Figure 48-143: A55_CLUSTER_CTIINEN0 Register Diagram

Table 48-141: A55_CLUSTER_CTIINEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN0_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 0 to output channel <x> enable.	
		0	Input trigger 0 will not generate an event on output channel <x>.
		1	Input trigger 0 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 1 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

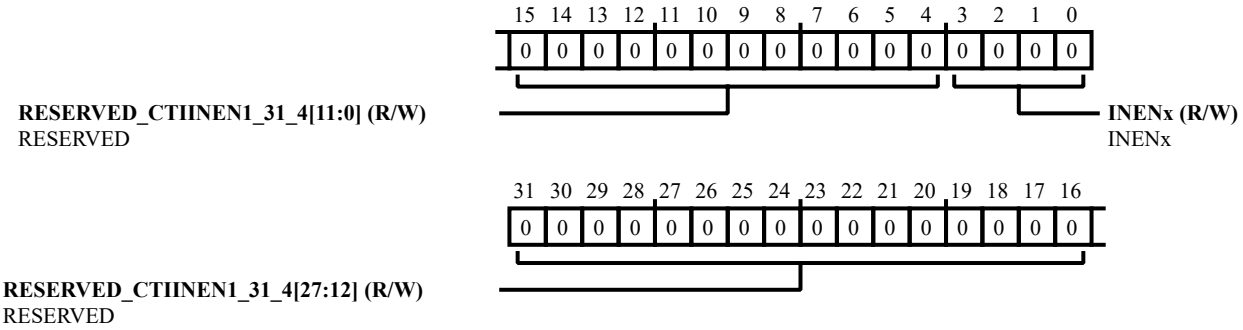


Figure 48-144: A55_CLUSTER_CTIINEN1 Register Diagram

Table 48-142: A55_CLUSTER_CTIINEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN1_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 1 to output channel <x> enable.	
		0	Input trigger 1 will not generate an event on output channel <x>.
		1	Input trigger 1 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 2 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

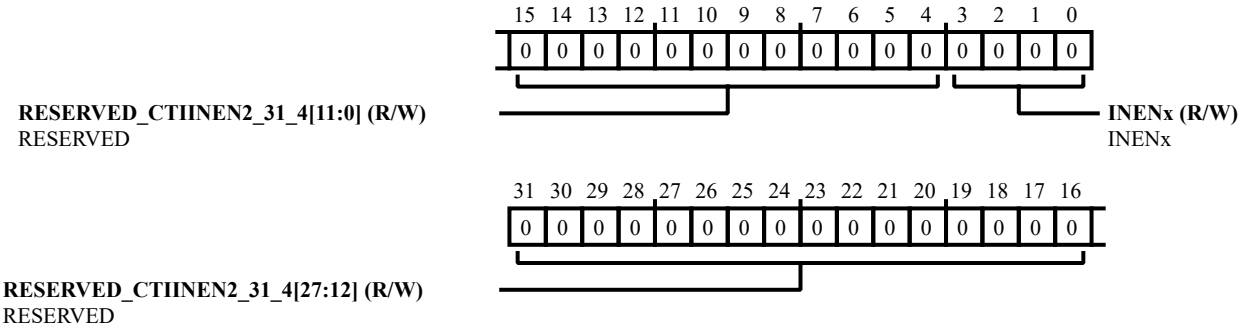


Figure 48-145: A55_CLUSTER_CTIINEN2 Register Diagram

Table 48-143: A55_CLUSTER_CTIINEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN2_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 2 to output channel <x> enable.	
		0	Input trigger 2 will not generate an event on output channel <x>.
		1	Input trigger 2 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 3 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

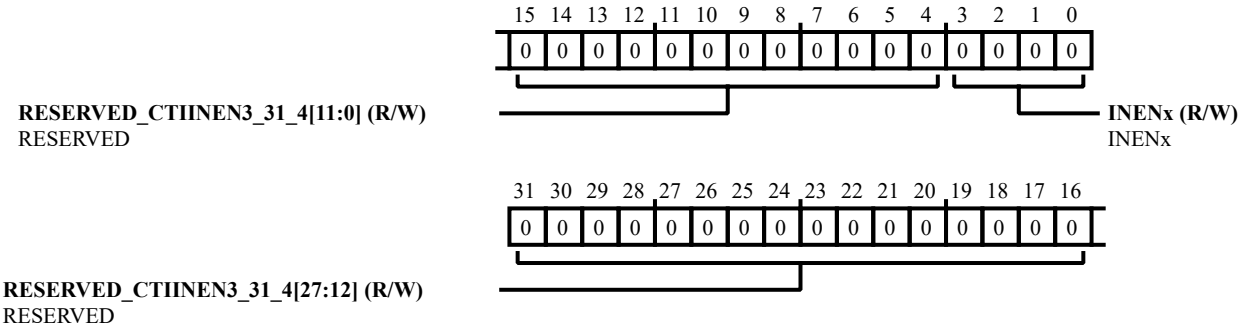


Figure 48-146: A55_CLUSTER_CTIINEN3 Register Diagram

Table 48-144: A55_CLUSTER_CTIINEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN3_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 3 to output channel <x> enable.	
		0	Input trigger 3 will not generate an event on output channel <x>.
		1	Input trigger 3 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 4 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

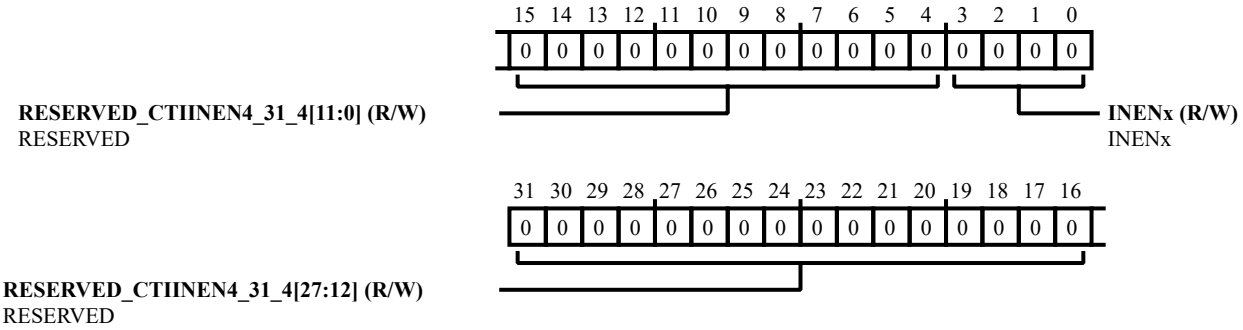


Figure 48-147: A55_CLUSTER_CTIINEN4 Register Diagram

Table 48-145: A55_CLUSTER_CTIINEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN4_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 4 to output channel <x> enable.	
		0	Input trigger 4 will not generate an event on output channel <x>.
		1	Input trigger 4 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 5 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

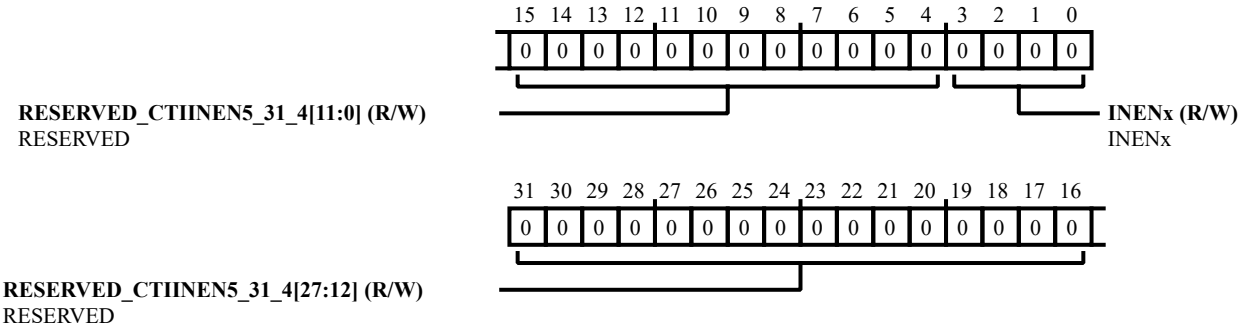


Figure 48-148: A55_CLUSTER_CTIINEN5 Register Diagram

Table 48-146: A55_CLUSTER_CTIINEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN5_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 5 to output channel <x> enable.	
		0	Input trigger 5 will not generate an event on output channel <x>.
		1	Input trigger 5 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 6 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

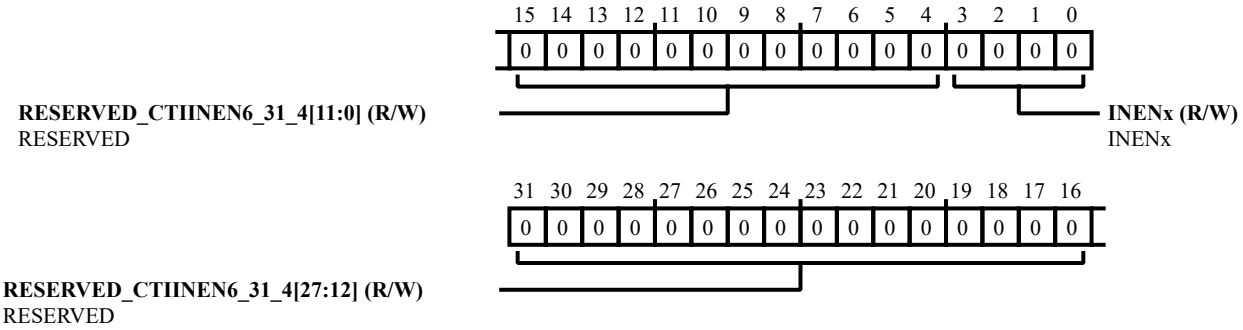


Figure 48-149: A55_CLUSTER_CTIINEN6 Register Diagram

Table 48-147: A55_CLUSTER_CTIINEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN6_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 6 to output channel <x> enable.	
		0	Input trigger 6 will not generate an event on output channel <x>.
		1	Input trigger 6 will generate an event on output channel <x>.

CTI Input Trigger to Output Channel Enable 7 Register

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

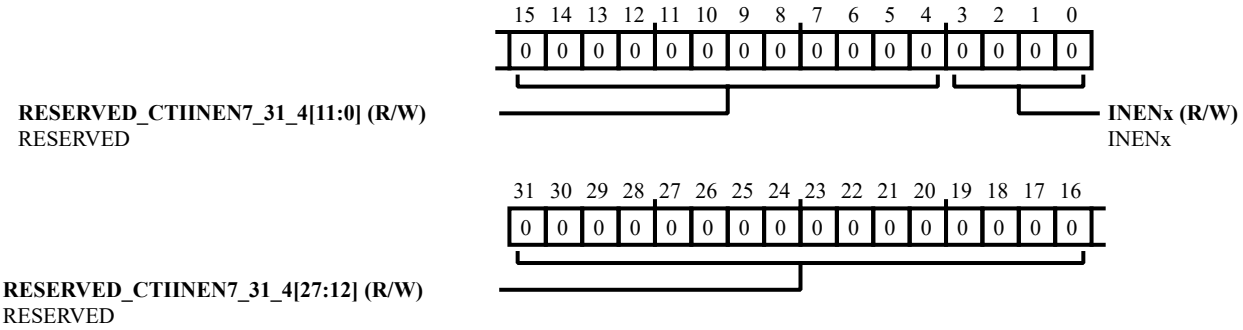


Figure 48-150: A55_CLUSTER_CTIINEN7 Register Diagram

Table 48-148: A55_CLUSTER_CTIINEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTII-NEN7_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	INENX	INENx. Input trigger 7 to output channel <x> enable.	
		0	Input trigger 7 will not generate an event on output channel <x>.
		1	Input trigger 7 will generate an event on output channel <x>.

CTI Output Trigger Acknowledge Register

Can be used to deactivate the output triggers.

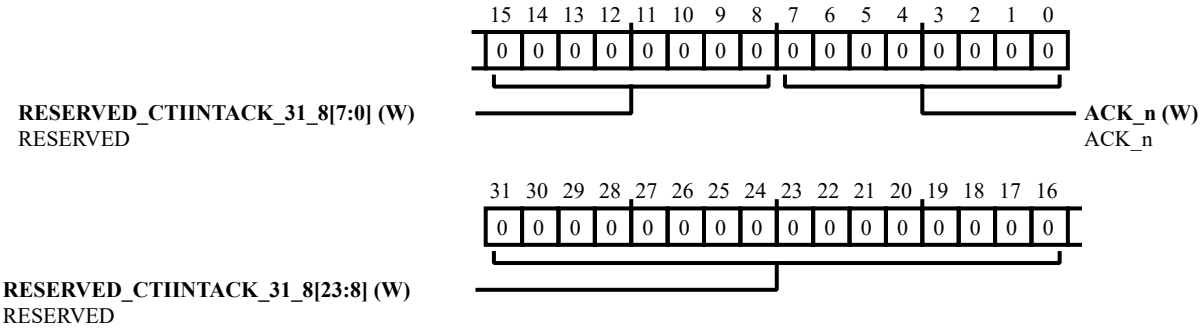


Figure 48-151: A55_CLUSTER_CTIINTACK Register Diagram

Table 48-149: A55_CLUSTER_CTIINTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (RX/W)	RESERVED_CTIIN-TACK_31_8	RESERVED. Reserved RES0.	
		0	Reserved.
7:0 (RX/W)	ACK_N	ACK_n. Can be used to create soft acknowledges for output triggers.	
		0	No effect.
		1	Deactivate the trigger.

CTI Integration mode Control Register

Enables the CTI to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

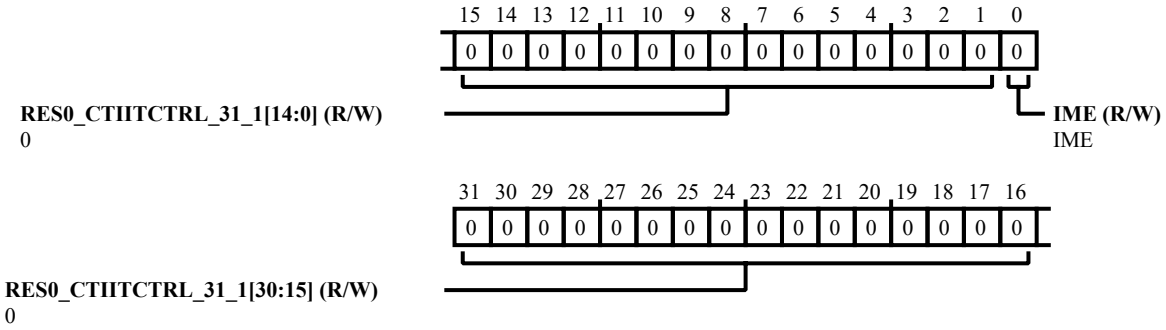


Figure 48-152: A55_CLUSTER_CTIITCTRL Register Diagram

Table 48-150: A55_CLUSTER_CTIITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_CTIITCTRL_31_1	0. Reserved, RES0.	
0 (R/W)	IME	IME. Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED. 0 Normal operation. 1 Integration mode enabled.	
		0	Normal operation.
		1	Integration mode enabled.

CTI Lock Access Register

Allows or disallows access to the CTI registers through a memory-mapped interface.

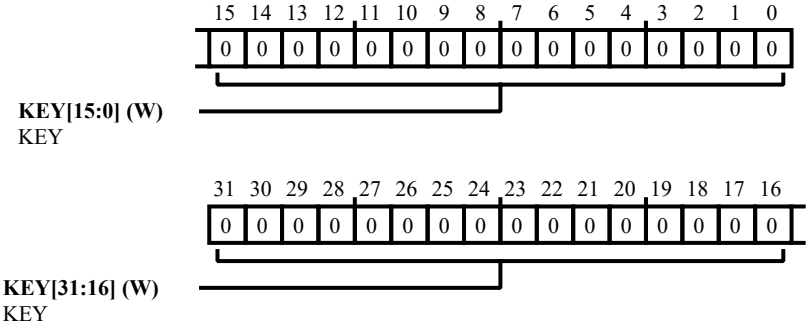


Figure 48-153: A55_CLUSTER_CTILAR Register Diagram

Table 48-151: A55_CLUSTER_CTILAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

CTI Lock Status Register

Indicates the current status of the Software Lock for CTI registers.

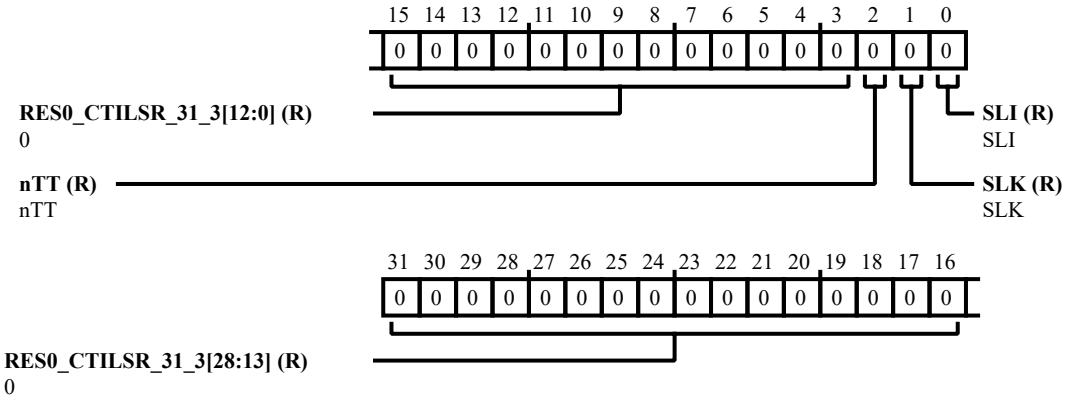


Figure 48-154: A55_CLUSTER_CTILSR Register Diagram

Table 48-152: A55_CLUSTER_CTILSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_CTILSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0. For memory-mapped accesses when the Software Lock is implemented, possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

Table 48-152: A55_CLUSTER_CTILSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SLI	SLI. Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are: 0 Software Lock not implemented or not memory-mapped access. 1 Software Lock implemented and memory-mapped access.	
		0	Software Lock not implemented or not memory-mapped access.
		1	Software Lock implemented and memory-mapped access.

CTI Input Channel to Output Trigger Enable 0 Register

Defines which input channels generate output trigger n.

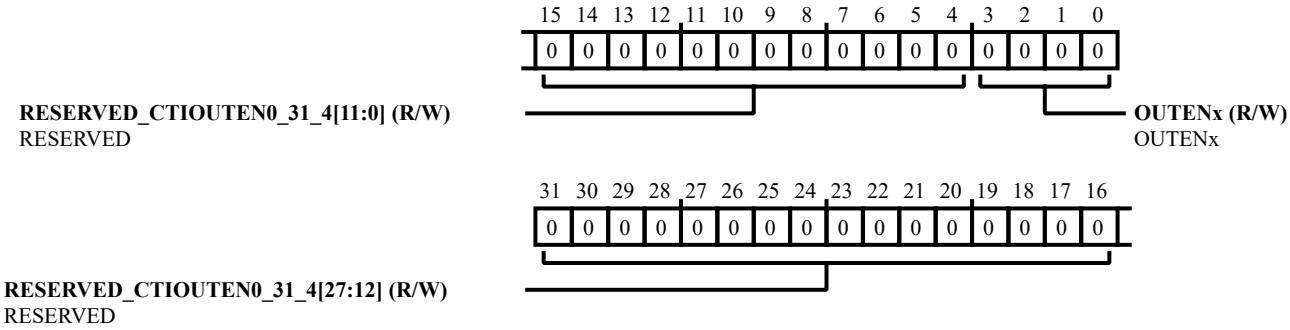


Figure 48-155: A55_CLUSTER_CTIOUTEN0 Register Diagram

Table 48-153: A55_CLUSTER_CTIOUTEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN0_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 0 enable.	
		0	An event on input channel <x> will not cause output trigger 0 to be asserted.
		1	An event on input channel <x> will cause output trigger 0 to be asserted.

CTI Input Channel to Output Trigger Enable 1 Register

Defines which input channels generate output trigger n.

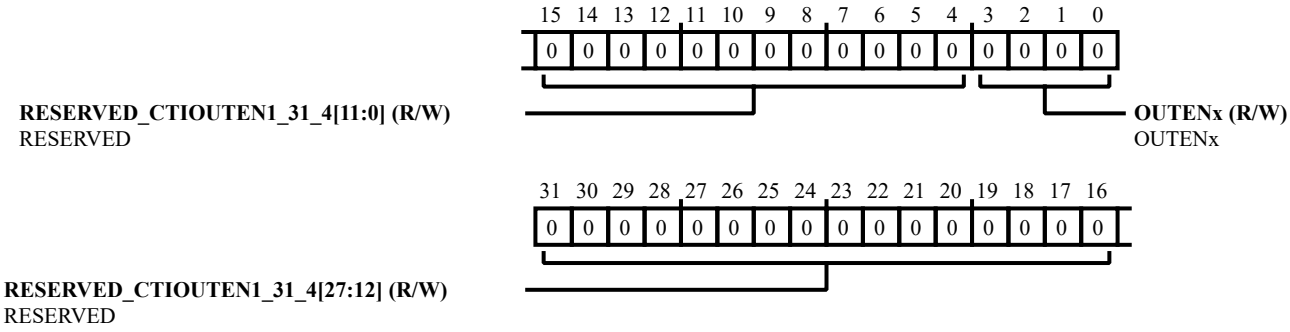


Figure 48-156: A55_CLUSTER_CTIOUTEN1 Register Diagram

Table 48-154: A55_CLUSTER_CTIOUTEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN1_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 1 enable.	
		0	An event on input channel <x> will not cause output trigger 1 to be asserted.
		1	An event on input channel <x> will cause output trigger 1 to be asserted.

CTI Input Channel to Output Trigger Enable 2 Register

Defines which input channels generate output trigger n.

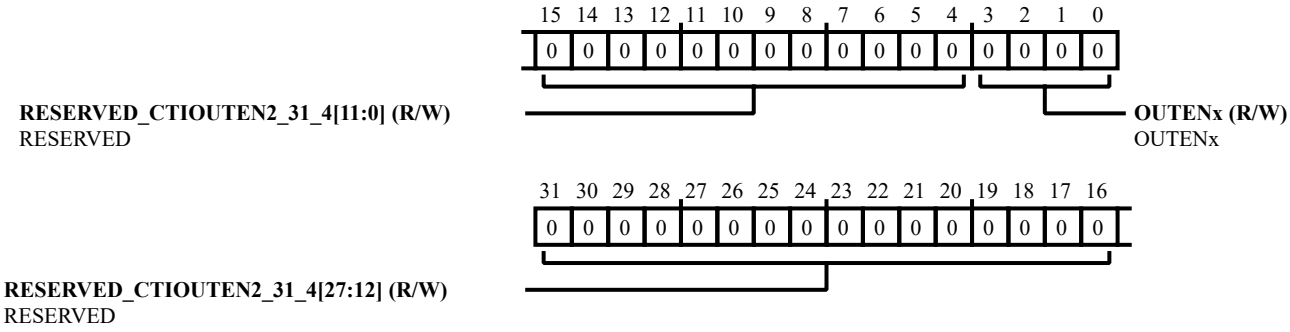


Figure 48-157: A55_CLUSTER_CTIOUTEN2 Register Diagram

Table 48-155: A55_CLUSTER_CTIOUTEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN2_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 2 enable.	
		0	An event on input channel <x> will not cause output trigger 2 to be asserted.
		1	An event on input channel <x> will cause output trigger 2 to be asserted.

CTI Input Channel to Output Trigger Enable 3 Register

Defines which input channels generate output trigger n.

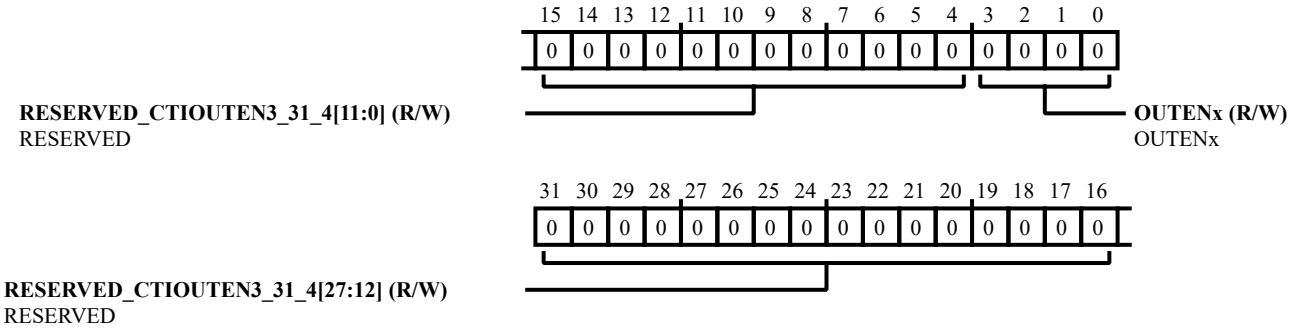


Figure 48-158: A55_CLUSTER_CTIOUTEN3 Register Diagram

Table 48-156: A55_CLUSTER_CTIOUTEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN3_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 3 enable.	
		0	An event on input channel <x> will not cause output trigger 3 to be asserted.
		1	An event on input channel <x> will cause output trigger 3 to be asserted.

CTI Input Channel to Output Trigger Enable 4 Register

Defines which input channels generate output trigger n.

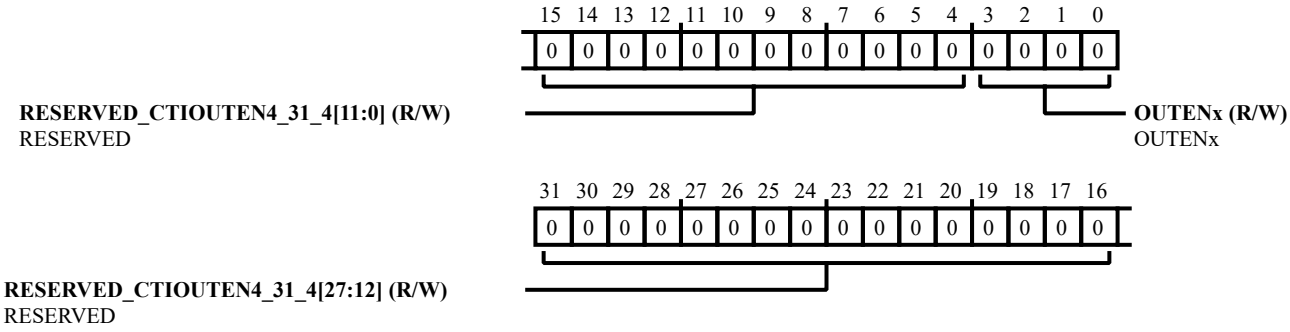


Figure 48-159: A55_CLUSTER_CTIOUTEN4 Register Diagram

Table 48-157: A55_CLUSTER_CTIOUTEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN4_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 4 enable.	
		0	An event on input channel <x> will not cause output trigger 4 to be asserted.
		1	An event on input channel <x> will cause output trigger 4 to be asserted.

CTI Input Channel to Output Trigger Enable 5 Register

Defines which input channels generate output trigger n.

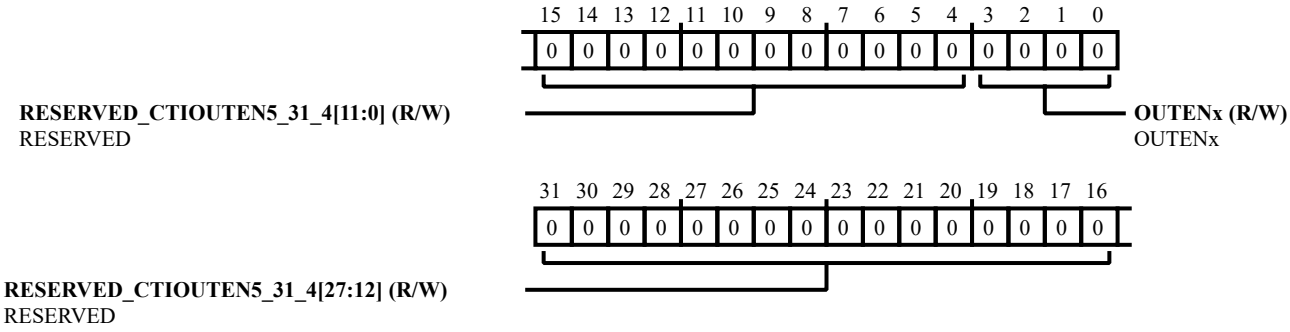


Figure 48-160: A55_CLUSTER_CTIOUTEN5 Register Diagram

Table 48-158: A55_CLUSTER_CTIOUTEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN5_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 5 enable.	
		0	An event on input channel <x> will not cause output trigger 5 to be asserted.
		1	An event on input channel <x> will cause output trigger 5 to be asserted.

CTI Input Channel to Output Trigger Enable 6 Register

Defines which input channels generate output trigger n.

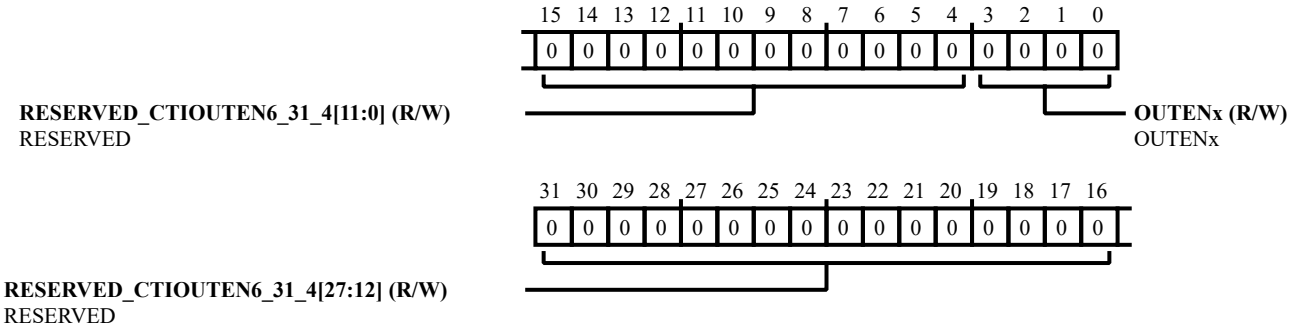


Figure 48-161: A55_CLUSTER_CTIOUTEN6 Register Diagram

Table 48-159: A55_CLUSTER_CTIOUTEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN6_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 6 enable.	
		0	An event on input channel <x> will not cause output trigger 6 to be asserted.
		1	An event on input channel <x> will cause output trigger 6 to be asserted.

CTI Input Channel to Output Trigger Enable 7 Register

Defines which input channels generate output trigger n.

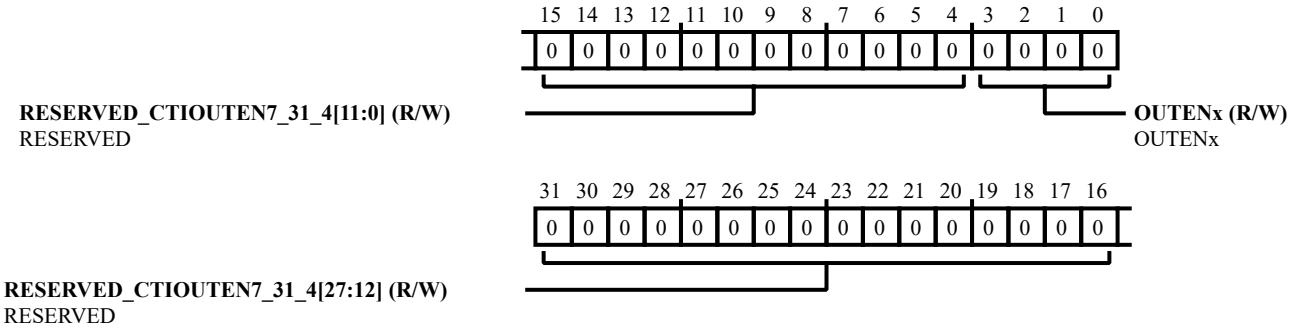


Figure 48-162: A55_CLUSTER_CTIOUTEN7 Register Diagram

Table 48-160: A55_CLUSTER_CTIOUTEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_CTIOUT-EN7_31_4	RESERVED. Reserved RES0.	
		0	Reserved.
3:0 (R/W)	OUTENX	OUTENx. Input channel <x> to output trigger 7 enable.	
		0	An event on input channel <x> will not cause output trigger 7 to be asserted.
		1	An event on input channel <x> will cause output trigger 7 to be asserted.

CTI Peripheral Identification 0 Register

Provides information to identify a CTI component.

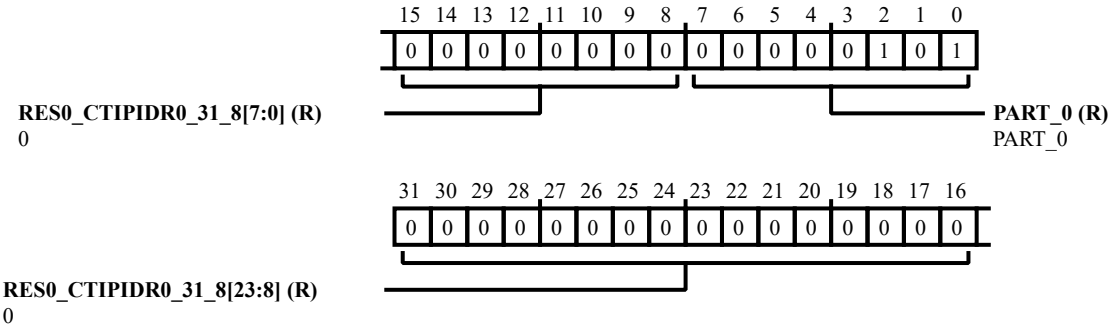


Figure 48-163: A55_CLUSTER_CTIPIDR0 Register Diagram

Table 48-161: A55_CLUSTER_CTIPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, least significant byte.

CTI Peripheral Identification 1 Register

Provides information to identify a CTI component.

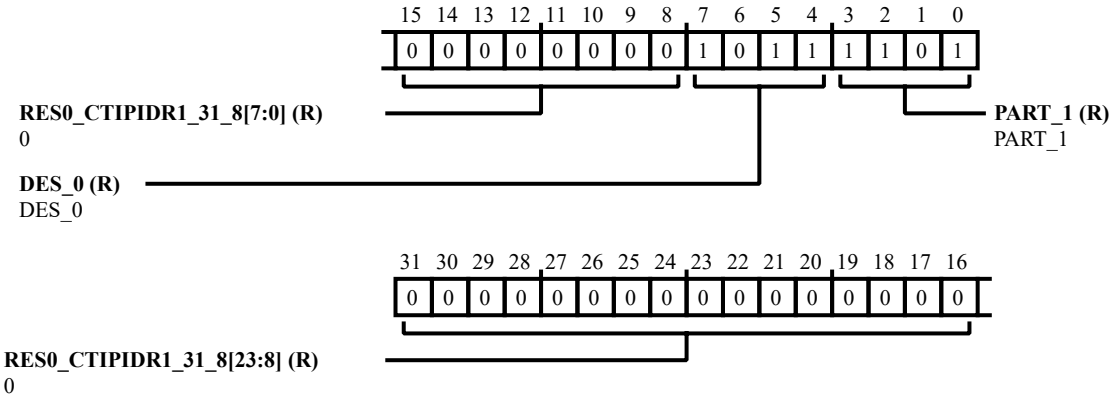


Figure 48-164: A55_CLUSTER_CTIPIDR1 Register Diagram

Table 48-162: A55_CLUSTER_CTIPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, most significant nibble.

CTI Peripheral Identification 2 Register

Provides information to identify a CTI component.

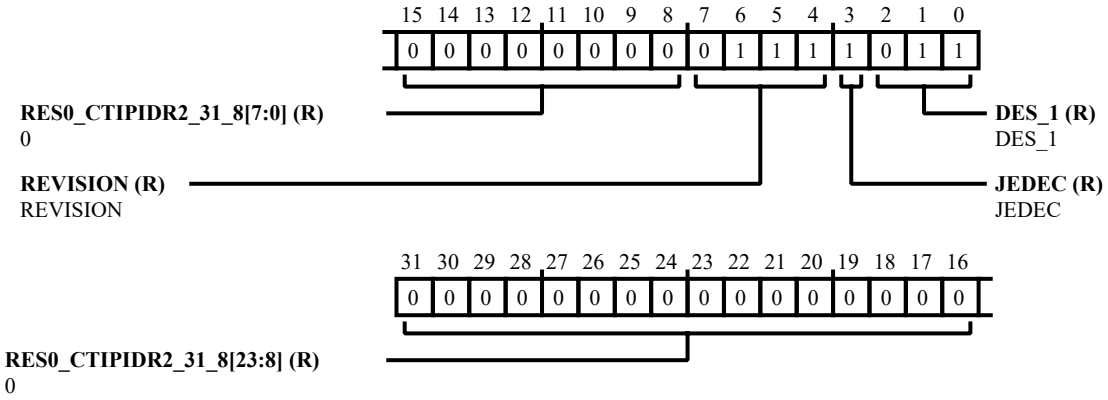


Figure 48-165: A55_CLUSTER_CTIPIDR2 Register Diagram

Table 48-163: A55_CLUSTER_CTIPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. Part major revision. Parts can also use this field to extend Part number to 16-bits.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

CTI Peripheral Identification 3 Register

Provides information to identify a CTI component.

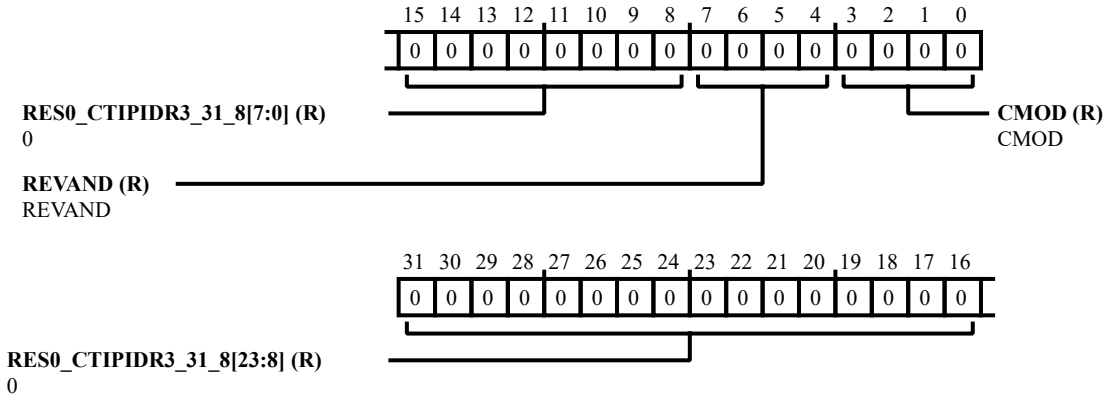


Figure 48-166: A55_CLUSTER_CTIPIDR3 Register Diagram

Table 48-164: A55_CLUSTER_CTIPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. Part minor revision. Parts using CTIPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

CTI Peripheral Identification 4 Register

Provides information to identify a CTI component.

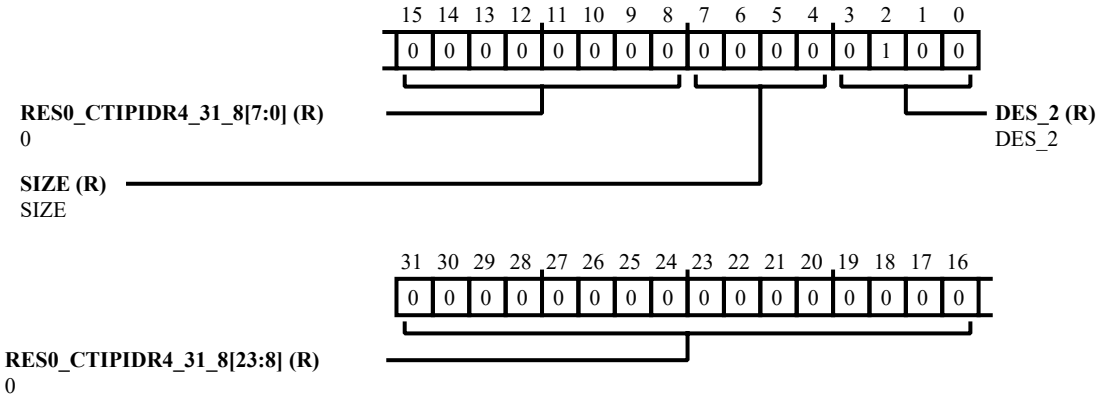


Figure 48-167: A55_CLUSTER_CTIPIDR4 Register Diagram

Table 48-165: A55_CLUSTER_CTIPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_CTIPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

CTI Peripheral Identification 5 Register

Provides information to identify a CTI component.

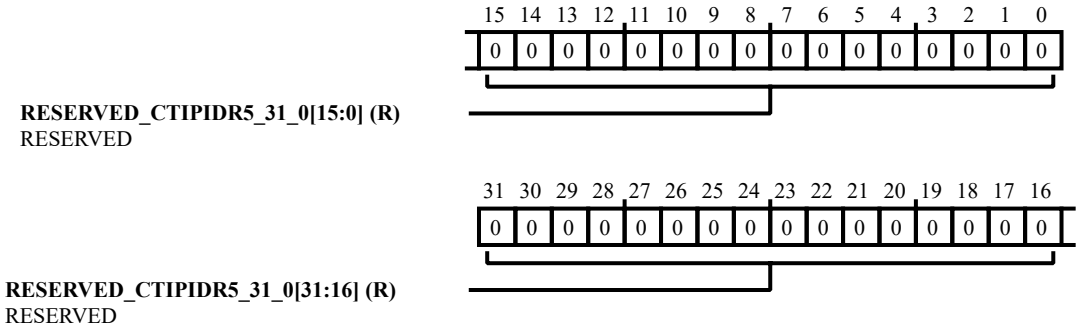


Figure 48-168: A55_CLUSTER_CTIPIDR5 Register Diagram

Table 48-166: A55_CLUSTER_CTIPIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_CTI-PIDR5_31_0	RESERVED. Reserved RES0.

CTI Peripheral Identification 6 Register

Provides information to identify a CTI component.

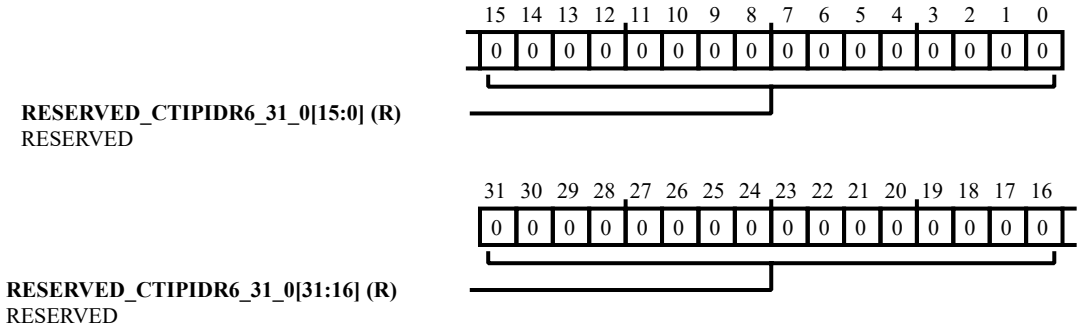


Figure 48-169: A55_CLUSTER_CTIPIDR6 Register Diagram

Table 48-167: A55_CLUSTER_CTIPIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_CTI-PIDR6_31_0	RESERVED. Reserved RES0.

CTI Peripheral Identification 7 Register

Provides information to identify a CTI component.

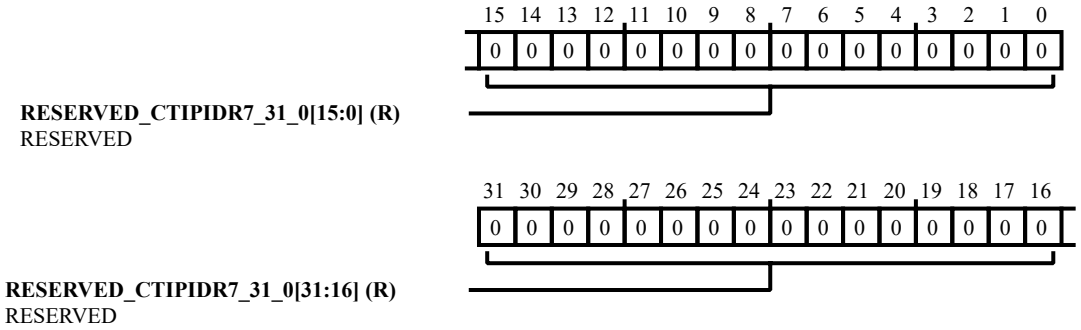


Figure 48-170: A55_CLUSTER_CTIPIDR7 Register Diagram

Table 48-168: A55_CLUSTER_CTIPIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_CTI-PIDR7_31_0	RESERVED. Reserved RES0.

CTI Trigger In Status Register

Provides the status of the trigger inputs.

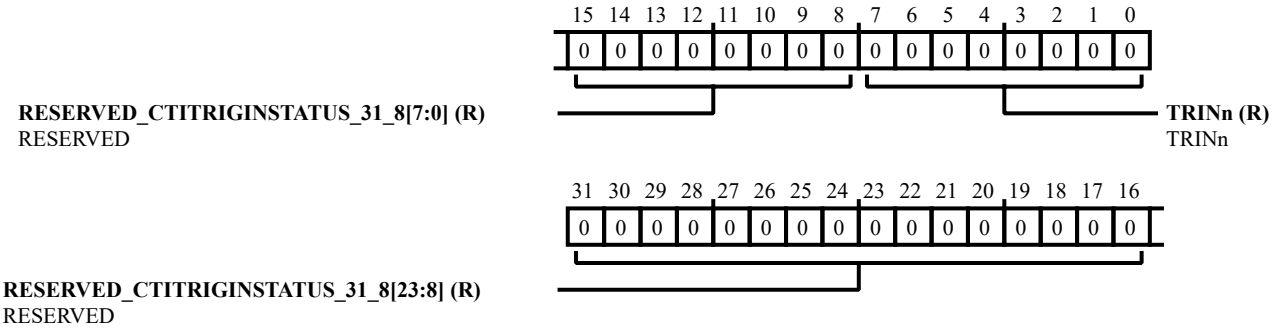


Figure 48-171: A55_CLUSTER_CTITRIGINSTATUS Register Diagram

Table 48-169: A55_CLUSTER_CTITRIGINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RESERVED_CTITRIGIN-STATUS_31_8	RESERVED. Reserved RES0.	
		0	Reserved.
7:0 (R/NW)	TRINN	TRINn. Provides the status of the trigger inputs.	
		0	Input trigger n is inactive.
		1	Input trigger n is active.

CTI Trigger Out Status Register

Provides the raw status of the trigger outputs, after processing by any IMPLEMENTATION DEFINED trigger interface logic. For output triggers that are self-acknowledging, this is only meaningful if the CTI implements multicycle channel events.

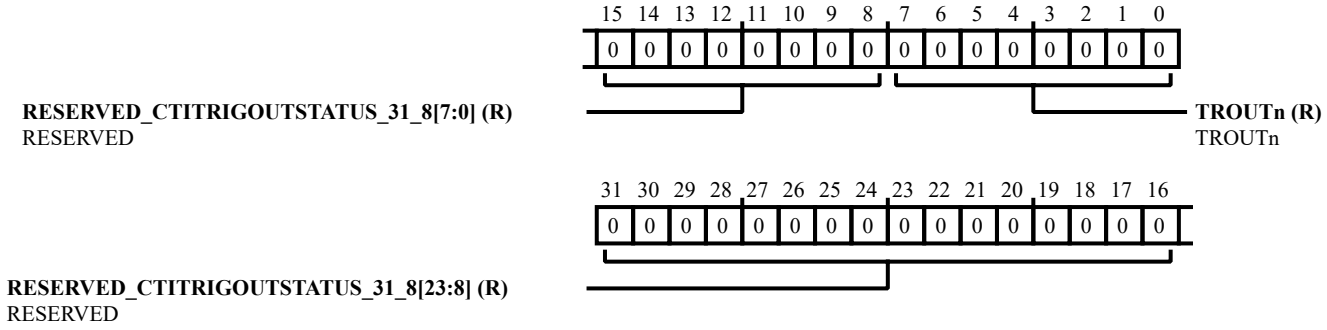


Figure 48-172: A55_CLUSTER_CTITRIGOUTSTATUS Register Diagram

Table 48-170: A55_CLUSTER_CTITRIGOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RESERVED_CTITRI-GOUTSTATUS_31_8	RESERVED. Reserved RES0.	
		0	Reserved.
7:0 (R/NW)	TROUTN	TROUTn. Provides the status of the trigger outputs.	
		0	Output trigger n is inactive.
		1	Output trigger n is active.

Debug Authentication Status Register

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

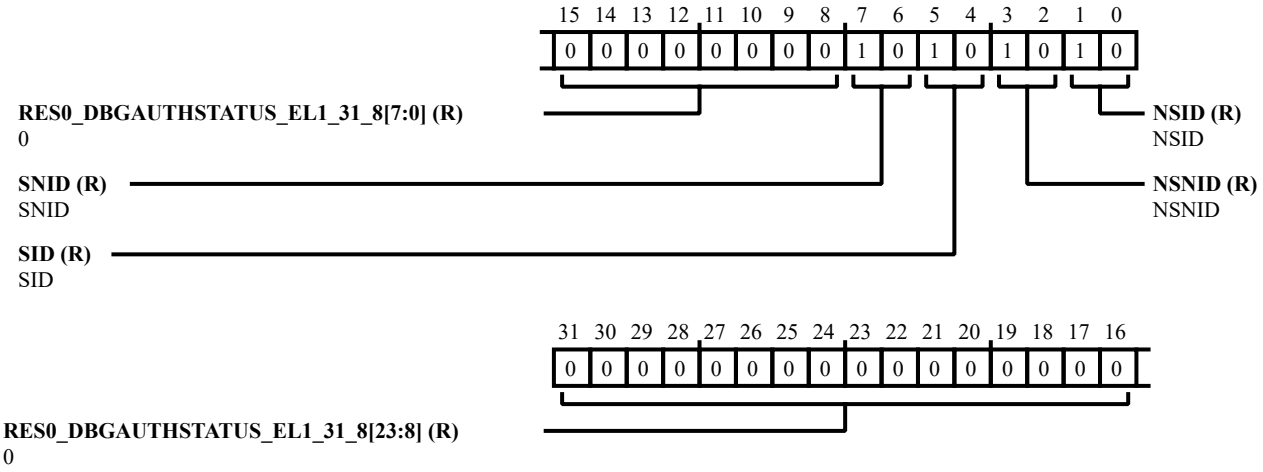


Figure 48-173: A55_CLUSTER_DBGAUTHSTATUS_EL1 Register Diagram

Table 48-171: A55_CLUSTER_DBGAUTHSTATUS_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_DBGAUTHSTATUS_EL1_31_8	0. Reserved, RES0.	
7:6 (R/NW)	SNID	SNID. Secure non-invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state. 10 Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
		2	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

Table 48-171: A55_CLUSTER_DBGAUTHSTATUS_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	SID	SID. Secure invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state. 10 Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
		2	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.
3:2 (R/NW)	NSNID	NSNID. Non-secure non-invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Secure state. 10 Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
		2	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.
1:0 (R/NW)	NSID	NSID. Non-secure invasive debug. Possible values of this field are: 00 Not implemented. EL3 is not implemented and the implemented Security state is Secure state. 10 Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE. 11 Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE. Other values are reserved.	
		0	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
		2	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
		3	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

External Debug Component Identification 0 Register

Provides information to identify an external debug component.

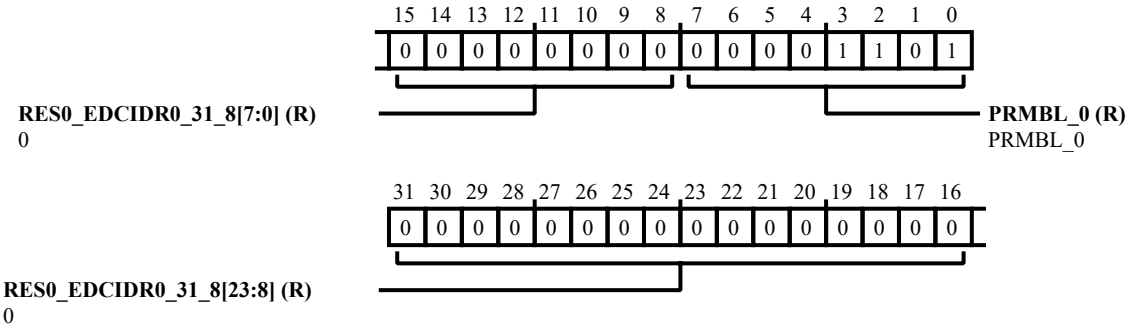


Figure 48-174: A55_CLUSTER_EDCIDR0 Register Diagram

Table 48-172: A55_CLUSTER_EDCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

External Debug Component Identification 1 Register

Provides information to identify an external debug component.

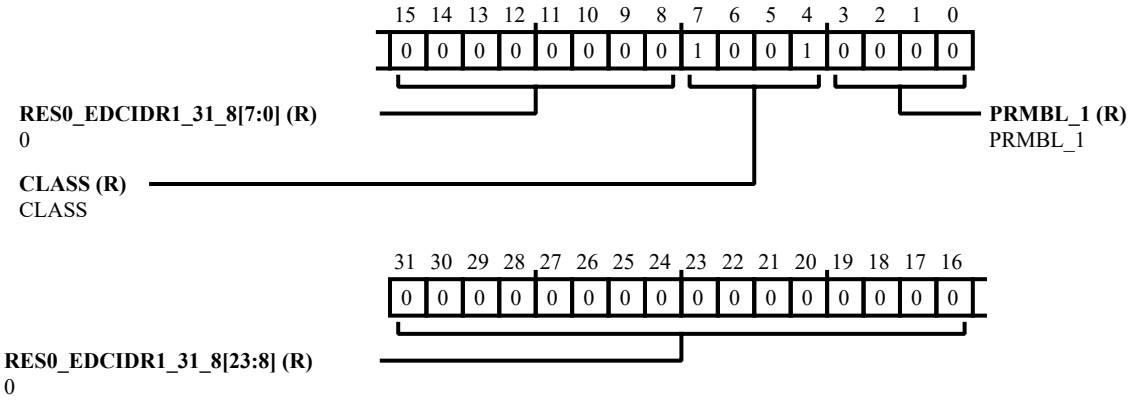


Figure 48-175: A55_CLUSTER_EDCIDR1 Register Diagram

Table 48-173: A55_CLUSTER_EDCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, debug component.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. RAZ.

External Debug Component Identification 2 Register

Provides information to identify an external debug component.

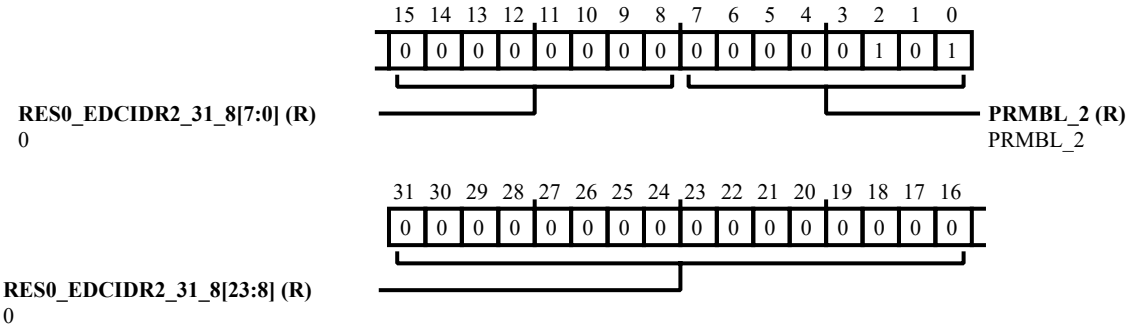


Figure 48-176: A55_CLUSTER_EDCIDR2 Register Diagram

Table 48-174: A55_CLUSTER_EDCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

External Debug Component Identification 3 Register

Provides information to identify an external debug component.

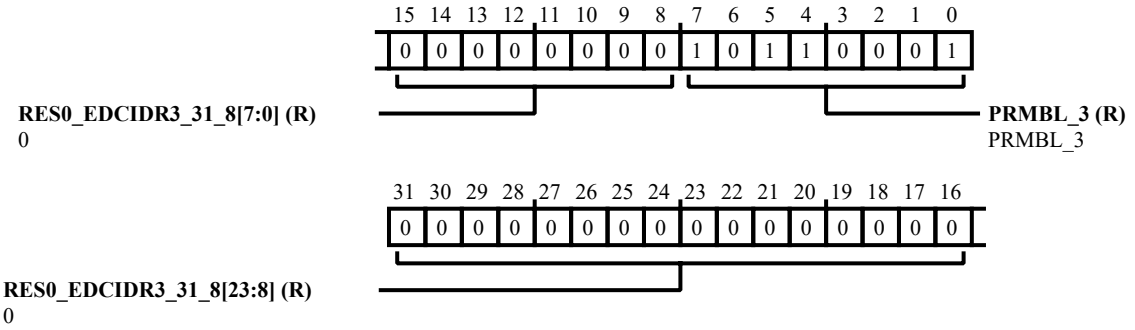


Figure 48-177: A55_CLUSTER_EDCIDR3 Register Diagram

Table 48-175: A55_CLUSTER_EDCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDCIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

External Debug Device Affinity 0 Register

Copy of the low half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

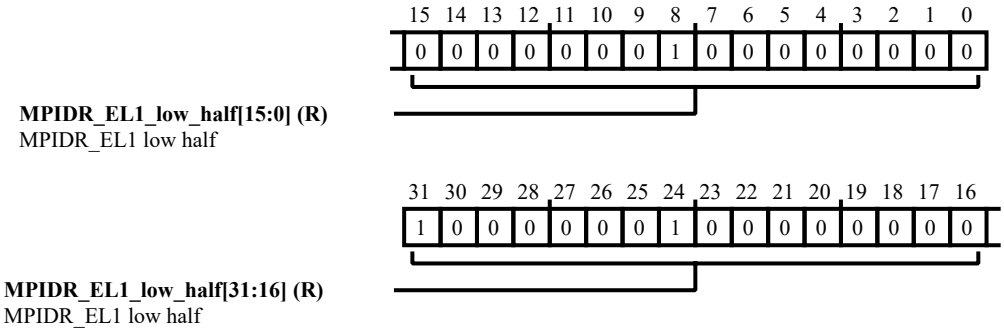


Figure 48-178: A55_CLUSTER_EDDEVAFF0 Register Diagram

Table 48-176: A55_CLUSTER_EDDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_LOW_HALF	MPIDR_EL1 low half. MPIDR_EL1 low half. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented Exception level.

External Debug Device Affinity 1 Register

Copy of the high half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

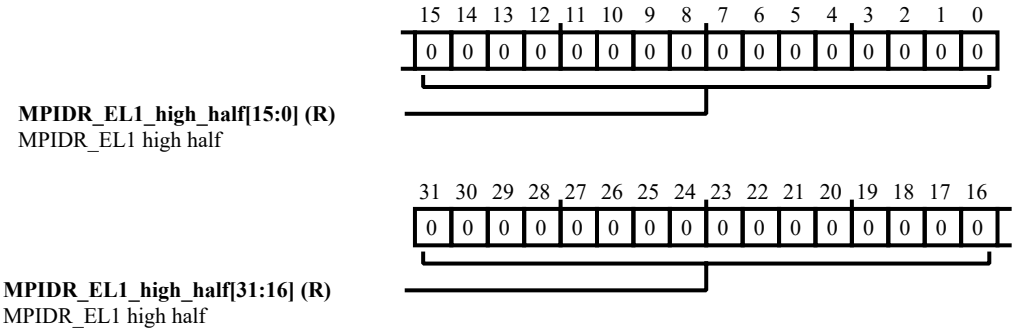


Figure 48-179: A55_CLUSTER_EDDEVAFF1 Register Diagram

Table 48-177: A55_CLUSTER_EDDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_HIGH_HALF	MPIDR_EL1 high half. MPIDR_EL1 high half. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented Exception level.

External Debug Device Architecture Register

Identifies the programmers' model architecture of the external debug component.

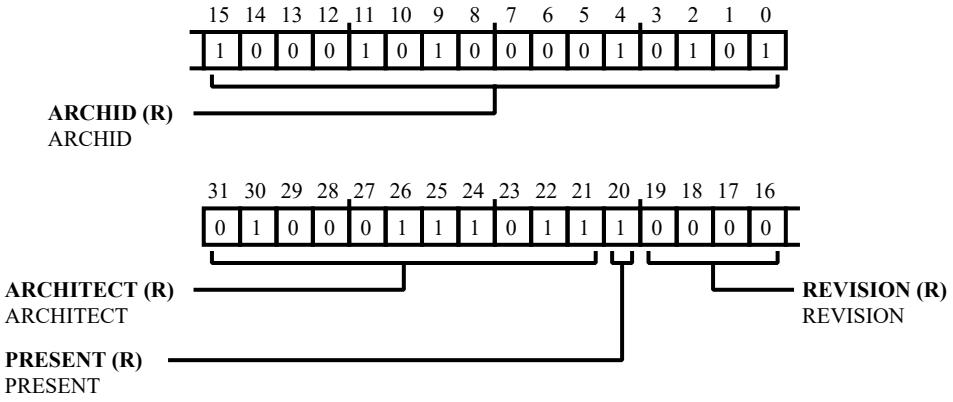


Figure 48-180: A55_CLUSTER_EDDEVARCH Register Diagram

Table 48-178: A55_CLUSTER_EDDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For debug, this is ARM Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present. This field is 1 in ARMv8.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision. For debug, the revision defined by ARMv8 is 0x0. All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided. For debug: Bits [15:12] are the architecture version:In ARMv8-A, this is 0x6.In ARMv8.1, this is 0x7.In ARMv8.2, this is0x8. Bits [11:0] are the architecture part number, 0xA15. This corresponds to the ARMv8 debug architecture version.

External Debug Device ID 0 Register

Provides extra information for external debuggers about features of the debug implementation.

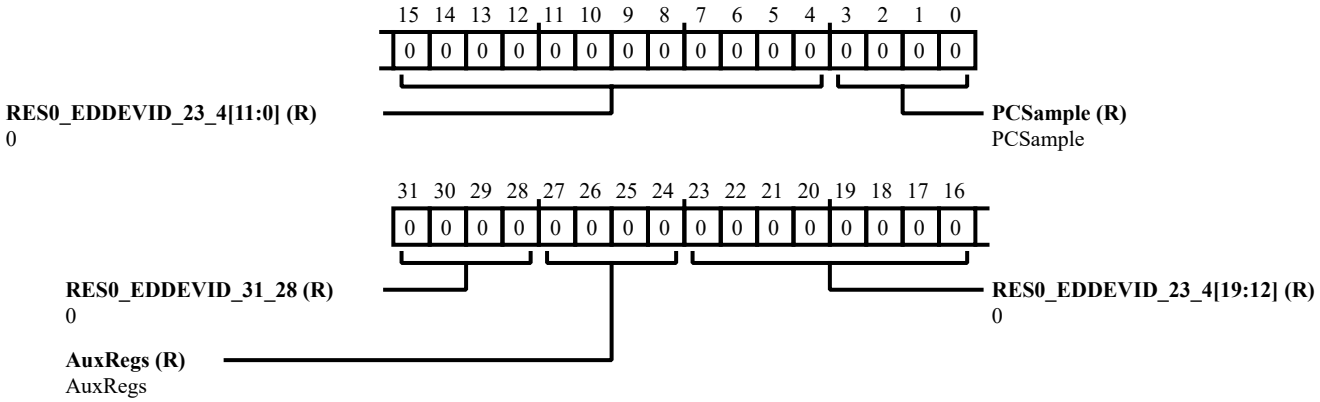


Figure 48-181: A55_CLUSTER_EDDEVID Register Diagram

Table 48-179: A55_CLUSTER_EDDEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_EDDEVID_31_28	0. Reserved, RES0.	
27:24 (R/NW)	AUXREGS	AuxRegs. Indicates support for Auxiliary registers. Permitted values for this field are: 0000 None supported. 0001 Support for External Debug Auxiliary Control Register, EDACR. All other values are reserved.	
		0	None supported.
		1	Support for External Debug Auxiliary Control Register, EDACR.
23:4 (R/NW)	RES0_EDDEVID_23_4	0. Reserved, RES0.	

Table 48-179: A55_CLUSTER_EDDEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/NW)	PCSAMPLE	PCSample. Indicates the level of PC Sample-based profiling support using external debug registers 40 through 43. Permitted values of this field in ARMv8 are: 0000 Architecture-defined form of PC Sample-based profiling not implemented. 0010 EDPCSR and EDCIDSR are implemented (only permitted if EL3 and EL2 are not implemented). 0011 EDPCSR, EDCIDSR, and EDVIDSR are implemented. All other values are reserved. In ARMv8.2 implementations, this field reads as 0b0000.	
		0	Architecture-defined form of PC Sample-based profiling not implemented.
		2	EDPCSR and EDCIDSR are implemented (only permitted if EL3 and EL2 are not implemented).
		3	EDPCSR, EDCIDSR, and EDVIDSR are implemented.

External Debug Device ID 1 Register

Provides extra information for external debuggers about features of the debug implementation.

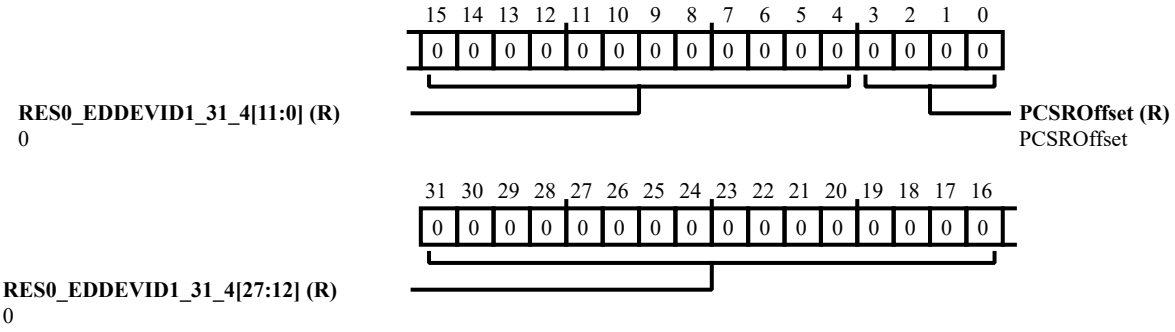


Figure 48-182: A55_CLUSTER_EDDEVID1 Register Diagram

Table 48-180: A55_CLUSTER_EDDEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_EDDEVID1_31_4	0. Reserved, RES0.	
3:0 (R/NW)	PCSROFFSET	PCSROffset. This field indicates the offset applied to PC samples returned by reads of EDPCSR. Permitted values of this field in ARMv8 are: 0000 EDPCSR not implemented. 0010 EDPCSR implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state. In ARMv8.2 implementations, this field reads as 0b0000.	
		0	EDPCSR not implemented.
		2	EDPCSR implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state.

External Debug Device ID 2 Register

Reserved for future descriptions of features of the debug implementation.

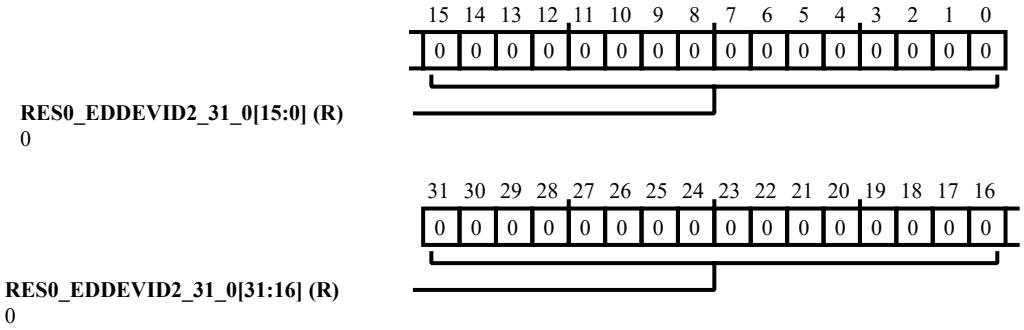


Figure 48-183: A55_CLUSTER_EDDEVID2 Register Diagram

Table 48-181: A55_CLUSTER_EDDEVID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_EDDEVID2_31_0	0. Reserved, RES0.

External Debug Device Type Register

Indicates to a debugger that this component is part of a PEs debug logic.

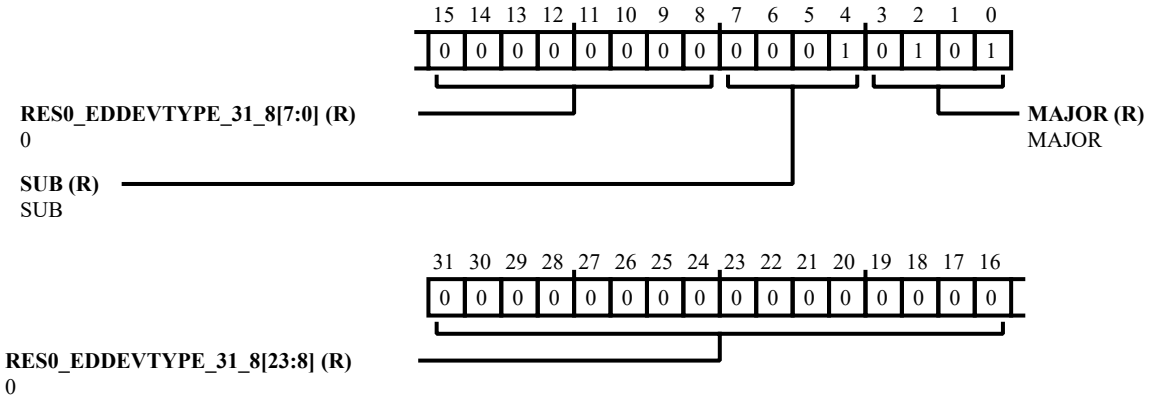


Figure 48-184: A55_CLUSTER_EDDEVTYPE Register Diagram

Table 48-182: A55_CLUSTER_EDDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDDEVTYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Subtype. Must read as 0x1 to indicate this is a component within a PE.
3:0 (R/NW)	MAJOR	MAJOR. Major type. Must read as 0x5 to indicate this is a debug logic component.

External Debug Feature Register

Provides top level information about the debug system. Debuggers must use EDDEVARCH to determine the Debug architecture version. For general information about the interpretation of the ID registers see .

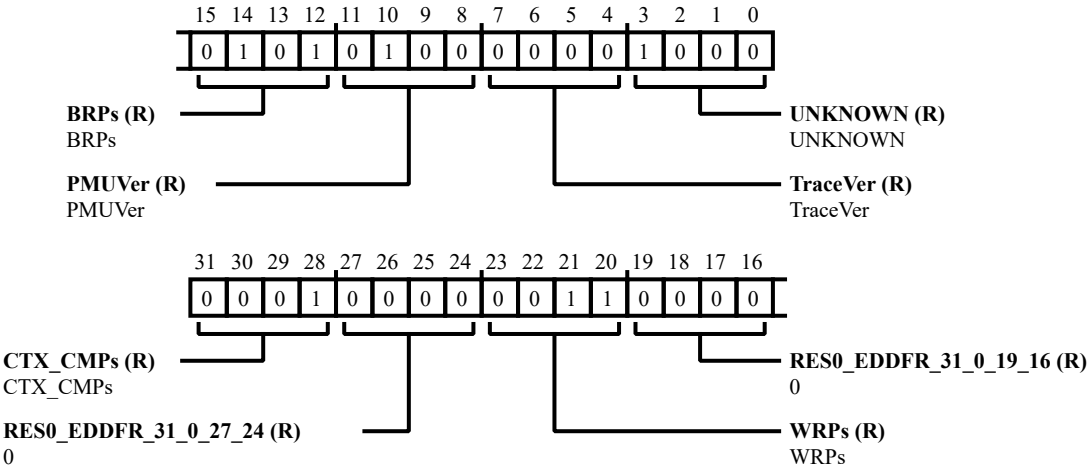


Figure 48-185: A55_CLUSTER_EDDFR_31_0 Register Diagram

Table 48-183: A55_CLUSTER_EDDFR_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/NW)	CTX_CMPS	CTX_CMPs. Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.CTX_CMPs.
27:24 (R/NW)	RES0_EDDFR_31_0_27_24	0. Reserved, RES0.
23:20 (R/NW)	WRPS	WRPs. Number of watchpoints, minus 1. The value of 0b0000 is reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.WRPs.
19:16 (R/NW)	RES0_EDDFR_31_0_19_16	0. Reserved, RES0.
15:12 (R/NW)	BRPS	BRPs. Number of breakpoints, minus 1. The value of 0b0000 is reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.BRPs.

Table 48-183: A55_CLUSTER_EDDFR_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/NW)	PMUVER	PMUVer. Performance Monitors Extension version. Indicates whether System register interface to Performance Monitors extension is implemented. Defined values are: 0000 Performance Monitors Extension System registers not implemented. 0001 Performance Monitors Extension System registers implemented, PMUv3. 0100 Performance Monitors Extension System registers implemented, PMUv3, with a 16-bit evtCount field, and if EL2 is implemented, the addition of the MDCR_EL2.HPMD bit. 1111 IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported. All other values are reserved. In ARMv8-A the permitted values are 0b0000, 0b0001 and 0b1111. In ARMv8.1 the permitted values are 0b0000, 0b0100 and 0b1111. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.PMUVer.	
		0	Performance Monitors Extension System registers not implemented.
		1	Performance Monitors Extension System registers implemented, PMUv3.
		4	Performance Monitors Extension System registers implemented, PMUv3, with a 16-bit evtCount field, and if EL2 is implemented, the addition of the MDCR_EL2.HPMD bit.
		15	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported.
7:4 (R/NW)	TRACEVER	TraceVer. Trace support. Indicates whether System register interface to a trace macrocell is implemented. Defined values are: 0000 Trace macrocell System registers not implemented. 0001 Trace macrocell System registers implemented. All other values are reserved. A value of 0b0000 only indicates that no System register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a System register interface. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64DFR0_EL1.TraceVer.	
		0	Trace macrocell System registers not implemented.
		1	Trace macrocell System registers implemented.
3:0 (R/NW)	UNKNOWN	UNKNOWN. Reserved, UNKNOWN.	

External Debug Feature Register

Provides top level information about the debug system. Debuggers must use EDDEVARCH to determine the Debug architecture version. For general information about the interpretation of the ID registers see .

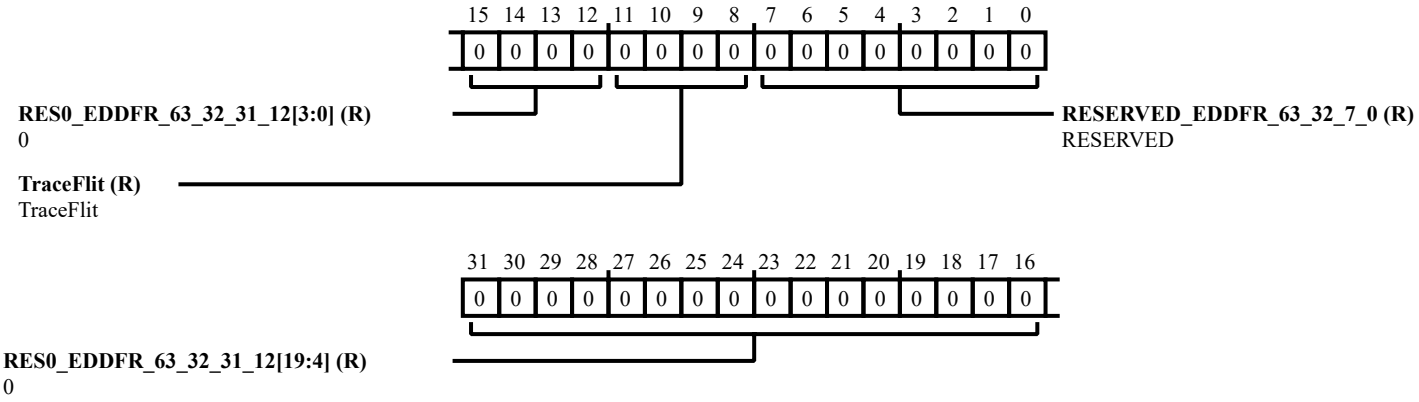


Figure 48-186: A55_CLUSTER_EDDFR_63_32 Register Diagram

Table 48-184: A55_CLUSTER_EDDFR_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_EDDFR_63_32_31_12	0. Reserved, RES0.
11:8 (R/NW)	TRACEFLIT	TraceFlit. ARMv8.4 Self-hosted Trace Extension implemented.
7:0 (R/NW)	RESERVED_EDDFR_63_32_7_0	RESERVED. Reserved. This field reads as an UNKNOWN value.

External Debug Execution Control Register

Controls Halting debug events.

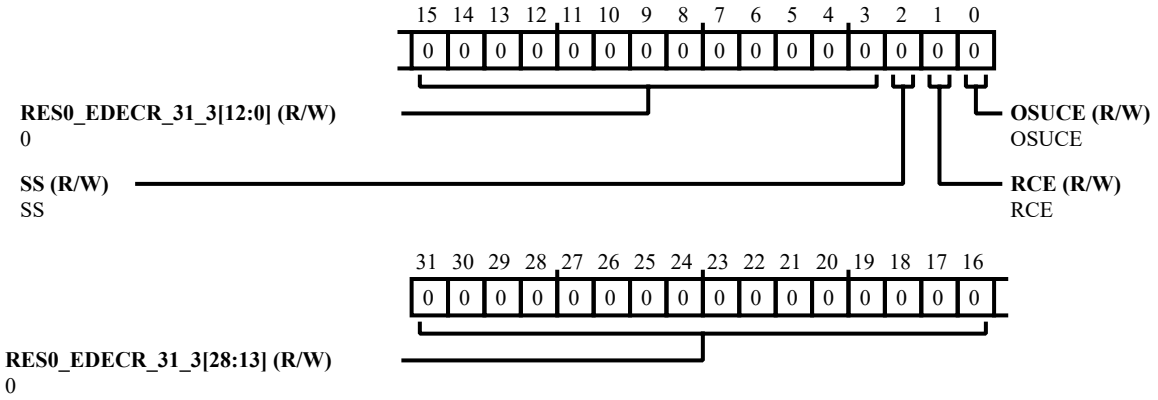


Figure 48-187: A55_CLUSTER_EDECR Register Diagram

Table 48-185: A55_CLUSTER_EDECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/W)	RES0_EDECR_31_3	0. Reserved, RES0.	
2 (R/W)	SS	SS. Halting step enable. Possible values of this field are: 0 Halting step debug event disabled. 1 Halting step debug event enabled. If the value of EDECR.SS is changed when the PE is in Non-debug state, behavior is CONSTRAINED UNPREDICTABLE as described in .	
		0	Halting step debug event disabled.
		1	Halting step debug event enabled.
1 (R/W)	RCE	RCE. Reset Catch enable. Possible values of this field are: 0 Reset Catch debug event disabled. 1 Reset Catch debug event enabled.	
		0	Reset Catch debug event disabled.
		1	Reset Catch debug event enabled.
0 (R/W)	OSUCE	OSUCE. OS Unlock Catch enabled. Possible values of this field are: 0 OS Unlock Catch debug event disabled. 1 OS Unlock Catch debug event enabled.	
		0	OS Unlock Catch debug event disabled.
		1	OS Unlock Catch debug event enabled.

External Debug Lock Access Register

Allows or disallows access to the external debug registers through a memory-mapped interface.

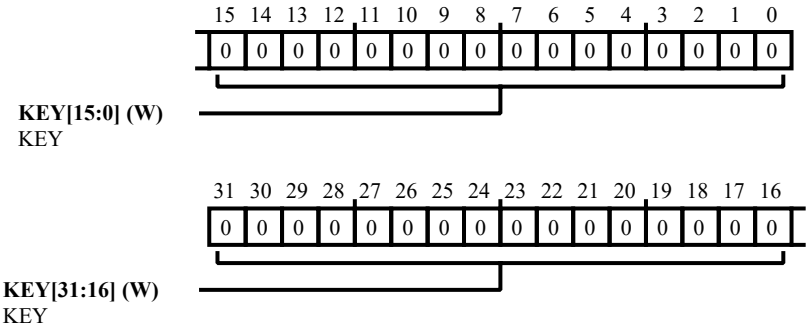


Figure 48-188: A55_CLUSTER_EDLAR Register Diagram

Table 48-186: A55_CLUSTER_EDLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

External Debug Lock Status Register

Indicates the current status of the software lock for external debug registers.

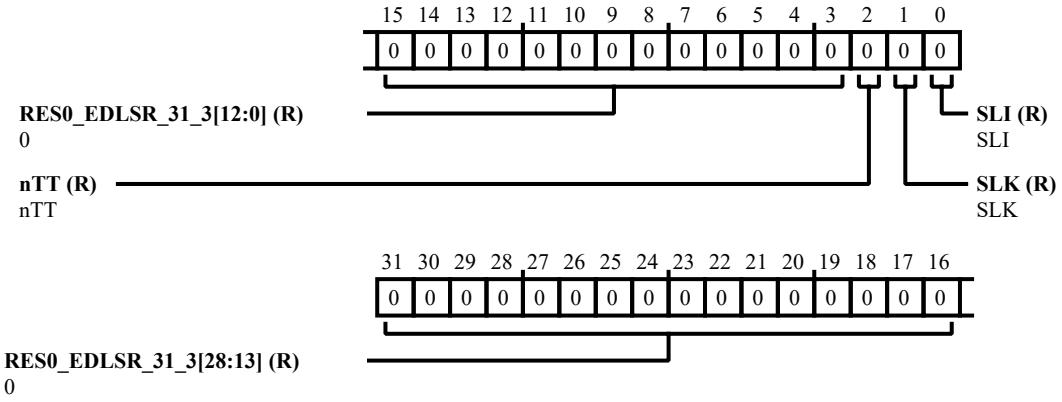


Figure 48-189: A55_CLUSTER_EDLSR Register Diagram

Table 48-187: A55_CLUSTER_EDLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_EDLSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0. For memory-mapped accesses when the Software Lock is implemented, possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

Table 48-187: A55_CLUSTER_EDLSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SLI	SLI. Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are: 0 Software Lock not implemented or not memory-mapped access. 1 Software Lock implemented and memory-mapped access.	
		0	Software Lock not implemented or not memory-mapped access.
		1	Software Lock implemented and memory-mapped access.

External Debug Processor Feature Register

Provides information about implemented PE features. For general information about the interpretation of the ID registers see .

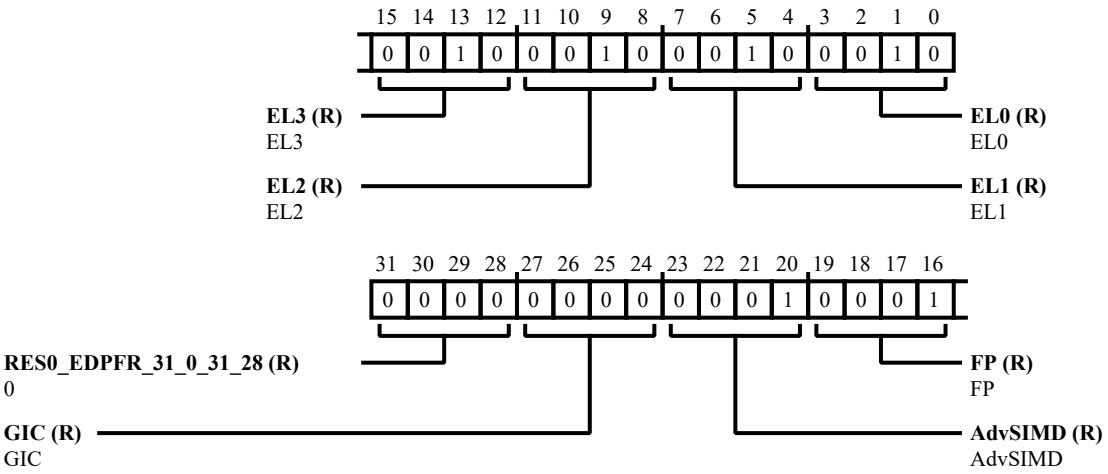


Figure 48-190: A55_CLUSTER_EDPFR_31_0 Register Diagram

Table 48-188: A55_CLUSTER_EDPFR_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_EDPFR_31_0_31_28	0. Reserved, RES0.	
27:24 (R/NW)	GIC	GIC. System register GIC interface support. Defined values are: 0000 No System register interface to the GIC is supported. 0001 System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.GIC.	
		0	No System register interface to the GIC is supported.
		1	System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported.
23:20 (R/NW)	ADVSIMD	AdvSIMD. Advanced SIMD. Defined values are: 0000 Advanced SIMD is implemented. 1111 Advanced SIMD is not implemented. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.AdvSIMD.	
		0	Advanced SIMD is implemented.
		15	Advanced SIMD is not implemented.

Table 48-188: A55_CLUSTER_EDPFR_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/NW)	FP	FP. Floating-point. Defined values are: 0000 Floating-point is implemented. 1111 Floating-point is not implemented. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.FP.	
		0	Floating-point is implemented.
		15	Floating-point is not implemented.
15:12 (R/NW)	EL3	EL3. AArch64 EL3 Exception level handling. Defined values are: 0000 EL3 is not implemented or cannot be executed in AArch64 state. 0001 EL3 can be executed in AArch64 state only. 0010 EL3 can be executed in either AArch64 or AArch32 state. When the value of EDAA32PFR.EL3 is non-zero, this field must be 0000. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL3.	
		0	EL3 is not implemented or cannot be executed in AArch64 state.
		1	EL3 can be executed in AArch64 state only.
		2	EL3 can be executed in either AArch64 or AArch32 state.
11:8 (R/NW)	EL2	EL2. AArch64 EL2 Exception level handling. Defined values are: 0000 EL2 is not implemented or cannot be executed in AArch64 state. 0001 EL2 can be executed in AArch64 state only. 0010 EL2 can be executed in either AArch64 or AArch32 state. When the value of EDAA32PFR.EL2 is non-zero, this field must be 0000. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL2.	
		0	EL2 is not implemented or cannot be executed in AArch64 state.
		1	EL2 can be executed in AArch64 state only.
		2	EL2 can be executed in either AArch64 or AArch32 state.

Table 48-188: A55_CLUSTER_EDPFR_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	EL1	EL1. AArch64 EL1 Exception level handling. Defined values are: 0000 EL1 can be executed in AArch32 state only. 0001 EL1 can be executed in AArch64 state only. 0010 EL1 can be executed in either AArch64 or AArch32 state. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL1.	
		0	EL1 can be executed in AArch32 state only.
		1	EL1 can be executed in AArch64 state only.
		2	EL1 can be executed in either AArch64 or AArch32 state.
3:0 (R/NW)	EL0	EL0. AArch64 EL0 Exception level handling. Defined values are: 0000 EL0 can be executed in AArch32 state only. 0001 EL0 can be executed in AArch64 state only. 0010 EL0 can be executed in either AArch64 or AArch32 state. All other values are reserved. In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of ID_AA64PFR0_EL1.EL0.	
		0	EL0 can be executed in AArch32 state only.
		1	EL0 can be executed in AArch64 state only.
		2	EL0 can be executed in either AArch64 or AArch32 state.

External Debug Processor Feature Register

Provides information about implemented PE features. For general information about the interpretation of the ID registers see .

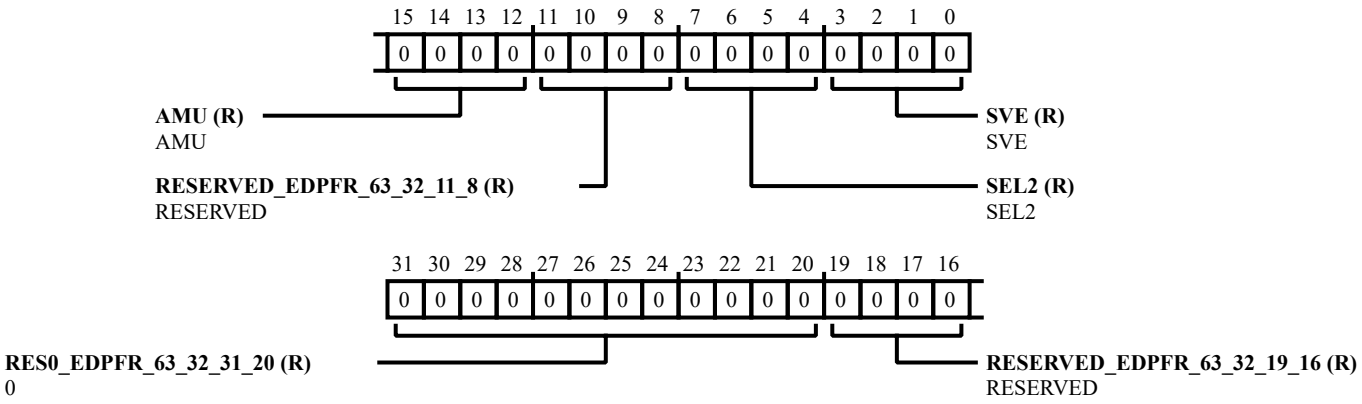


Figure 48-191: A55_CLUSTER_EDPFR_63_32 Register Diagram

Table 48-189: A55_CLUSTER_EDPFR_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:20 (R/NW)	RES0_EDPFR_63_32_31_20	0. Reserved, RES0.
19:16 (R/NW)	RESERVED_EDPFR_63_32_19_16	RESERVED. Reserved. This field reads as an UNKNOWN value.
15:12 (R/NW)	AMU	AMU. AMU.
11:8 (R/NW)	RESERVED_EDPFR_63_32_11_8	RESERVED. Reserved. This field reads as an UNKNOWN value.
7:4 (R/NW)	SEL2	SEL2. Secure EL2 implemented.
3:0 (R/NW)	SVE	SVE. SVE.

External Debug Peripheral Identification 0 Register

Provides information to identify an external debug component.

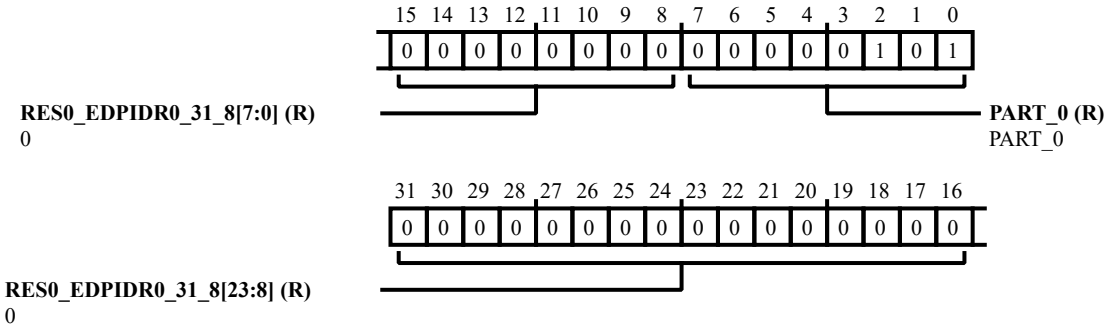


Figure 48-192: A55_CLUSTER_EDPIDR0 Register Diagram

Table 48-190: A55_CLUSTER_EDPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, least significant byte.

External Debug Peripheral Identification 1 Register

Provides information to identify an external debug component.

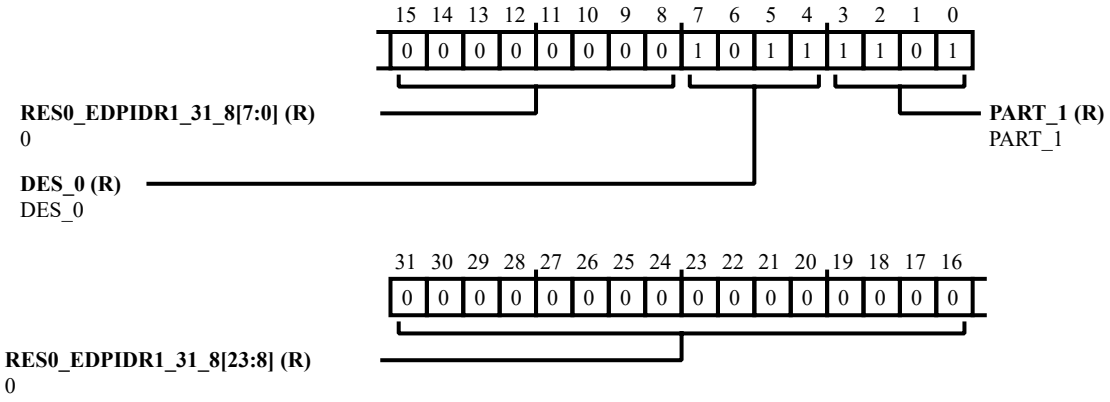


Figure 48-193: A55_CLUSTER_EDPIDR1 Register Diagram

Table 48-191: A55_CLUSTER_EDPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, most significant nibble.

External Debug Peripheral Identification 2 Register

Provides information to identify an external debug component.

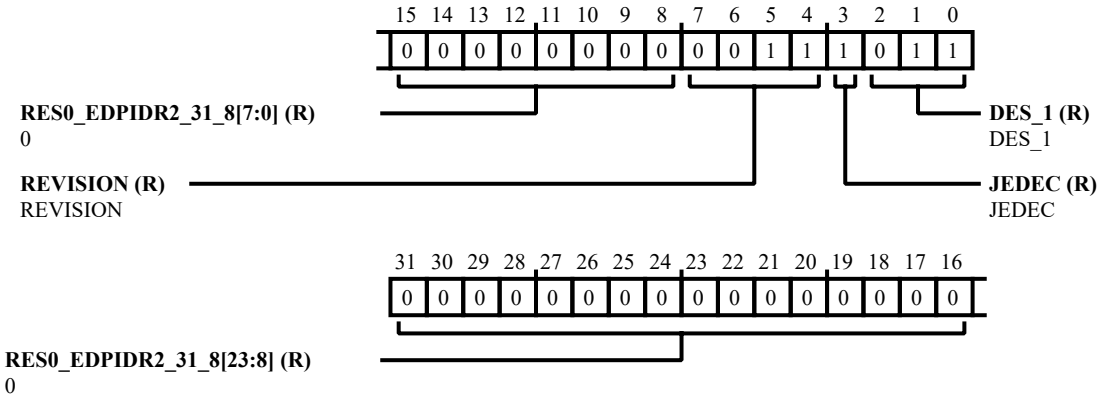


Figure 48-194: A55_CLUSTER_EDPIDR2 Register Diagram

Table 48-192: A55_CLUSTER_EDPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. Part major revision. Parts can also use this field to extend Part number to 16-bits.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

External Debug Peripheral Identification 3 Register

Provides information to identify an external debug component.

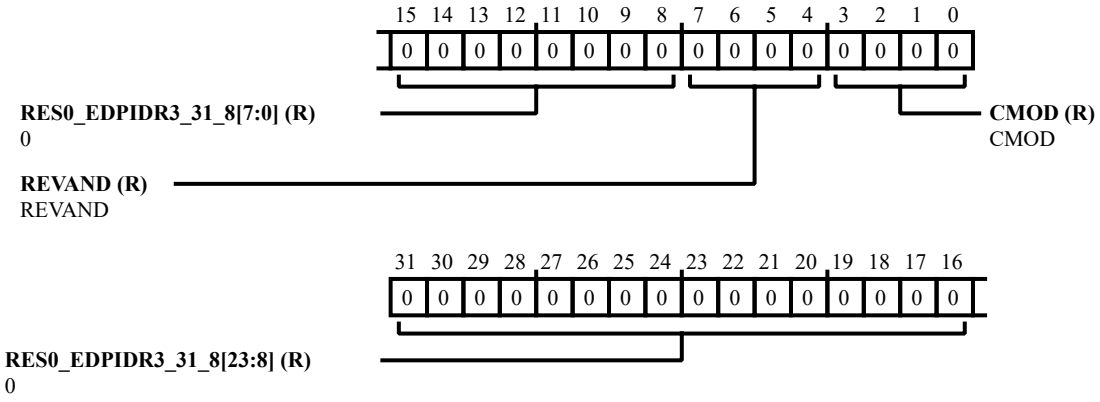


Figure 48-195: A55_CLUSTER_EDPIDR3 Register Diagram

Table 48-193: A55_CLUSTER_EDPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. Part minor revision. Parts using EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

External Debug Peripheral Identification 4 Register

Provides information to identify an external debug component.

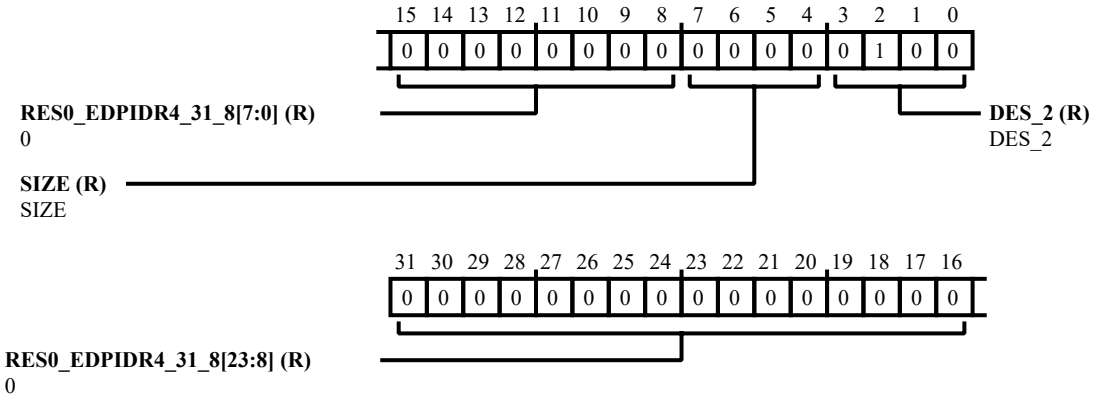


Figure 48-196: A55_CLUSTER_EDPIDR4 Register Diagram

Table 48-194: A55_CLUSTER_EDPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_EDPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

External Debug Peripheral Identification 5 Register

Provides information to identify an external debug component.

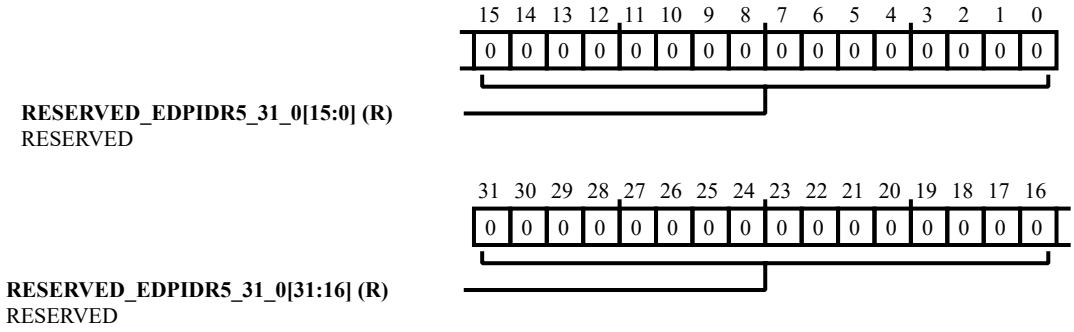


Figure 48-197: A55_CLUSTER_EDPIDR5 Register Diagram

Table 48-195: A55_CLUSTER_EDPIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_ED-PIDR5_31_0	RESERVED. Reserved RES0.

External Debug Peripheral Identification 6 Register

Provides information to identify an external debug component.

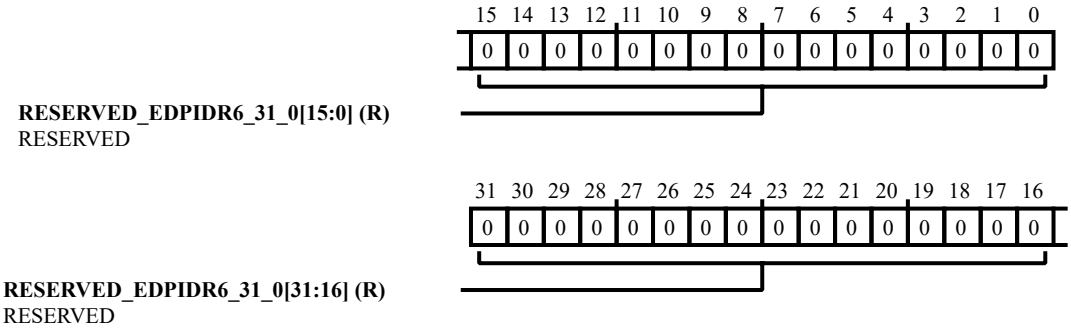


Figure 48-198: A55_CLUSTER_EDPIDR6 Register Diagram

Table 48-196: A55_CLUSTER_EDPIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_ED-PIDR6_31_0	RESERVED. Reserved RES0.

External Debug Peripheral Identification 7 Register

Provides information to identify an external debug component.

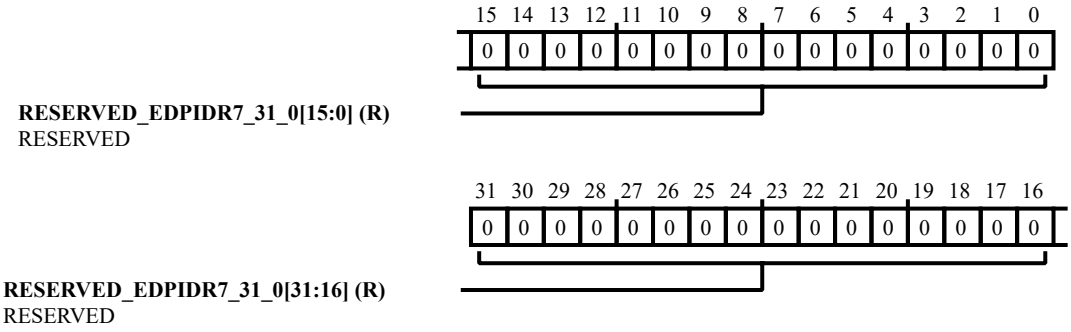


Figure 48-199: A55_CLUSTER_EDPIDR7 Register Diagram

Table 48-197: A55_CLUSTER_EDPIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_ED-PIDR7_31_0	RESERVED. Reserved RES0.

Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

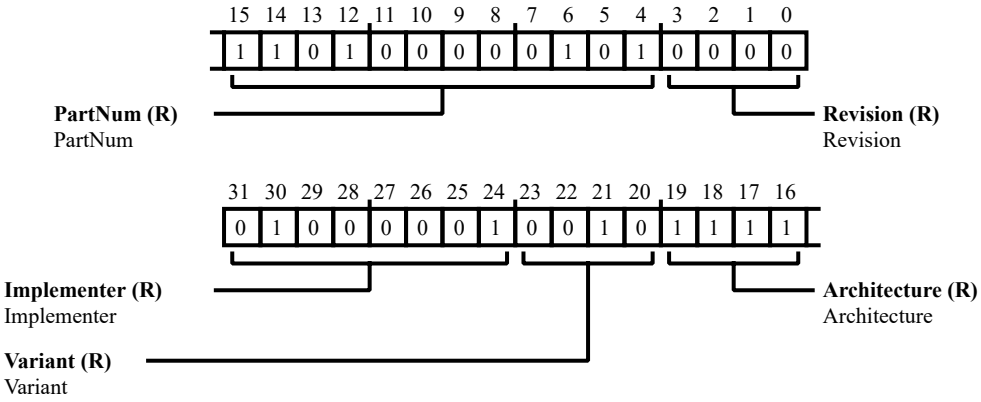


Figure 48-200: A55_CLUSTER_MIDR_EL1 Register Diagram

Table 48-198: A55_CLUSTER_MIDR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/NW)	IMPLEMENTER	Implementer. The Implementer code. This field must hold an implementer code that has been assigned by ARM.	
23:20 (R/NW)	VARIANT	Variant. An IMPLEMENTATION DEFINED variant number. Typically this field is used to distinguish between different product variants or major revisions of a product.	
19:16 (R/NW)	ARCHITECTURE	Architecture. The Architecture used.	
		1	ARMv4.
		2	ARMv4T.
		3	ARMv5 (obsolete).
		4	ARMv5T.
		5	ARMv5TE.
		6	ARMv5TEJ.
		7	ARMv6.
		15	Defined by CPUID scheme.

Table 48-198: A55_CLUSTER_MIDR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:4 (R/NW)	PARTNUM	PartNum. An IMPLEMENTATION DEFINED primary part number for the device. On processors implemented by ARM if the top four bits of the primary part number are 0x0 or 0x7 the variant and architecture are encoded differently.
3:0 (R/NW)	REVISION	Revision. An IMPLEMENTATION DEFINED revision number for the device.

Trace Authentication Status Register

Returns the level of tracing that the trace unit can support.

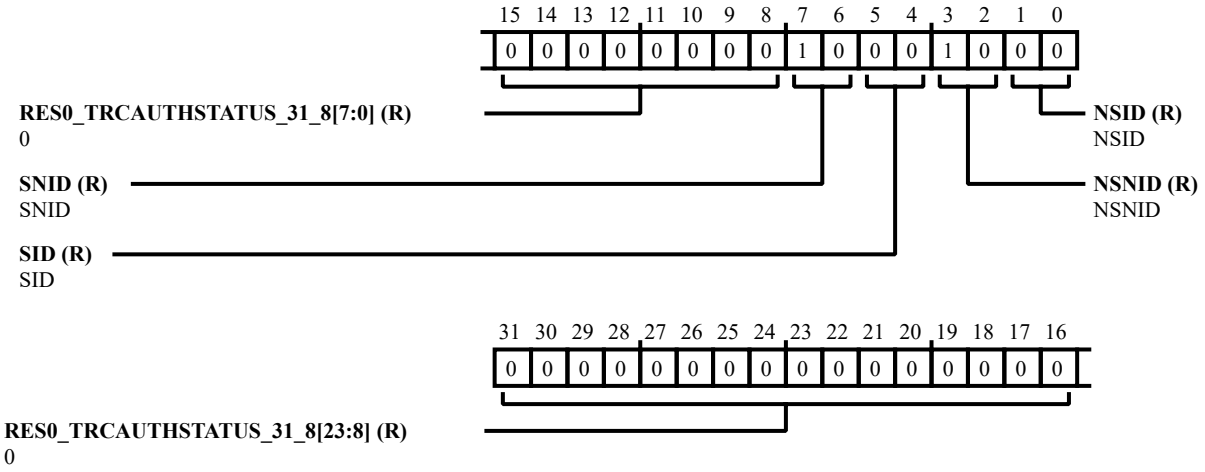


Figure 48-201: A55_CLUSTER_TRCAUTHSTATUS Register Diagram

Table 48-199: A55_CLUSTER_TRCAUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_TRCAUTHSTA-TUS_31_8	0. Reserved, RES0.	
7:6 (R/NW)	SNID	SNID. Indicates whether the system enables the trace unit to support Secure non-invasive debug: 00 The trace unit does not implement support for Secure non-invasive debug. 01 Reserved. 10 Secure non-invasive debug is disabled. 11 Secure non-invasive debug is enabled.	
		0	The trace unit does not implement support for Secure non-invasive debug.
		1	Reserved.
		2	Secure non-invasive debug is disabled.
		3	Secure non-invasive debug is enabled.
5:4 (R/NW)	SID	SID. Indicates whether the trace unit supports Secure invasive debug: 00 The trace unit does not support Secure invasive debug. All other values are reserved.	
		0	The trace unit does not support Secure invasive debug.

Table 48-199: A55_CLUSTER_TRCAUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/NW)	NSNID	NSNID. Indicates whether the system enables the trace unit to support Non-secure non-invasive debug: 00 The trace unit does not implement support for Non-secure non-invasive debug. 01 Reserved. 10 Non-secure non-invasive debug is disabled. 11 Non-secure non-invasive debug is enabled.	
		0	The trace unit does not implement support for Non-secure non-invasive debug.
		1	Reserved.
		2	Non-secure non-invasive debug is disabled.
		3	Non-secure non-invasive debug is enabled.
1:0 (R/NW)	NSID	NSID. Indicates whether the trace unit supports Non-secure invasive debug: 00 The trace unit does not support Non-secure invasive debug. All other values are reserved.	
		0	The trace unit does not support Non-secure invasive debug.

Trace Component ID 0 Register

Return byte 0 of the CoreSight preamble information.

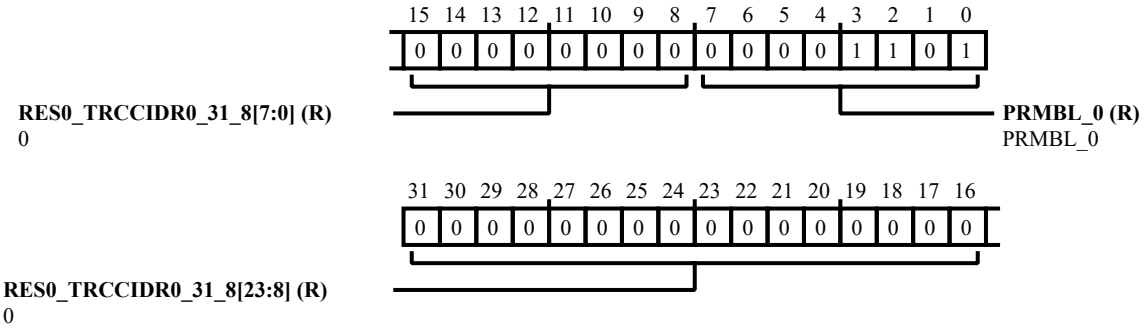


Figure 48-202: A55_CLUSTER_TRCCIDR0 Register Diagram

Table 48-200: A55_CLUSTER_TRCCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

Trace Component ID 1 Register

Return byte 1 of the CoreSight preamble information.

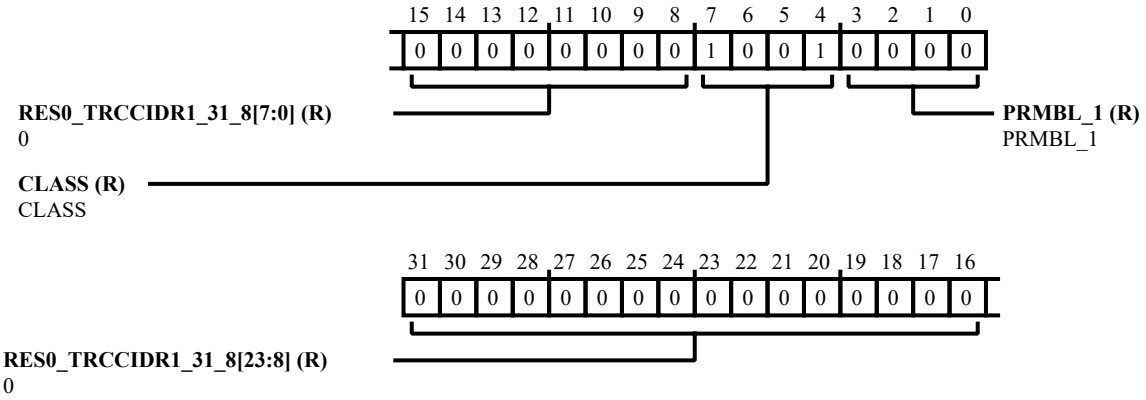


Figure 48-203: A55_CLUSTER_TRCCIDR1 Register Diagram

Table 48-201: A55_CLUSTER_TRCCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, to indicate that the ETM is a debug component, with CoreSight architecture compliant management registers.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. Must read as 0x0.

Trace Component ID 2 Register

Return byte 2 of the CoreSight preamble information.

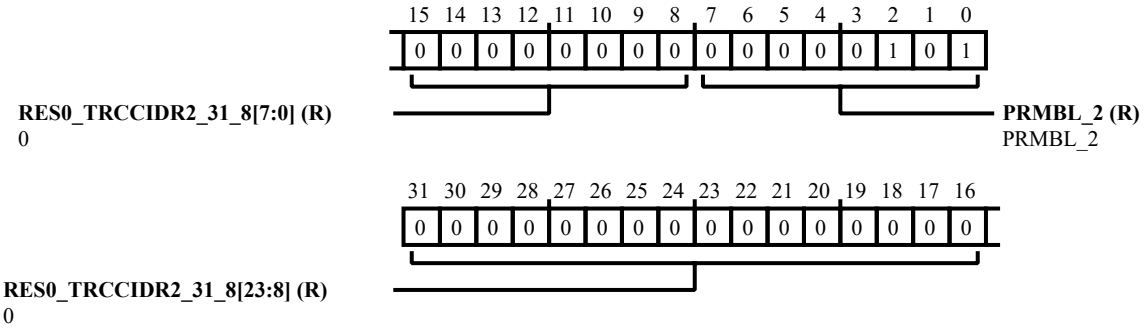


Figure 48-204: A55_CLUSTER_TRCCIDR2 Register Diagram

Table 48-202: A55_CLUSTER_TRCCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

Trace Component ID 3 Register

Return byte 3 of the CoreSight preamble information.

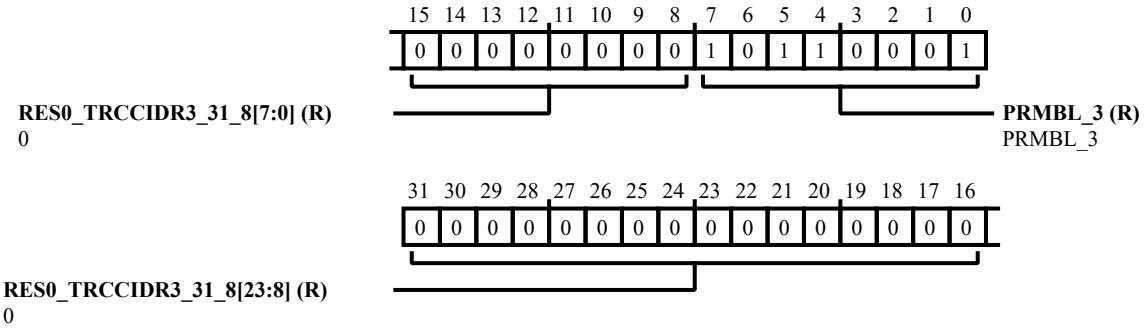


Figure 48-205: A55_CLUSTER_TRCCIDR3 Register Diagram

Table 48-203: A55_CLUSTER_TRCCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCCIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

Trace Device Affinity 0 Register

Returns the value of the lower 32 bits of MPIDR_EL1 as seen from the highest implemented exception level, unaffected by VMPIDR_EL2. If the trace unit is shared between multiple processors, the value in this register indicates the MPIDR value of the processor that TRCPROCSELR selects.

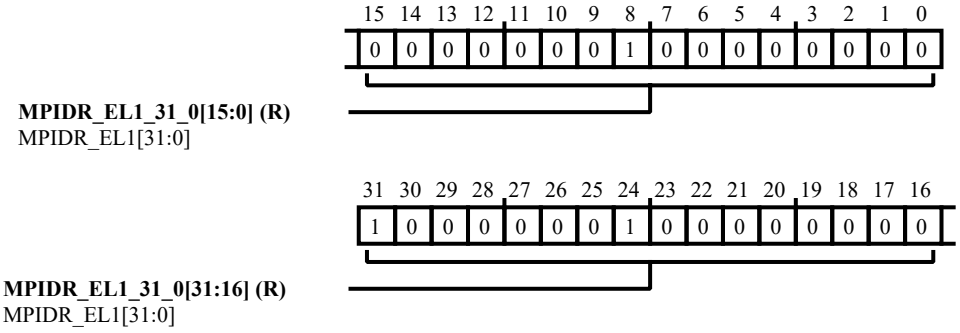


Figure 48-206: A55_CLUSTER_TRCDEVAFF0 Register Diagram

Table 48-204: A55_CLUSTER_TRCDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_31_0	MPIDR_EL1[31:0]. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented exception level.

Trace Device Affinity 1 Register

Returns the value of the upper 32 bits of MPIDR_EL1 as seen from the highest implemented exception level, unaffected by VMPIDR_EL2. If the trace unit is shared between multiple processors, the value in this register indicates the MPIDR value of the processor that TRCPROCSELR selects.

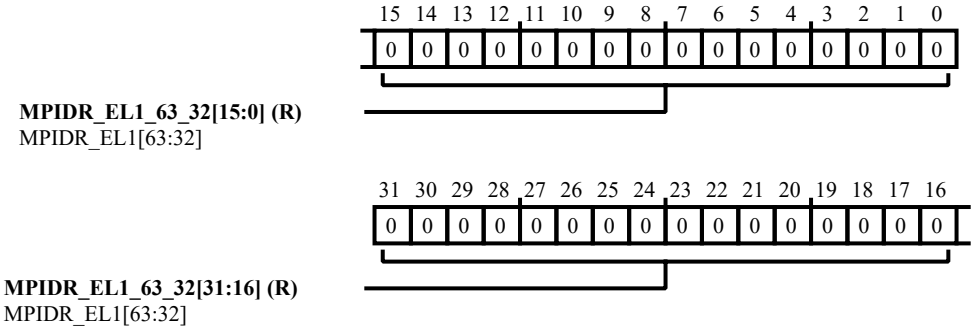


Figure 48-207: A55_CLUSTER_TRCDEVAFF1 Register Diagram

Table 48-205: A55_CLUSTER_TRCDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_63_32	MPIDR_EL1[63:32]. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented exception level.

Trace Device Architecture Register

Identifies the trace unit as an ETMv4 component. This enables debuggers to make some use of a component that conforms to the architecture definition but whose part number is not necessarily recognized.

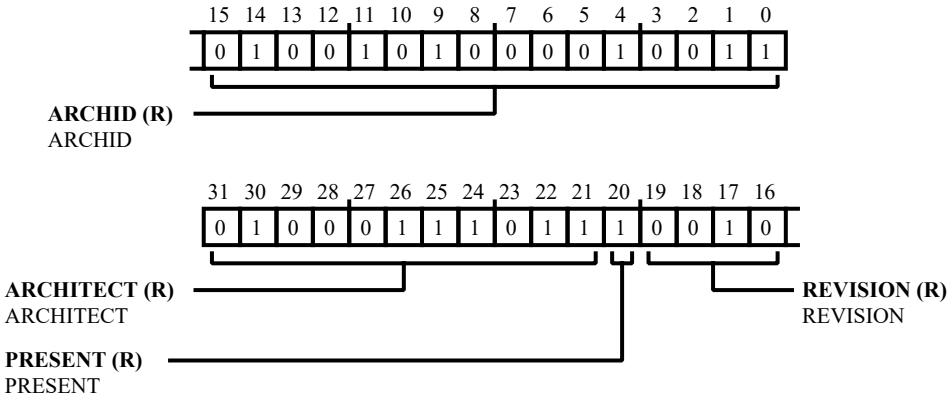


Figure 48-208: A55_CLUSTER_TRCDEVARCH Register Diagram

Table 48-206: A55_CLUSTER_TRCDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For trace, this is ARM Limited.Bits [31:28] are the JEP 106 continuation code, 0x4.Bits [27:21] are the JEP 106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present.This field is RAO.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision.For trace, the revision defined by ETMv4 is 0x0.All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be a v8-A debug component. For architectures defined by ARM this is further subdivided.For trace, bits [15:12] are the architecture version, 0x4; bits [11:0] are the architecture part number, 0xA13.This corresponds to trace architecture version ETMv4.

Trace Device ID Register

Returns an IMPLEMENTATION DEFINED CoreSight component capabilities field.

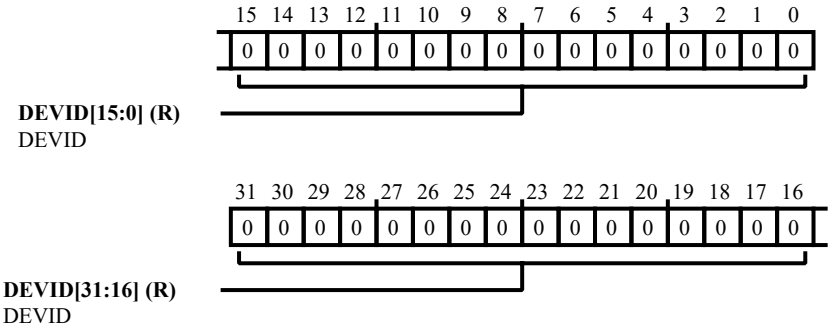


Figure 48-209: A55_CLUSTER_TRCDEVID Register Diagram

Table 48-207: A55_CLUSTER_TRCDEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVID	DEVID. Indicates the capabilities of the trace unit. The implemented width of this field and its bit assignments are IMPLEMENTATION DEFINED. Unimplemented bits are RAZ/WI.If a component is configurable then ARM recommends that this field can also indicate which configuration options are implemented that differ from the standard configuration.

Trace Device Type Register

Indicates what type of CoreSight device the trace unit is.

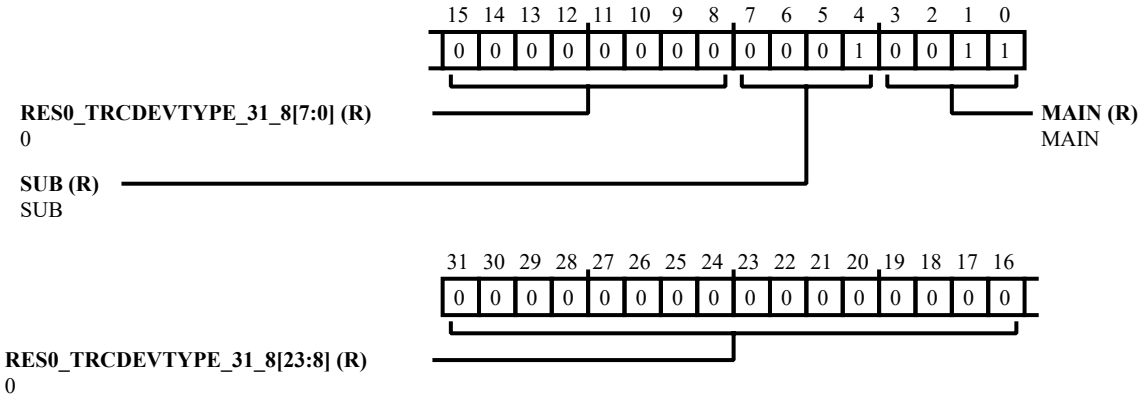


Figure 48-210: A55_CLUSTER_TRCDEVTYPE Register Diagram

Table 48-208: A55_CLUSTER_TRCDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCDEV-TYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Returns 0x1, to indicate that the ETM generates processor trace.All other values are reserved.
3:0 (R/NW)	MAIN	MAIN. Returns 0x3, to indicate that the ETM is a trace source.All other values are reserved.

Trace Software Lock Access Register

Controls whether the Software Lock is locked. When the Software Lock is locked, the trace unit ignores write accesses from the memory-mapped interface to all ETM registers other than the TRCLAR, and does not change the TRCPDSR.STICKYPD bit if a read access occurs to the TRCPDSR on the memory-mapped interface.

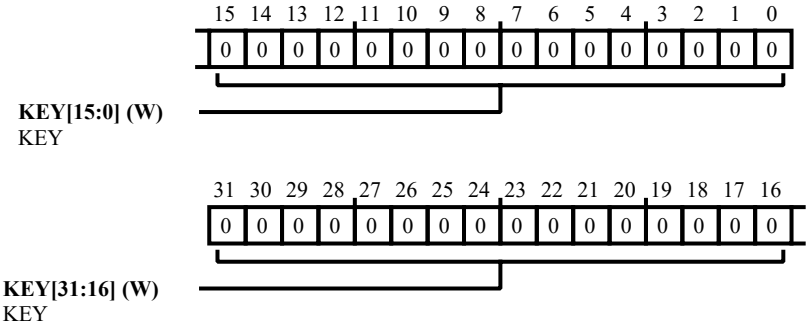


Figure 48-211: A55_CLUSTER_TRCLAR Register Diagram

Table 48-209: A55_CLUSTER_TRCLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Writing the key value 0xC5ACCE55 to this field clears the lock, enabling write accesses to this component's registers through a memory-mapped interface.Writing any other value to this register sets the lock, disabling write accesses to this component's registers through a memory mapped interface.Software can use the Software Lock to prevent accidental modification of the trace unit registers by software being debugged. For example, software that accidentally initializes an incorrect region of memory might disable the trace unit and make it impossible to trace the software.

Trace Software Lock Status Register

Trace Software Lock Status Register.

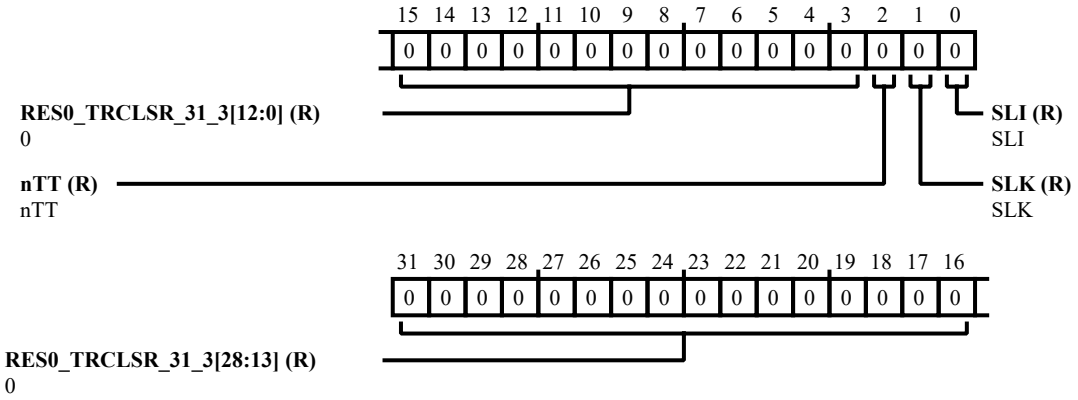


Figure 48-212: A55_CLUSTER_TRCLSR Register Diagram

Table 48-210: A55_CLUSTER_TRCLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_TRCLSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software lock status for this component. Possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.
0 (R/NW)	SLI	SLI. Software lock implemented. RAO.	

Trace PowerDown Control Register

Requests the system to provide power to the trace unit.

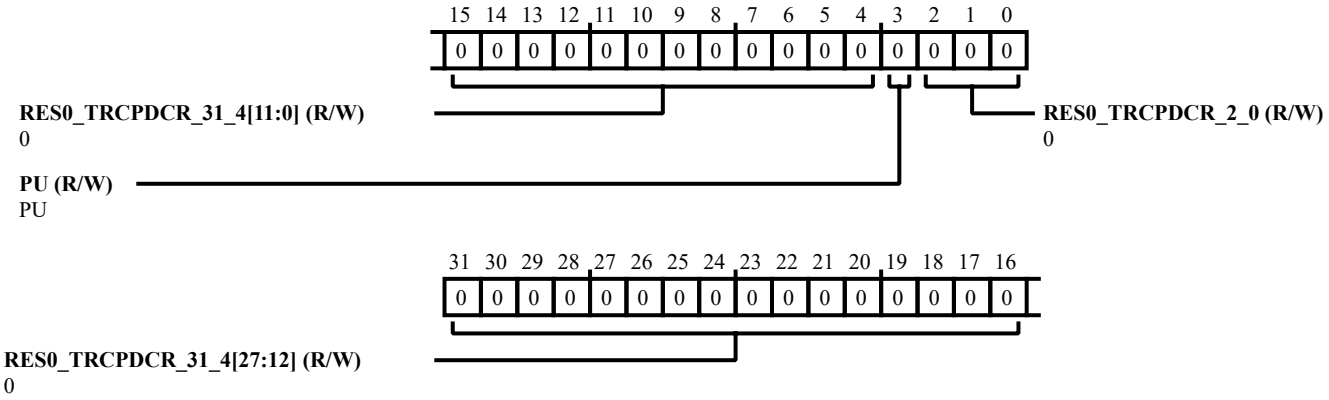


Figure 48-213: A55_CLUSTER_TRCPDCR Register Diagram

Table 48-211: A55_CLUSTER_TRCPDCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RES0_TRCPDCR_31_4	0. Reserved, RES0.	
3 (R/W)	PU	PU. Powerup request bit: 0 The system can remove power from the trace unit. The TRCPDSR indicates if the trace unit is powered down. 1 The system must provide power to the trace unit. Typically, a trace unit drives a signal representing the value of this bit to a power controller to request that the trace unit core power domain is powered up. However, if the trace unit and the processor are in the same power domain then the implementation might combine the PU status with a signal from the processor.	
		0	The system can remove power from the trace unit. The TRCPDSR indicates if the trace unit is powered down.
		1	The system must provide power to the trace unit.
2:0 (R/W)	RES0_TRCPDCR_2_0	0. Reserved, RES0.	

Trace Peripheral ID 0 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

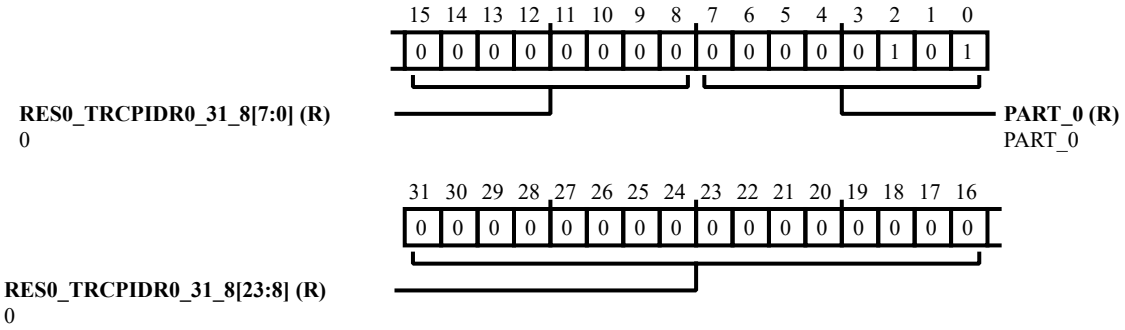


Figure 48-214: A55_CLUSTER_TRCPIDR0 Register Diagram

Table 48-212: A55_CLUSTER_TRCPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, bits[7:0].

Trace Peripheral ID 1 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

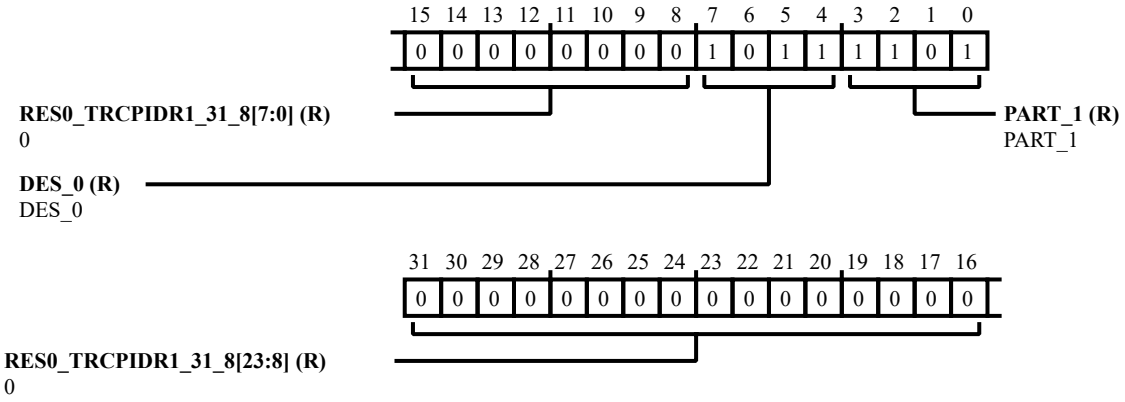


Figure 48-215: A55_CLUSTER_TRCPIDR1 Register Diagram

Table 48-213: A55_CLUSTER_TRCPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, bits[3:0] of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, bits[11:8].

Trace Peripheral ID 2 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

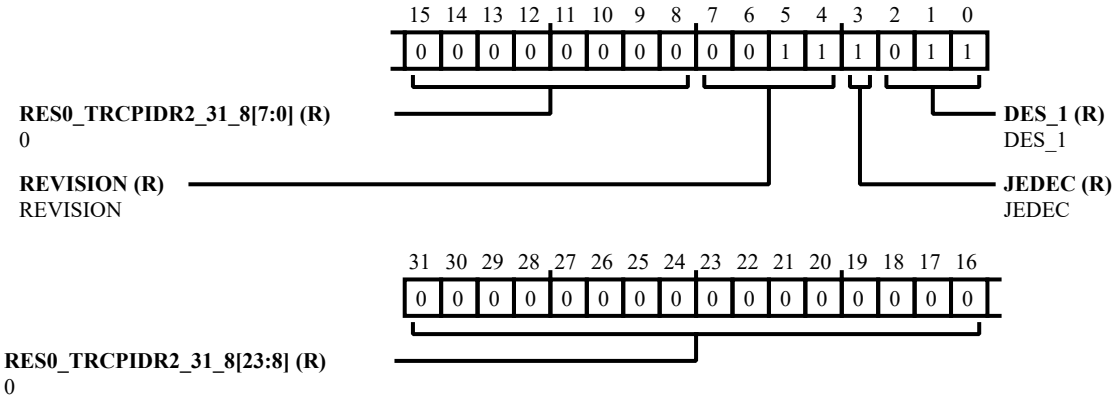


Figure 48-216: A55_CLUSTER_TRCPIDR2 Register Diagram

Table 48-214: A55_CLUSTER_TRCPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. The IMPLEMENTATION DEFINED revision number for the ETM implementation. See also TRCIDR1.REVISION.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

Trace Peripheral ID 3 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

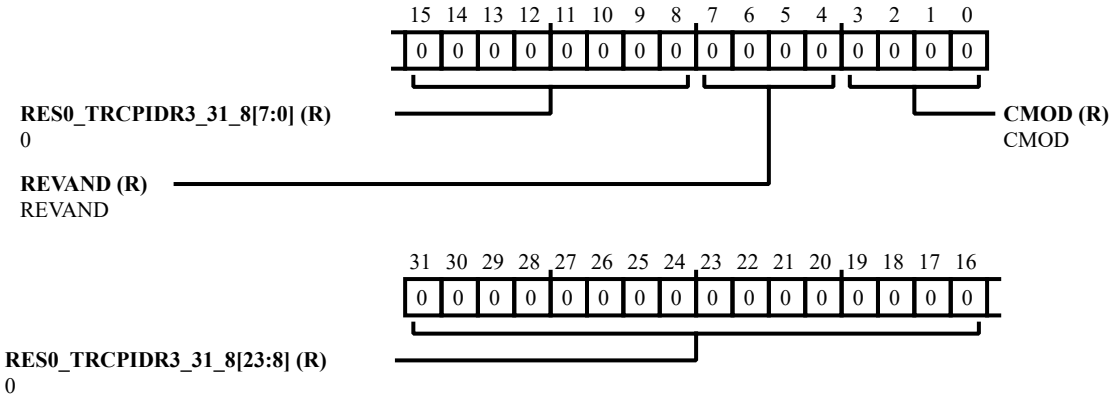


Figure 48-217: A55_CLUSTER_TRCPIDR3 Register Diagram

Table 48-215: A55_CLUSTER_TRCPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. The IMPLEMENTATION DEFINED manufacturing revision number for the implementation. After silicon is available, if metal fixes are necessary, the manufacturer can alter the top metal layer so that this field can indicate any post-fab silicon changes.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

Trace Peripheral ID 4 Register

Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the ETM.

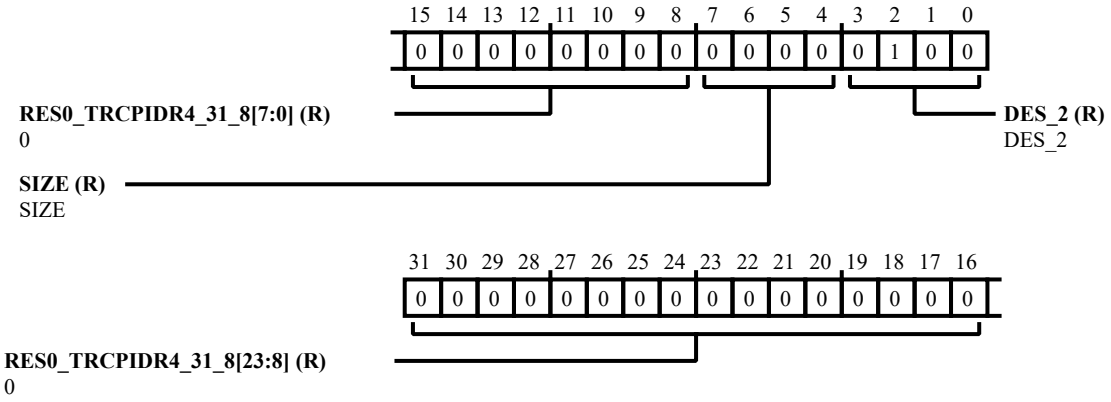


Figure 48-218: A55_CLUSTER_TRCPIDR4 Register Diagram

Table 48-216: A55_CLUSTER_TRCPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_TRCPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RES0. This indicates that the ETM memory map occupies 4KB.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code. For ARM Limited, this field is 0b0100.

Performance Monitors Authentication Status Register

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for Performance Monitors.

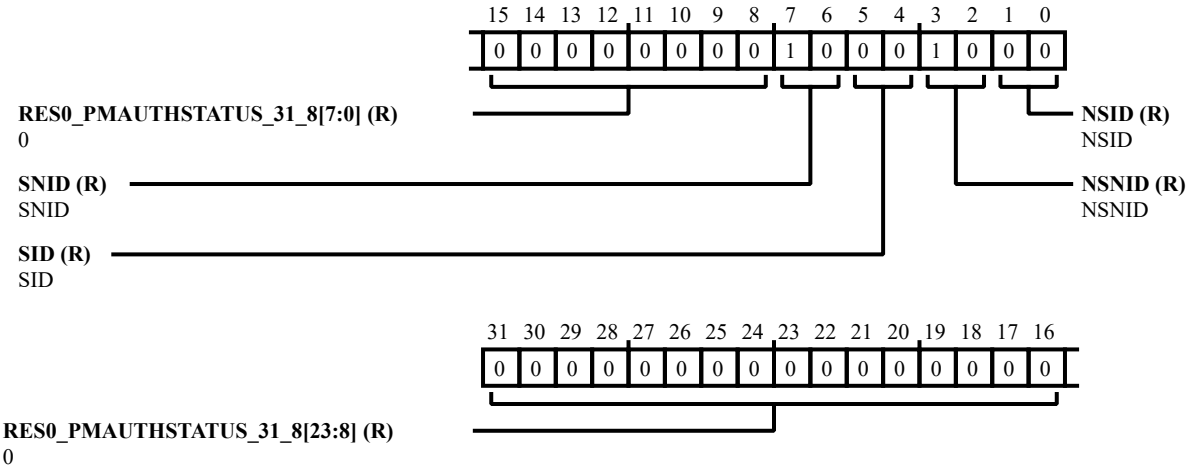


Figure 48-219: A55_CLUSTER_PMAUTHSTATUS Register Diagram

Table 48-217: A55_CLUSTER_PMAUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_PMAUTHSTATUS_31_8	0. Reserved, RES0.	
7:6 (R/NW)	SNID	SNID. Holds the same value as DBGAUTHSTATUS_EL1.SNID.	
5:4 (R/NW)	SID	SID. Secure invasive debug. Possible values of this field are: 00 Not implemented. All other values are reserved.	
		0	Not implemented.
3:2 (R/NW)	NSNID	NSNID. Holds the same value as DBGAUTHSTATUS_EL1.NSNID.	
1:0 (R/NW)	NSID	NSID. Non-secure invasive debug. Possible values of this field are: 00 Not implemented. All other values are reserved.	
		0	Not implemented.

Performance Monitors Component Identification 0 Register

Provides information to identify a Performance Monitor component.

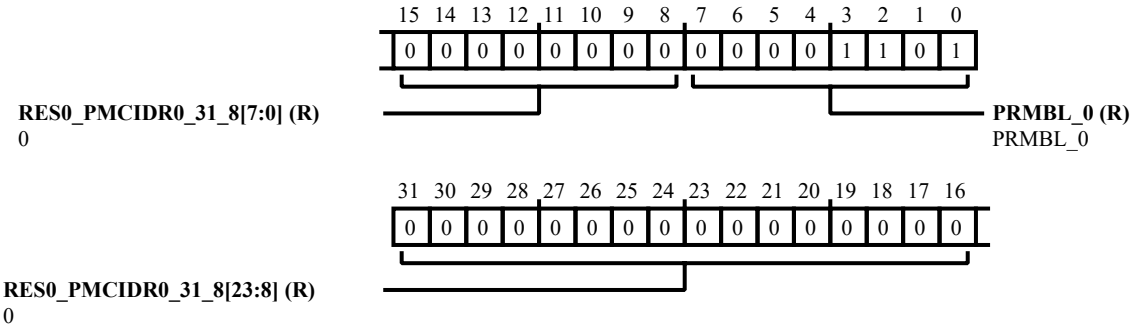


Figure 48-220: A55_CLUSTER_PMCIDR0 Register Diagram

Table 48-218: A55_CLUSTER_PMCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_0	PRMBL_0. Preamble. Must read as 0x0D.

Performance Monitors Component Identification 1 Register

Provides information to identify a Performance Monitor component.

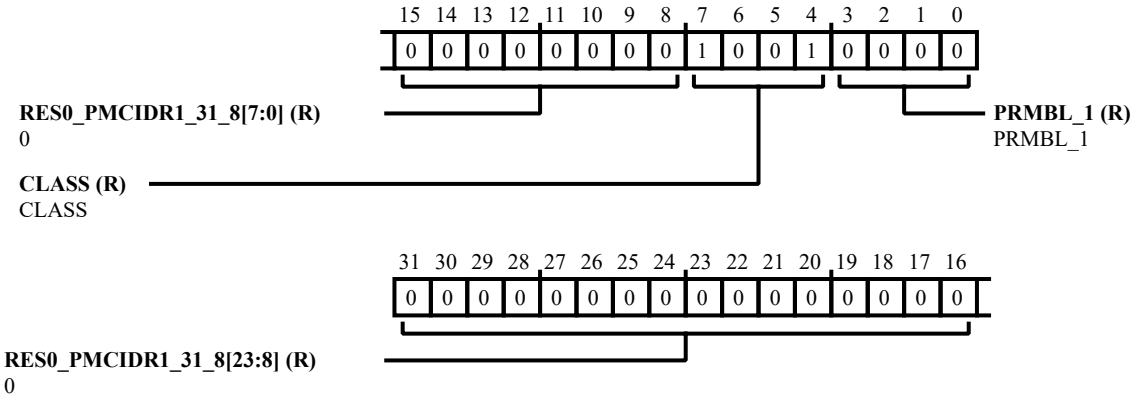


Figure 48-221: A55_CLUSTER_PMCIDR1 Register Diagram

Table 48-219: A55_CLUSTER_PMCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	CLASS	CLASS. Component class. Reads as 0x9, debug component.
3:0 (R/NW)	PRMBL_1	PRMBL_1. Preamble. RAZ.

Performance Monitors Component Identification 2 Register

Provides information to identify a Performance Monitor component.

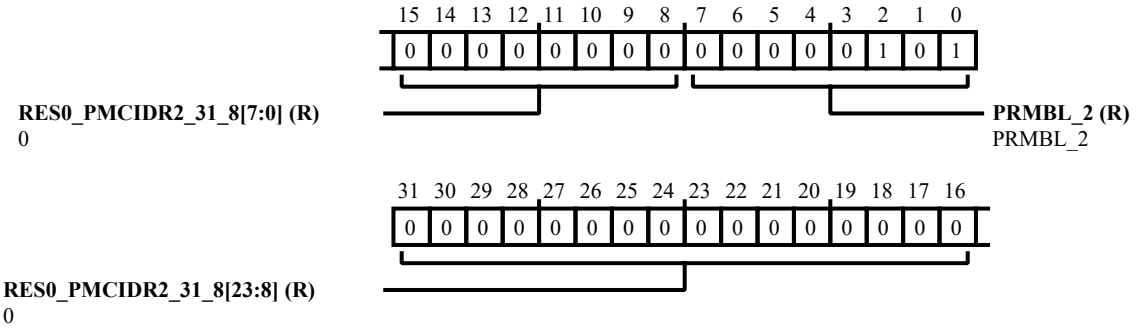


Figure 48-222: A55_CLUSTER_PMCIDR2 Register Diagram

Table 48-220: A55_CLUSTER_PMCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR2_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_2	PRMBL_2. Preamble. Must read as 0x05.

Performance Monitors Component Identification 3 Register

Provides information to identify a Performance Monitor component.

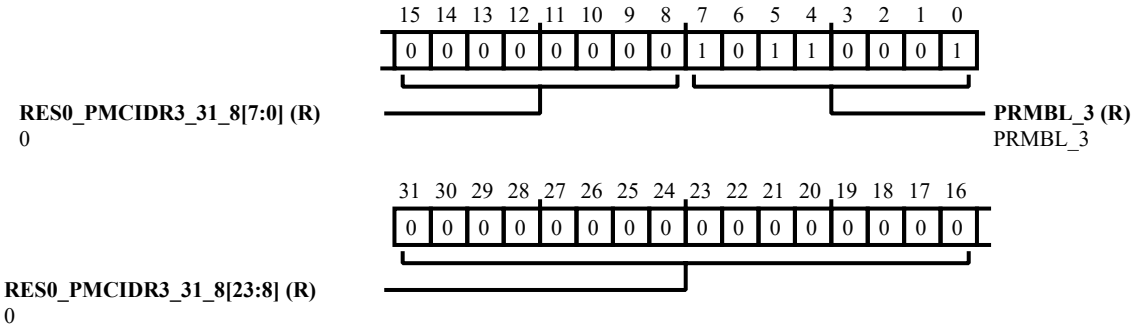


Figure 48-223: A55_CLUSTER_PMCIDR3 Register Diagram

Table 48-221: A55_CLUSTER_PMCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMCIDR3_31_8	0. Reserved, RES0.
7:0 (R/NW)	PRMBL_3	PRMBL_3. Preamble. Must read as 0xB1.

Performance Monitors Device Affinity 0 Register

Copy of the low half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

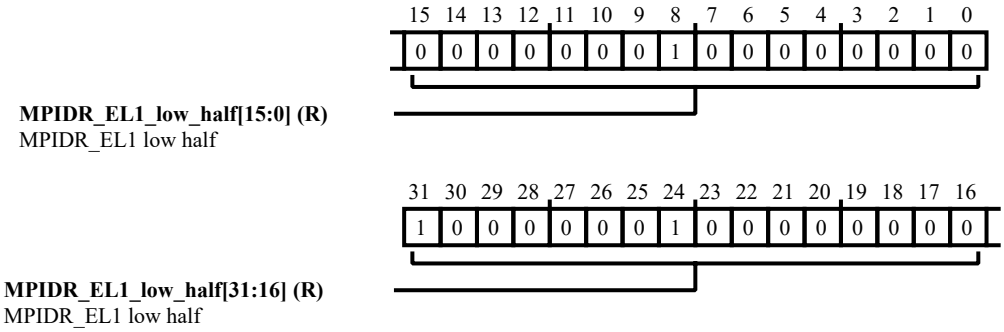


Figure 48-224: A55_CLUSTER_PMDEVAFF0 Register Diagram

Table 48-222: A55_CLUSTER_PMDEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_LOW_HALF	MPIDR_EL1 low half. MPIDR_EL1 low half. Read-only copy of the low half of MPIDR_EL1, as seen from the highest implemented Exception level.

Performance Monitors Device Affinity 1 Register

Copy of the high half of the PE MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

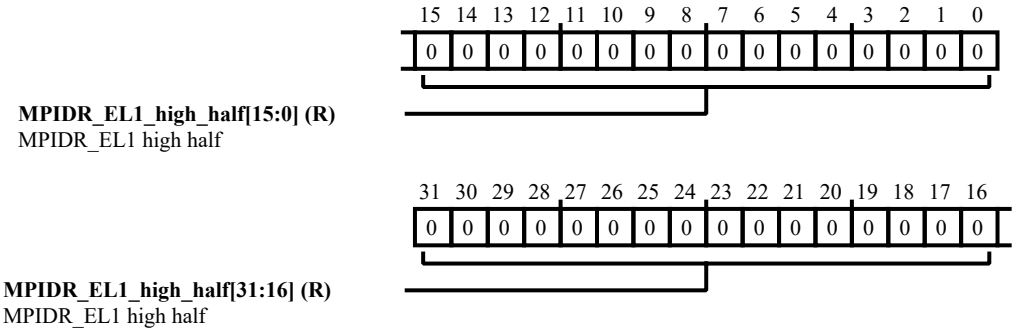


Figure 48-225: A55_CLUSTER_PMDEVAFF1 Register Diagram

Table 48-223: A55_CLUSTER_PMDEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPIDR_EL1_HIGH_HALF	MPIDR_EL1 high half. MPIDR_EL1 high half. Read-only copy of the high half of MPIDR_EL1, as seen from the highest implemented Exception level.

Performance Monitors Device Architecture Register

Identifies the programmers' model architecture of the Performance Monitor component.

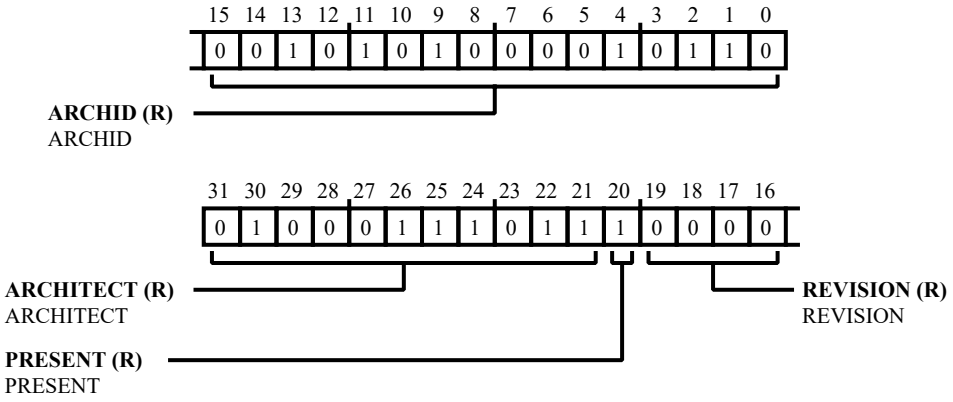


Figure 48-226: A55_CLUSTER_PMDEVARCH Register Diagram

Table 48-224: A55_CLUSTER_PMDEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	ARCHITECT	ARCHITECT. Defines the architecture of the component. For Performance Monitors, this is ARM Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.
20 (R/NW)	PRESENT	PRESENT. When set to 1, indicates that the DEVARCH is present. This field is 1 in ARMv8.
19:16 (R/NW)	REVISION	REVISION. Defines the architecture revision. For architectures defined by ARM this is the minor revision. For Performance Monitors, the revision defined by ARMv8 is 0x0. All other values are reserved.
15:0 (R/NW)	ARCHID	ARCHID. Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided. For Performance Monitors: Bits [15:12] are the architecture version, 0x2. Bits [11:0] are the architecture part number, 0xA16. This corresponds to Performance Monitors architecture version PMUv3. The PMUv3 memory-mapped programmers' model can be used by devices other than ARMv8 processors. Software must determine whether the PMU is attached to an ARMv8 processor by using the PMDEVAFF0 and PMDEVAFF1 registers to discover the affinity of the PMU to any ARMv8 processors.

Performance Monitors Device Type Register

Indicates to a debugger that this component is part of a PEs performance monitor interface.

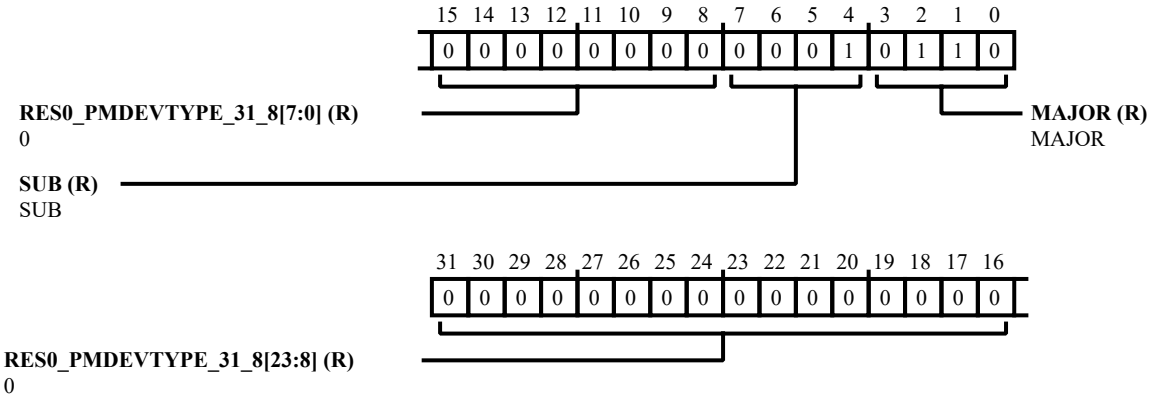


Figure 48-227: A55_CLUSTER_PMDEVTYPE Register Diagram

Table 48-225: A55_CLUSTER_PMDEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMDEVTYPE_31_8	0. Reserved, RES0.
7:4 (R/NW)	SUB	SUB. Subtype. Must read as 0x1 to indicate this is a component within a PE.
3:0 (R/NW)	MAJOR	MAJOR. Major type. Must read as 0x6 to indicate this is a performance monitor component.

Performance Monitors Lock Access Register

Allows or disallows access to the Performance Monitors registers through a memory-mapped interface.

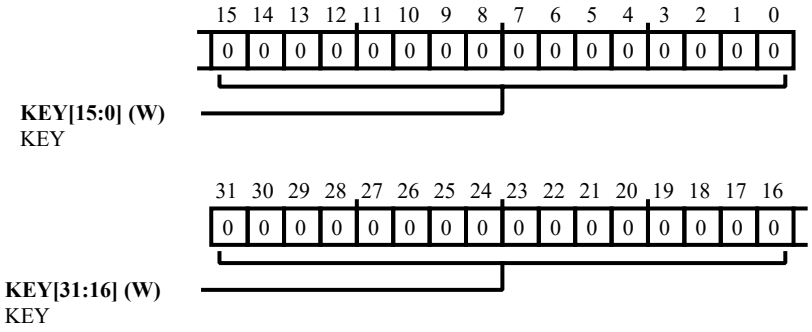


Figure 48-228: A55_CLUSTER_PMLAR Register Diagram

Table 48-226: A55_CLUSTER_PMLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	KEY	KEY. Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

Performance Monitors Lock Status Register

Indicates the current status of the software lock for Performance Monitors registers.

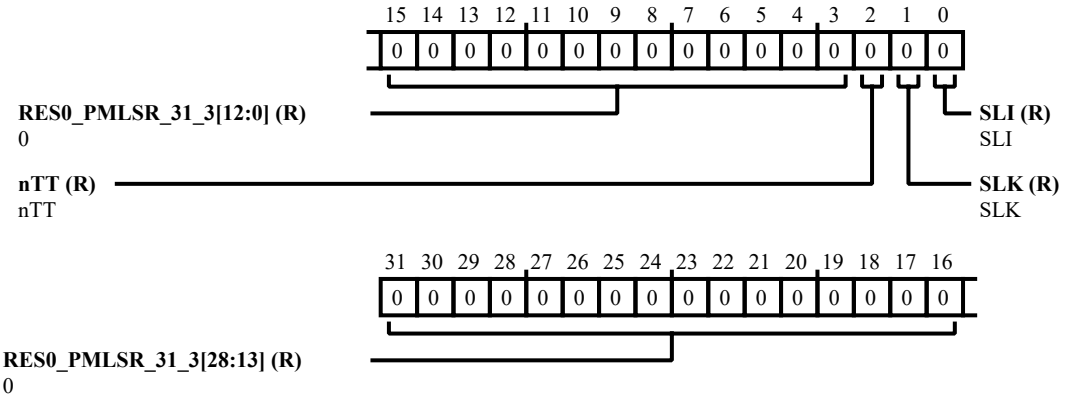


Figure 48-229: A55_CLUSTER_PMLSR Register Diagram

Table 48-227: A55_CLUSTER_PMLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_PMLSR_31_3	0. Reserved, RES0.	
2 (R/NW)	NTT	nTT. Not thirty-two bit access required. RAZ.	
1 (R/NW)	SLK	SLK. Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0. For memory-mapped accesses when the software lock is implemented, possible values of this field are: 0 Lock clear. Writes are permitted to this component's registers. 1 Lock set. Writes to this component's registers are ignored, and reads have no side effects.	
		0	Lock clear. Writes are permitted to this component's registers.
		1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

Table 48-227: A55_CLUSTER_PMLSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SLI	SLI. Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are: 0 Software Lock not implemented or not memory-mapped access. 1 Software Lock implemented and memory-mapped access.	
		0	Software Lock not implemented or not memory-mapped access.
		1	Software Lock implemented and memory-mapped access.

Performance Monitors Peripheral Identification 0 Register

Provides information to identify a Performance Monitor component.

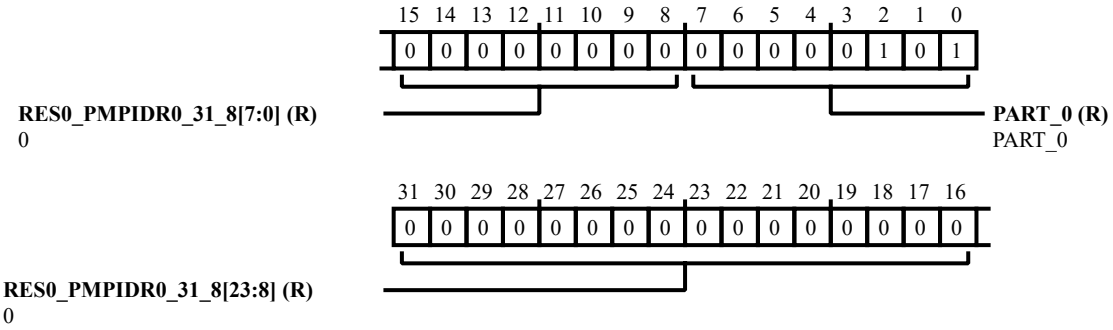


Figure 48-230: A55_CLUSTER_PMPIDR0 Register Diagram

Table 48-228: A55_CLUSTER_PMPIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR0_31_8	0. Reserved, RES0.
7:0 (R/NW)	PART_0	PART_0. Part number, least significant byte.

Performance Monitors Peripheral Identification 1 Register

Provides information to identify a Performance Monitor component.

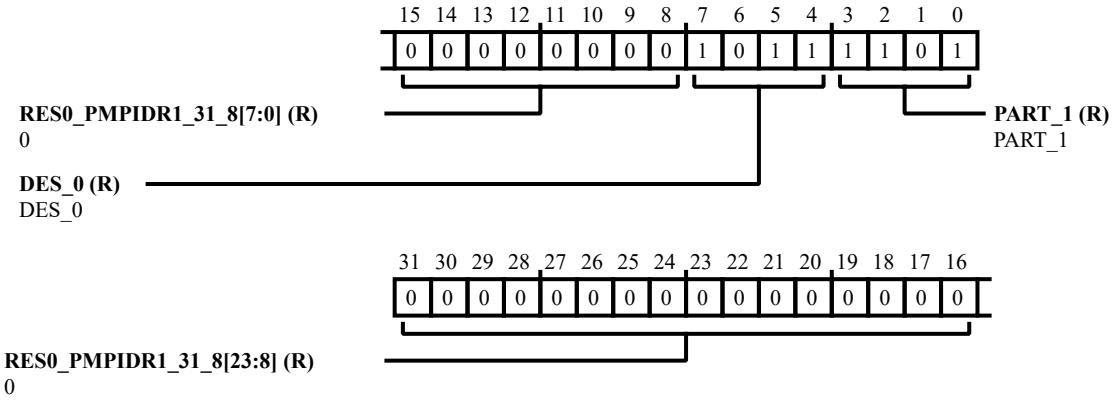


Figure 48-231: A55_CLUSTER_PMPIDR1 Register Diagram

Table 48-229: A55_CLUSTER_PMPIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR1_31_8	0. Reserved, RES0.
7:4 (R/NW)	DES_0	DES_0. Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.
3:0 (R/NW)	PART_1	PART_1. Part number, most significant nibble.

Performance Monitors Peripheral Identification 2 Register

Provides information to identify a Performance Monitor component.

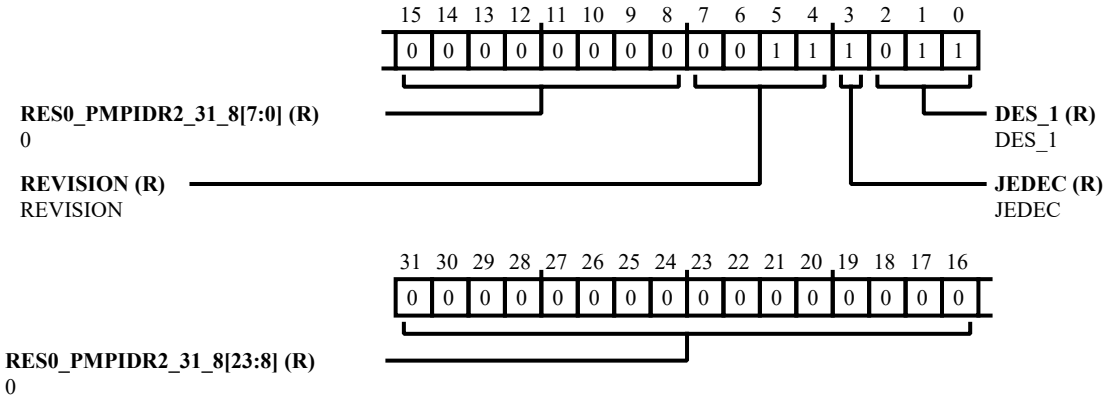


Figure 48-232: A55_CLUSTER_PMPIDR2 Register Diagram

Table 48-230: A55_CLUSTER_PMPIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR2_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVISION	REVISION. Part major revision. Parts can also use this field to extend Part number to 16-bits.
3 (R/NW)	JEDEC	JEDEC. RAO. Indicates a JEP106 identity code is used.
2:0 (R/NW)	DES_1	DES_1. Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

Performance Monitors Peripheral Identification 3 Register

Provides information to identify a Performance Monitor component.

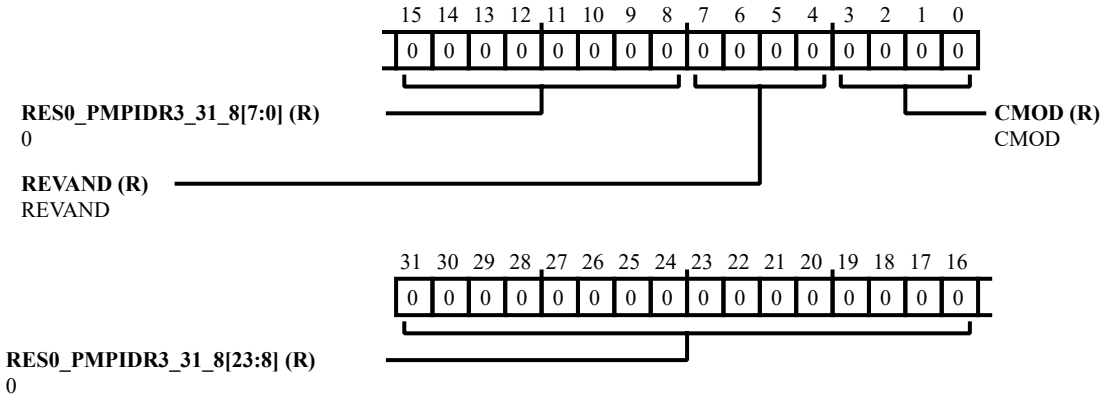


Figure 48-233: A55_CLUSTER_PMPIDR3 Register Diagram

Table 48-231: A55_CLUSTER_PMPIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR3_31_8	0. Reserved, RES0.
7:4 (R/NW)	REVAND	REVAND. Part minor revision. Parts using PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.
3:0 (R/NW)	CMOD	CMOD. Customer modified. Indicates someone other than the Designer has modified the component.

Performance Monitors Peripheral Identification 4 Register

Provides information to identify a Performance Monitor component.

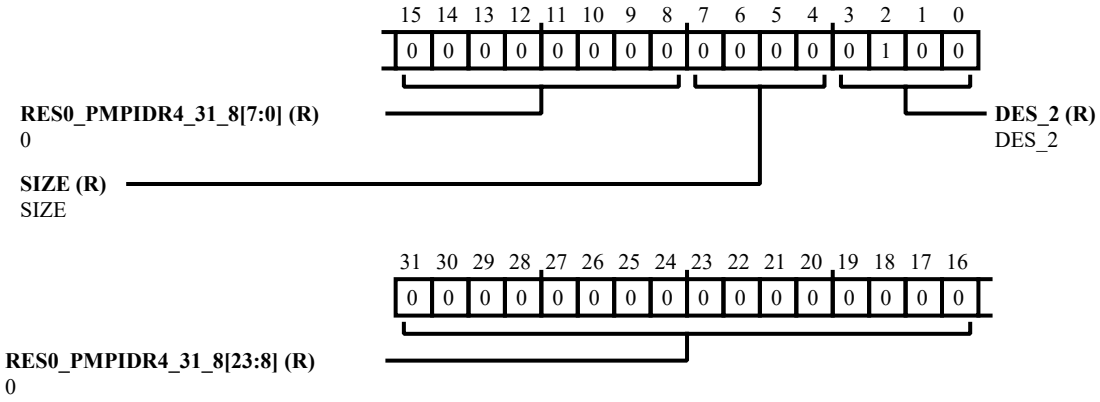


Figure 48-234: A55_CLUSTER_PMPIDR4 Register Diagram

Table 48-232: A55_CLUSTER_PMPIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_PMPIDR4_31_8	0. Reserved, RES0.
7:4 (R/NW)	SIZE	SIZE. Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
3:0 (R/NW)	DES_2	DES_2. Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

Performance Monitors Peripheral Identification 5 Register

Provides information to identify a Performance Monitor component.

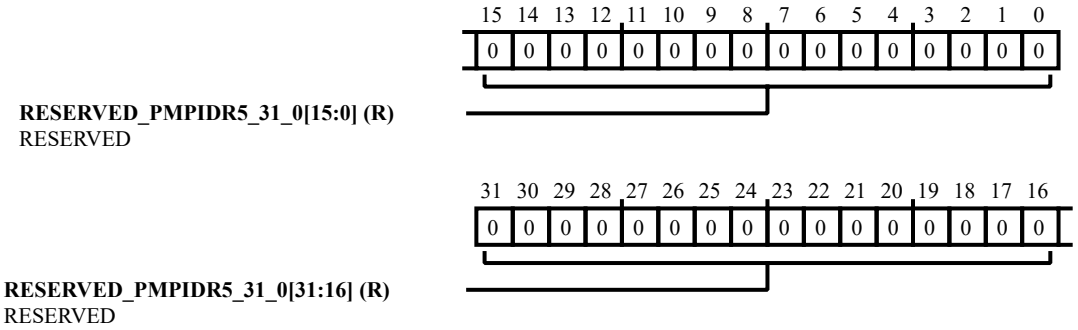


Figure 48-235: A55_CLUSTER_PMPIDR5 Register Diagram

Table 48-233: A55_CLUSTER_PMPIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_PMPIDR5_31_0	RESERVED. Reserved RES0.

Performance Monitors Peripheral Identification 6 Register

Provides information to identify a Performance Monitor component.

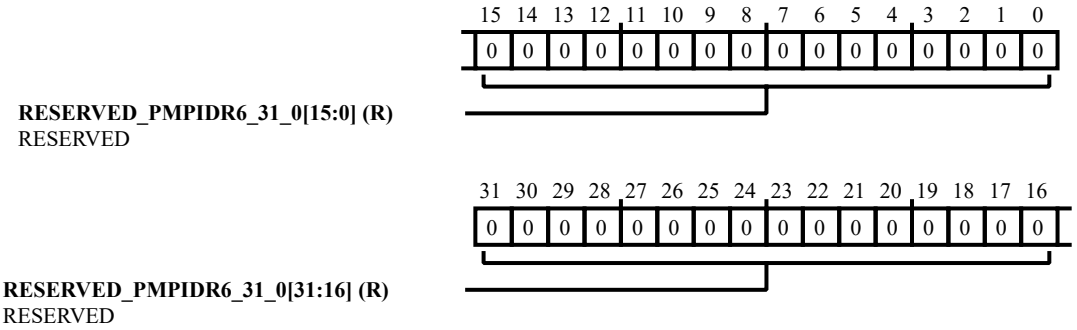


Figure 48-236: A55_CLUSTER_PMPIDR6 Register Diagram

Table 48-234: A55_CLUSTER_PMPIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_PMPIDR6_31_0	RESERVED. Reserved RES0.

Performance Monitors Peripheral Identification 7 Register

Provides information to identify a Performance Monitor component.

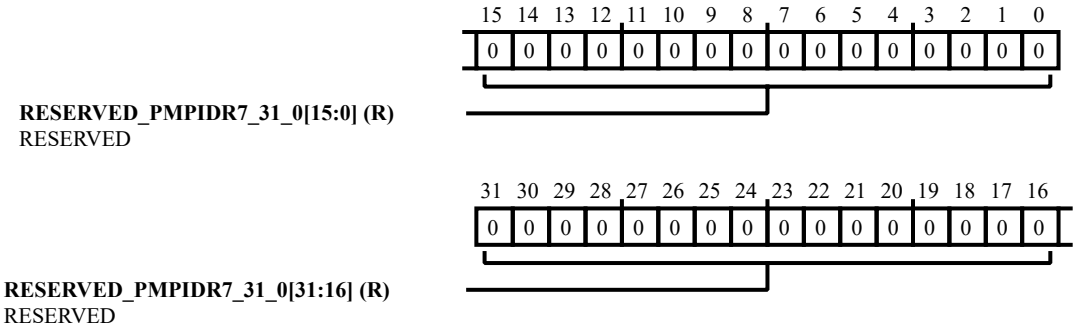


Figure 48-237: A55_CLUSTER_PMPIDR7 Register Diagram

Table 48-235: A55_CLUSTER_PMPIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RESERVED_PMPIDR7_31_0	RESERVED. Reserved RES0.

Authentication Status Register

AUTHSTATUS reports the required security level and status of the authentication interface. Where functionality changes on a given security level, the change in status must be reported in this register. The effect of each debug level being enabled or disabled is specific to each AP.

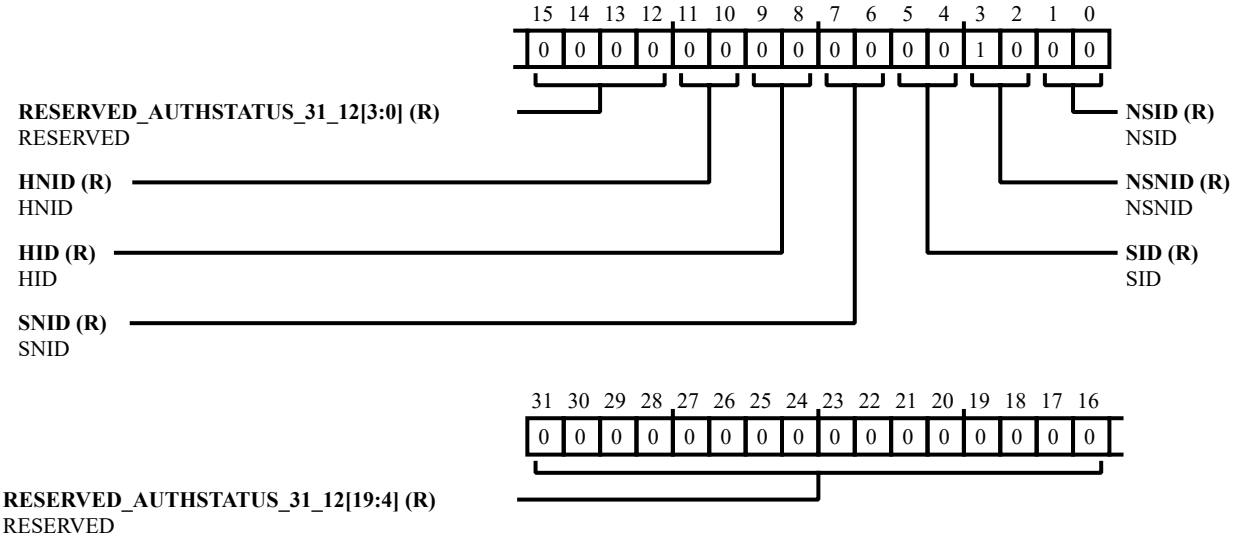


Figure 48-238: A55_CLUSTER_AUTHSTATUS Register Diagram

Table 48-236: A55_CLUSTER_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RESERVED_AUTHSTATUS_31_12	RESERVED. Reserved RES0.

Table 48-236: A55_CLUSTER_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/NW)	HNID	HNID. Configuration of hypervisor noninvasive debug.	
		0	Separate controls for hypervisor noninvasive debug are not implemented, or no hypervisor is implemented. For ARMv7 PEs that implement the Virtualization Extensions, and for ARMv8 PEs that implement EL2, if separate controls for hypervisor debug visibility are not implemented, the hypervisor debug visibility is indicated by the relevant Non-secure debug visibility fields NSNID and NSID. See the relevant ARM Architecture Manual for more information about Virtualization Extensions and EL2.
		1	Reserved.
		2	Hypervisor noninvasive debug is implemented and disabled.
		3	If (HIDEN | HNIDEN) & (DBGEN | NIDEN) == TRUE.
9:8 (R/NW)	HID	HID. Configuration of hypervisor invasive debug.	
		0	Separate controls for hypervisor invasive debug are not implemented, or no hypervisor is implemented. For ARMv7 PEs that implement the Virtualization Extensions, and for ARMv8 PEs that implement EL2, if separate controls for hypervisor debug visibility are not implemented, the hypervisor debug visibility is indicated by the relevant Non-secure debug visibility fields NSNID and NSID. See the relevant ARM Architecture Manual for more information about Virtualization Extensions and EL2.
		1	Reserved.
		2	Hypervisor invasive debug is implemented and disabled.
		3	If (HIDEN & DBGEN) == TRUE.

Table 48-236: A55_CLUSTER_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	SNID. If Secure noninvasive debug is not supported set this field to 0b00. If Secure noninvasive debug is supported, set this field to 0b11 or 0b10.	
		0	Debug level is not supported.
		1	Reserved.
		2	In all other cases.
		3	If (SPIDEN | SPNIDEN) & (DBGEN | NIDEN) == TRUE.
5:4 (R/NW)	SID	SID. If Secure invasive debug is not supported set this field to 0b00. If Secure invasive debug is supported, set this field to 0b11 or 0b10.	
		0	Debug level is not supported.
		1	Reserved.
		2	In all other cases.
		3	If (SPIDEN & DBGEN) == TRUE.
3:2 (R/NW)	NSNID	NSNID. If Non-secure noninvasive debug is not supported, set this field to 0b00. If Non-secure noninvasive debug is supported, set this field to 0b11 or 0b10.	
		0	Debug level is not supported.
		1	Reserved.
		2	In all other cases.
		3	If (NIDEN | DBGEN) == TRUE.
1:0 (R/NW)	NSID	NSID. If Non-secure invasive debug is not supported, set this field to 0b00. If Non-secure invasive debug is supported, set this field to 0b11 or 0b10.	
		0	Debug level is not supported.
		1	Reserved.
		2	In all other cases.
		3	If DBGEN == TRUE.

Component Identification 0 Register

Component Identification Register.

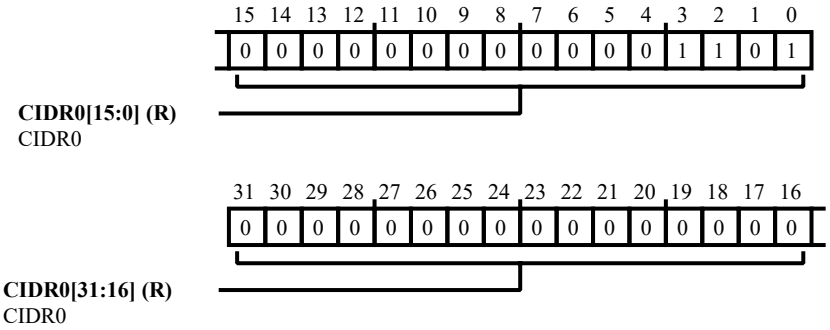


Figure 48-239: A55_CLUSTER_CIDR0 Register Diagram

Table 48-237: A55_CLUSTER_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CIDR0	CIDR0. CIDR0.

Component Identification 1 Register

Component Identification Register.

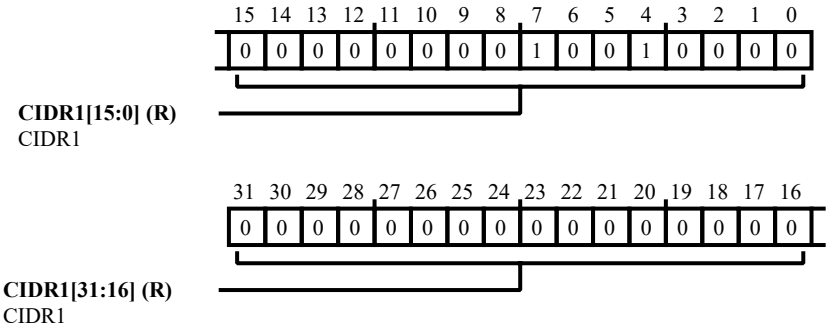


Figure 48-240: A55_CLUSTER_CIDR1 Register Diagram

Table 48-238: A55_CLUSTER_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CIDR1	CIDR1. CIDR1.

Component Identification 2 Register

Component Identification Register.

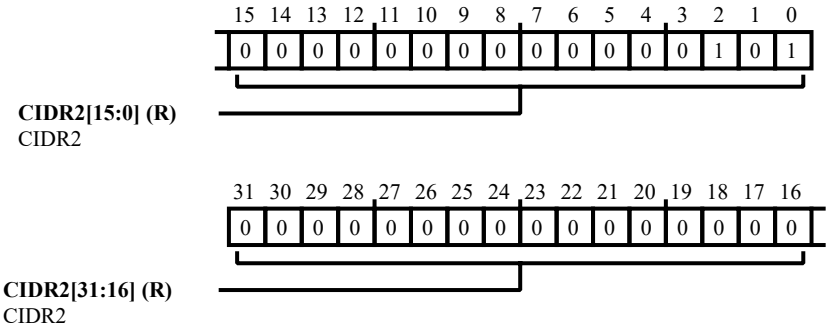


Figure 48-241: A55_CLUSTER_CIDR2 Register Diagram

Table 48-239: A55_CLUSTER_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CIDR2	CIDR2. CIDR2.

Component Identification 3 Register

Component Identification Register.

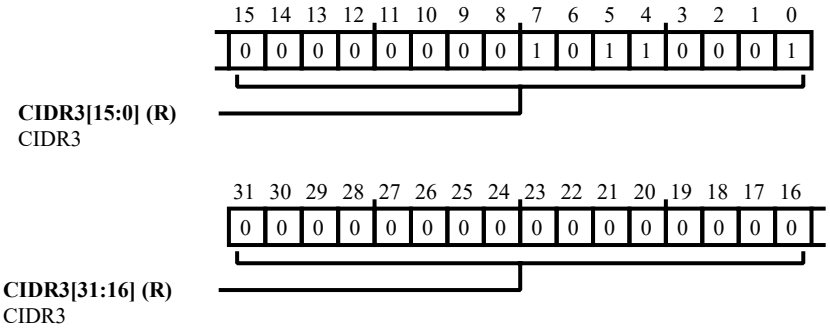


Figure 48-242: A55_CLUSTER_CIDR3 Register Diagram

Table 48-240: A55_CLUSTER_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CIDR3	CIDR3. CIDR3.

Debug Power Control 0 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

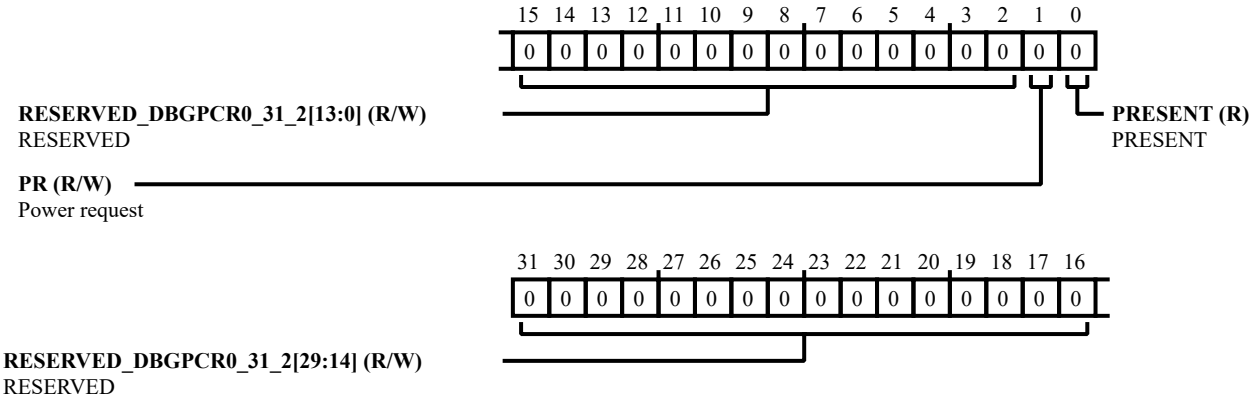


Figure 48-243: A55_CLUSTER_DBGPCR0 Register Diagram

Table 48-241: A55_CLUSTER_DBGPCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR0_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 1 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

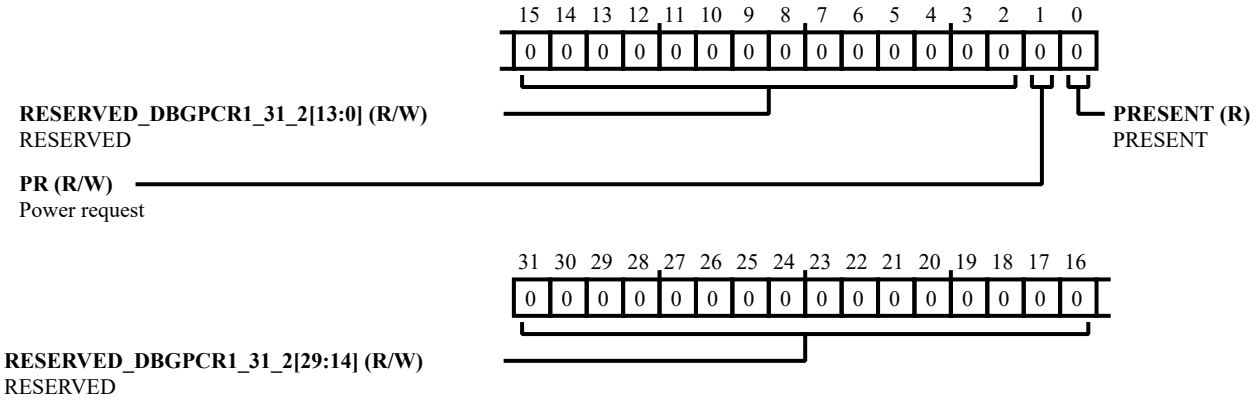


Figure 48-244: A55_CLUSTER_DBGPCR1 Register Diagram

Table 48-242: A55_CLUSTER_DBGPCR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR1_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 2 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

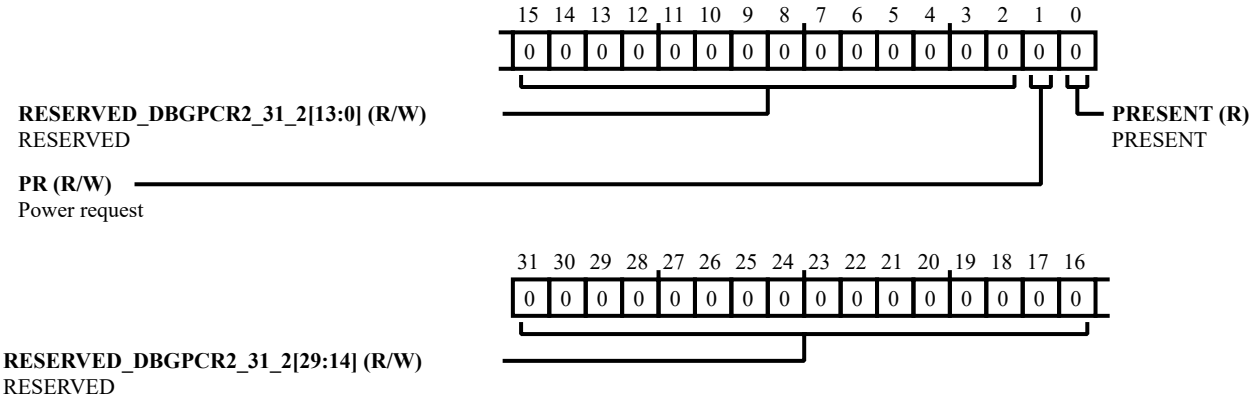


Figure 48-245: A55_CLUSTER_DBGPCR2 Register Diagram

Table 48-243: A55_CLUSTER_DBGPCR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR2_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 3 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

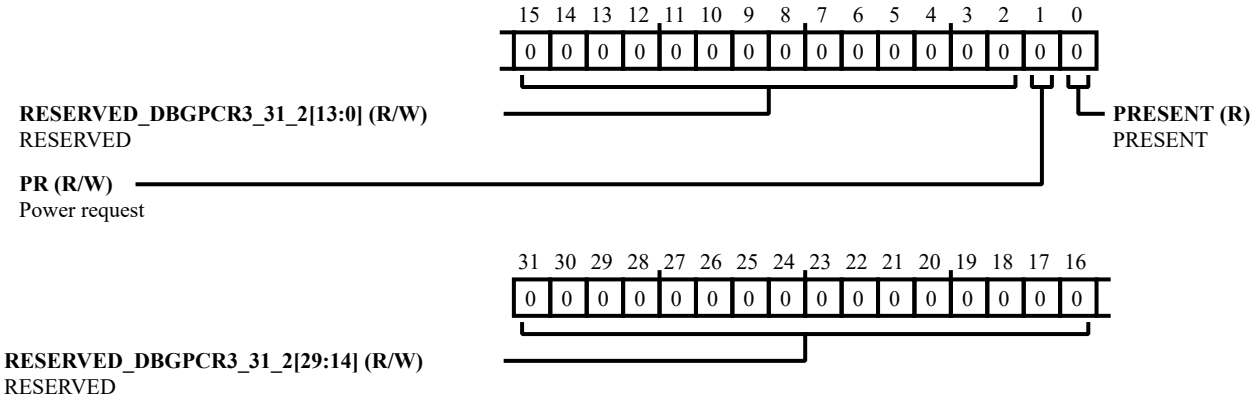


Figure 48-246: A55_CLUSTER_DBGPCR3 Register Diagram

Table 48-244: A55_CLUSTER_DBGPCR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR3_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 4 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

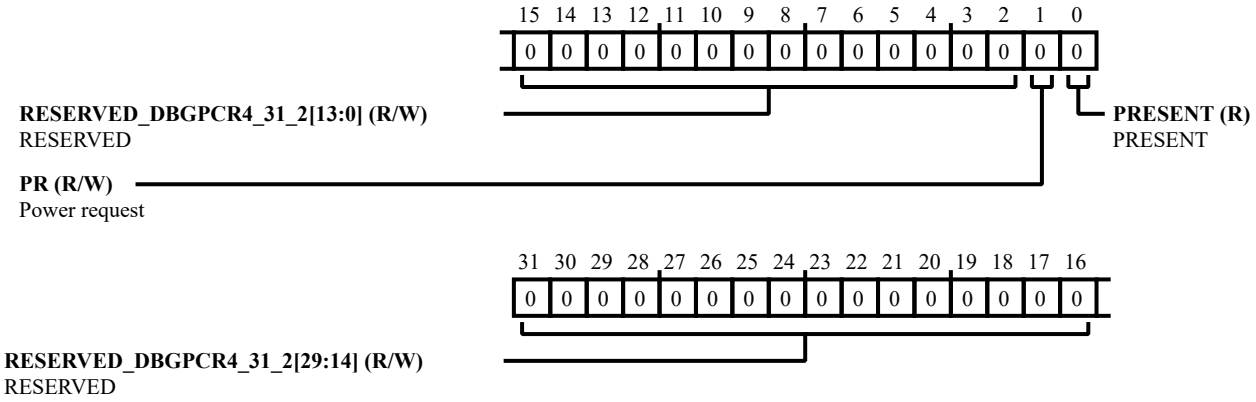


Figure 48-247: A55_CLUSTER_DBGPCR4 Register Diagram

Table 48-245: A55_CLUSTER_DBGPCR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR4_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 5 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

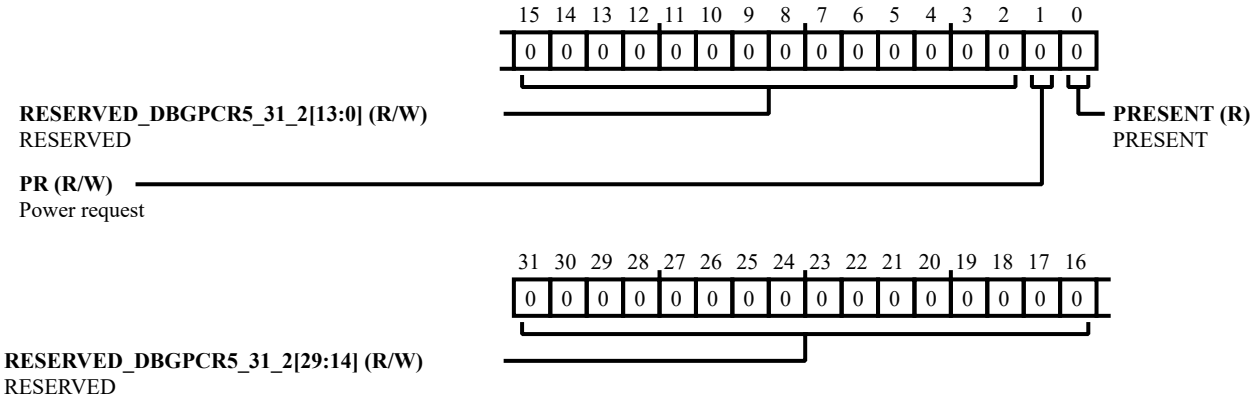


Figure 48-248: A55_CLUSTER_DBGPCR5 Register Diagram

Table 48-246: A55_CLUSTER_DBGPCR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR5_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 6 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

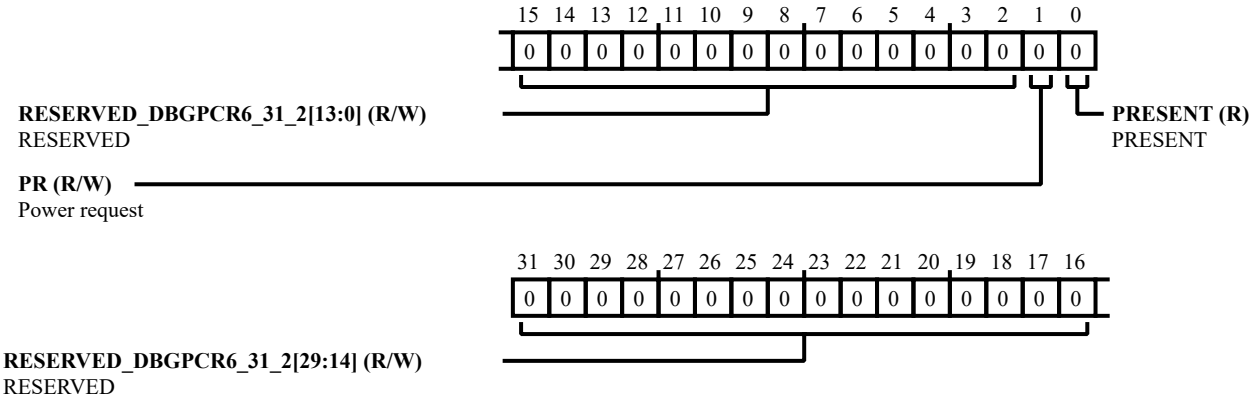


Figure 48-249: A55_CLUSTER_DBGPCR6 Register Diagram

Table 48-247: A55_CLUSTER_DBGPCR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR6_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 7 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

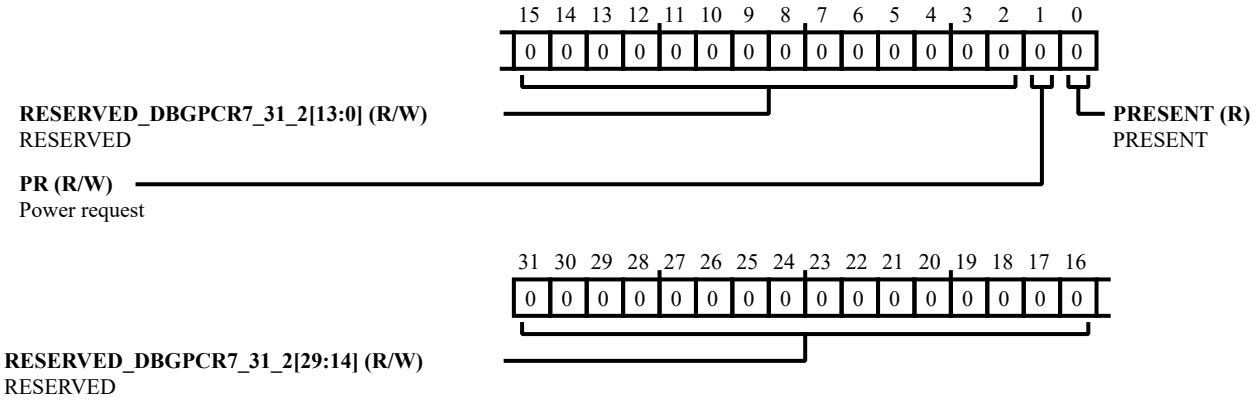


Figure 48-250: A55_CLUSTER_DBGPCR7 Register Diagram

Table 48-248: A55_CLUSTER_DBGPCR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR7_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Control 8 Register

DBGPCRn is used to indicate whether a power request mechanism is implemented for debug power domain n. Request power to debug power domain n.

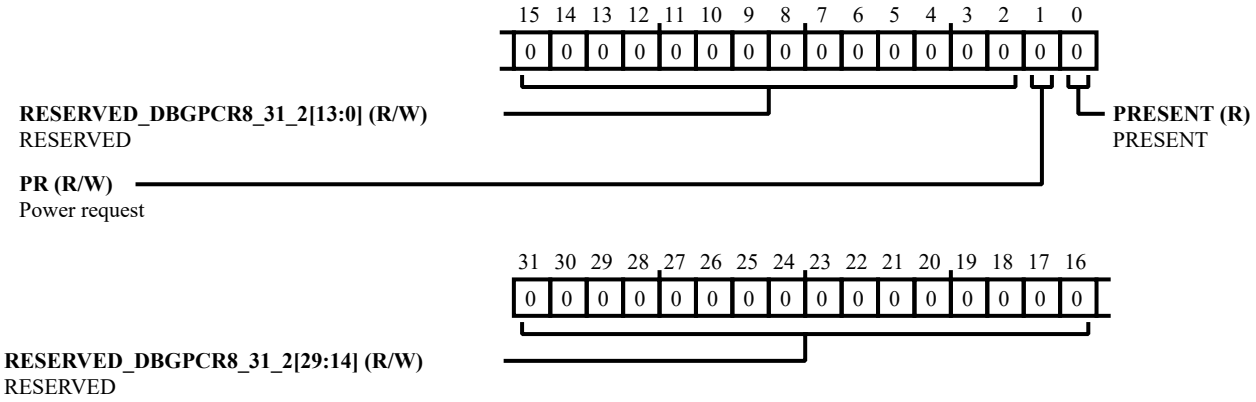


Figure 48-251: A55_CLUSTER_DBGPCR8 Register Diagram

Table 48-249: A55_CLUSTER_DBGPCR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RESERVED_DBGPCR8_31_2	RESERVED. Reserved RES0.	
1 (R/W)	PR	Power request. This field is Reserved if DBGPCRn.PRESENT is b0.	
		0	Power not requested for debug power domain n.
		1	Power requested for debug power domain n.
0 (R/NW)	PRESENT	PRESENT. Power request implemented.	
		0	Power request not implemented for debug power domain n.
		1	Power request implemented for debug power domain n. DBGPCRn.PR and DBGPSRn are implemented.

Debug Power Status 0 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

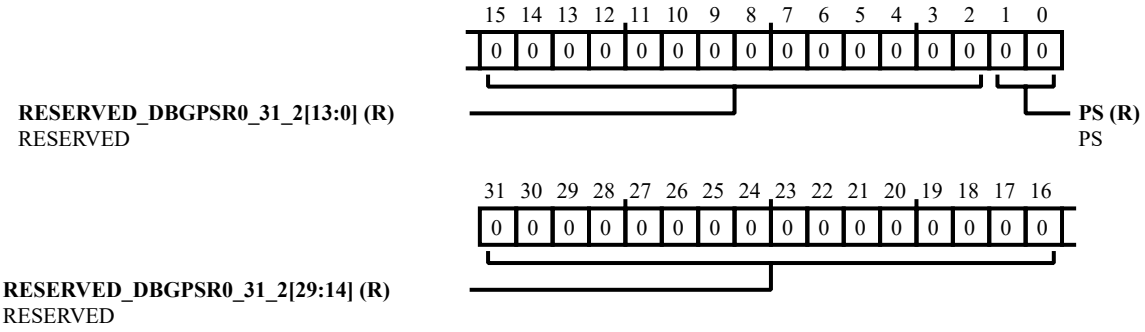


Figure 48-252: A55_CLUSTER_DBGPSR0 Register Diagram

Table 48-250: A55_CLUSTER_DBGPSR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR0_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 1 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

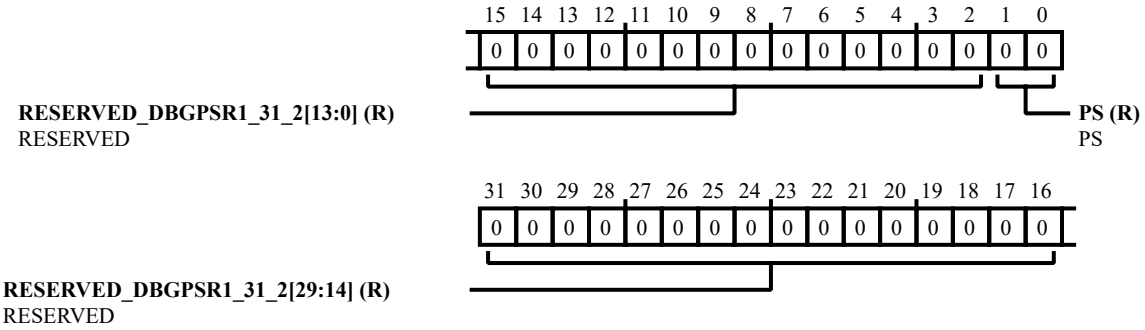


Figure 48-253: A55_CLUSTER_DBGPSR1 Register Diagram

Table 48-251: A55_CLUSTER_DBGPSR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR1_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 2 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

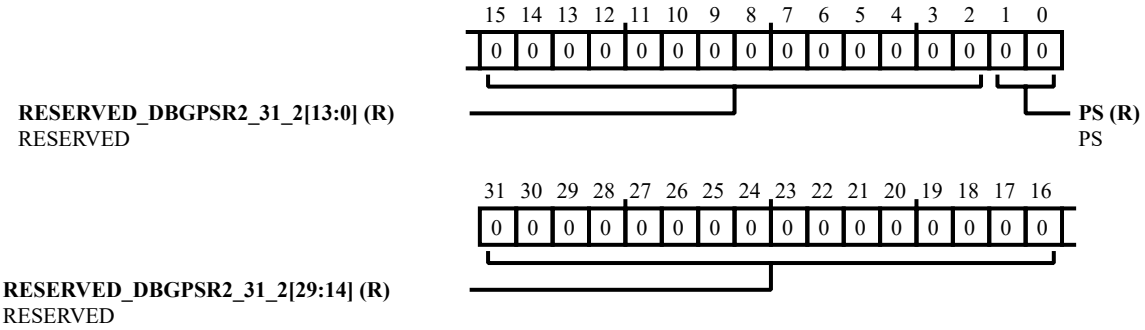


Figure 48-254: A55_CLUSTER_DBGPSR2 Register Diagram

Table 48-252: A55_CLUSTER_DBGPSR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR2_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 3 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

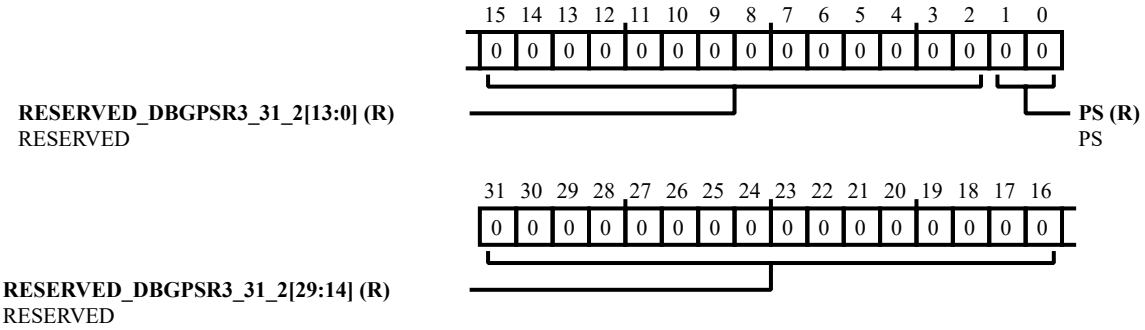


Figure 48-255: A55_CLUSTER_DBGPSR3 Register Diagram

Table 48-253: A55_CLUSTER_DBGPSR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR3_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 4 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

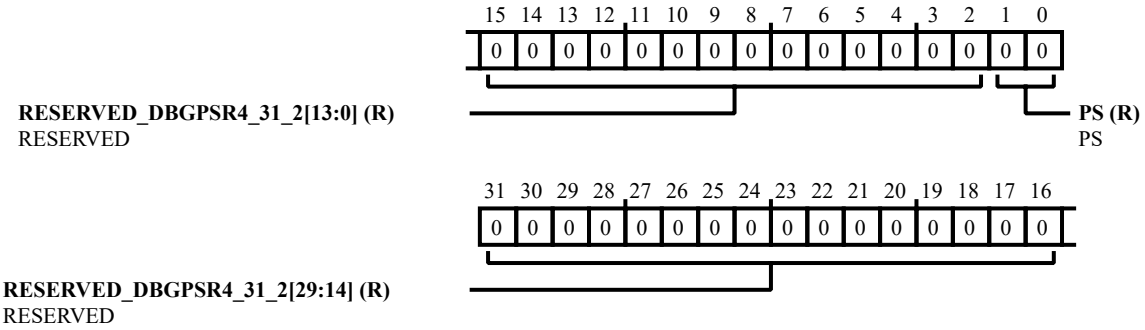


Figure 48-256: A55_CLUSTER_DBGPSR4 Register Diagram

Table 48-254: A55_CLUSTER_DBGPSR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR4_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 5 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

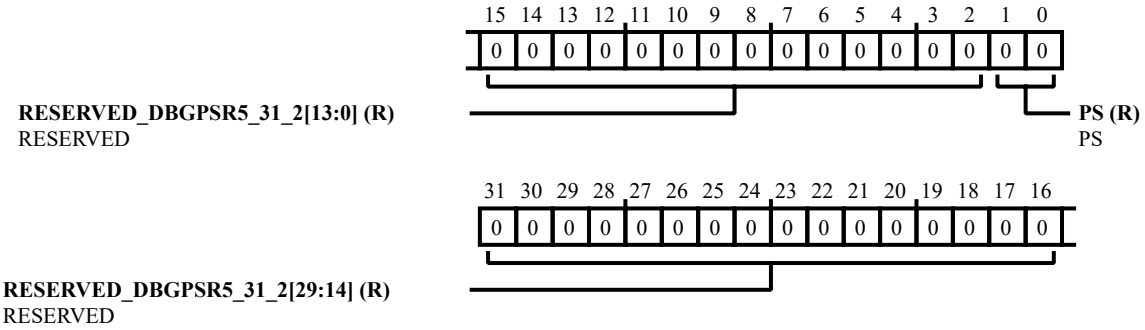


Figure 48-257: A55_CLUSTER_DBGPSR5 Register Diagram

Table 48-255: A55_CLUSTER_DBGPSR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR5_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 6 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

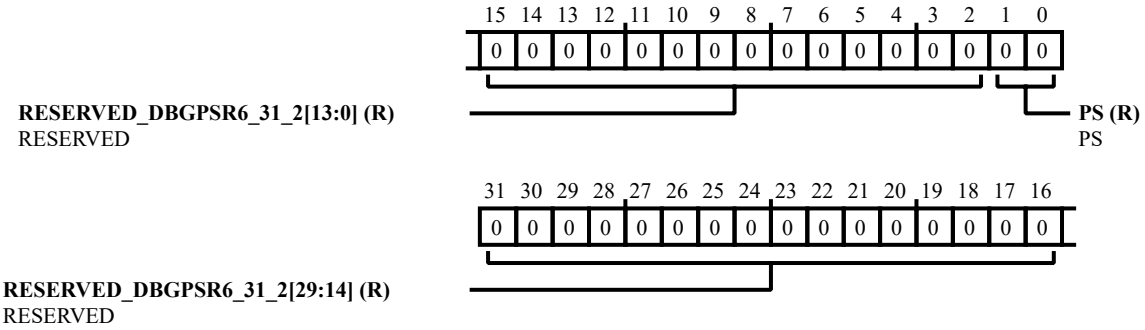


Figure 48-258: A55_CLUSTER_DBGPSR6 Register Diagram

Table 48-256: A55_CLUSTER_DBGPSR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR6_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 7 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

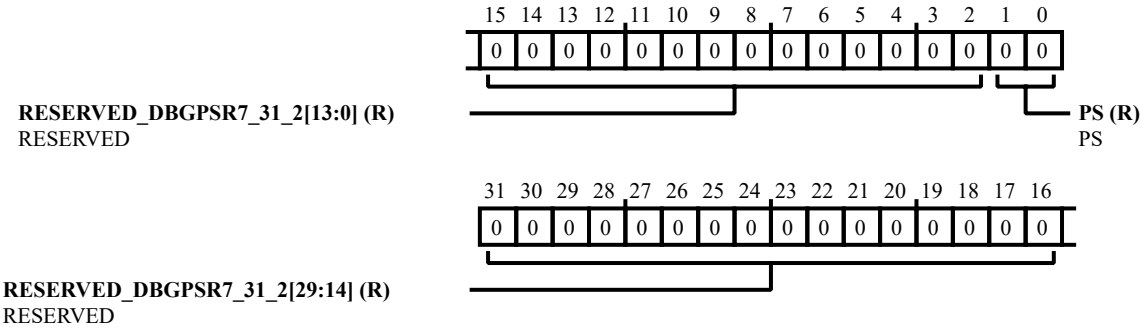


Figure 48-259: A55_CLUSTER_DBGPSR7 Register Diagram

Table 48-257: A55_CLUSTER_DBGPSR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR7_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Debug Power Status 8 Register

DBGPSRn is used to indicate the current power status for debug power domain n.

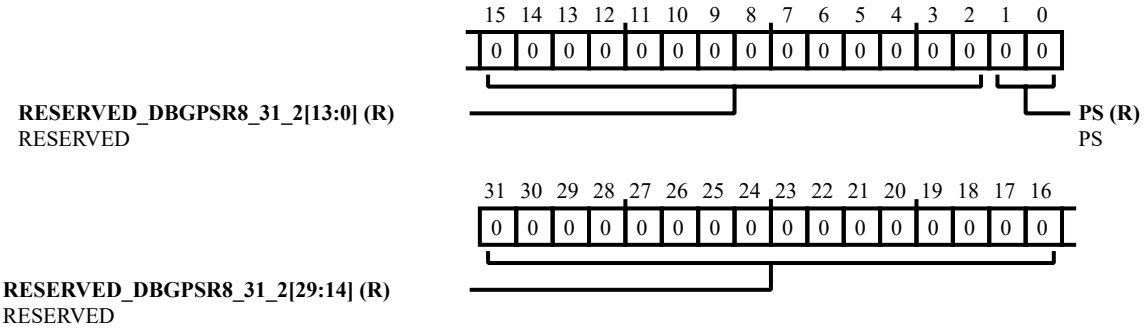


Figure 48-260: A55_CLUSTER_DBGPSR8 Register Diagram

Table 48-258: A55_CLUSTER_DBGPSR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RESERVED_DBGPSR8_31_2	RESERVED. Reserved RES0.	
1:0 (R/NW)	PS	PS. Power status.	
		0	Debug power domain n might not be powered.
		1	Debug power domain n is powered.
		2	Reserved.
		3	Debug power domain n is powered and must remain powered until DBGPCRn.PR is set to b0.

Device Architecture Register

Indicates the component is a ROM table.

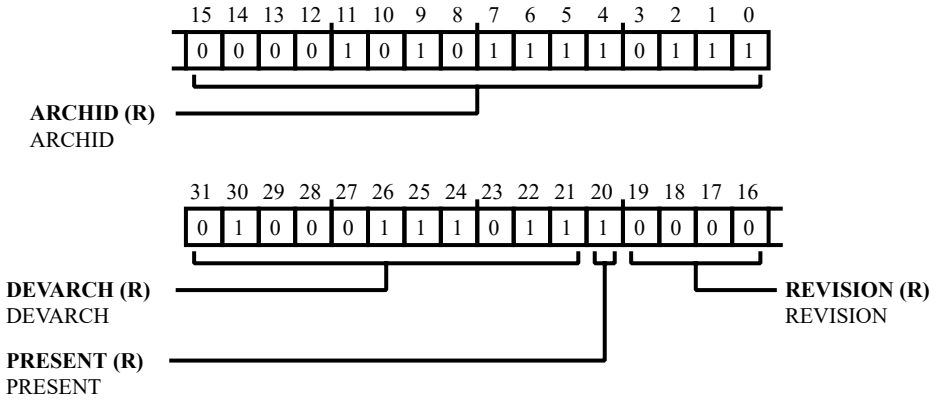


Figure 48-261: A55_CLUSTER_DEVARCH Register Diagram

Table 48-259: A55_CLUSTER_DEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	DEVARCH	DEVARCH. ARCHITECT.
20 (R/NW)	PRESENT	PRESENT. PRESENT.
19:16 (R/NW)	REVISION	REVISION. REVISION.
15:0 (R/NW)	ARCHID	ARCHID. indicating a ROM table v0.

Device ID Register

DEVID.

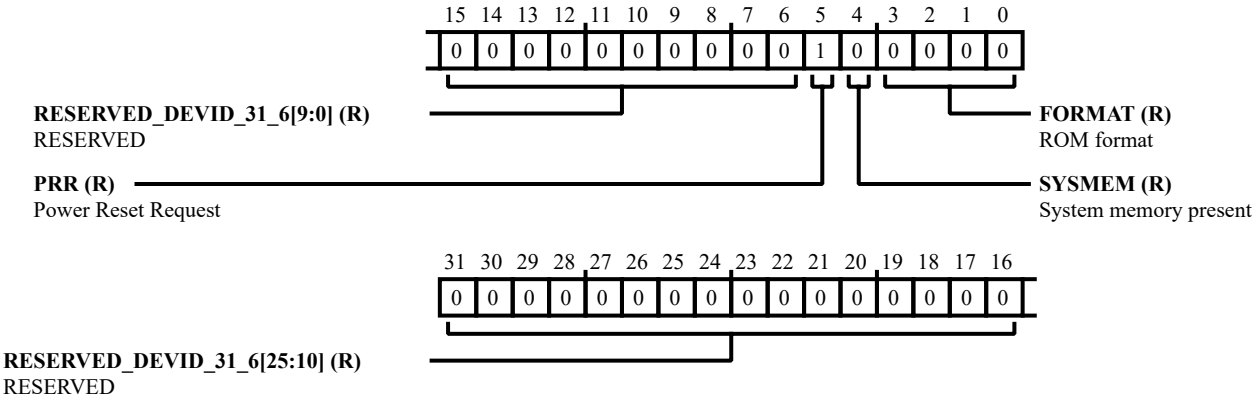


Figure 48-262: A55_CLUSTER_DEVID Register Diagram

Table 48-260: A55_CLUSTER_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:6 (R/NW)	RESERVED_DEVID_31_6	RESERVED. Reserved RES0.	
5 (R/NW)	PRR	Power Reset Request. Power Reset Request functionality included.	
		0	Power Reset Request functionality not included. If any ROM table entries contain power domain IDs, then a GPR must be present and pointed to by this ROM table. The GPR provides controls to these power domains. PRIDR0 is not implemented.
		1	Power Reset Request functionality included. If any ROM table entries contain power domain IDs, the controls to these power domains is in the Power Reset Request functionality included in this ROM table. PRIDR0 is implemented.
4 (R/NW)	SYSMEM	System memory present. Indicates whether system memory is present on the bus that connects to the ROM Table.	
		0	System memory not present on bus. This is a dedicated debug bus.
		1	System memory is also present on this bus.

Table 48-260: A55_CLUSTER_DEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/NW)	FORMAT	ROM format. ROM format.	
		0	32-bit Format 0.
		1	64-bit Format 1.

Device Type Register

DEVTYPE.

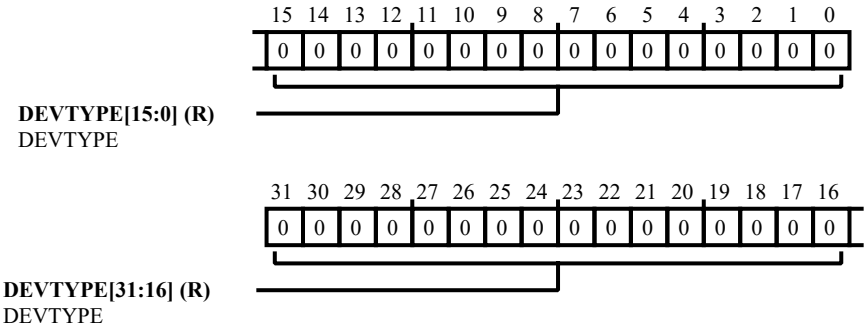


Figure 48-263: A55_CLUSTER_DEVTYPE Register Diagram

Table 48-261: A55_CLUSTER_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVTYPE	DEVTYPE. DEVTYPE.

Peripheral Identification 0 Register

Peripheral Identification Register.

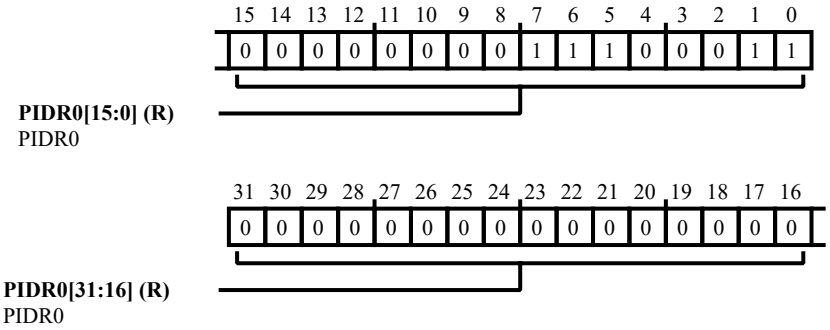


Figure 48-264: A55_CLUSTER_PIDR0 Register Diagram

Table 48-262: A55_CLUSTER_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR0	PIDR0. PIDR0.

Peripheral Identification 1 Register

Peripheral Identification Register.

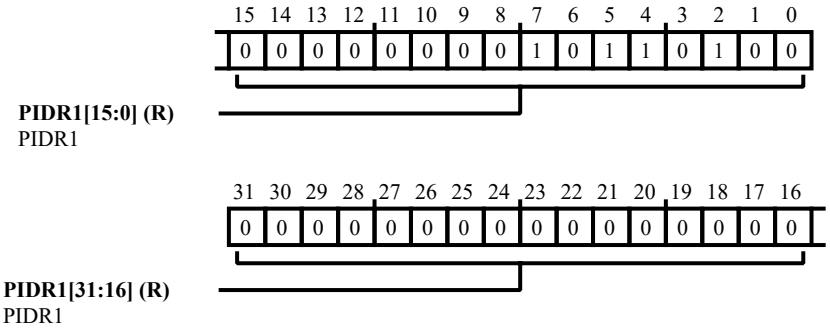


Figure 48-265: A55_CLUSTER_PIDR1 Register Diagram

Table 48-263: A55_CLUSTER_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR1	PIDR1. PIDR1.

Peripheral Identification 2 Register

Peripheral Identification Register.

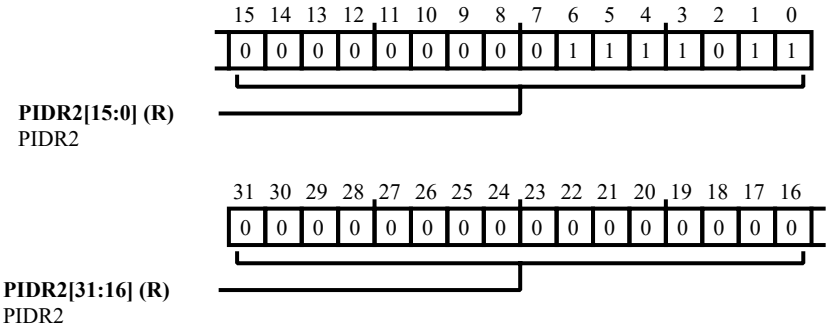


Figure 48-266: A55_CLUSTER_PIDR2 Register Diagram

Table 48-264: A55_CLUSTER_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR2	PIDR2. PIDR2.

Peripheral Identification 3 Register

Peripheral Identification Register.

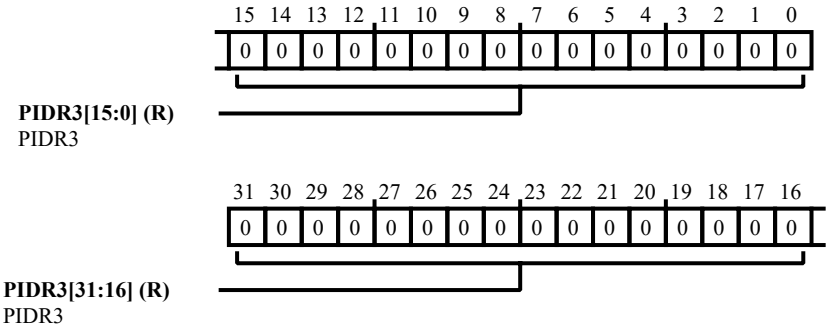


Figure 48-267: A55_CLUSTER_PIDR3 Register Diagram

Table 48-265: A55_CLUSTER_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR3	PIDR3. PIDR3.

Peripheral Identification 4 Register

Peripheral Identification Register.

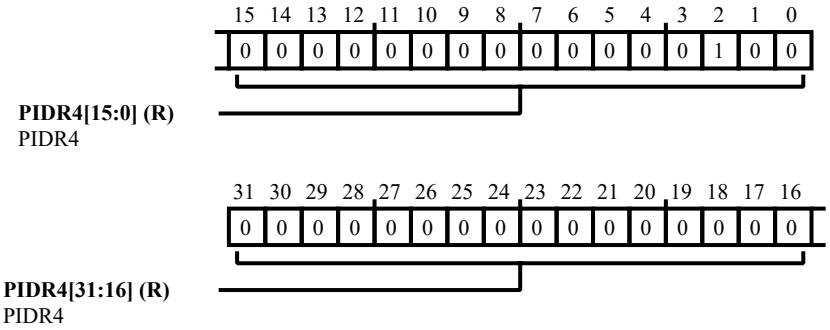


Figure 48-268: A55_CLUSTER_PIDR4 Register Diagram

Table 48-266: A55_CLUSTER_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR4	PIDR4. PIDR4.

Peripheral Identification 5 Register

Peripheral Identification Register.

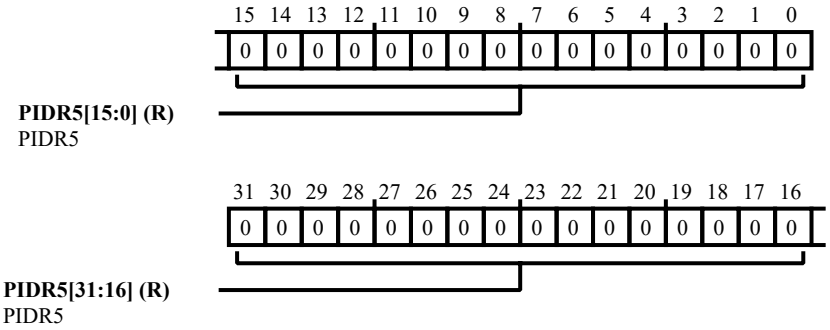


Figure 48-269: A55_CLUSTER_PIDR5 Register Diagram

Table 48-267: A55_CLUSTER_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR5	PIDR5. PIDR5.

Peripheral Identification 6 Register

Peripheral Identification Register.

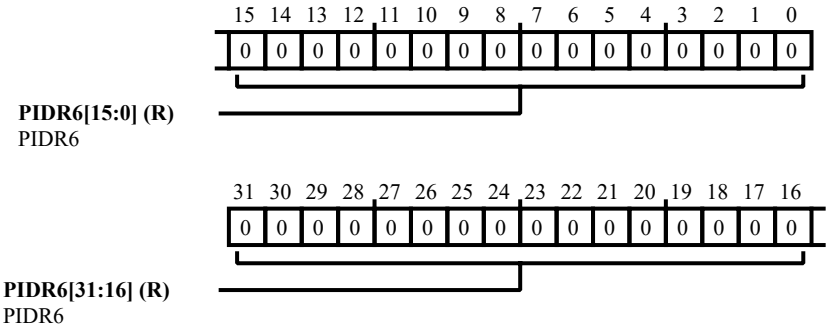


Figure 48-270: A55_CLUSTER_PIDR6 Register Diagram

Table 48-268: A55_CLUSTER_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR6	PIDR6. PIDR6.

Peripheral Identification 7 Register

Peripheral Identification Register.

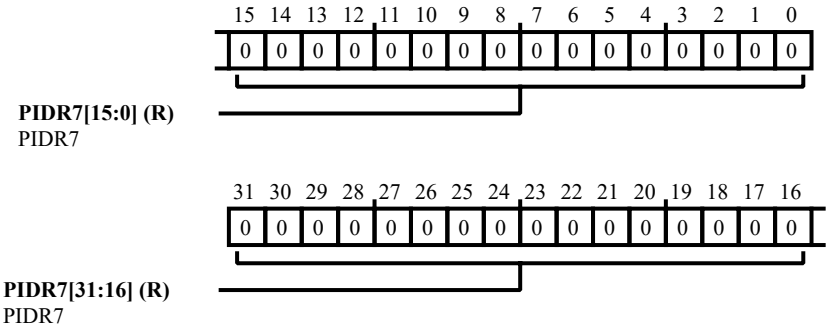


Figure 48-271: A55_CLUSTER_PIDR7 Register Diagram

Table 48-269: A55_CLUSTER_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PIDR7	PIDR7. PIDR7.

Power Reset Identification 0 Register

PRIDR0 indicates the features of the power request functionality.

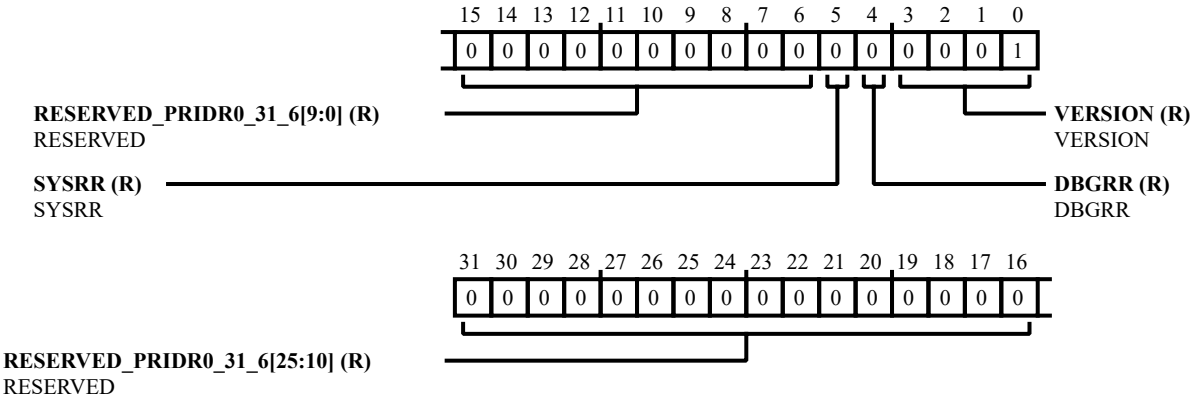


Figure 48-272: A55_CLUSTER_PRIDR0 Register Diagram

Table 48-270: A55_CLUSTER_PRIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:6 (R/NW)	RESERVED_PRIDR0_31_6	RESERVED. Reserved RES0.	
5 (R/NW)	SYSRR	SYSRR. SYSRR.	
		0	System reset request functionality not implemented.
		1	System reset request functionality implemented. SYSRSTRR and SYSRSTAR implemented.
4 (R/NW)	DBGRR	DBGRR. Indicates whether the debug reset request functionality is present.	
		0	Debug reset request functionality not implemented.
		1	Debug reset request functionality implemented. DBGRSTRR and DBGRSTAR implemented.
3:0 (R/NW)	VERSION	VERSION. Power request functionality version.	
		0	Power request functionality not implemented.
		1	Power request version 0 implemented.

ROM Table Entry 0 Register

ROMENTRYn.

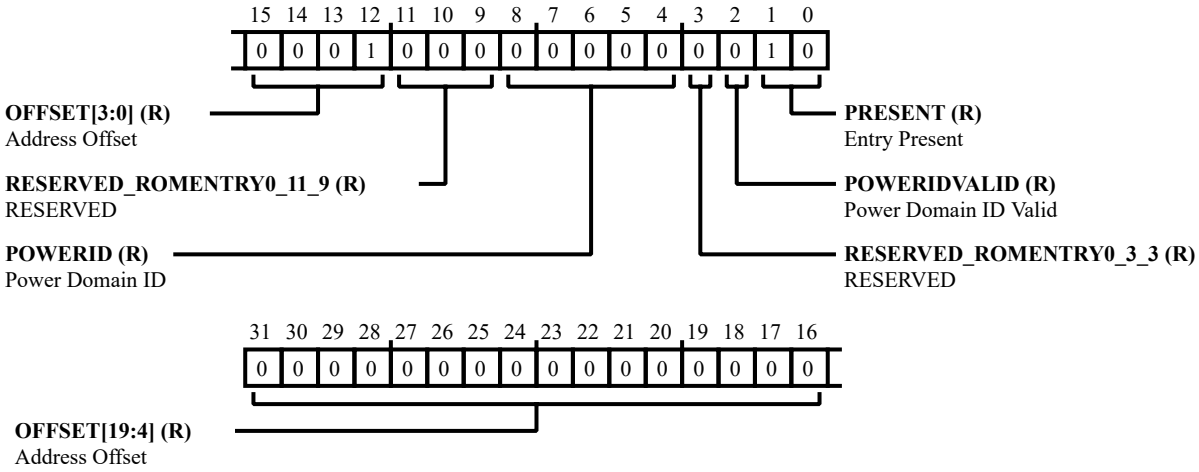


Figure 48-273: A55_CLUSTER_ROMENTRY0 Register Diagram

Table 48-271: A55_CLUSTER_ROMENTRY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY0_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY0_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-271: A55_CLUSTER_ROMENTRY0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 1 Register

ROMENTRYn.

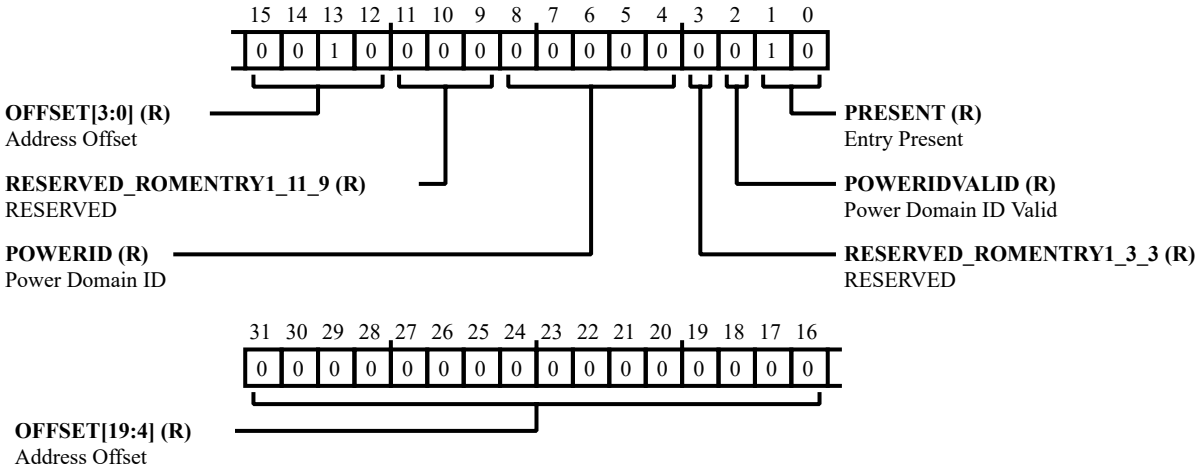


Figure 48-274: A55_CLUSTER_ROMENTRY1 Register Diagram

Table 48-272: A55_CLUSTER_ROMENTRY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY1_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY1_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-272: A55_CLUSTER_ROMENTRY1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 10 Register

ROMENTRYn.

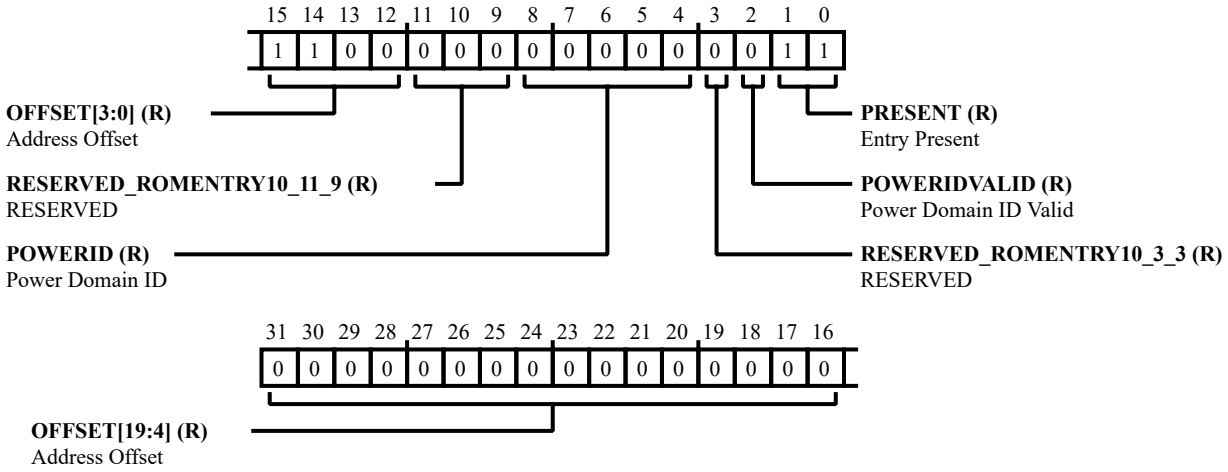


Figure 48-275: A55_CLUSTER_ROMENTRY10 Register Diagram

Table 48-273: A55_CLUSTER_ROMENTRY10 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY10_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY10_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-273: A55_CLUSTER_ROMENTRY10 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 11 Register

ROMENTRYn.

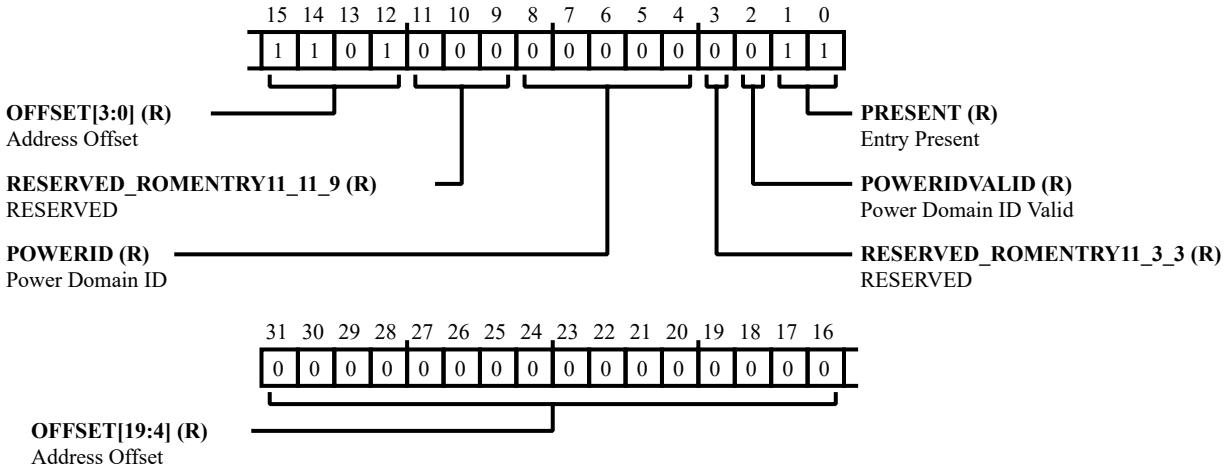


Figure 48-276: A55_CLUSTER_ROMENTRY11 Register Diagram

Table 48-274: A55_CLUSTER_ROMENTRY11 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY11_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY11_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-274: A55_CLUSTER_ROMENTRY11 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 12 Register

ROMENTRYn.

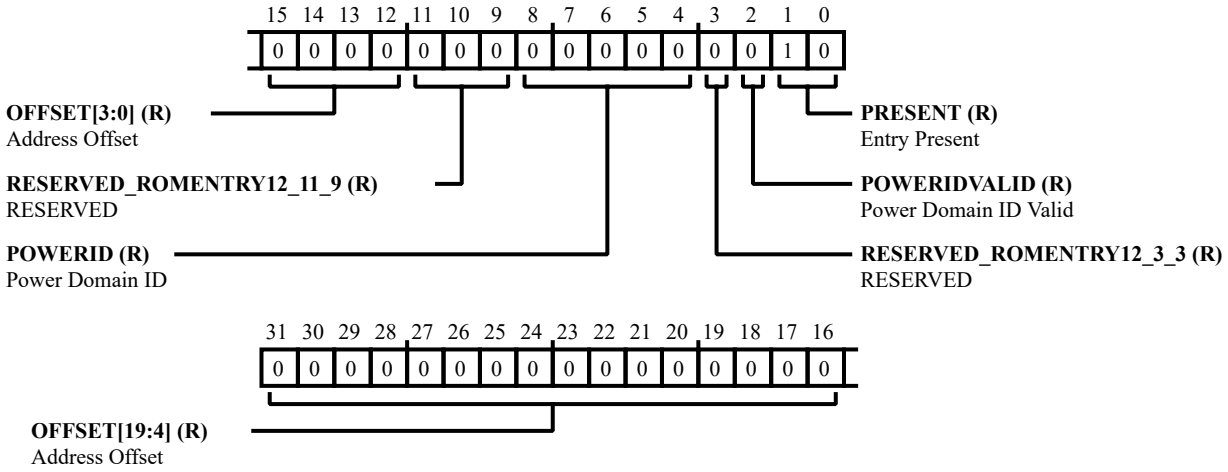


Figure 48-277: A55_CLUSTER_ROMENTRY12 Register Diagram

Table 48-275: A55_CLUSTER_ROMENTRY12 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY12_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY12_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-275: A55_CLUSTER_ROMENTRY12 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 13 Register

ROMENTRYn.

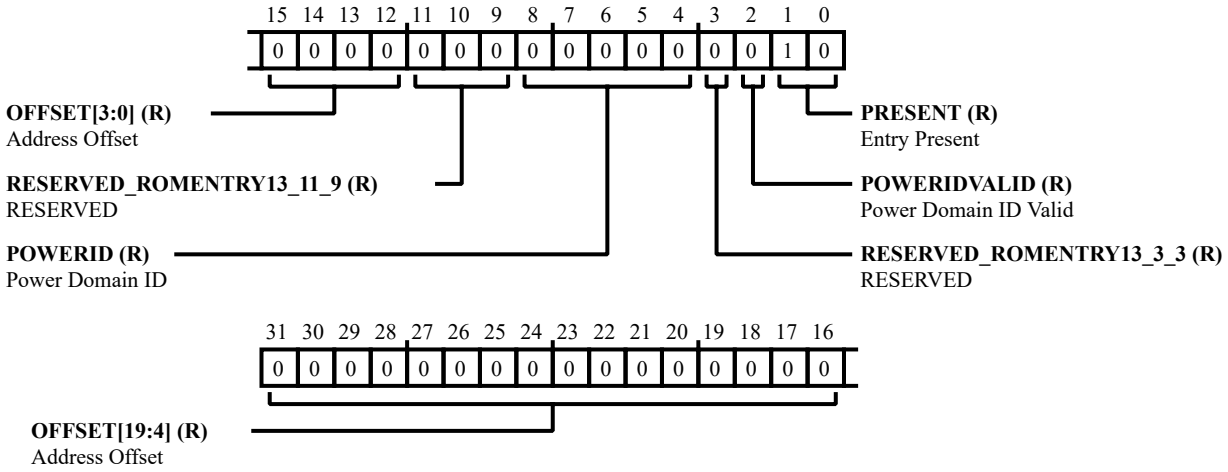


Figure 48-278: A55_CLUSTER_ROMENTRY13 Register Diagram

Table 48-276: A55_CLUSTER_ROMENTRY13 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY13_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY13_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-276: A55_CLUSTER_ROMENTRY13 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 14 Register

ROMENTRYn.

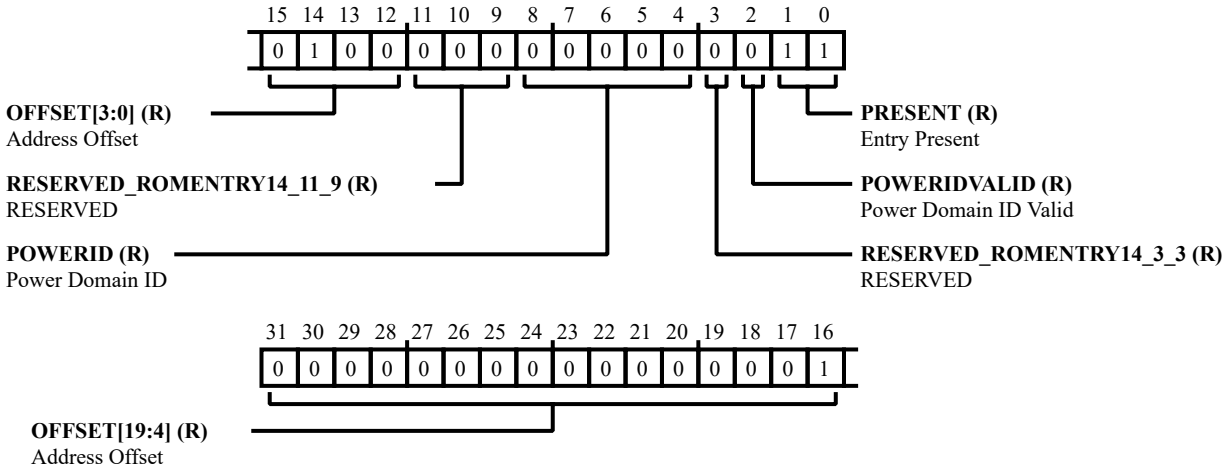


Figure 48-279: A55_CLUSTER_ROMENTRY14 Register Diagram

Table 48-277: A55_CLUSTER_ROMENTRY14 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY14_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY14_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-277: A55_CLUSTER_ROMENTRY14 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 15 Register

ROMENTRYn.

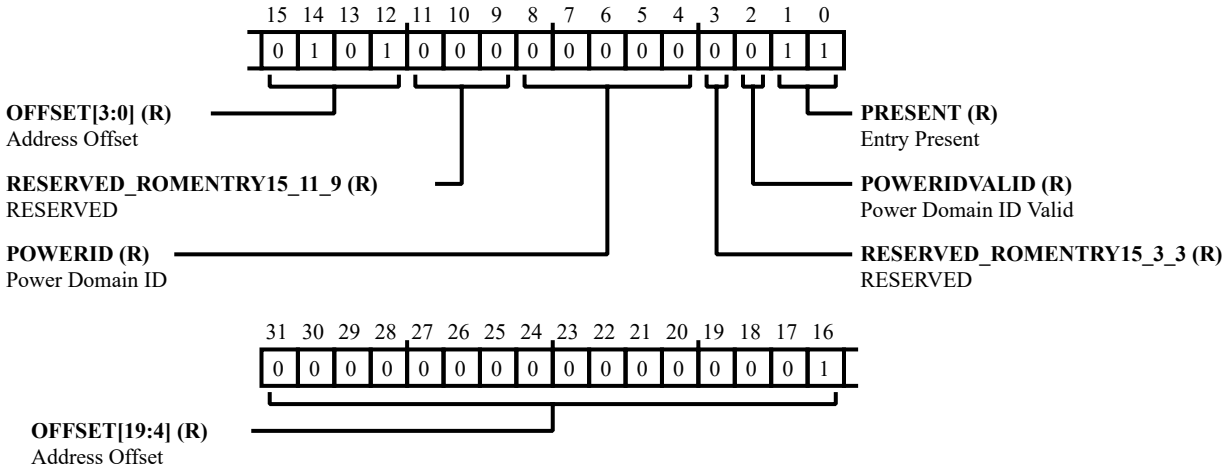


Figure 48-280: A55_CLUSTER_ROMENTRY15 Register Diagram

Table 48-278: A55_CLUSTER_ROMENTRY15 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY15_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY15_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-278: A55_CLUSTER_ROMENTRY15 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 16 Register

ROMENTRYn.

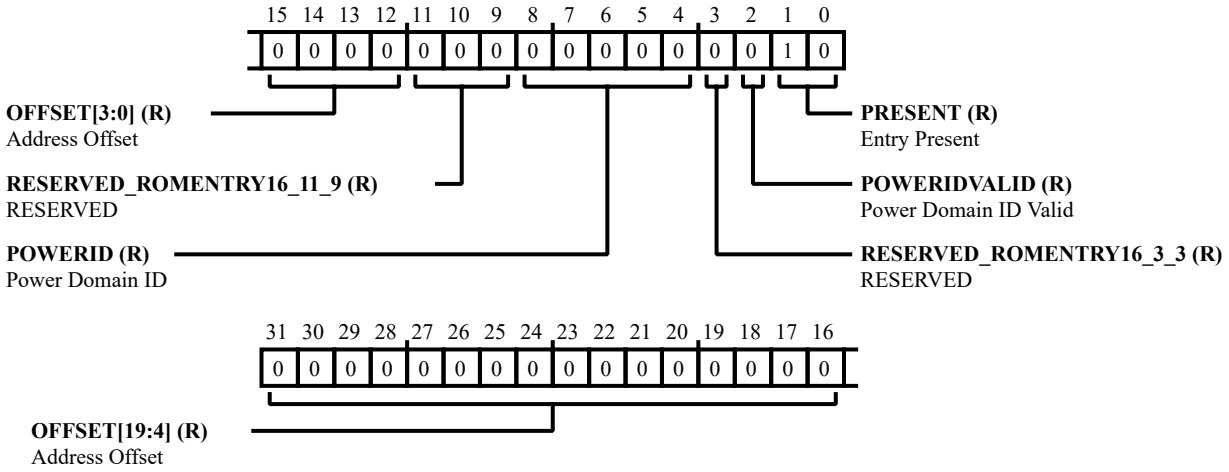


Figure 48-281: A55_CLUSTER_ROMENTRY16 Register Diagram

Table 48-279: A55_CLUSTER_ROMENTRY16 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY16_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY16_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-279: A55_CLUSTER_ROMENTRY16 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 17 Register

ROMENTRYn.

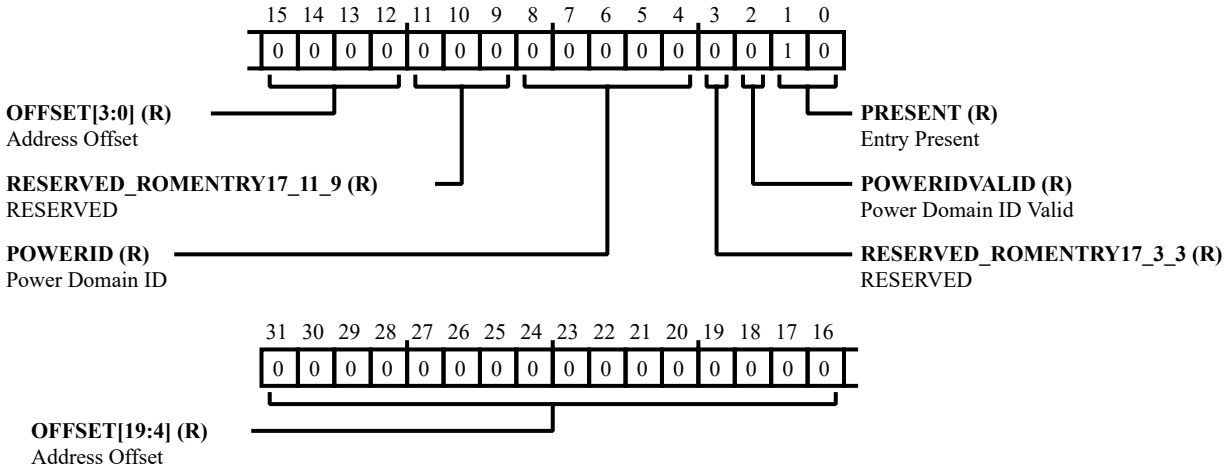


Figure 48-282: A55_CLUSTER_ROMENTRY17 Register Diagram

Table 48-280: A55_CLUSTER_ROMENTRY17 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY17_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY17_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-280: A55_CLUSTER_ROMENTRY17 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 18 Register

ROMENTRYn.

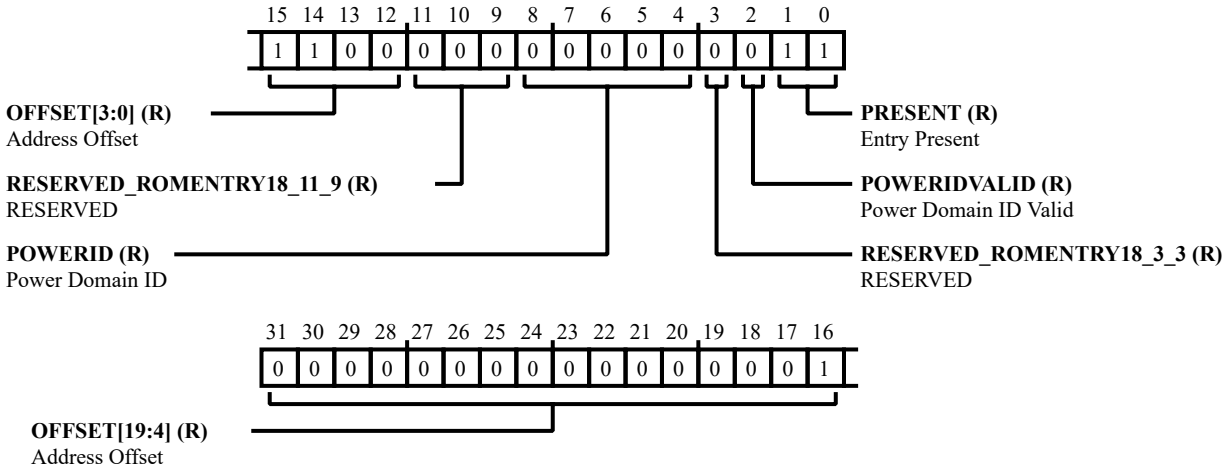


Figure 48-283: A55_CLUSTER_ROMENTRY18 Register Diagram

Table 48-281: A55_CLUSTER_ROMENTRY18 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY18_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY18_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-281: A55_CLUSTER_ROMENTRY18 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 19 Register

ROMENTRYn.

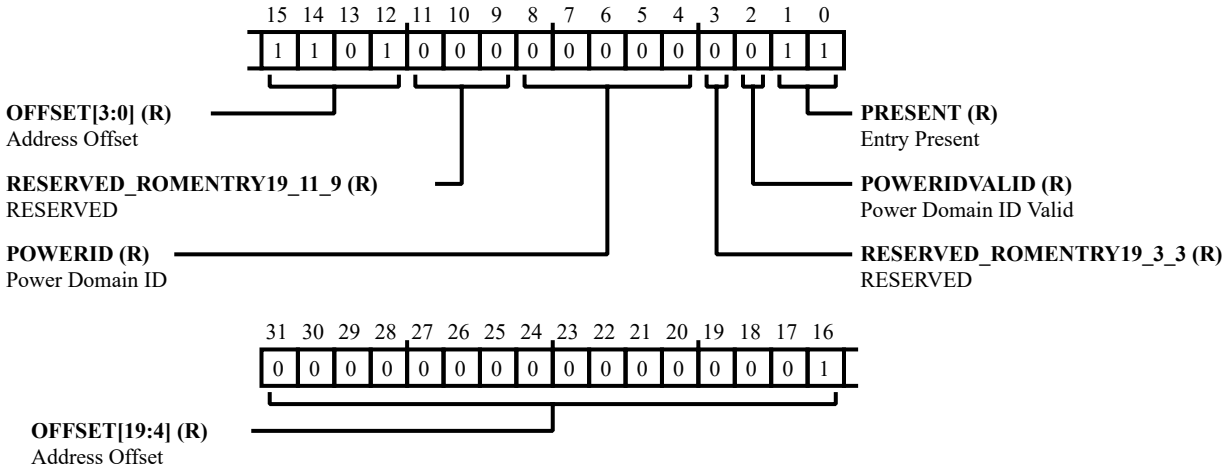


Figure 48-284: A55_CLUSTER_ROMENTRY19 Register Diagram

Table 48-282: A55_CLUSTER_ROMENTRY19 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY19_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY19_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-282: A55_CLUSTER_ROMENTRY19 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 2 Register

ROMENTRYn.

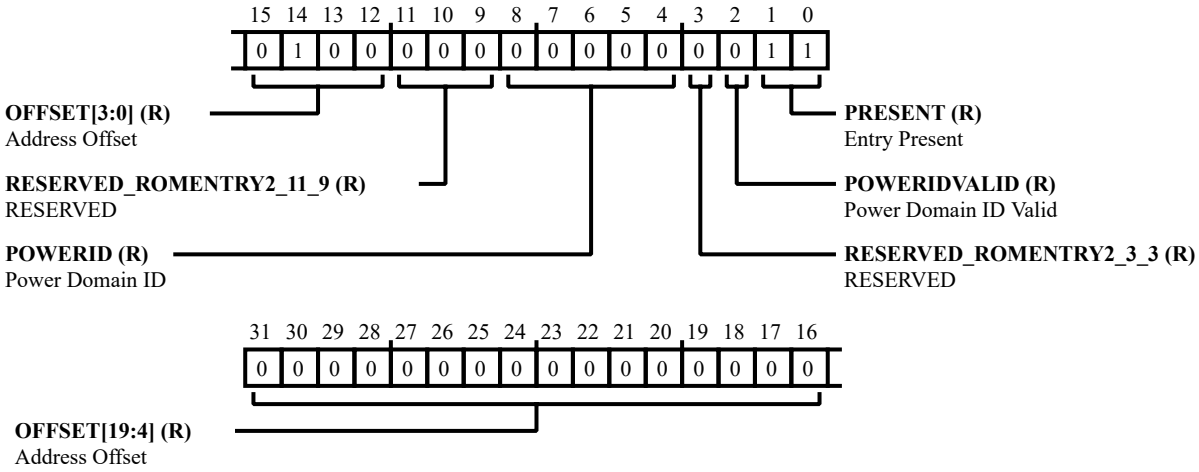


Figure 48-285: A55_CLUSTER_ROMENTRY2 Register Diagram

Table 48-283: A55_CLUSTER_ROMENTRY2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY2_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY2_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-283: A55_CLUSTER_ROMENTRY2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 20 Register

ROMENTRYn.

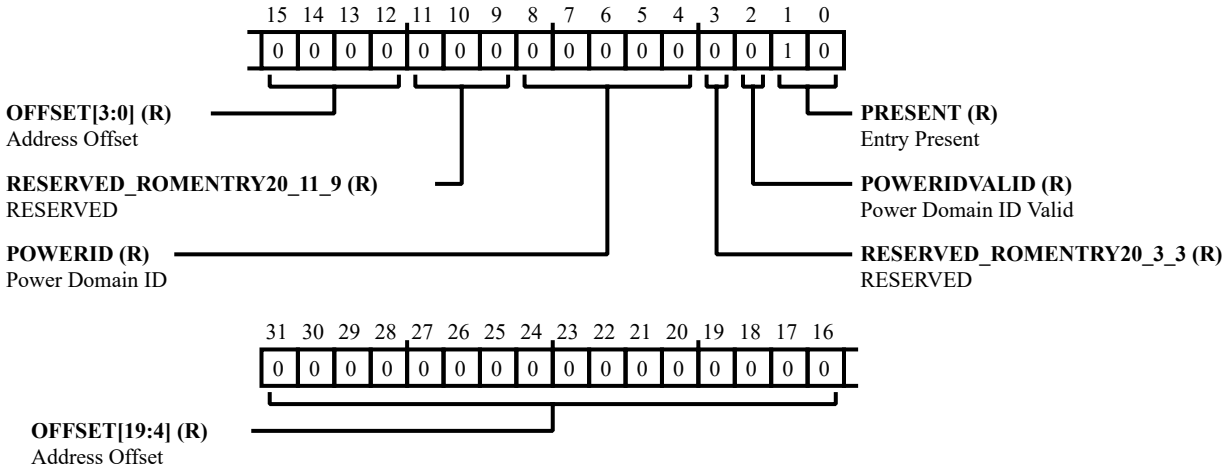


Figure 48-286: A55_CLUSTER_ROMENTRY20 Register Diagram

Table 48-284: A55_CLUSTER_ROMENTRY20 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY20_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY20_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-284: A55_CLUSTER_ROMENTRY20 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 21 Register

ROMENTRYn.

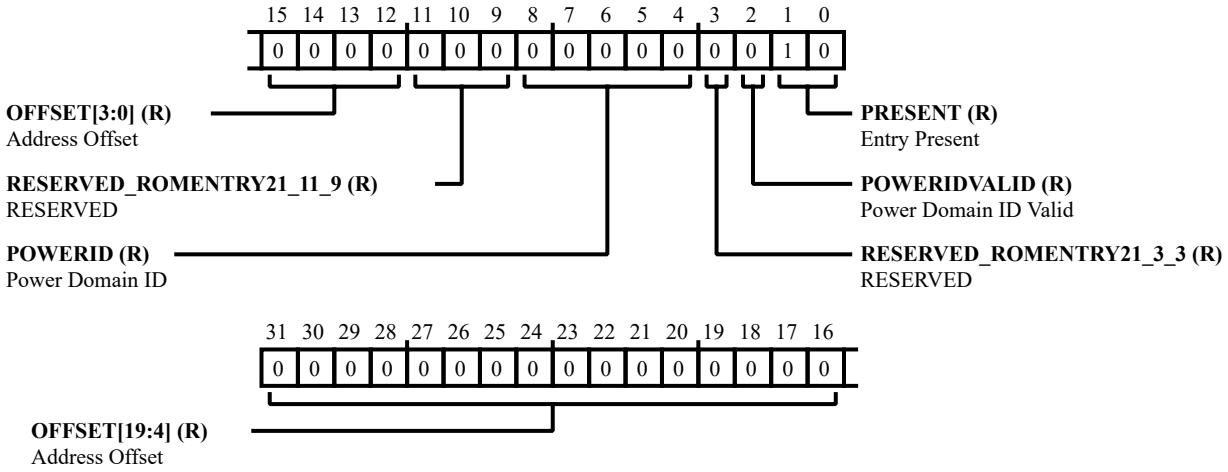


Figure 48-287: A55_CLUSTER_ROMENTRY21 Register Diagram

Table 48-285: A55_CLUSTER_ROMENTRY21 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY21_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY21_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-285: A55_CLUSTER_ROMENTRY21 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 22 Register

ROMENTRYn.

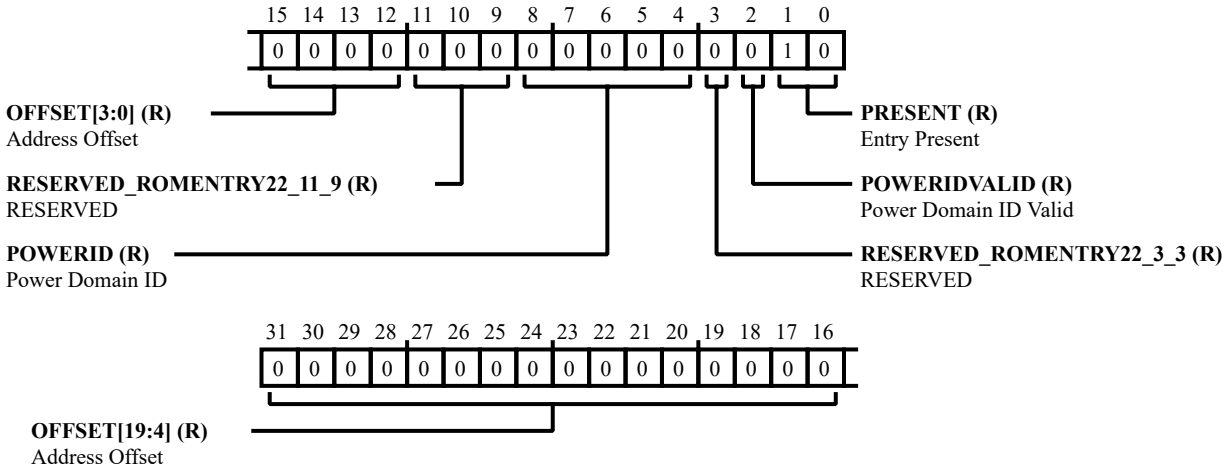


Figure 48-288: A55_CLUSTER_ROMENTRY22 Register Diagram

Table 48-286: A55_CLUSTER_ROMENTRY22 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY22_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY22_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-286: A55_CLUSTER_ROMENTRY22 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 23 Register

ROMENTRYn.

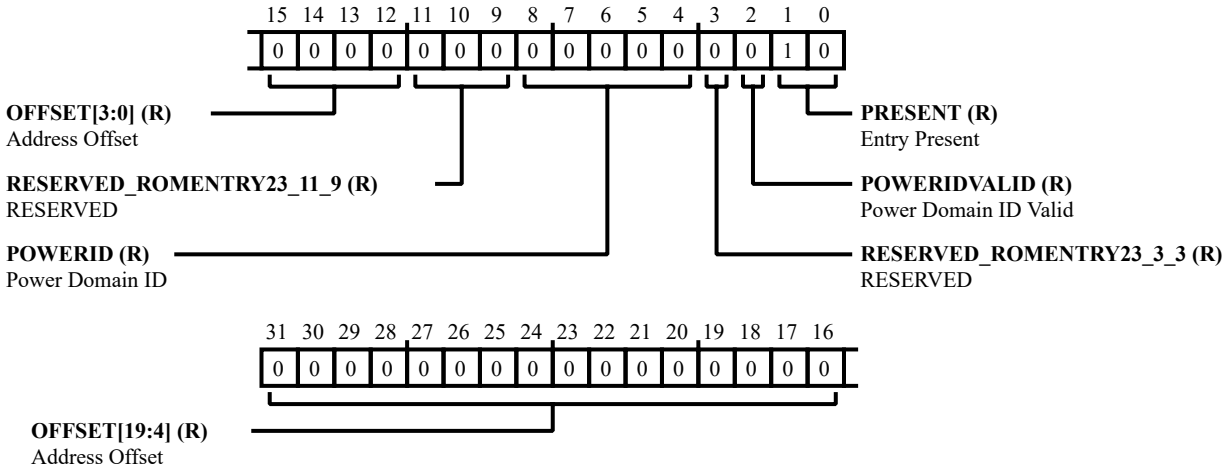


Figure 48-289: A55_CLUSTER_ROMENTRY23 Register Diagram

Table 48-287: A55_CLUSTER_ROMENTRY23 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY23_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY23_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-287: A55_CLUSTER_ROMENTRY23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 24 Register

ROMENTRYn.

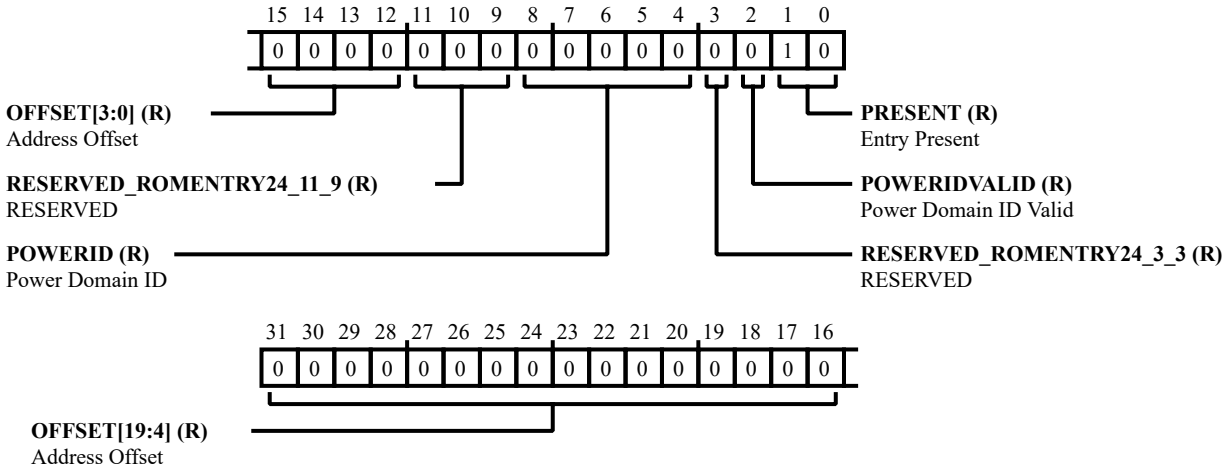


Figure 48-290: A55_CLUSTER_ROMENTRY24 Register Diagram

Table 48-288: A55_CLUSTER_ROMENTRY24 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY24_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY24_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-288: A55_CLUSTER_ROMENTRY24 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 25 Register

ROMENTRYn.

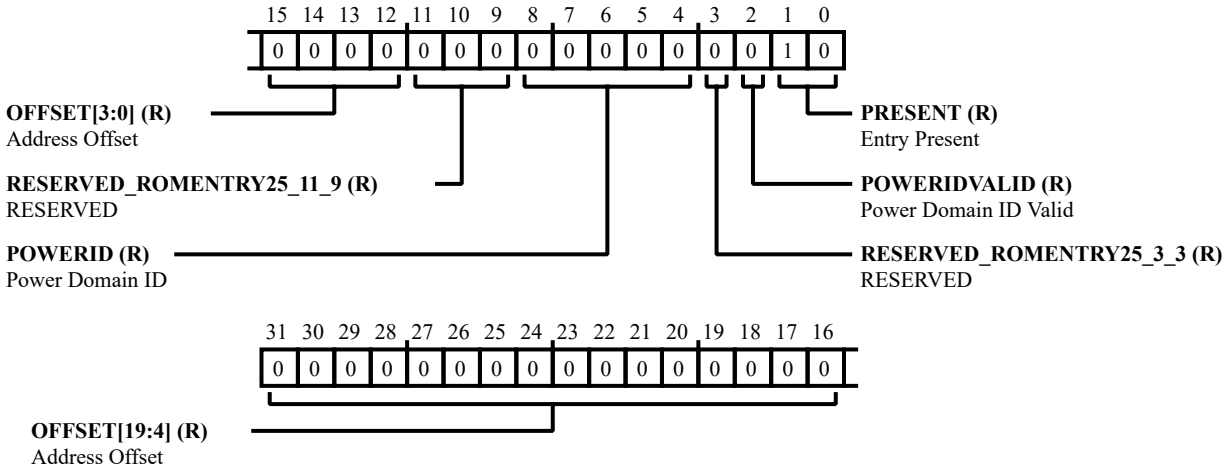


Figure 48-291: A55_CLUSTER_ROMENTRY25 Register Diagram

Table 48-289: A55_CLUSTER_ROMENTRY25 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY25_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY25_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-289: A55_CLUSTER_ROMENTRY25 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 26 Register

ROMENTRYn.

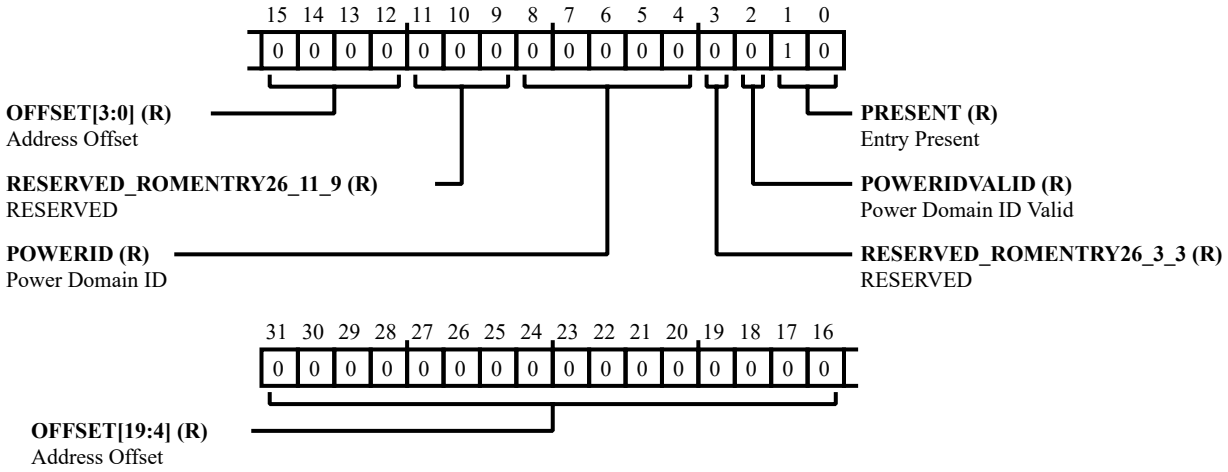


Figure 48-292: A55_CLUSTER_ROMENTRY26 Register Diagram

Table 48-290: A55_CLUSTER_ROMENTRY26 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY26_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY26_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-290: A55_CLUSTER_ROMENTRY26 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 27 Register

ROMENTRYn.

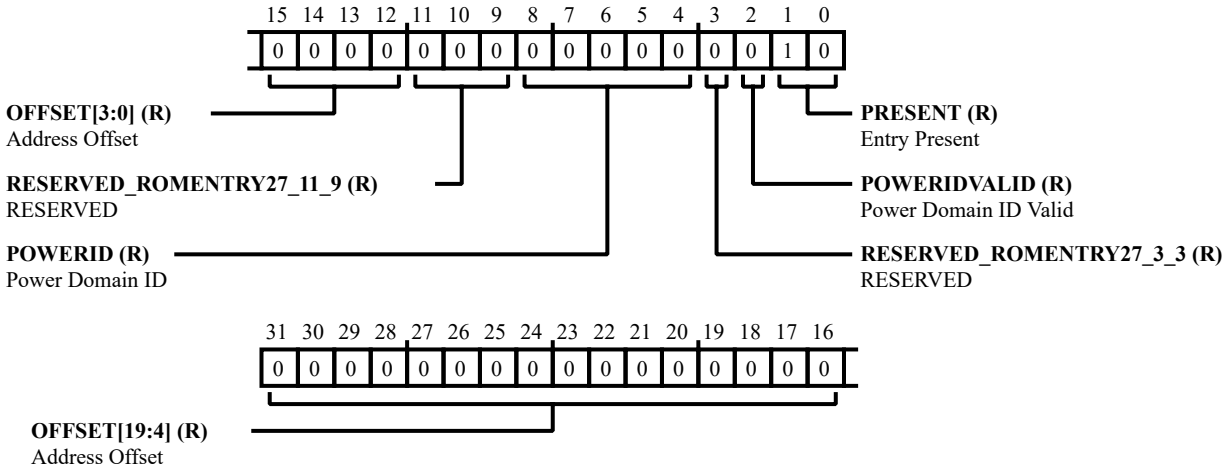


Figure 48-293: A55_CLUSTER_ROMENTRY27 Register Diagram

Table 48-291: A55_CLUSTER_ROMENTRY27 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY27_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY27_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-291: A55_CLUSTER_ROMENTRY27 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 28 Register

ROMENTRYn.

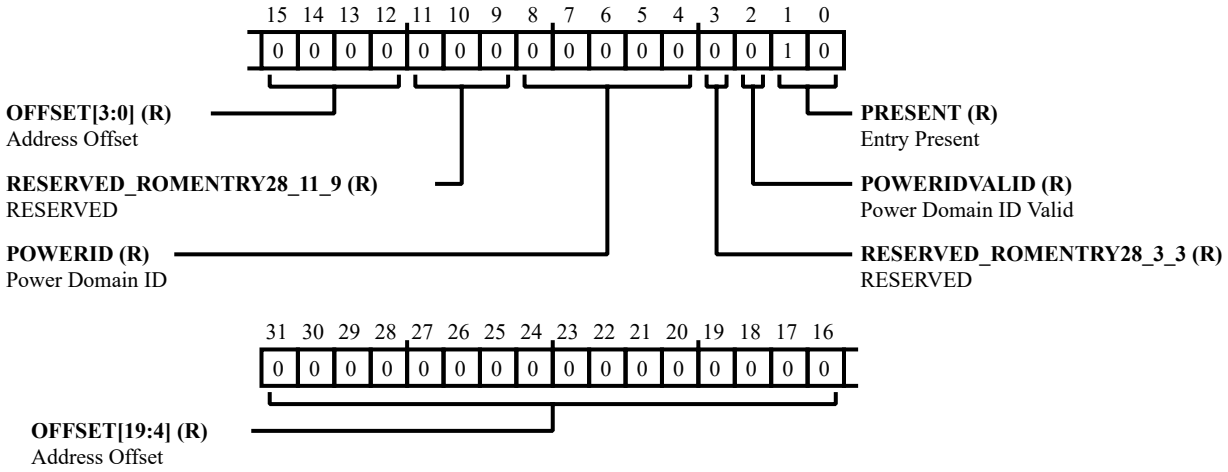


Figure 48-294: A55_CLUSTER_ROMENTRY28 Register Diagram

Table 48-292: A55_CLUSTER_ROMENTRY28 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY28_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY28_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-292: A55_CLUSTER_ROMENTRY28 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 29 Register

ROMENTRYn.

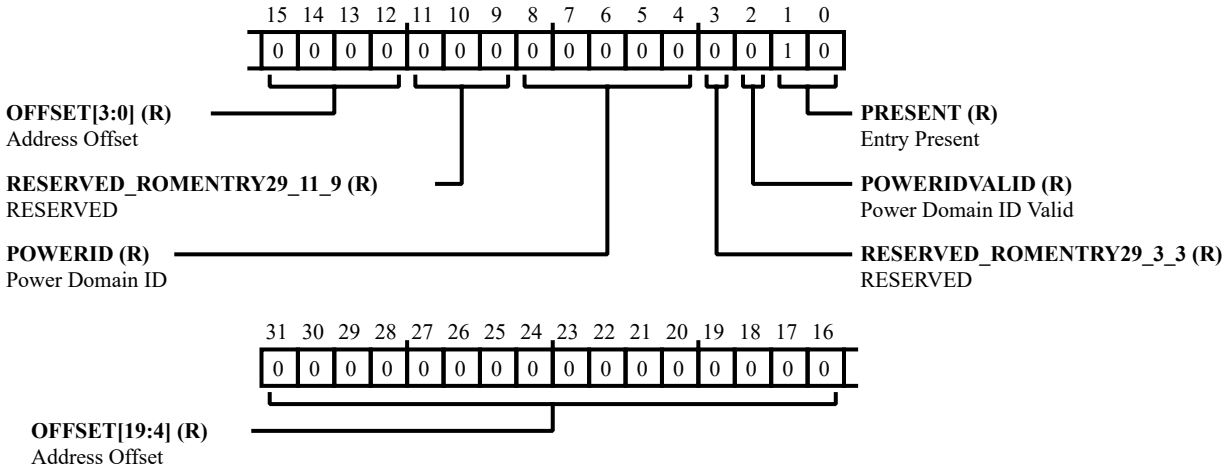


Figure 48-295: A55_CLUSTER_ROMENTRY29 Register Diagram

Table 48-293: A55_CLUSTER_ROMENTRY29 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY29_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY29_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-293: A55_CLUSTER_ROMENTRY29 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 3 Register

ROMENTRYn.

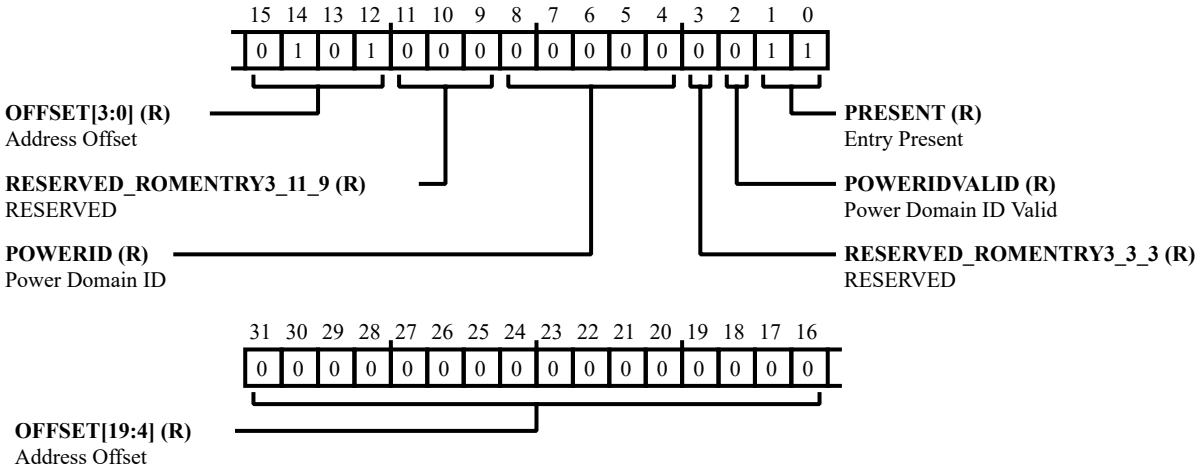


Figure 48-296: A55_CLUSTER_ROMENTRY3 Register Diagram

Table 48-294: A55_CLUSTER_ROMENTRY3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY3_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY3_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-294: A55_CLUSTER_ROMENTRY3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 30 Register

ROMENTRYn.

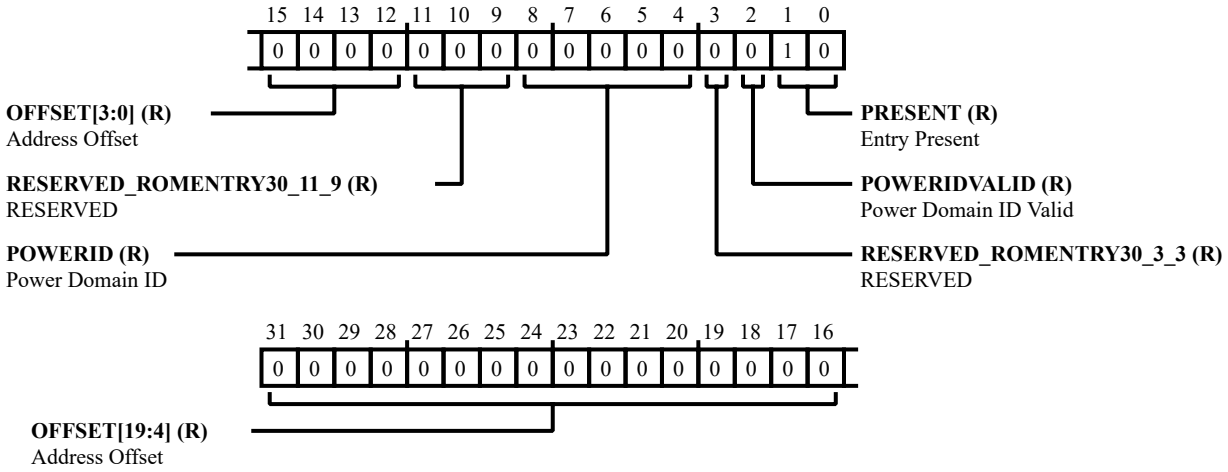


Figure 48-297: A55_CLUSTER_ROMENTRY30 Register Diagram

Table 48-295: A55_CLUSTER_ROMENTRY30 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY30_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY30_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-295: A55_CLUSTER_ROMENTRY30 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 31 Register

ROMENTRYn.

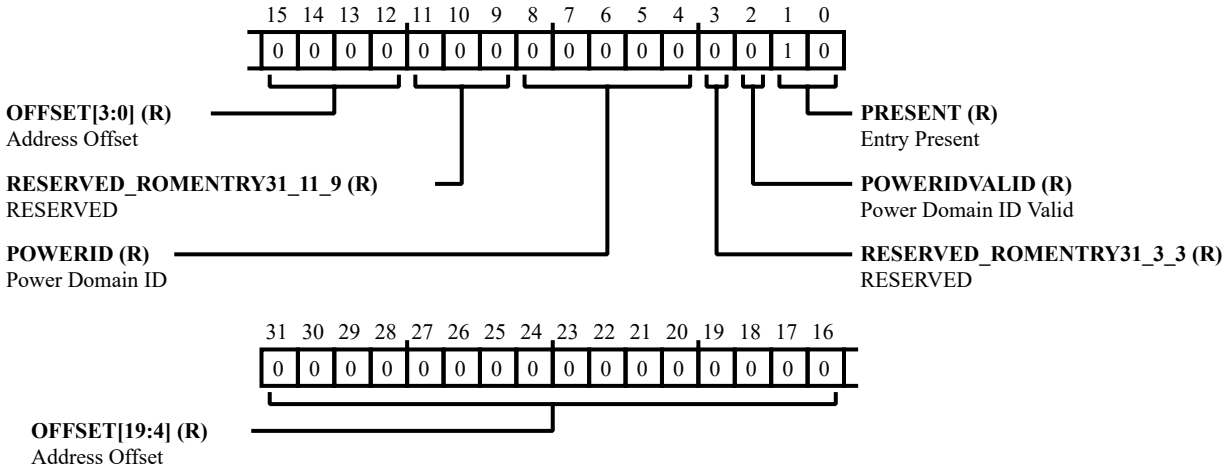


Figure 48-298: A55_CLUSTER_ROMENTRY31 Register Diagram

Table 48-296: A55_CLUSTER_ROMENTRY31 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY31_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY31_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-296: A55_CLUSTER_ROMENTRY31 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 32 Register

ROMENTRYn.

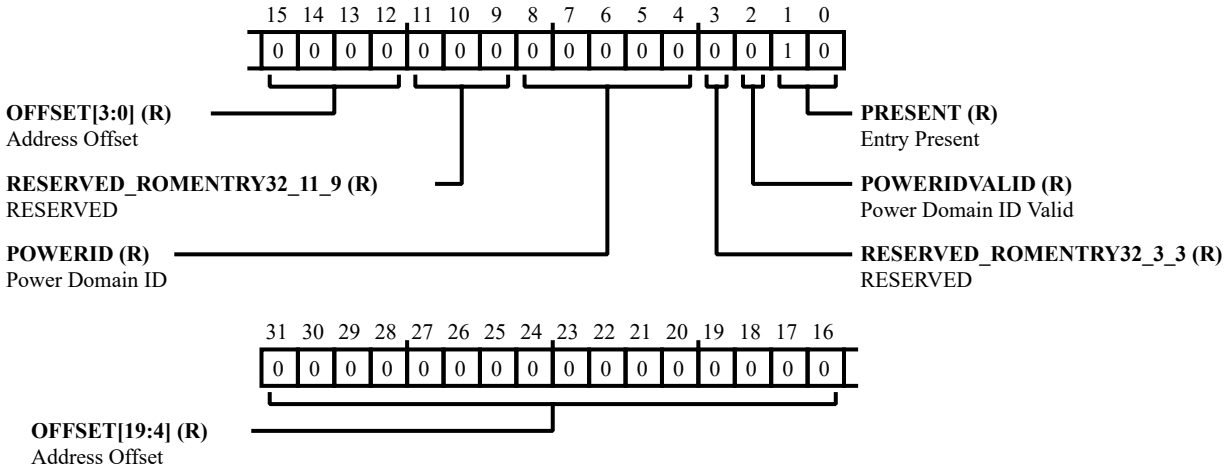


Figure 48-299: A55_CLUSTER_ROMENTRY32 Register Diagram

Table 48-297: A55_CLUSTER_ROMENTRY32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY32_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY32_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-297: A55_CLUSTER_ROMENTRY32 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 33 Register

ROMENTRYn.

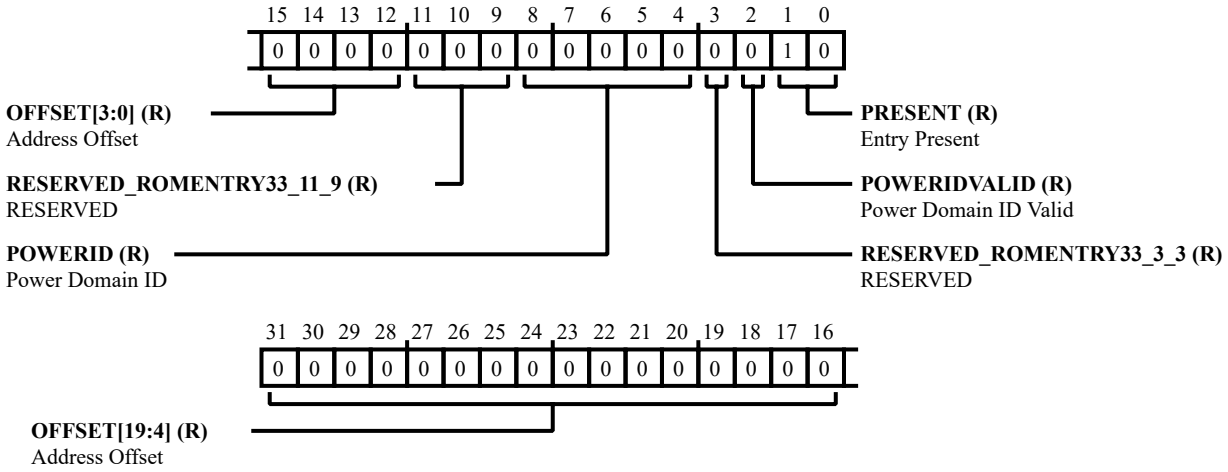


Figure 48-300: A55_CLUSTER_ROMENTRY33 Register Diagram

Table 48-298: A55_CLUSTER_ROMENTRY33 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY33_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY33_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-298: A55_CLUSTER_ROMENTRY33 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 34 Register

ROMENTRYn.

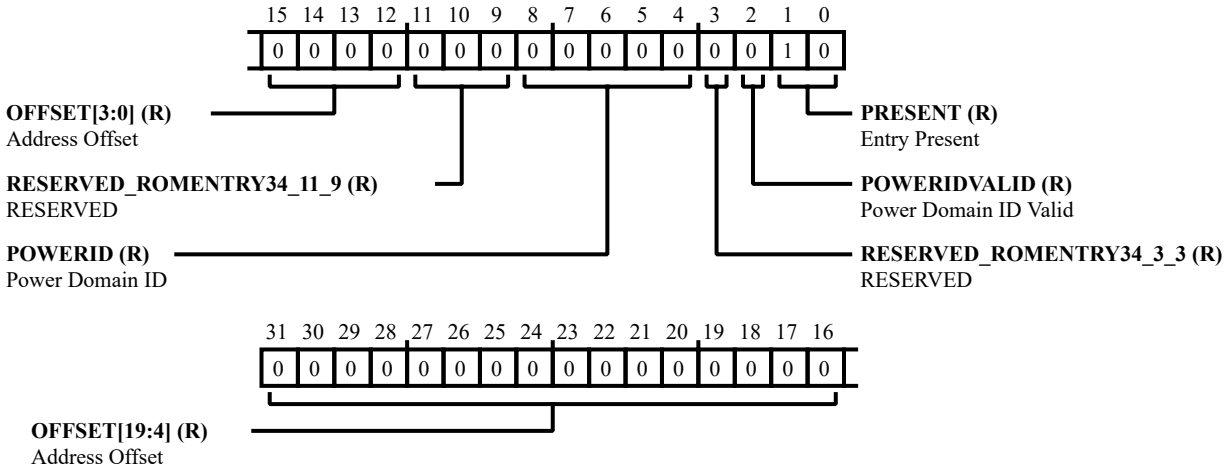


Figure 48-301: A55_CLUSTER_ROMENTRY34 Register Diagram

Table 48-299: A55_CLUSTER_ROMENTRY34 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY34_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY34_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-299: A55_CLUSTER_ROMENTRY34 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 35 Register

ROMENTRYn.

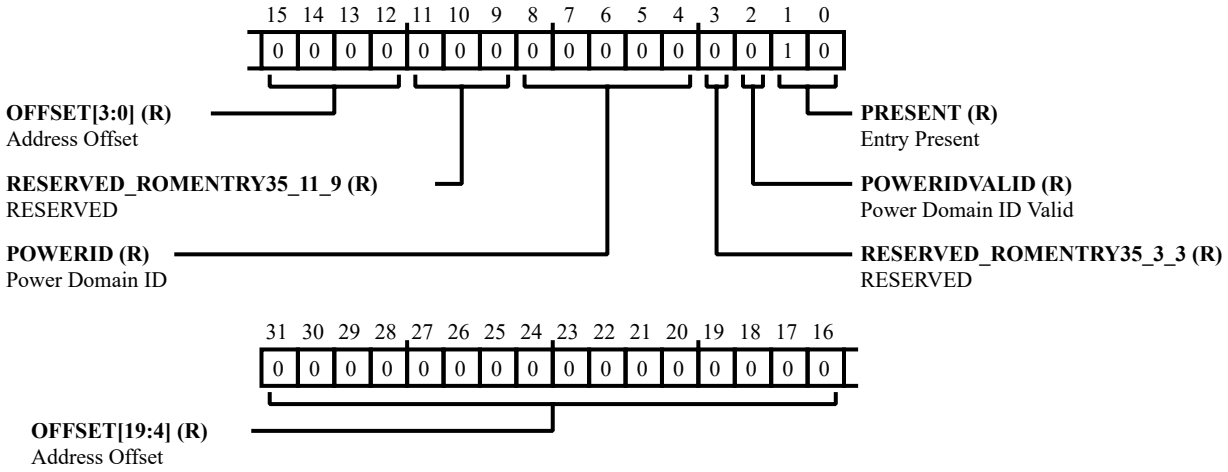


Figure 48-302: A55_CLUSTER_ROMENTRY35 Register Diagram

Table 48-300: A55_CLUSTER_ROMENTRY35 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY35_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY35_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-300: A55_CLUSTER_ROMENTRY35 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 36 Register

ROMENTRYn.

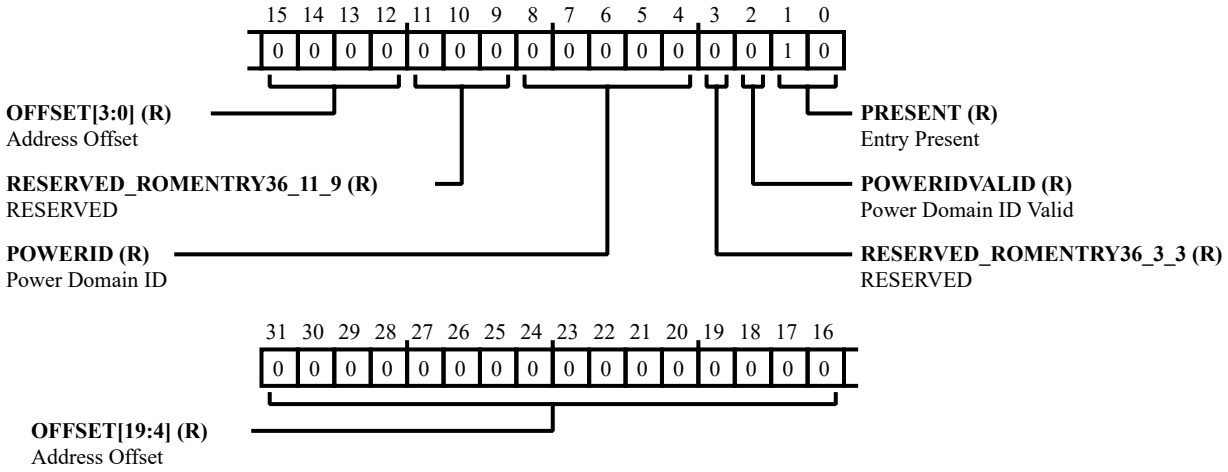


Figure 48-303: A55_CLUSTER_ROMENTRY36 Register Diagram

Table 48-301: A55_CLUSTER_ROMENTRY36 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY36_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY36_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-301: A55_CLUSTER_ROMENTRY36 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 37 Register

ROMENTRYn.

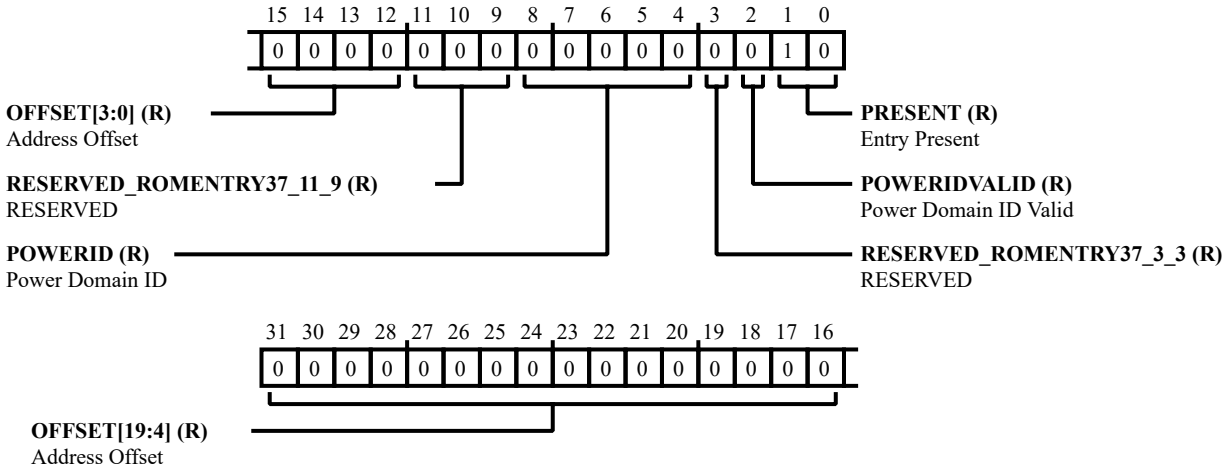


Figure 48-304: A55_CLUSTER_ROMENTRY37 Register Diagram

Table 48-302: A55_CLUSTER_ROMENTRY37 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY37_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY37_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-302: A55_CLUSTER_ROMENTRY37 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 38 Register

ROMENTRYn.

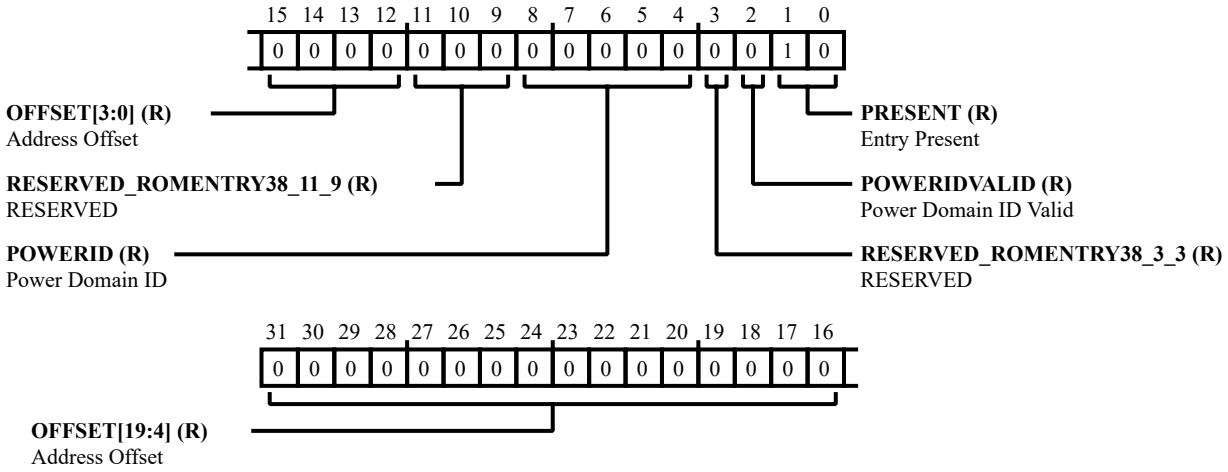


Figure 48-305: A55_CLUSTER_ROMENTRY38 Register Diagram

Table 48-303: A55_CLUSTER_ROMENTRY38 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY38_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY38_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-303: A55_CLUSTER_ROMENTRY38 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 39 Register

ROMENTRYn.

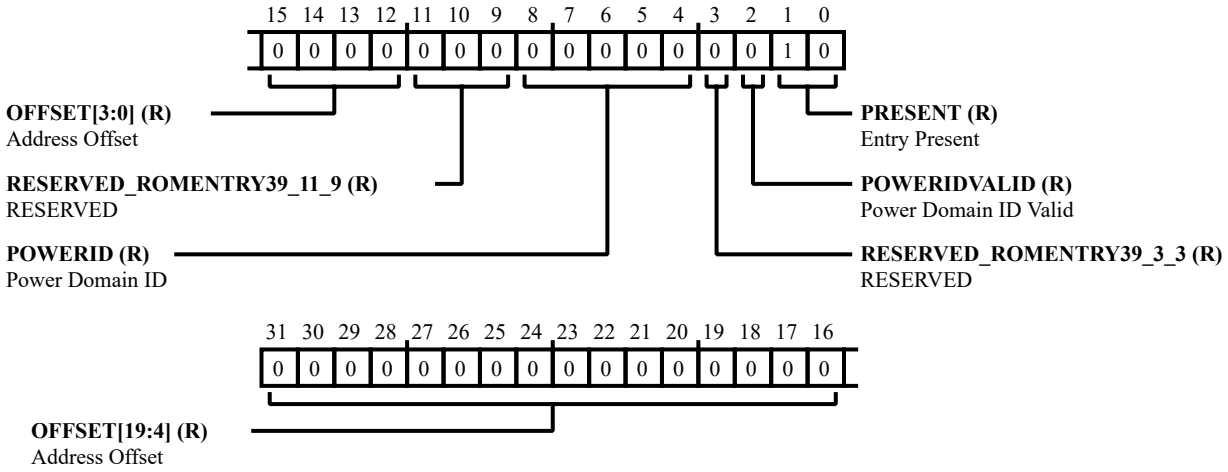


Figure 48-306: A55_CLUSTER_ROMENTRY39 Register Diagram

Table 48-304: A55_CLUSTER_ROMENTRY39 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY39_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY39_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-304: A55_CLUSTER_ROMENTRY39 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 4 Register

ROMENTRYn.

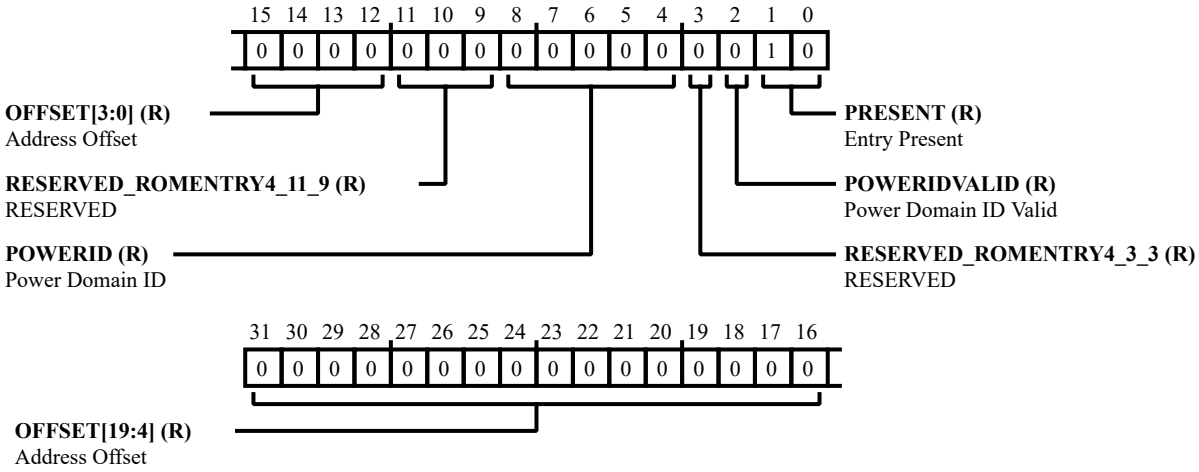


Figure 48-307: A55_CLUSTER_ROMENTRY4 Register Diagram

Table 48-305: A55_CLUSTER_ROMENTRY4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY4_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY4_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-305: A55_CLUSTER_ROMENTRY4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 40 Register

ROMENTRYn.

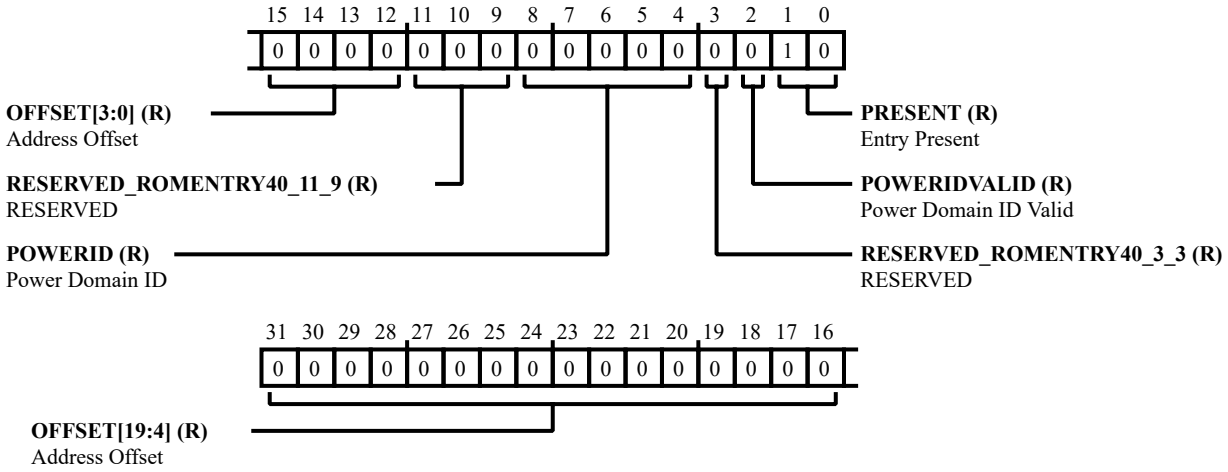


Figure 48-308: A55_CLUSTER_ROMENTRY40 Register Diagram

Table 48-306: A55_CLUSTER_ROMENTRY40 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY40_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY40_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-306: A55_CLUSTER_ROMENTRY40 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 41 Register

ROMENTRYn.

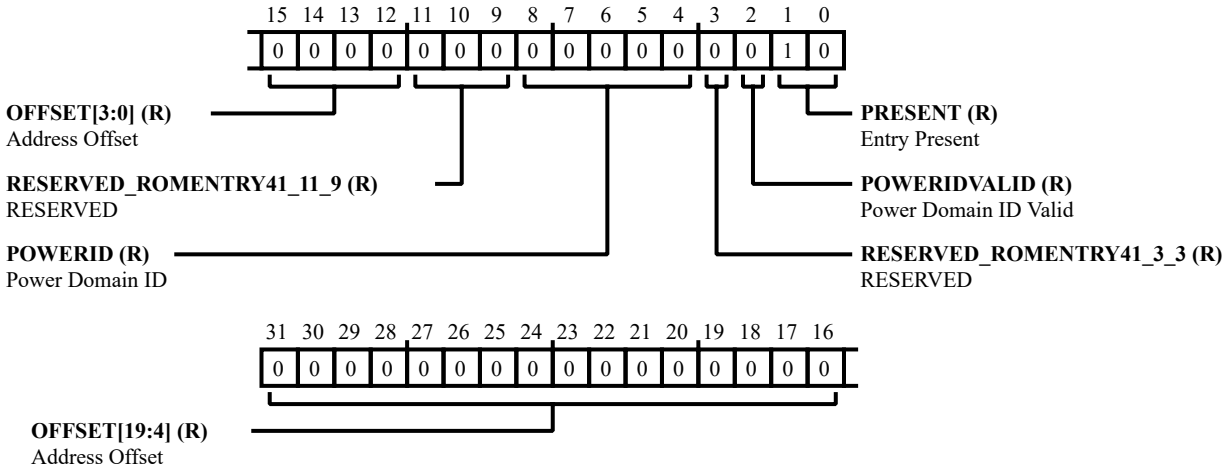


Figure 48-309: A55_CLUSTER_ROMENTRY41 Register Diagram

Table 48-307: A55_CLUSTER_ROMENTRY41 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY41_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY41_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-307: A55_CLUSTER_ROMENTRY41 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 42 Register

ROMENTRYn.

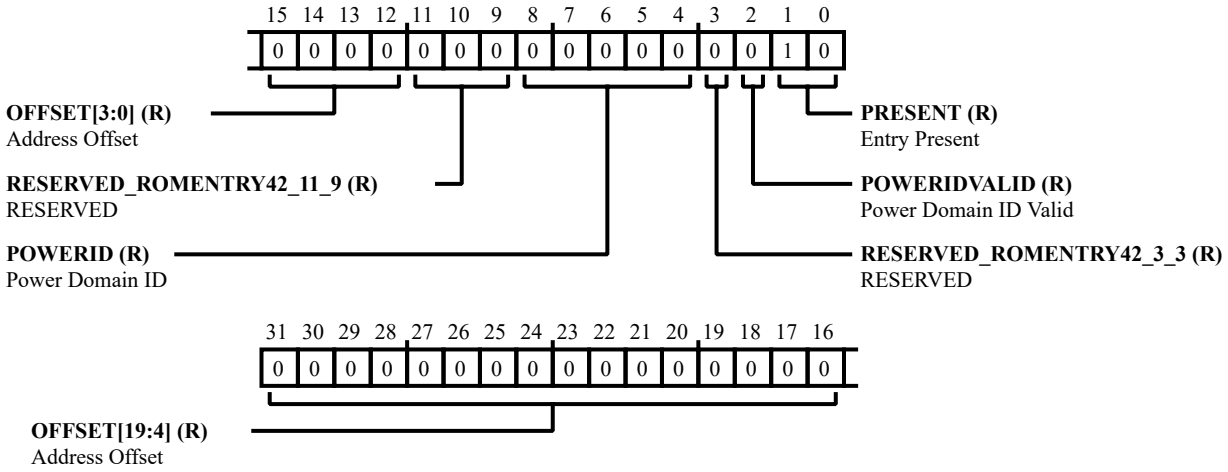


Figure 48-310: A55_CLUSTER_ROMENTRY42 Register Diagram

Table 48-308: A55_CLUSTER_ROMENTRY42 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY42_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY42_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-308: A55_CLUSTER_ROMENTRY42 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 43 Register

ROMENTRYn.

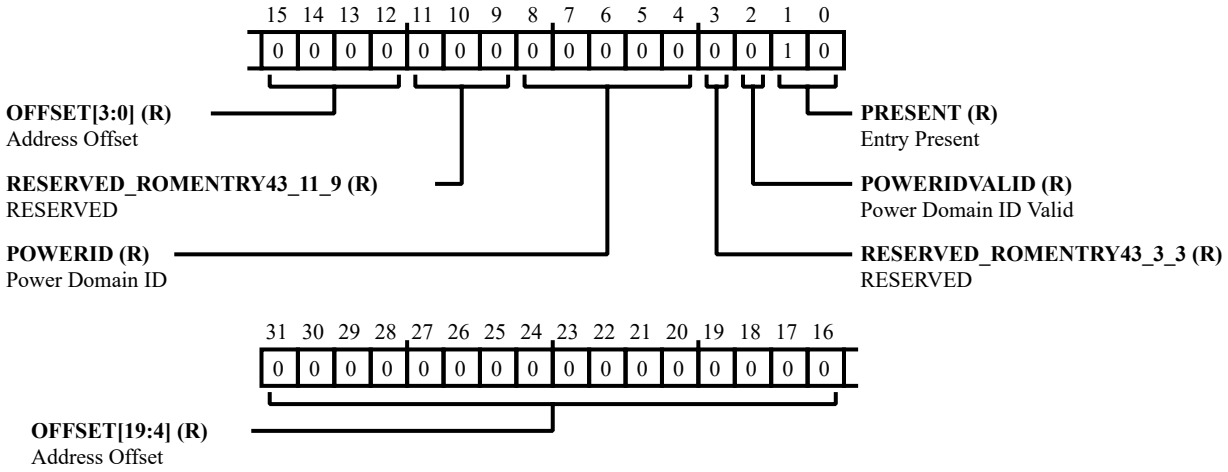


Figure 48-311: A55_CLUSTER_ROMENTRY43 Register Diagram

Table 48-309: A55_CLUSTER_ROMENTRY43 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY43_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY43_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-309: A55_CLUSTER_ROMENTRY43 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 44 Register

ROMENTRYn.

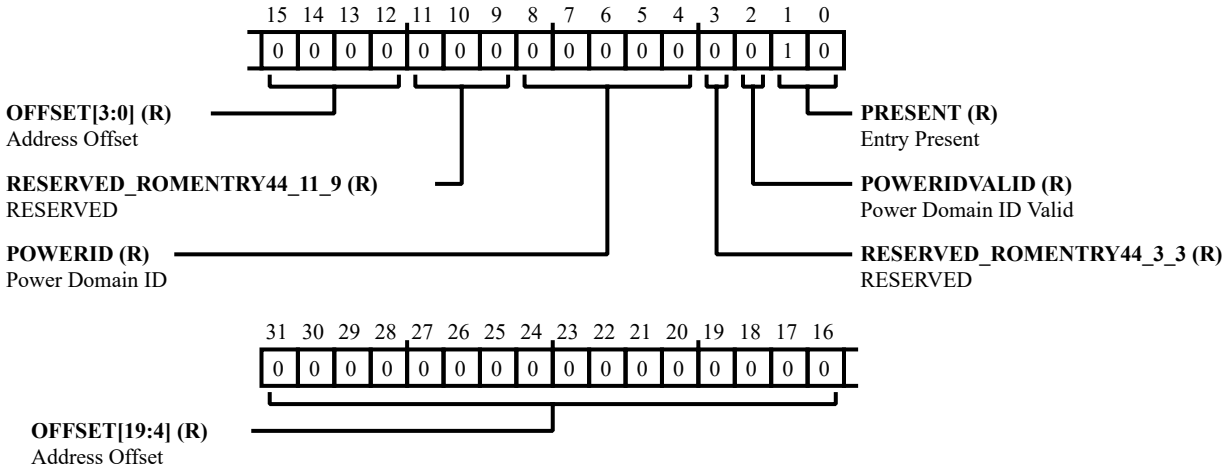


Figure 48-312: A55_CLUSTER_ROMENTRY44 Register Diagram

Table 48-310: A55_CLUSTER_ROMENTRY44 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY44_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY44_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-310: A55_CLUSTER_ROMENTRY44 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 45 Register

ROMENTRYn.

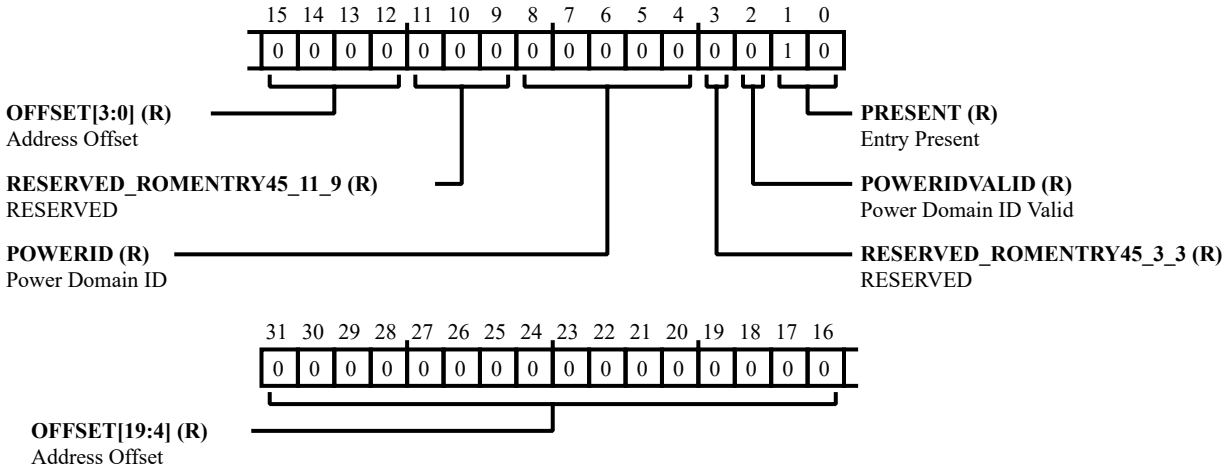


Figure 48-313: A55_CLUSTER_ROMENTRY45 Register Diagram

Table 48-311: A55_CLUSTER_ROMENTRY45 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY45_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY45_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-311: A55_CLUSTER_ROMENTRY45 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 46 Register

ROMENTRYn.

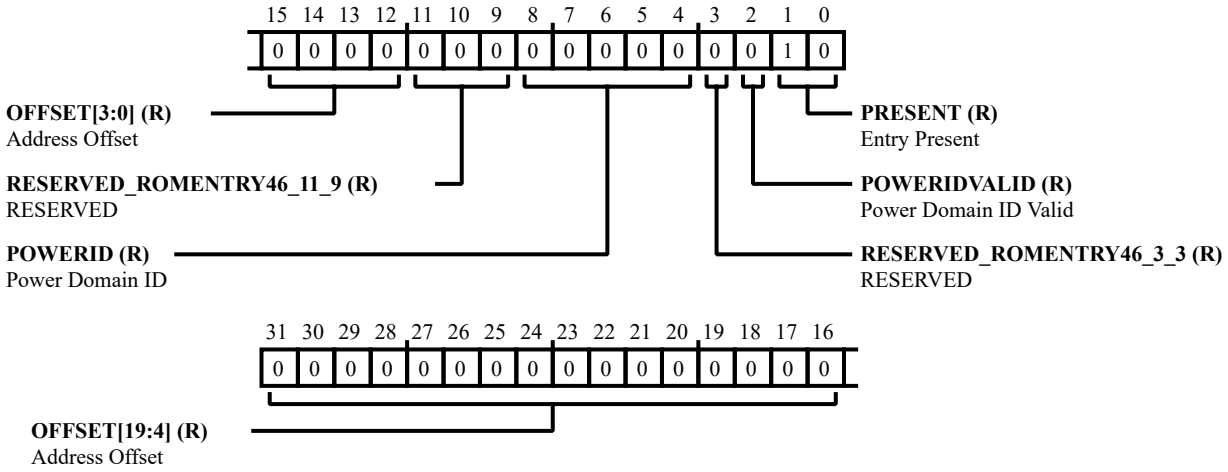


Figure 48-314: A55_CLUSTER_ROMENTRY46 Register Diagram

Table 48-312: A55_CLUSTER_ROMENTRY46 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY46_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY46_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-312: A55_CLUSTER_ROMENTRY46 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 47 Register

ROMENTRYn.

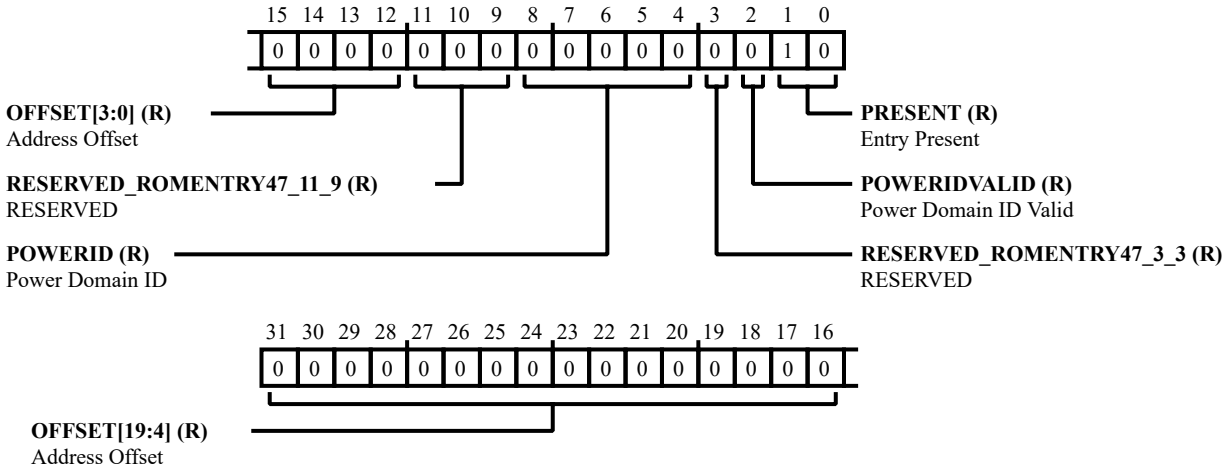


Figure 48-315: A55_CLUSTER_ROMENTRY47 Register Diagram

Table 48-313: A55_CLUSTER_ROMENTRY47 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY47_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY47_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-313: A55_CLUSTER_ROMENTRY47 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 48 Register

ROMENTRYn.

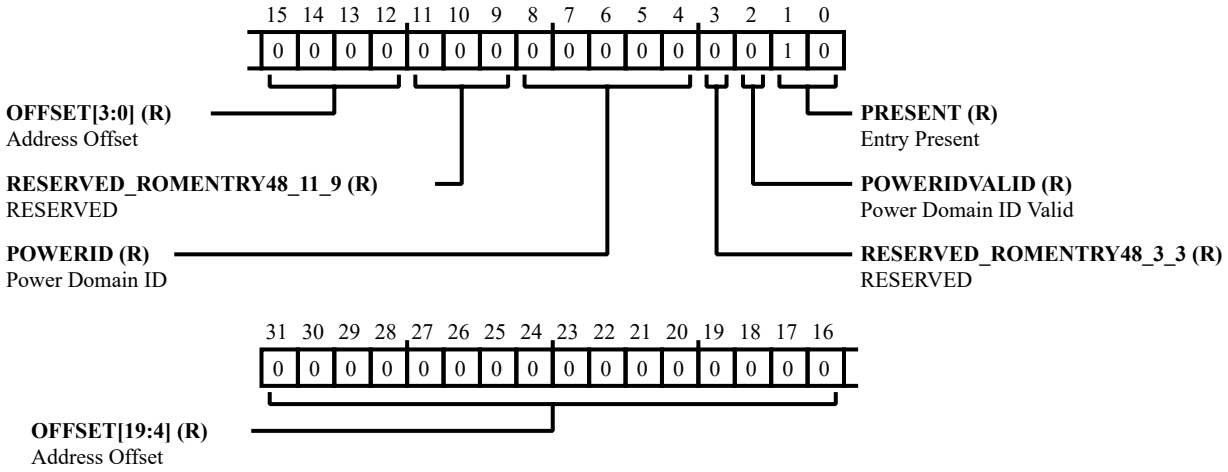


Figure 48-316: A55_CLUSTER_ROMENTRY48 Register Diagram

Table 48-314: A55_CLUSTER_ROMENTRY48 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY48_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY48_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-314: A55_CLUSTER_ROMENTRY48 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 49 Register

ROMENTRYn.

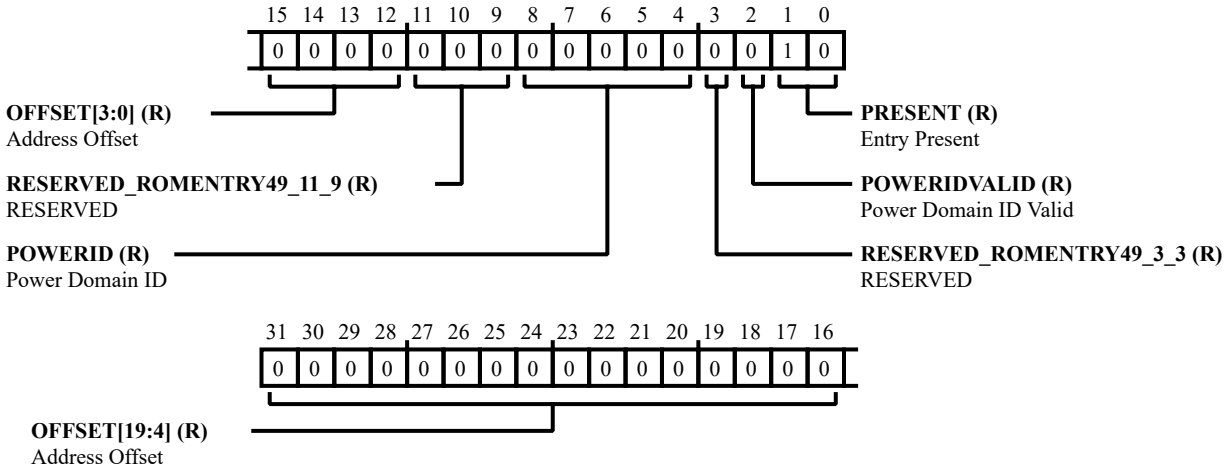


Figure 48-317: A55_CLUSTER_ROMENTRY49 Register Diagram

Table 48-315: A55_CLUSTER_ROMENTRY49 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY49_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY49_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-315: A55_CLUSTER_ROMENTRY49 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 5 Register

ROMENTRYn.

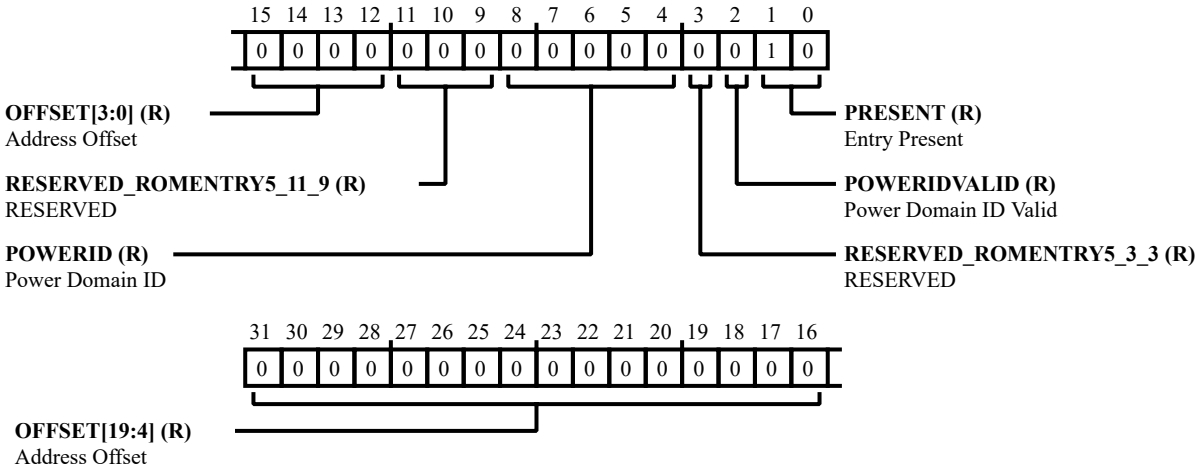


Figure 48-318: A55_CLUSTER_ROMENTRY5 Register Diagram

Table 48-316: A55_CLUSTER_ROMENTRY5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY5_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY5_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-316: A55_CLUSTER_ROMENTRY5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 50 Register

ROMENTRYn.

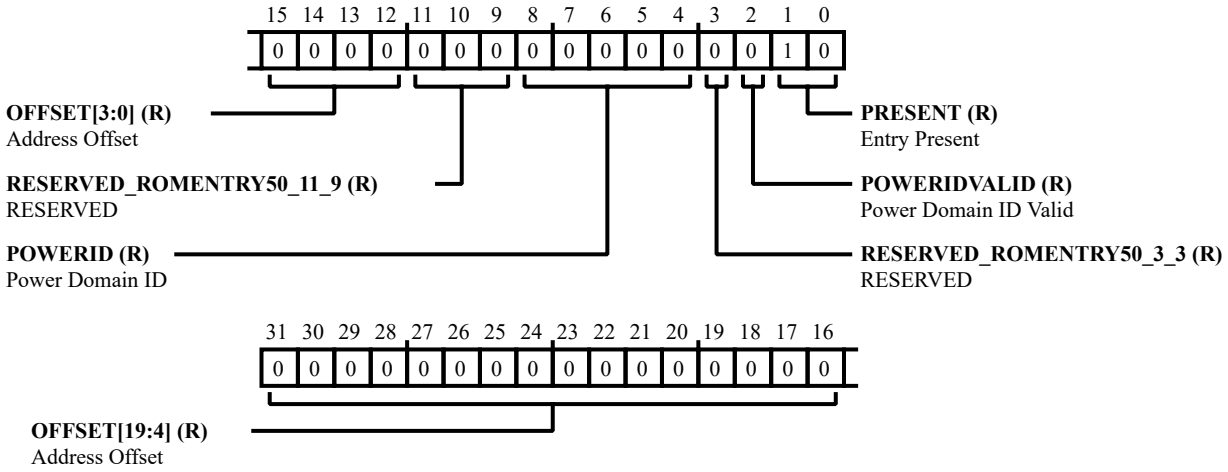


Figure 48-319: A55_CLUSTER_ROMENTRY50 Register Diagram

Table 48-317: A55_CLUSTER_ROMENTRY50 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY50_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY50_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-317: A55_CLUSTER_ROMENTRY50 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 51 Register

ROMENTRYn.

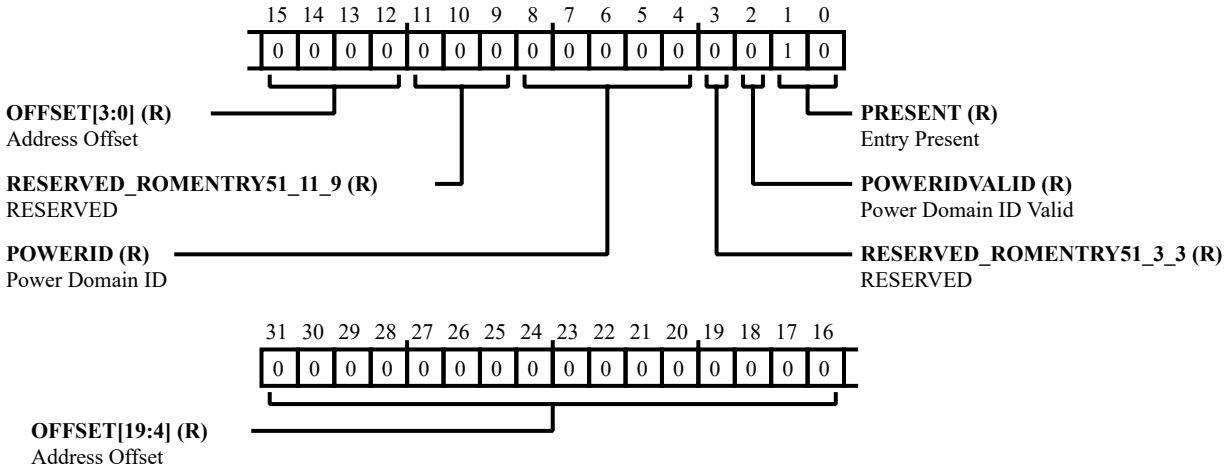


Figure 48-320: A55_CLUSTER_ROMENTRY51 Register Diagram

Table 48-318: A55_CLUSTER_ROMENTRY51 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY51_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY51_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-318: A55_CLUSTER_ROMENTRY51 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 52 Register

ROMENTRYn.

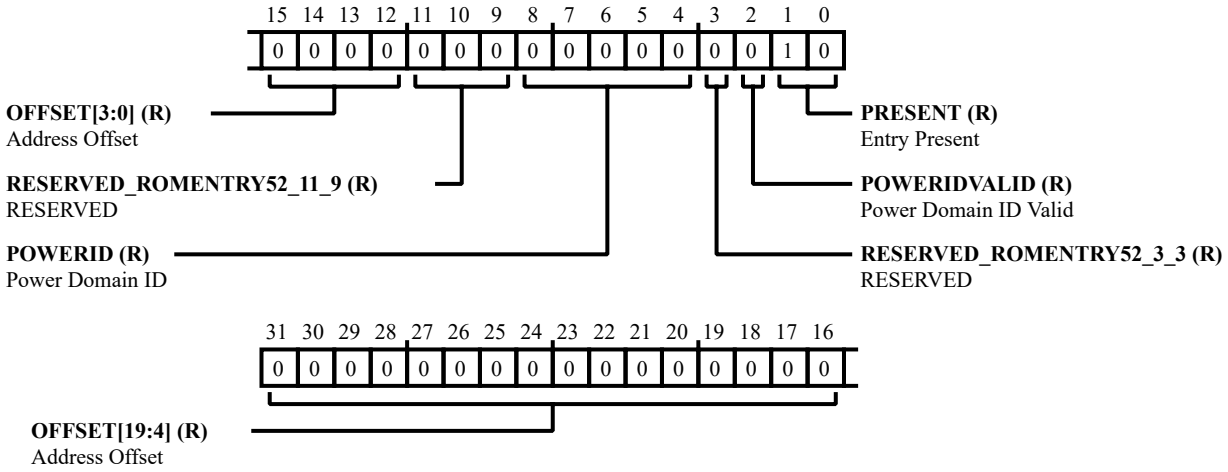


Figure 48-321: A55_CLUSTER_ROMENTRY52 Register Diagram

Table 48-319: A55_CLUSTER_ROMENTRY52 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY52_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY52_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-319: A55_CLUSTER_ROMENTRY52 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 53 Register

ROMENTRYn.

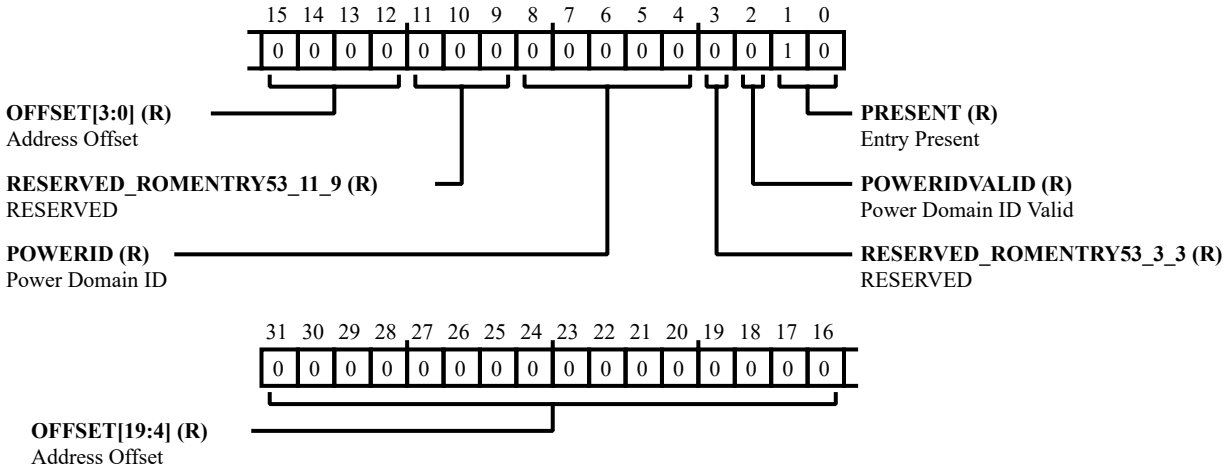


Figure 48-322: A55_CLUSTER_ROMENTRY53 Register Diagram

Table 48-320: A55_CLUSTER_ROMENTRY53 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY53_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY53_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-320: A55_CLUSTER_ROMENTRY53 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 54 Register

ROMENTRYn.

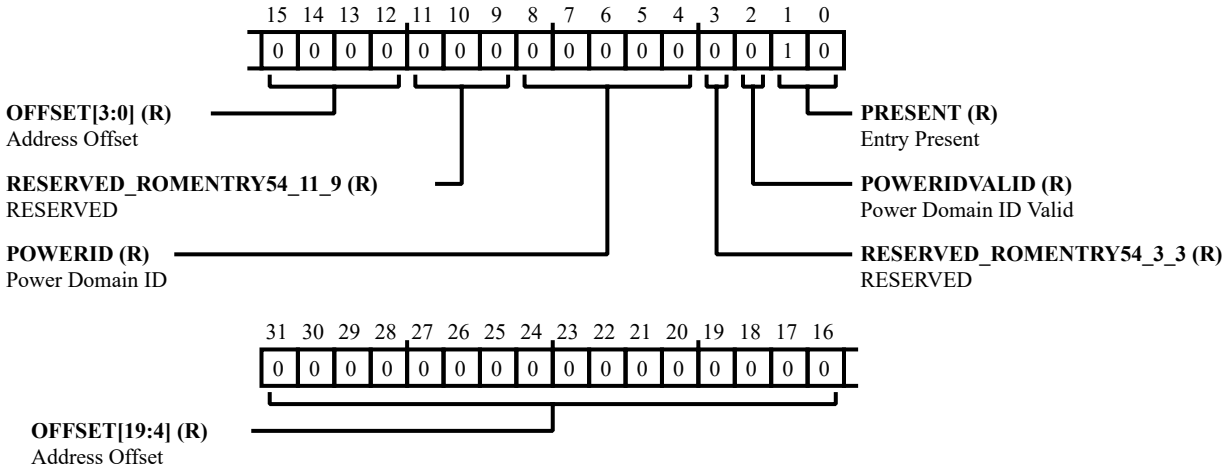


Figure 48-323: A55_CLUSTER_ROMENTRY54 Register Diagram

Table 48-321: A55_CLUSTER_ROMENTRY54 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY54_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY54_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-321: A55_CLUSTER_ROMENTRY54 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 55 Register

ROMENTRYn.

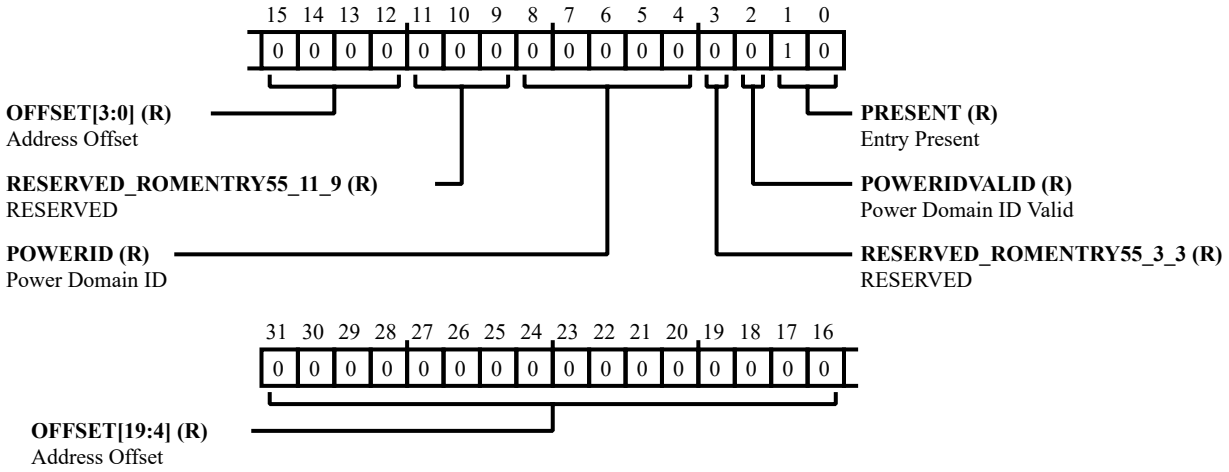


Figure 48-324: A55_CLUSTER_ROMENTRY55 Register Diagram

Table 48-322: A55_CLUSTER_ROMENTRY55 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY55_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY55_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-322: A55_CLUSTER_ROMENTRY55 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 56 Register

ROMENTRYn.

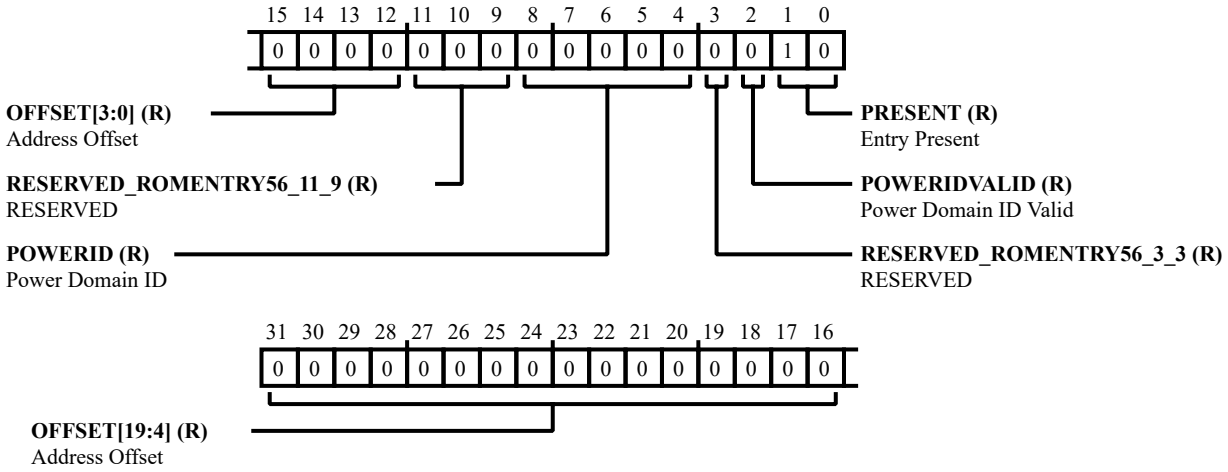


Figure 48-325: A55_CLUSTER_ROMENTRY56 Register Diagram

Table 48-323: A55_CLUSTER_ROMENTRY56 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY56_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY56_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-323: A55_CLUSTER_ROMENTRY56 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 57 Register

ROMENTRYn.

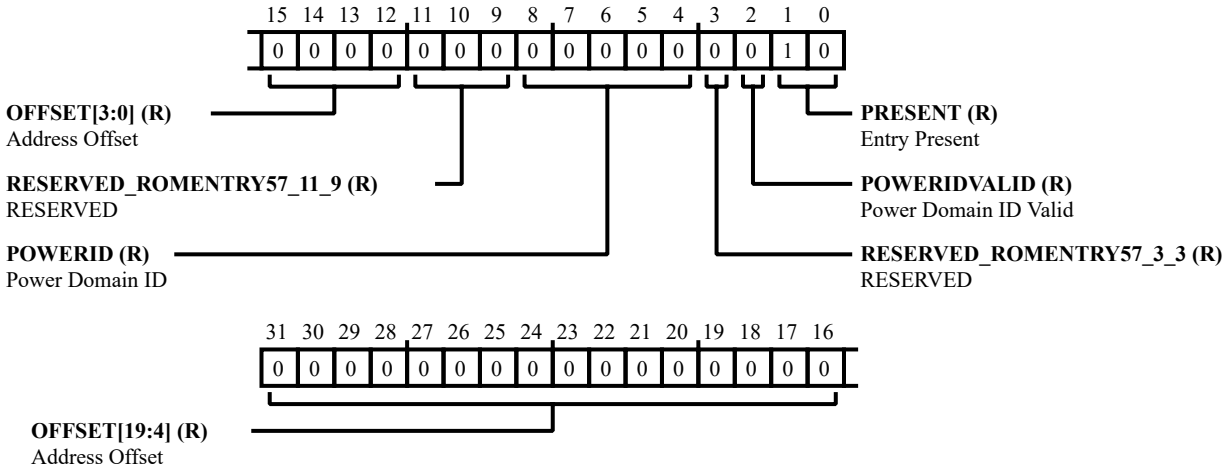


Figure 48-326: A55_CLUSTER_ROMENTRY57 Register Diagram

Table 48-324: A55_CLUSTER_ROMENTRY57 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY57_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY57_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-324: A55_CLUSTER_ROMENTRY57 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 58 Register

ROMENTRYn.

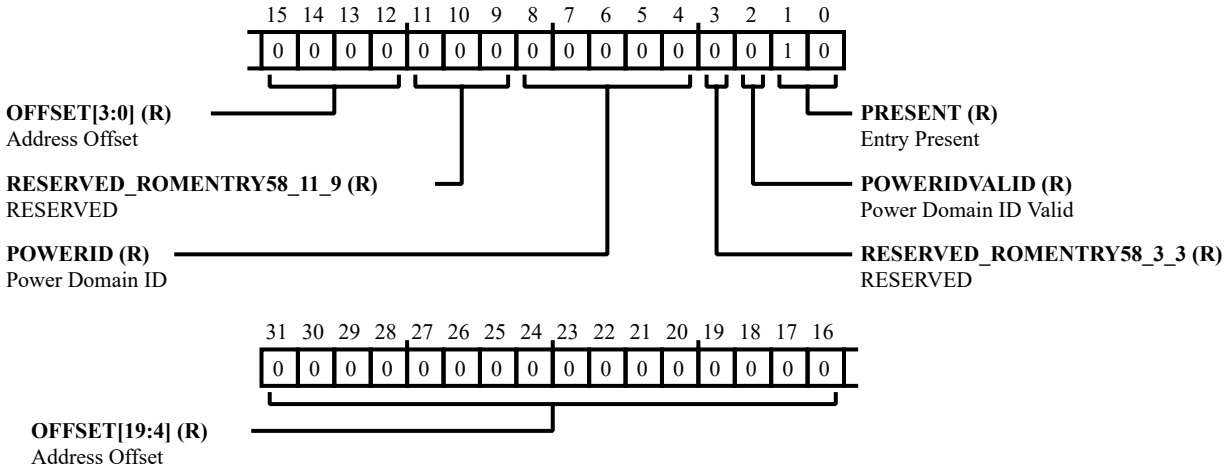


Figure 48-327: A55_CLUSTER_ROMENTRY58 Register Diagram

Table 48-325: A55_CLUSTER_ROMENTRY58 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY58_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY58_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-325: A55_CLUSTER_ROMENTRY58 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 59 Register

ROMENTRYn.

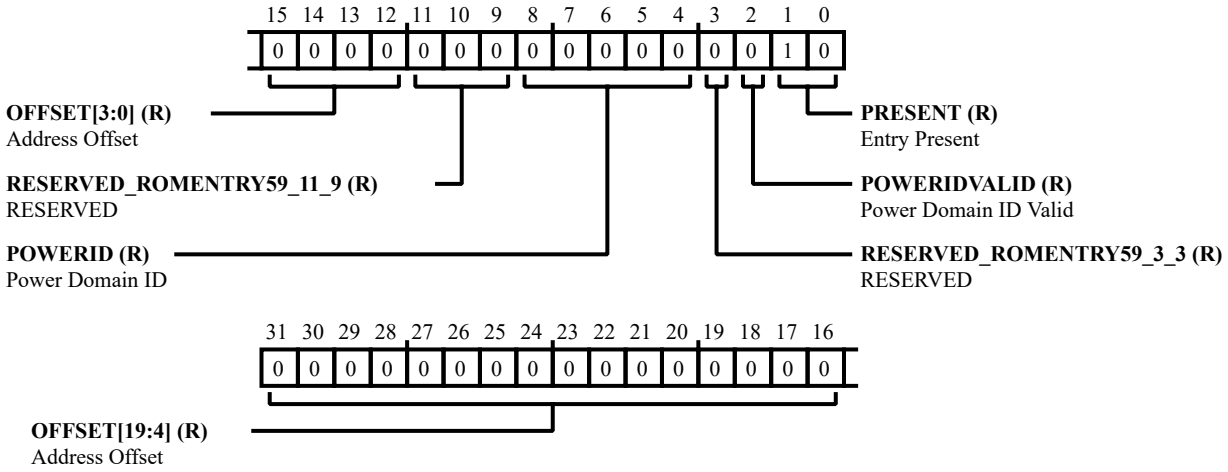


Figure 48-328: A55_CLUSTER_ROMENTRY59 Register Diagram

Table 48-326: A55_CLUSTER_ROMENTRY59 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY59_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY59_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-326: A55_CLUSTER_ROMENTRY59 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 6 Register

ROMENTRYn.

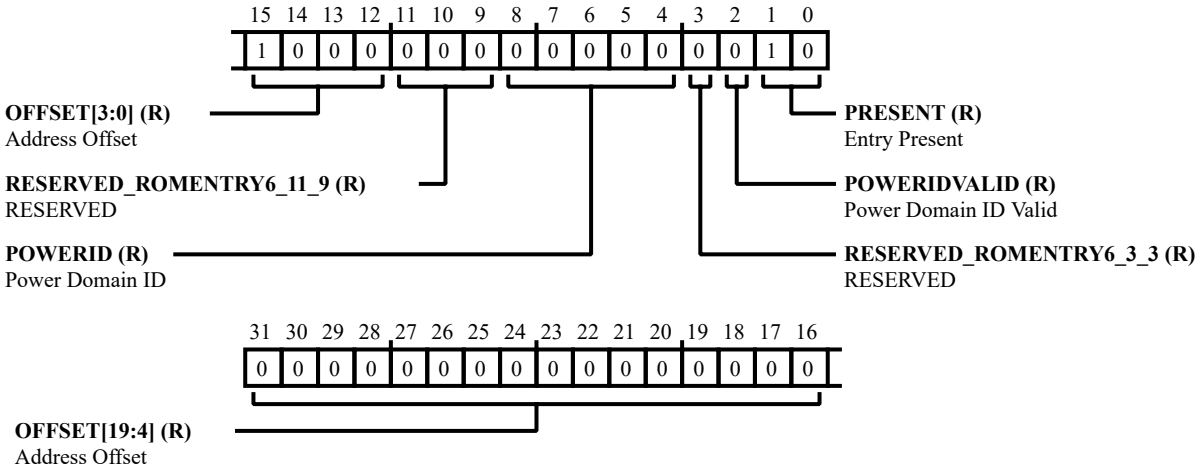


Figure 48-329: A55_CLUSTER_ROMENTRY6 Register Diagram

Table 48-327: A55_CLUSTER_ROMENTRY6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY6_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY6_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-327: A55_CLUSTER_ROMENTRY6 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 60 Register

ROMENTRYn.

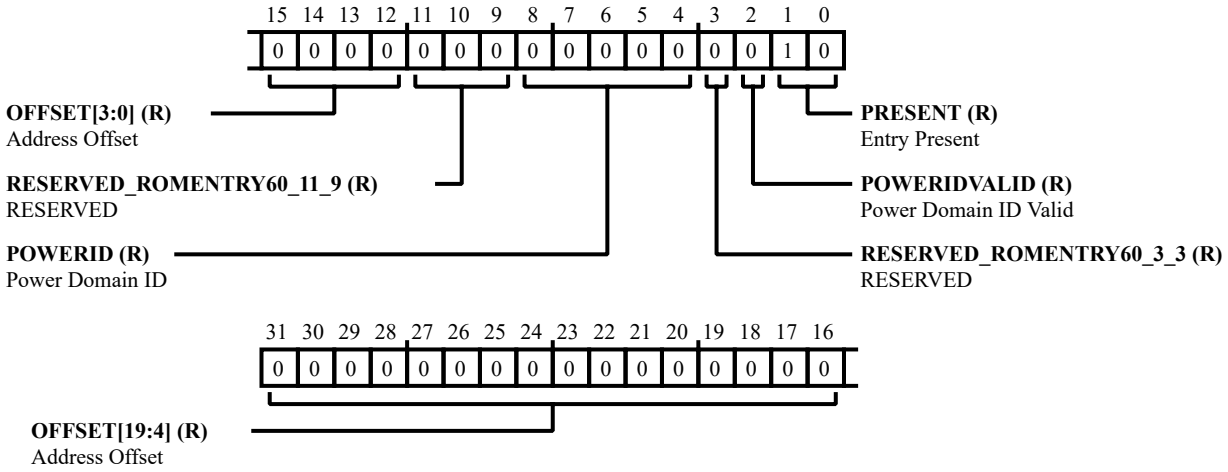


Figure 48-330: A55_CLUSTER_ROMENTRY60 Register Diagram

Table 48-328: A55_CLUSTER_ROMENTRY60 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY60_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY60_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-328: A55_CLUSTER_ROMENTRY60 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 61 Register

ROMENTRYn.

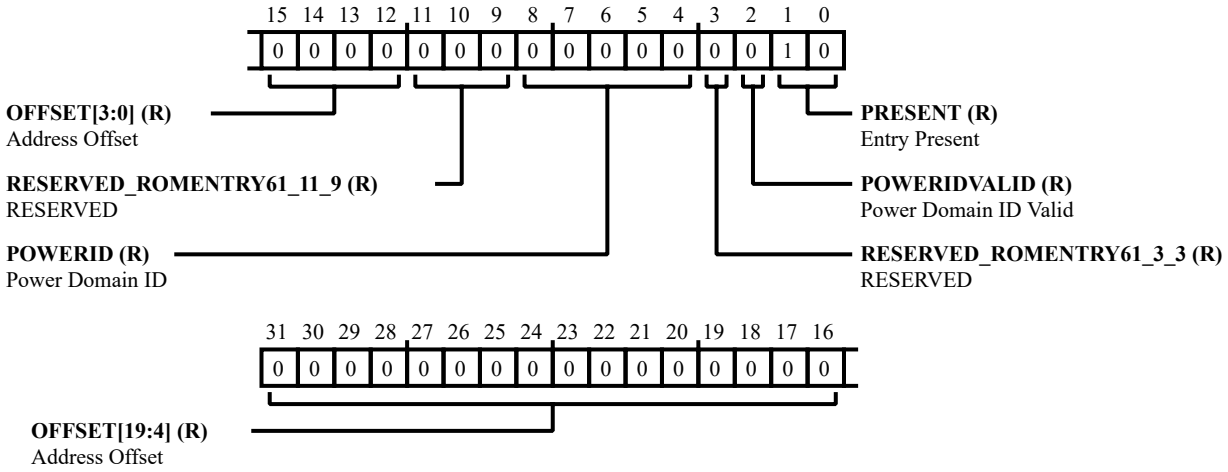


Figure 48-331: A55_CLUSTER_ROMENTRY61 Register Diagram

Table 48-329: A55_CLUSTER_ROMENTRY61 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY61_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY61_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-329: A55_CLUSTER_ROMENTRY61 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 62 Register

ROMENTRYn.

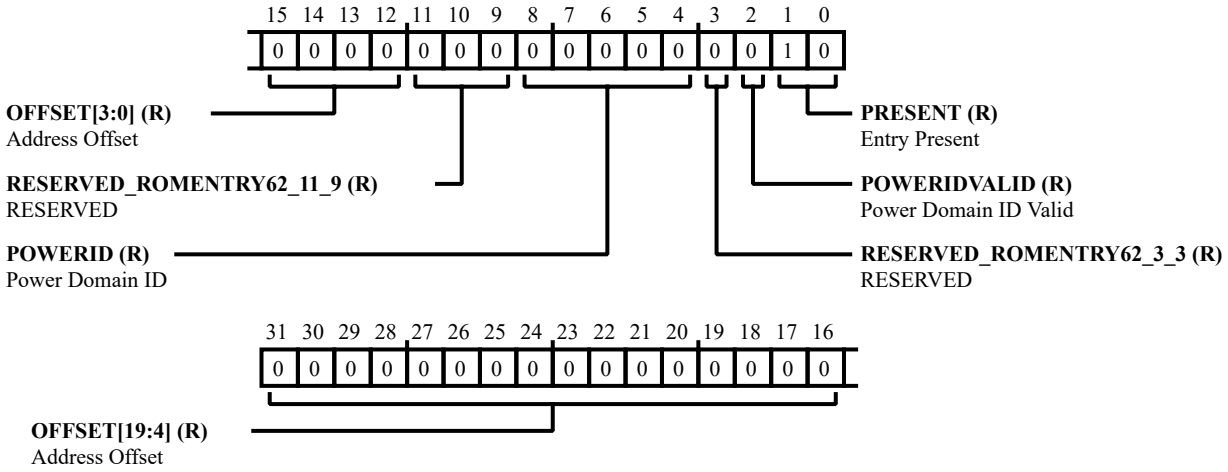


Figure 48-332: A55_CLUSTER_ROMENTRY62 Register Diagram

Table 48-330: A55_CLUSTER_ROMENTRY62 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY62_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY62_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-330: A55_CLUSTER_ROMENTRY62 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 63 Register

ROMENTRYn.

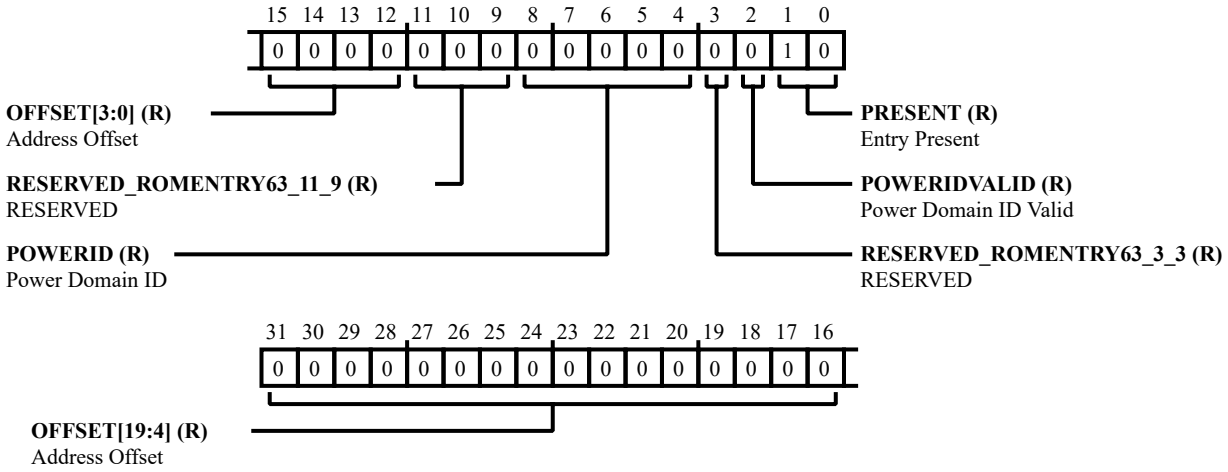


Figure 48-333: A55_CLUSTER_ROMENTRY63 Register Diagram

Table 48-331: A55_CLUSTER_ROMENTRY63 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY63_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY63_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-331: A55_CLUSTER_ROMENTRY63 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 64 Register

ROMENTRYn.

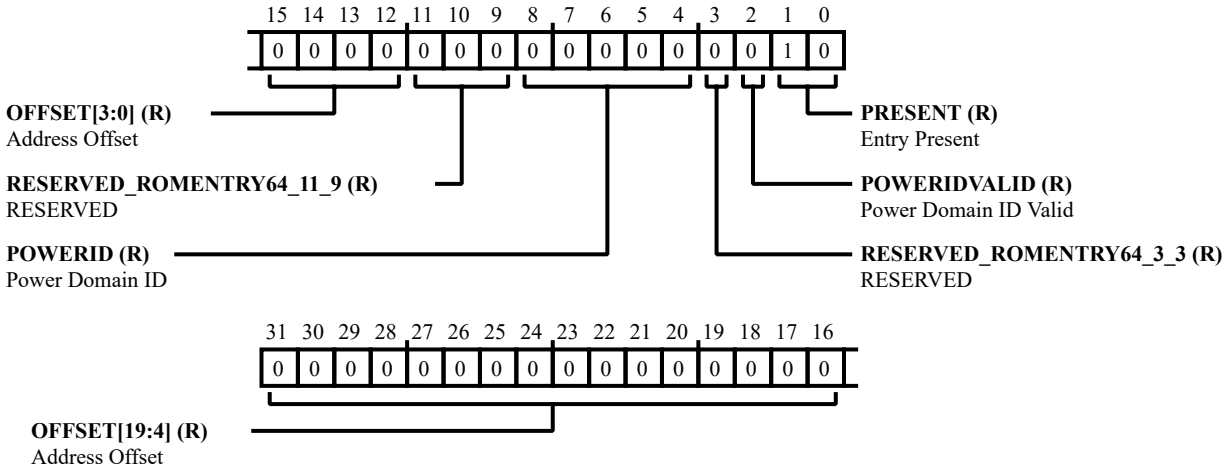


Figure 48-334: A55_CLUSTER_ROMENTRY64 Register Diagram

Table 48-332: A55_CLUSTER_ROMENTRY64 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY64_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY64_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-332: A55_CLUSTER_ROMENTRY64 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 65 Register

ROMENTRYn.

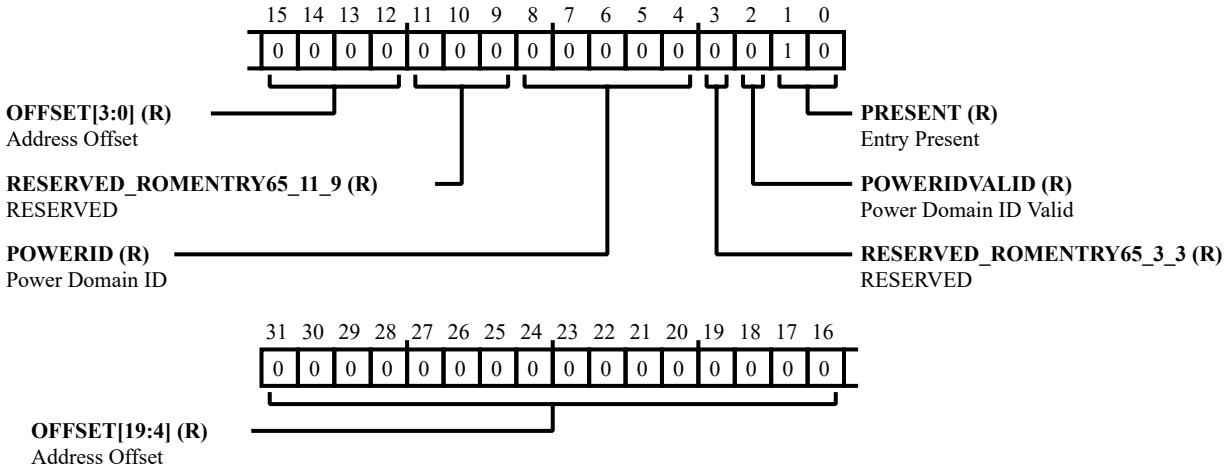


Figure 48-335: A55_CLUSTER_ROMENTRY65 Register Diagram

Table 48-333: A55_CLUSTER_ROMENTRY65 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY65_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY65_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-333: A55_CLUSTER_ROMENTRY65 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 66 Register

ROMENTRYn.

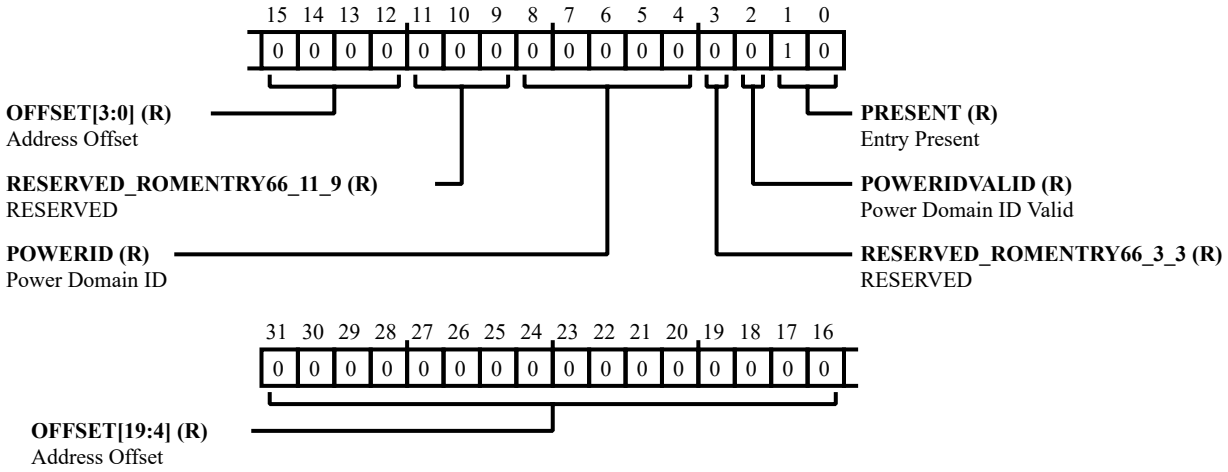


Figure 48-336: A55_CLUSTER_ROMENTRY66 Register Diagram

Table 48-334: A55_CLUSTER_ROMENTRY66 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY66_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY66_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-334: A55_CLUSTER_ROMENTRY66 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 67 Register

ROMENTRYn.

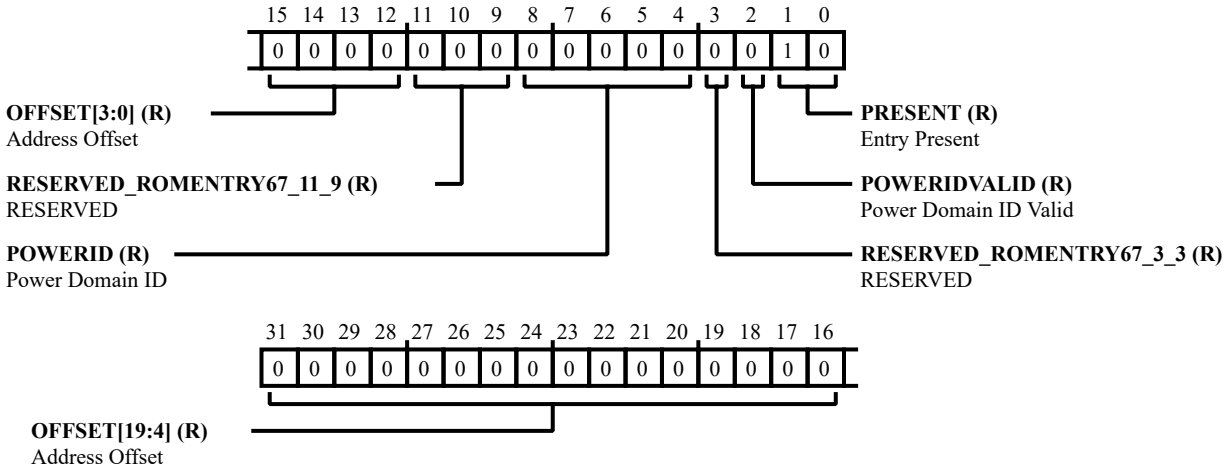


Figure 48-337: A55_CLUSTER_ROMENTRY67 Register Diagram

Table 48-335: A55_CLUSTER_ROMENTRY67 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY67_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY67_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-335: A55_CLUSTER_ROMENTRY67 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 68 Register

ROMENTRYn.

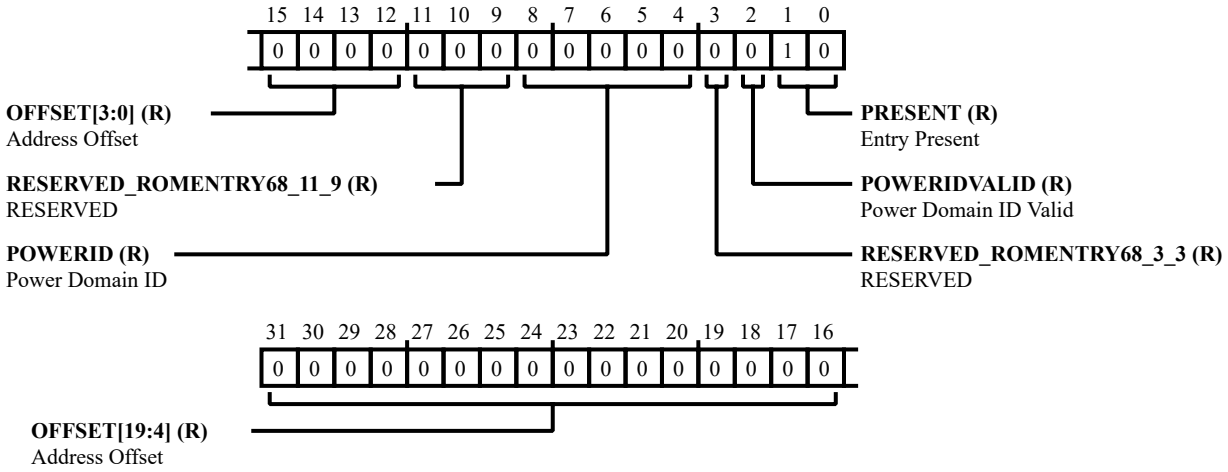


Figure 48-338: A55_CLUSTER_ROMENTRY68 Register Diagram

Table 48-336: A55_CLUSTER_ROMENTRY68 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY68_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY68_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-336: A55_CLUSTER_ROMENTRY68 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 69 Register

ROMENTRYn.

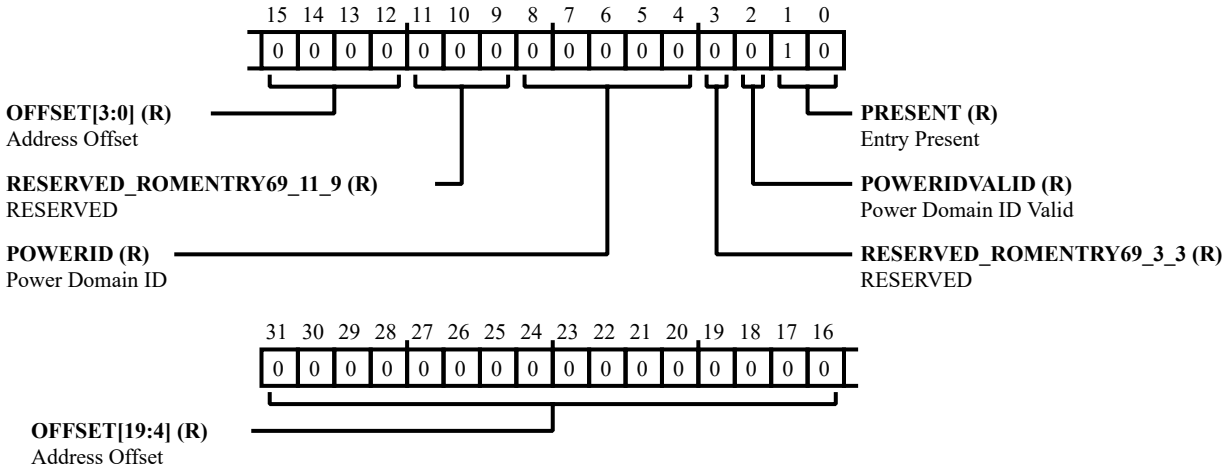


Figure 48-339: A55_CLUSTER_ROMENTRY69 Register Diagram

Table 48-337: A55_CLUSTER_ROMENTRY69 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY69_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY69_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-337: A55_CLUSTER_ROMENTRY69 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 7 Register

ROMENTRYn.

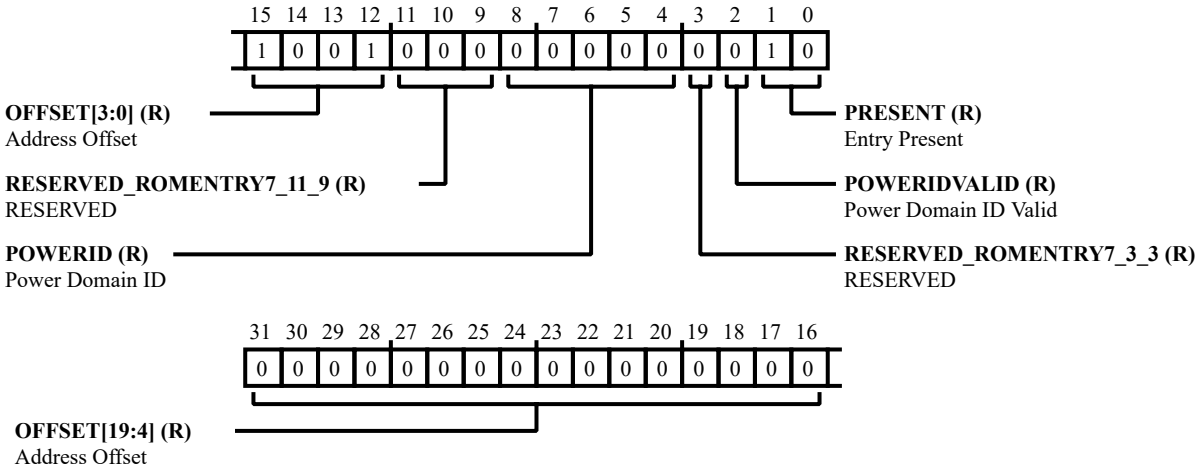


Figure 48-340: A55_CLUSTER_ROMENTRY7 Register Diagram

Table 48-338: A55_CLUSTER_ROMENTRY7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY7_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY7_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-338: A55_CLUSTER_ROMENTRY7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 70 Register

ROMENTRYn.

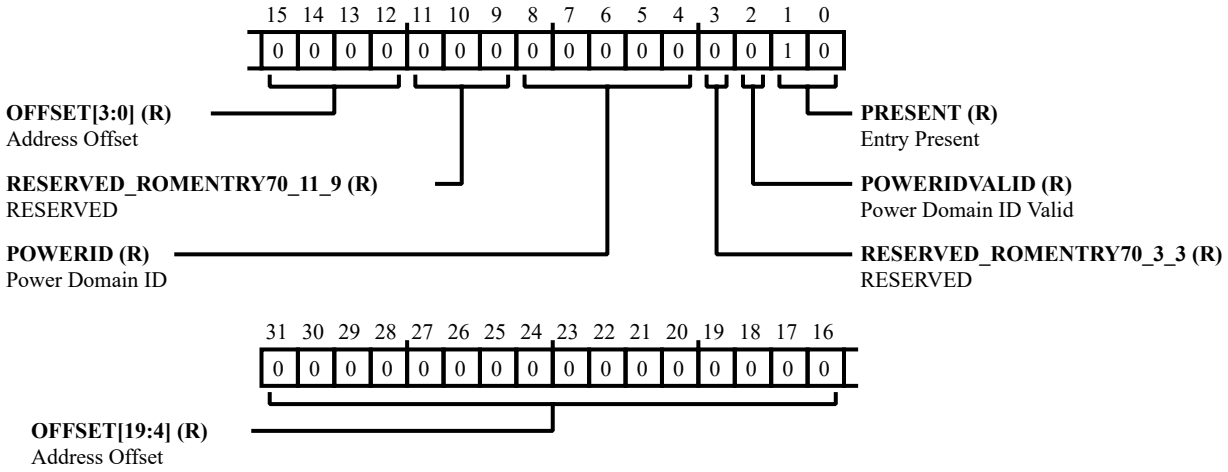


Figure 48-341: A55_CLUSTER_ROMENTRY70 Register Diagram

Table 48-339: A55_CLUSTER_ROMENTRY70 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY70_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY70_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-339: A55_CLUSTER_ROMENTRY70 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 71 Register

ROMENTRYn.

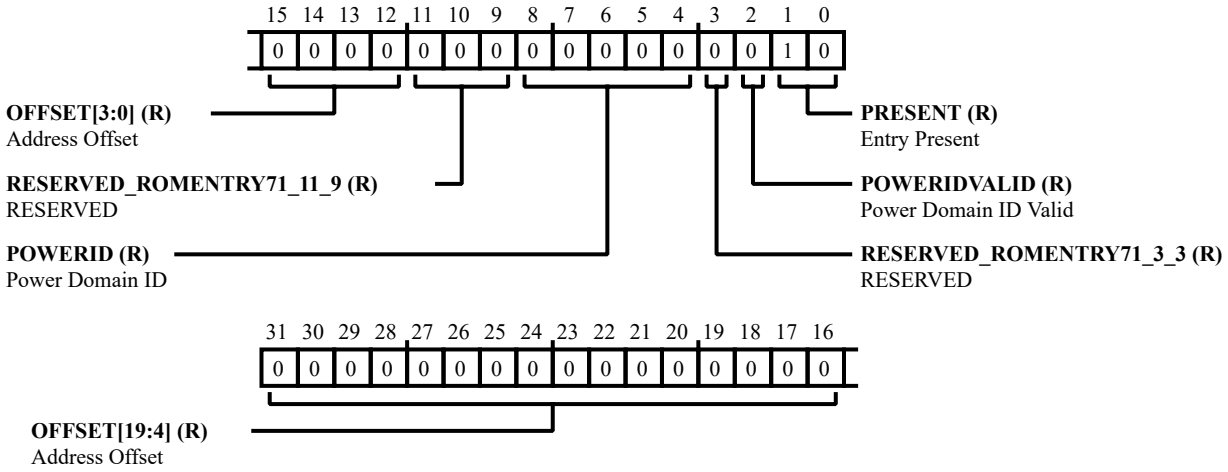


Figure 48-342: A55_CLUSTER_ROMENTRY71 Register Diagram

Table 48-340: A55_CLUSTER_ROMENTRY71 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY71_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY71_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-340: A55_CLUSTER_ROMENTRY71 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 72 Register

ROMENTRYn.

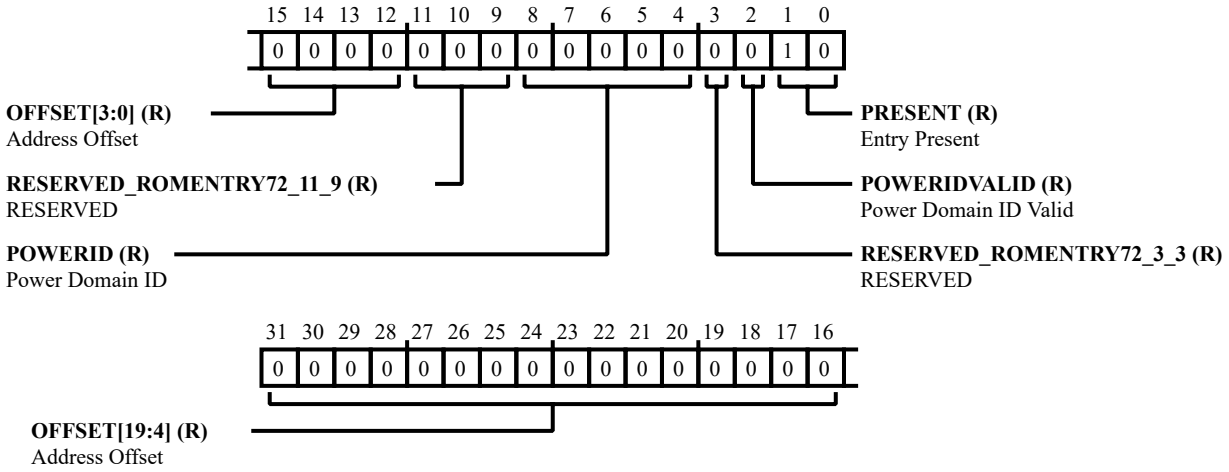


Figure 48-343: A55_CLUSTER_ROMENTRY72 Register Diagram

Table 48-341: A55_CLUSTER_ROMENTRY72 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY72_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY72_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-341: A55_CLUSTER_ROMENTRY72 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 73 Register

ROMENTRYn.

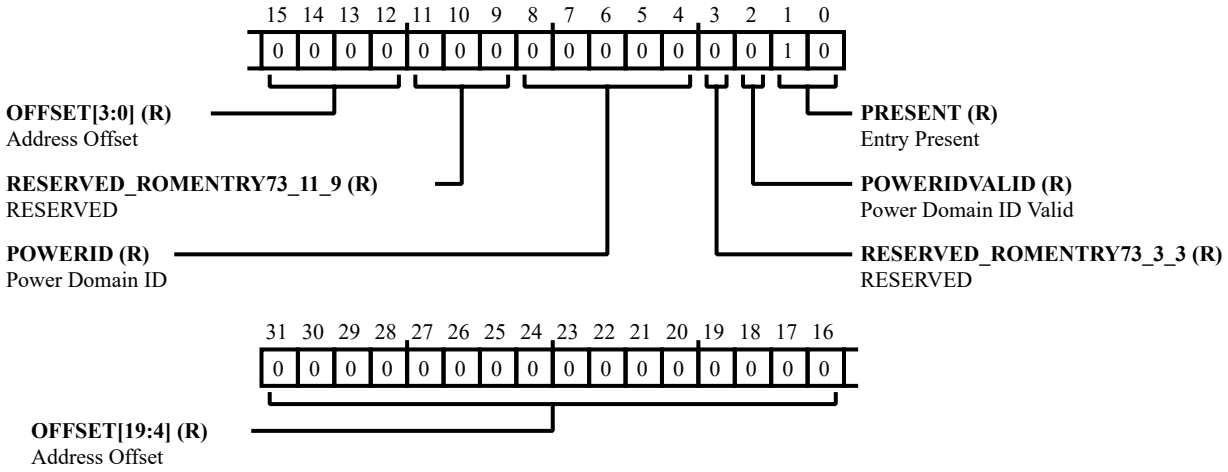


Figure 48-344: A55_CLUSTER_ROMENTRY73 Register Diagram

Table 48-342: A55_CLUSTER_ROMENTRY73 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY73_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY73_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-342: A55_CLUSTER_ROMENTRY73 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 74 Register

ROMENTRYn.

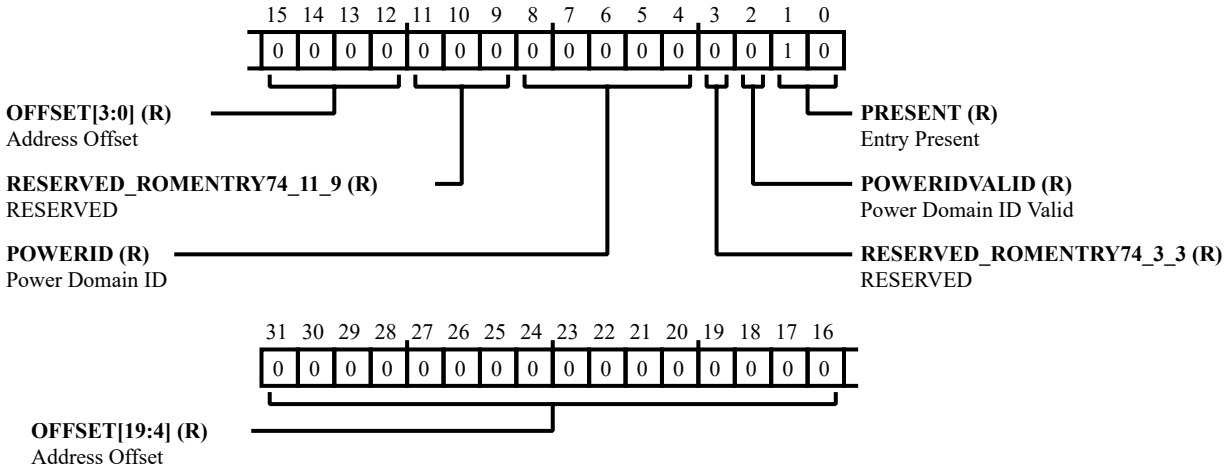


Figure 48-345: A55_CLUSTER_ROMENTRY74 Register Diagram

Table 48-343: A55_CLUSTER_ROMENTRY74 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY74_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY74_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-343: A55_CLUSTER_ROMENTRY74 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 75 Register

ROMENTRYn.

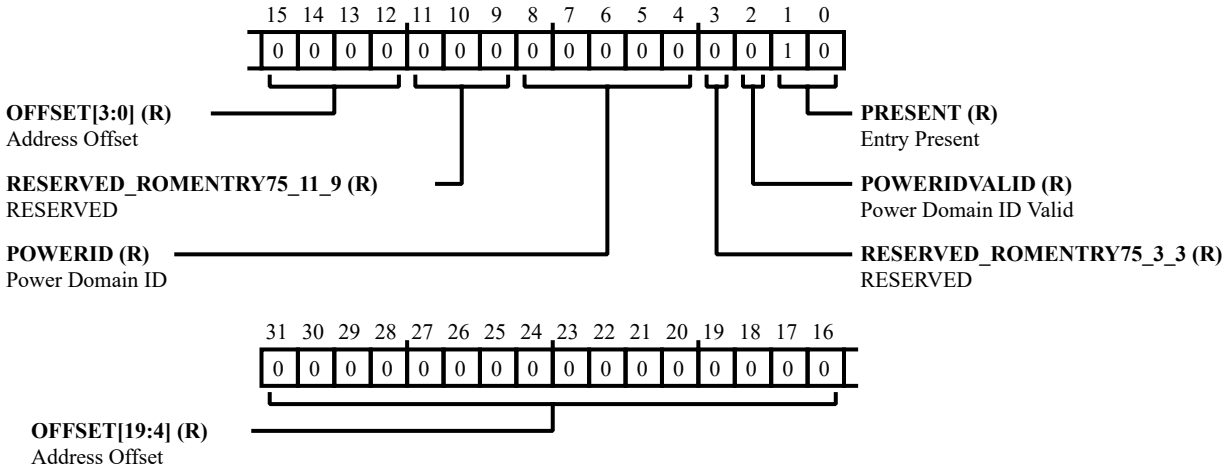


Figure 48-346: A55_CLUSTER_ROMENTRY75 Register Diagram

Table 48-344: A55_CLUSTER_ROMENTRY75 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY75_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY75_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-344: A55_CLUSTER_ROMENTRY75 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 76 Register

ROMENTRYn.

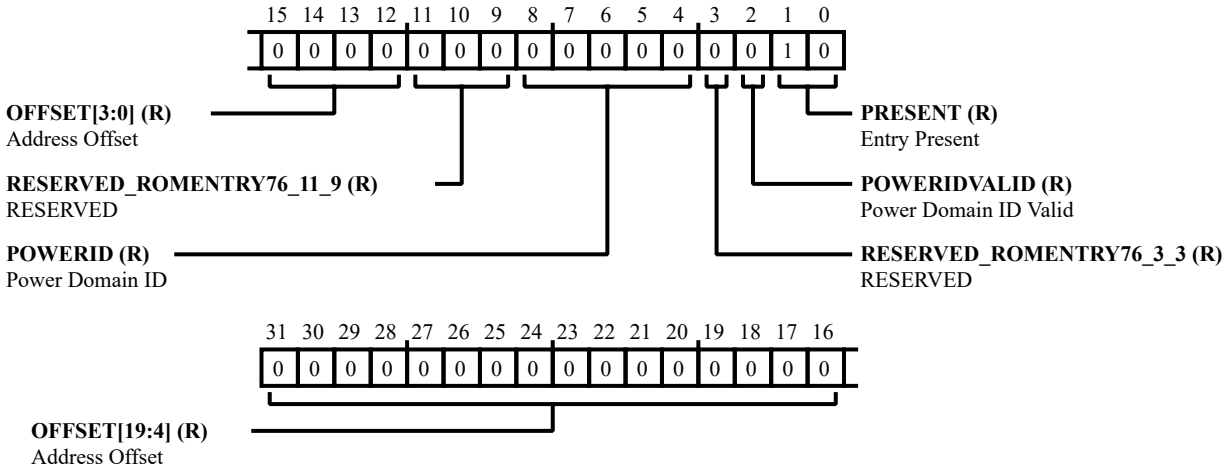


Figure 48-347: A55_CLUSTER_ROMENTRY76 Register Diagram

Table 48-345: A55_CLUSTER_ROMENTRY76 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY76_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY76_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-345: A55_CLUSTER_ROMENTRY76 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 77 Register

ROMENTRYn.

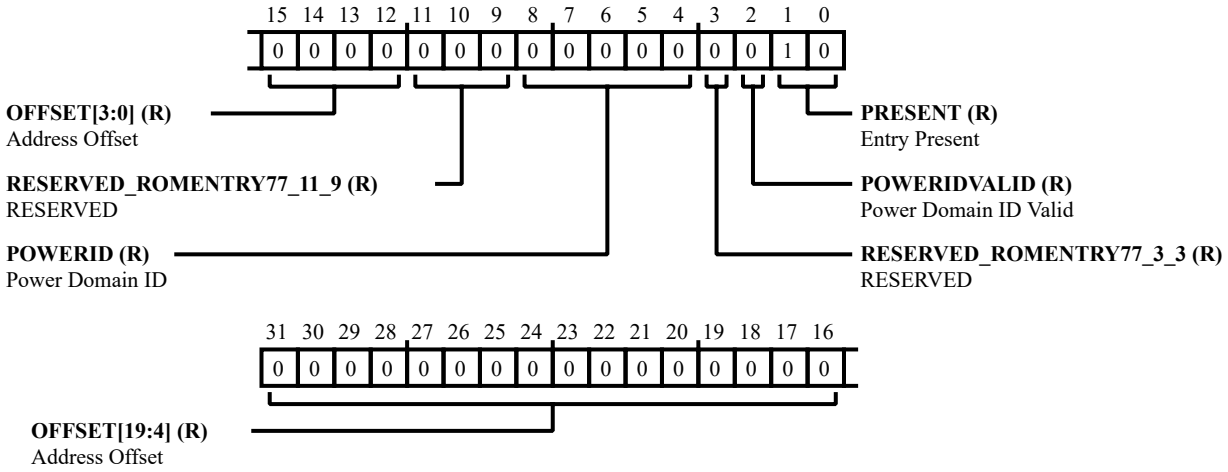


Figure 48-348: A55_CLUSTER_ROMENTRY77 Register Diagram

Table 48-346: A55_CLUSTER_ROMENTRY77 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY77_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY77_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-346: A55_CLUSTER_ROMENTRY77 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 78 Register

ROMENTRYn.

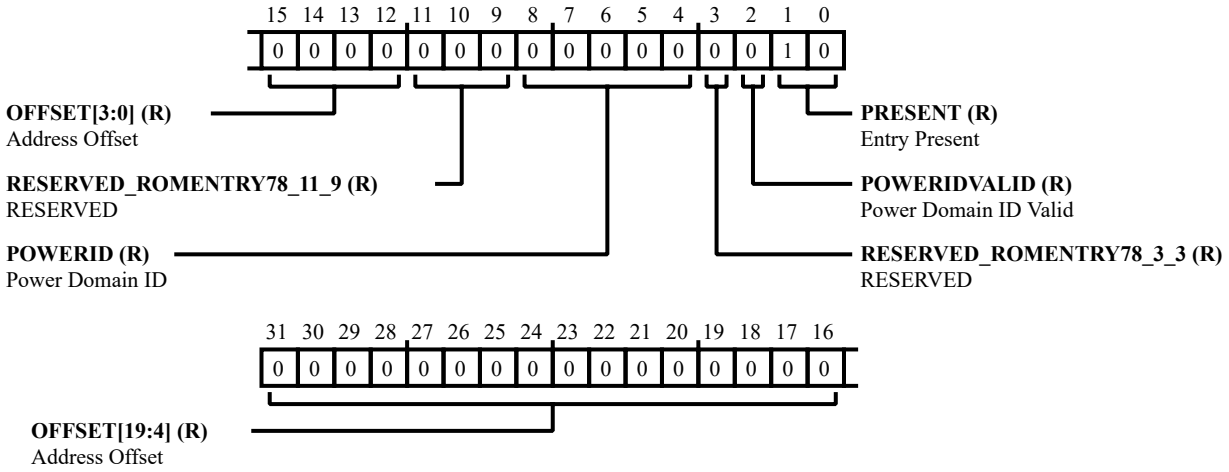


Figure 48-349: A55_CLUSTER_ROMENTRY78 Register Diagram

Table 48-347: A55_CLUSTER_ROMENTRY78 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY78_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY78_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-347: A55_CLUSTER_ROMENTRY78 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 79 Register

ROMENTRYn.

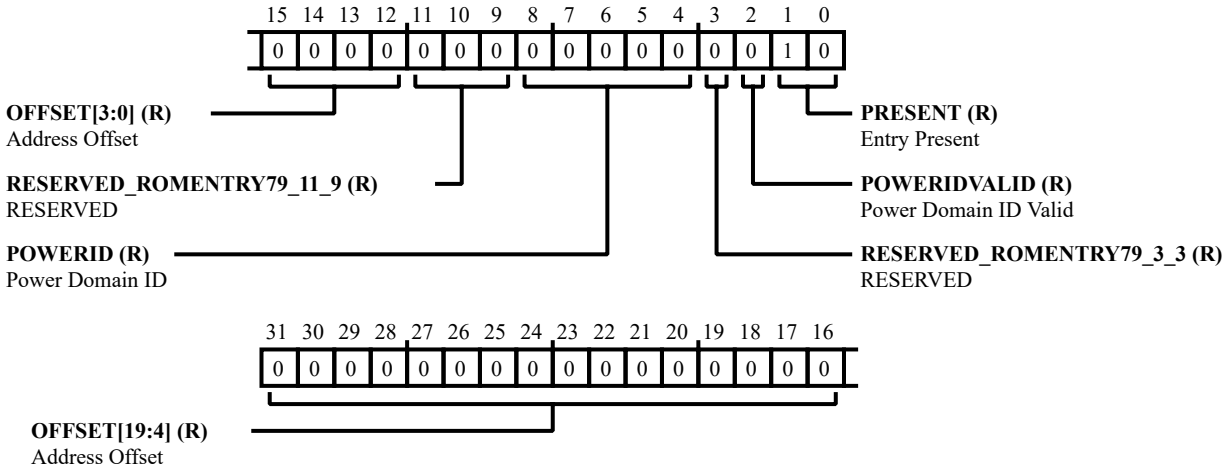


Figure 48-350: A55_CLUSTER_ROMENTRY79 Register Diagram

Table 48-348: A55_CLUSTER_ROMENTRY79 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY79_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY79_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-348: A55_CLUSTER_ROMENTRY79 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 8 Register

ROMENTRYn.

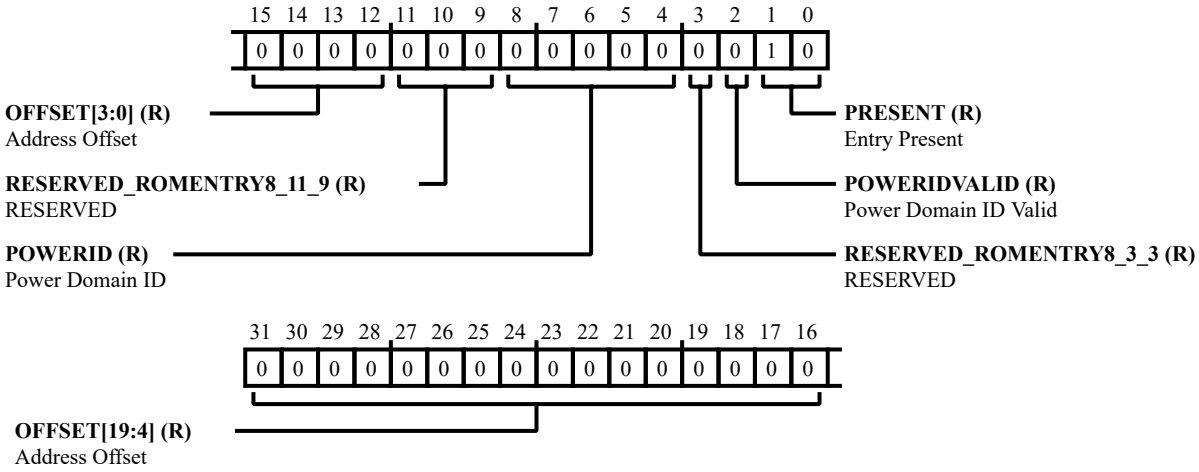


Figure 48-351: A55_CLUSTER_ROMENTRY8 Register Diagram

Table 48-349: A55_CLUSTER_ROMENTRY8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY8_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY8_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-349: A55_CLUSTER_ROMENTRY8 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 80 Register

ROMENTRYn.

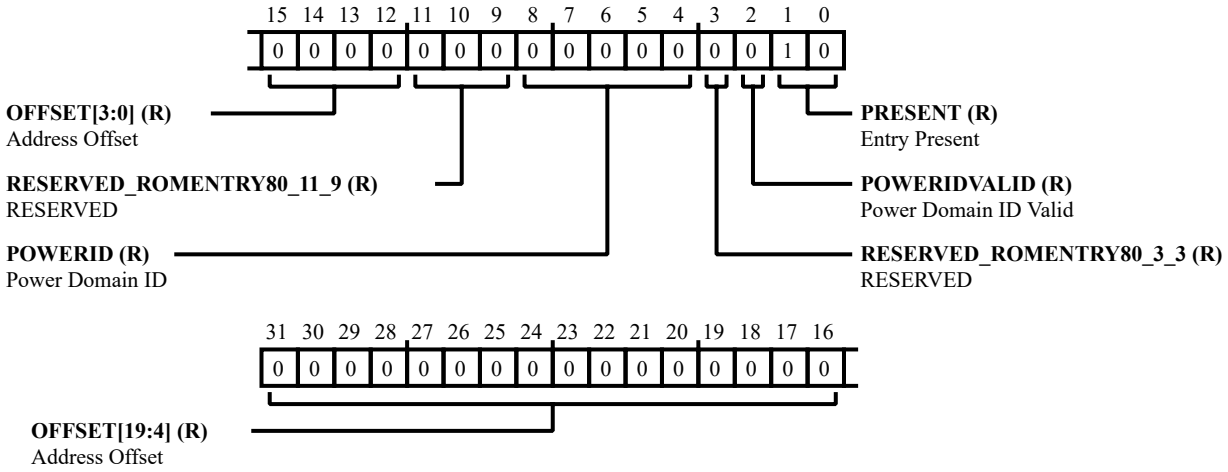


Figure 48-352: A55_CLUSTER_ROMENTRY80 Register Diagram

Table 48-350: A55_CLUSTER_ROMENTRY80 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY80_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY80_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-350: A55_CLUSTER_ROMENTRY80 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 81 Register

ROMENTRYn.

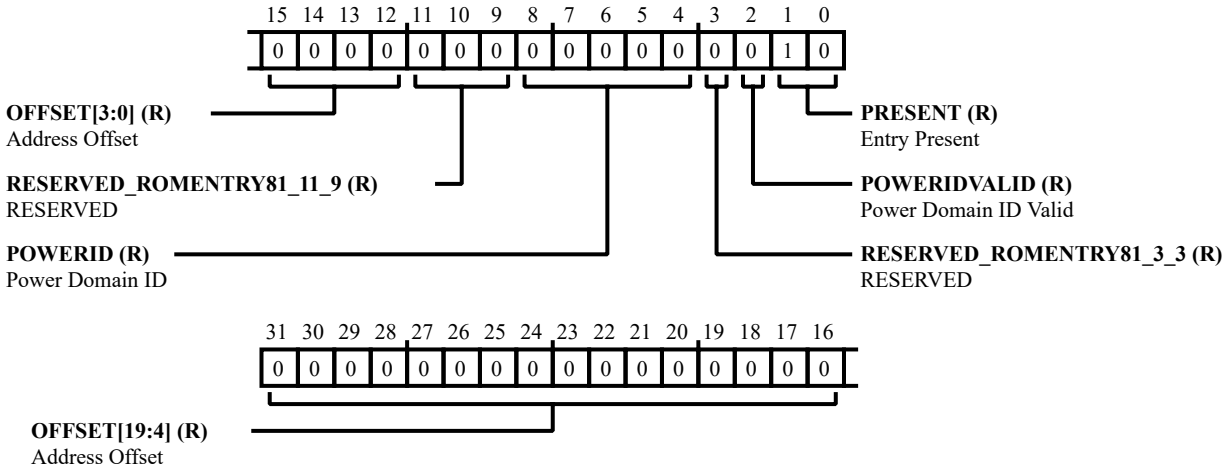


Figure 48-353: A55_CLUSTER_ROMENTRY81 Register Diagram

Table 48-351: A55_CLUSTER_ROMENTRY81 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY81_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY81_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-351: A55_CLUSTER_ROMENTRY81 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ROM Table Entry 9 Register

ROMENTRYn.

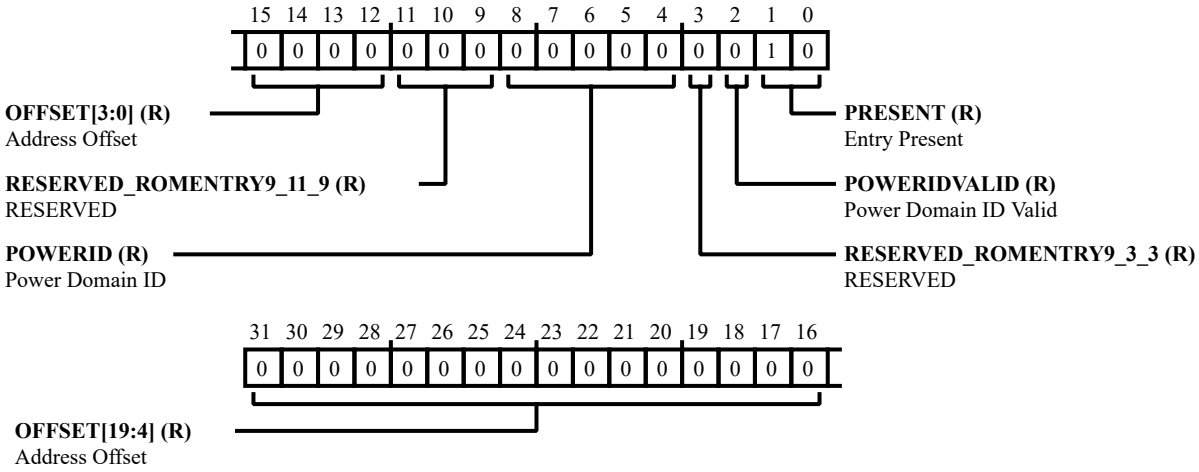


Figure 48-354: A55_CLUSTER_ROMENTRY9 Register Diagram

Table 48-352: A55_CLUSTER_ROMENTRY9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:12 (R/NW)	OFFSET	Address Offset. Base address of the highest 4KB block for that component, relative to the ROM address.	
11:9 (R/NW)	RESERVED_ROMEN-TRY9_11_9	RESERVED. Reserved RES0.	
		0	Reserved.
8:4 (R/NW)	POWERID	Power Domain ID. Indicates the power domain ID of the component.	
3 (R/NW)	RESERVED_ROMEN-TRY9_3_3	RESERVED. Reserved RES0.	
		0	Reserved.
2 (R/NW)	POWERIDVALID	Power Domain ID Valid. Indicates if the POWERID field valid.	
		0	Power domain ID not provided.
		1	Power domain ID provided in POWERID.

Table 48-352: A55_CLUSTER_ROMENTRY9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PRESENT	Entry Present. Set if corresponding unit exists.	
		0	ROM entry not present. All other fields in this entry must be zero. This is the final entry in the ROM table.
		1	Reserved.
		2	ROM entry not present. This is not the final entry in the ROM table. All other fields in this entry are UNKNOWN.
		3	ROM entry present.

ADSP-2184x SOC600CTI_TRACE Register Descriptions

Your module description, here. (SOC600CTI_TRACE) contains the following registers.

Table 48-353: ADSP-2184x SOC600CTI_TRACE Register List

Name	Description
SOC600CTI_TRACE_ASICCTRL	External Multiplexer Control register.
SOC600CTI_TRACE_AUTHSTATUS	Authentication Status Register.
SOC600CTI_TRACE_CIDR0	Component Identification Register 0.
SOC600CTI_TRACE_CIDR1	Component Identification Register 1.
SOC600CTI_TRACE_CIDR2	Component Identification Register 2.
SOC600CTI_TRACE_CIDR3	Component Identification Register 3.
SOC600CTI_TRACE_CLAIMCLR	Claim Tag Clear Register.
SOC600CTI_TRACE_CLAIMSET	Claim Tag Set Register.
SOC600CTI_TRACE_CTIAPPCLEAR	CTI Application Channel Clear register.
SOC600CTI_TRACE_CTIAPPPULSE	CTI Application Channel Pulse register.
SOC600CTI_TRACE_CTIAPPSET	CTI Application Channel Set register.
SOC600CTI_TRACE_CTICHINSTATUS	CTI Channel Input Status register.
SOC600CTI_TRACE_CTICHOUTSTATUS	CTI Channel Output Status register.
SOC600CTI_TRACE_CTICONTROL	CTI Control register.
SOC600CTI_TRACE_CTIGATE	Enable CTI Channel Gate register.
SOC600CTI_TRACE_CTIINEN0	CTI Trigger 0 to Channel Enable register.
SOC600CTI_TRACE_CTIINEN1	CTI Trigger 1 to Channel Enable register.

Table 48-353: ADSP-2184x SOC600CTI_TRACE Register List (Continued)

Name	Description
SOC600CTI_TRACE_CTIINEN2	CTI Trigger 2 to Channel Enable register.
SOC600CTI_TRACE_CTIINEN3	CTI Trigger 3 to Channel Enable register.
SOC600CTI_TRACE_CTIINEN4	CTI Trigger 4 to Channel Enable register.
SOC600CTI_TRACE_CTIINEN5	CTI Trigger 5 to Channel Enable register.
SOC600CTI_TRACE_CTIINEN6	CTI Trigger 6 to Channel Enable register.
SOC600CTI_TRACE_CTIINTACK	CTI Interrupt Acknowledge register.
SOC600CTI_TRACE_CTIOUTEN0	CTI Channel to Trigger 0 Enable register.
SOC600CTI_TRACE_CTIOUTEN1	CTI Channel to Trigger 1 Enable register.
SOC600CTI_TRACE_CTIOUTEN2	CTI Channel to Trigger 2 Enable register.
SOC600CTI_TRACE_CTIOUTEN3	CTI Channel to Trigger 3 Enable register.
SOC600CTI_TRACE_CTIOUTEN4	CTI Channel to Trigger 4 Enable register.
SOC600CTI_TRACE_CTIOUTEN5	CTI Channel to Trigger 5 Enable register.
SOC600CTI_TRACE_CTITRIGINSTATUS	CTI Trigger Input Status register.
SOC600CTI_TRACE_CTITRIGOUTSTATUS	CTI Trigger Output Status register.
SOC600CTI_TRACE_DEVAFF0	Device Affinity register 0.
SOC600CTI_TRACE_DEVAFF1	Device Affinity register 1.
SOC600CTI_TRACE_DEVARCH	Device Architecture Register.
SOC600CTI_TRACE_DEVID	Device Configuration Register.
SOC600CTI_TRACE_DEVTYPE	Device Type Identifier Register.
SOC600CTI_TRACE_ITCHIN	Integration Test Channel Input register.
SOC600CTI_TRACE_ITCHOUT	Integration Test Channel Output register.
SOC600CTI_TRACE_ITCTRL	Integration Mode Control Register.
SOC600CTI_TRACE_ITTRIGIN	Integration Test Trigger Input register.
SOC600CTI_TRACE_ITTRIGOUT	Integration Test Trigger Output register.
SOC600CTI_TRACE_PIDR0	Peripheral Identification Register 0.
SOC600CTI_TRACE_PIDR1	Peripheral Identification Register 1.
SOC600CTI_TRACE_PIDR2	Peripheral Identification Register 2.
SOC600CTI_TRACE_PIDR3	Peripheral Identification Register 3.
SOC600CTI_TRACE_PIDR4	Peripheral Identification Register 4.
SOC600CTI_TRACE_PIDR5	Peripheral Identification Register 5.
SOC600CTI_TRACE_PIDR6	Peripheral Identification Register 6.

Table 48-353: ADSP-2184x SOC600CTI_TRACE Register List (Continued)

Name	Description
SOC600CTI_TRACE_PIDR7	Peripheral Identification Register 7.

External Multiplexer Control register.

I/O port control.

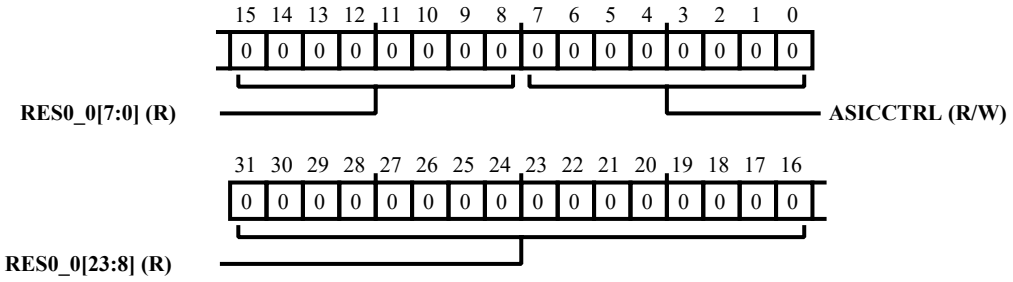


Figure 48-355: SOC600CTI_TRACE_ASICCTRL Register Diagram

Table 48-354: SOC600CTI_TRACE_ASICCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:0 (R/W)	ASICCTRL	. Set and clear external output signal.	
		0	Clear output bit to 0.
		1	Set output bit to 1.

Authentication Status Register.

Reports the current status of the authentication control signals.

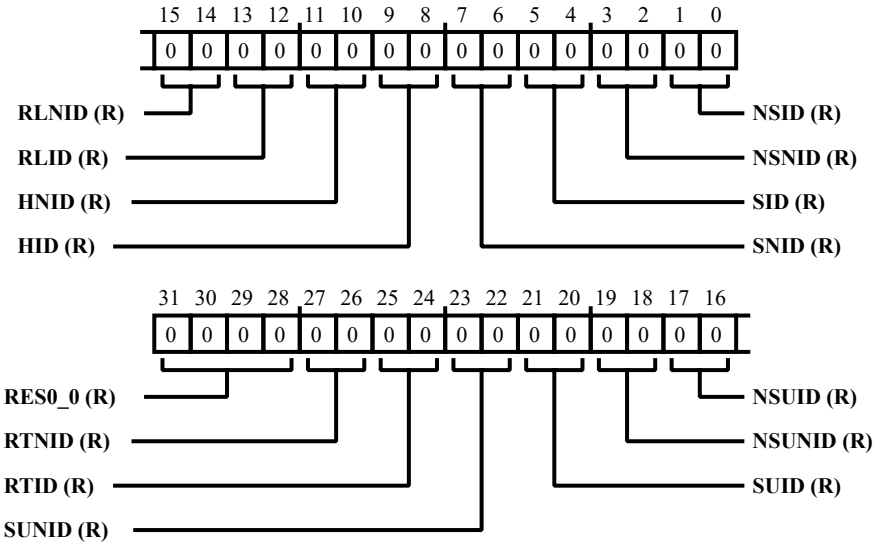


Figure 48-356: SOC600CTI_TRACE_AUTHSTATUS Register Diagram

Table 48-355: SOC600CTI_TRACE_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
27:26 (R/NW)	RTNID	. Root non-invasive debug.	
		0	Separate Root non-invasive debug enable not implemented or Root state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.

Table 48-355: SOC600CTI_TRACE_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	RTID	. Root invasive debug.	
		0	Separate Root invasive debug enable not implemented or Root state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
23:22 (R/NW)	SUNID	. Secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
21:20 (R/NW)	SUID	. Secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
19:18 (R/NW)	NSUNID	. Non-secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
17:16 (R/NW)	NSUID	. Non-secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-355: SOC600CTI_TRACE_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/NW)	RLNID	. Realm non-invasive debug.	
		0	Separate Realm non-invasive debug enable not implemented or Realm state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
13:12 (R/NW)	RLID	. Realm invasive debug.	
		0	Separate Realm invasive debug enable not implemented or Realm state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
11:10 (R/NW)	HNID	. Hypervisor non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
9:8 (R/NW)	HID	. Hypervisor invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-355: SOC600CTI_TRACE_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	. Secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
5:4 (R/NW)	SID	. Secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
3:2 (R/NW)	NSNID	. Non-secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
1:0 (R/NW)	NSID	. Non-secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

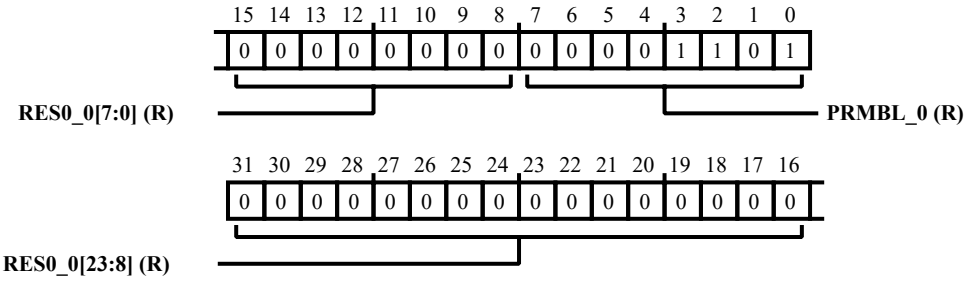


Figure 48-357: SOC600CTI_TRACE_CIDR0 Register Diagram

Table 48-356: SOC600CTI_TRACE_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

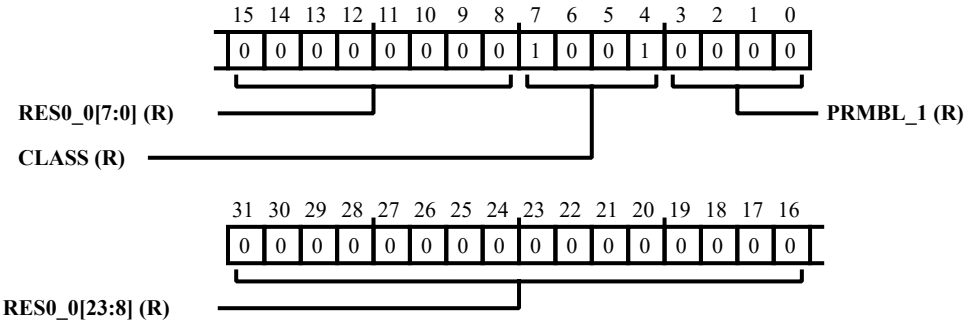


Figure 48-358: SOC600CTI_TRACE_CIDR1 Register Diagram

Table 48-357: SOC600CTI_TRACE_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

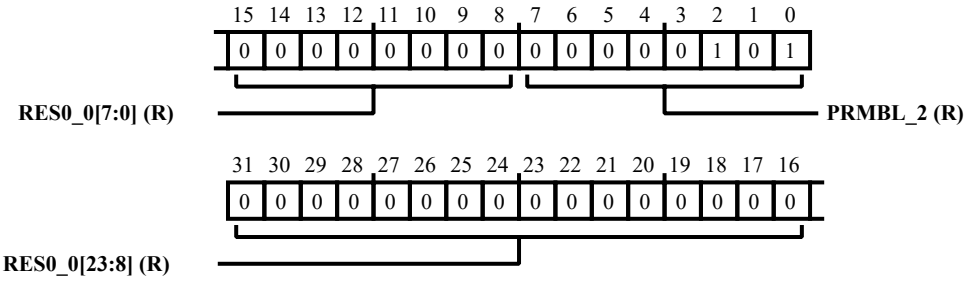


Figure 48-359: SOC600CTI_TRACE_CIDR2 Register Diagram

Table 48-358: SOC600CTI_TRACE_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

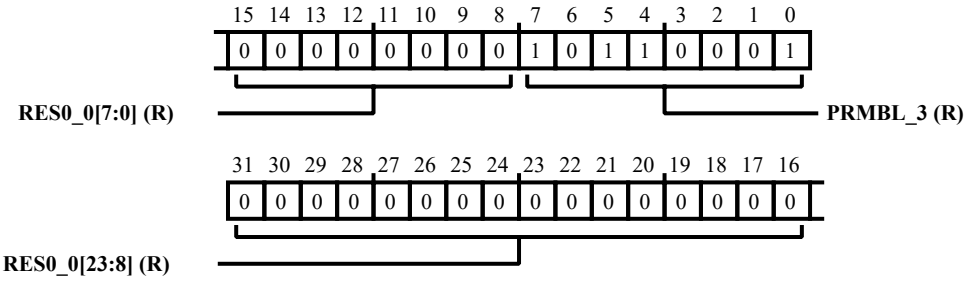


Figure 48-360: SOC600CTI_TRACE_CIDR3 Register Diagram

Table 48-359: SOC600CTI_TRACE_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Claim Tag Clear Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

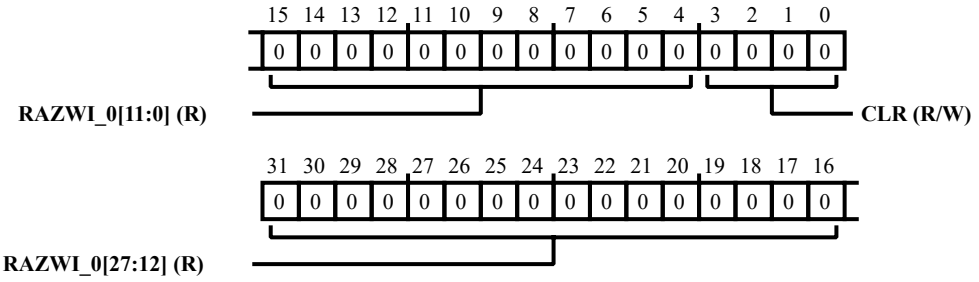


Figure 48-361: SOC600CTI_TRACE_CLAIMCLR Register Diagram

Table 48-360: SOC600CTI_TRACE_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	CLR	. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

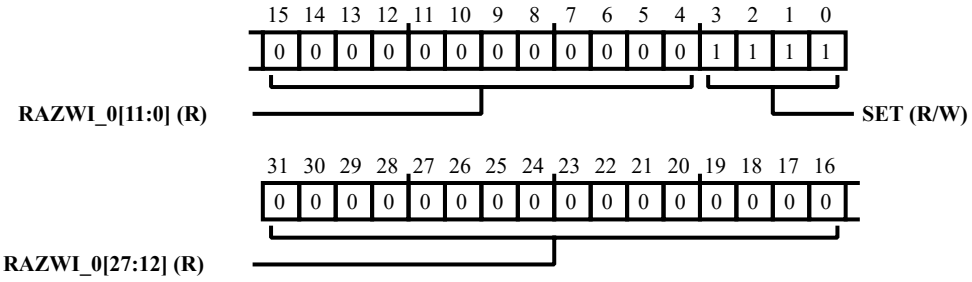


Figure 48-362: SOC600CTI_TRACE_CLAIMSET Register Diagram

Table 48-361: SOC600CTI_TRACE_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	SET	. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

CTI Application Channel Clear register.

The CTIAPPCLEAR register allows software to clear any channel output. Software can use this register to clear a channel event in place of a hardware source on a trigger input. This register must not be used in a system where all events are sent as single-cycle pulses. It is only retained for compatibility with older systems and software. The register is implemented before the CTIGATE register. Therefore, for the channel event to propagate outside the CTI, the corresponding CTIGATE bit must be set to 1.

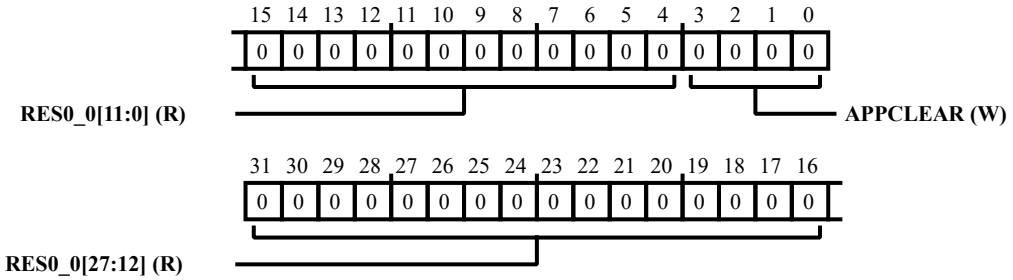


Figure 48-363: SOC600CTI_TRACE_CTIAPPCLEAR Register Diagram

Table 48-362: SOC600CTI_TRACE_CTIAPPCLEAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (RX/W)	APPCLEAR	. Clears the corresponding internal channel flag.	
		0	No effect.
		1	Clears the channel output.

CTI Application Channel Pulse register.

The application channel pulse register allows software to pulse any channel output. This register can be used by software to pulse a channel event in place of a hardware source on a trigger input. The register is implemented before the CTIGATE register so, for the channel event to propagate outside the CTI, it is necessary for the corresponding CTIGATE bit to be 1.

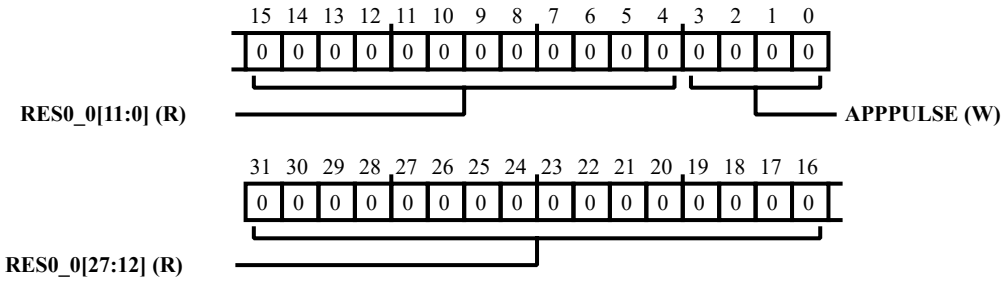


Figure 48-364: SOC600CTI_TRACE_CTIAPPPULSE Register Diagram

Table 48-363: SOC600CTI_TRACE_CTIAPPPULSE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (RX/W)	APPPULSE	. Pulses the channel outputs.	
		0	No effect.
		1	Pulse channel event for one clk cycle.

CTI Application Channel Set register.

The CTIAPPSET register allows software to set any channel output. Software can use this register to generate a channel event in place of a hardware source on a trigger input. This register must not be used in a system where all events are sent as single-cycle pulses. It is only retained for compatibility with older systems and software. The register is implemented before the CTIGATE register. Therefore, for the channel event to propagate outside the CTI, the corresponding CTIGATE bit must be set to 1.

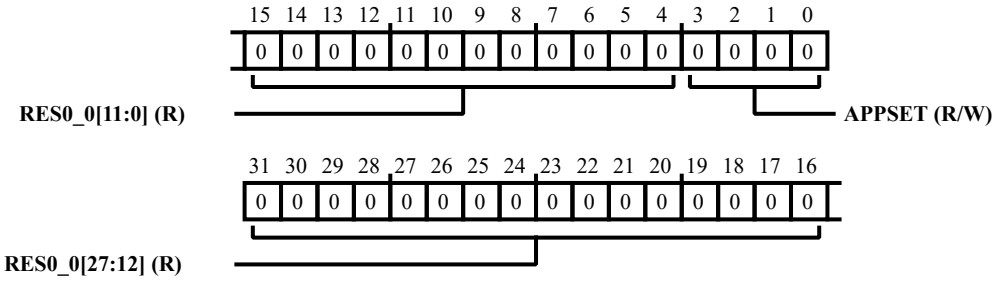


Figure 48-365: SOC600CTI_TRACE_CTIAPPSET Register Diagram

Table 48-364: SOC600CTI_TRACE_CTIAPPSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	APPSET	. Sets the corresponding internal channel flag:.	
		0	Read: application channel is inactive. Write: has no effect.
		1	Read: application channel is active. Write: sets the channel output.

CTI Channel Input Status register.

Channel input status. If the channel input is driven by a source that generates single cycle pulses, this register is generally read as 0.

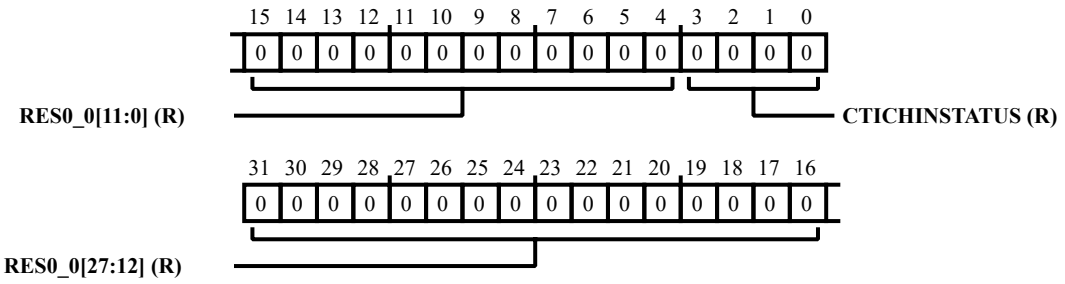


Figure 48-366: SOC600CTI_TRACE_CTICHINSTATUS Register Diagram

Table 48-365: SOC600CTI_TRACE_CTICHINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/NW)	CTICHINSTATUS	. Channel input status.	
		0	One bit per channel input. 0 means that the input is LOW.
		1	One bit per channel input. 1 means that the input is HIGH.

CTI Channel Output Status register.

Channel output status. The register only has meaning if the trigger source drives static levels, rather than pulses.

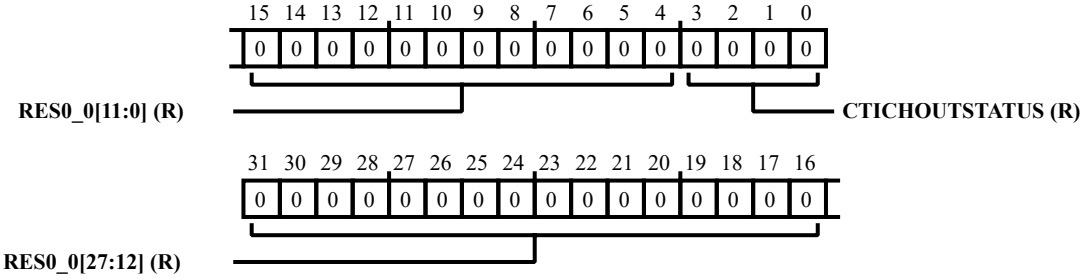


Figure 48-367: SOC600CTI_TRACE_CTICHOUTSTATUS Register Diagram

Table 48-366: SOC600CTI_TRACE_CTICHOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/NW)	CTICHOUTSTATUS	. Channel output status.	
		0	One bit per channel output. 0 means that the output is LOW.
		1	One bit per channel output. 1 means that the output is HIGH.

CTI Control register.

The CTICONTROL register enables and disables the CTI.

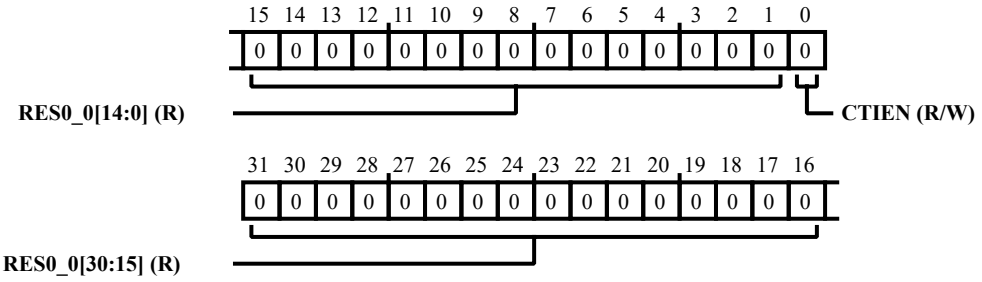


Figure 48-368: SOC600CTI_TRACE_CTICONTROL Register Diagram

Table 48-367: SOC600CTI_TRACE_CTICONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	CTIEN	. Enable control:.	
		0	CTI disabled.
		1	CTI enabled.

Enable CTI Channel Gate register.

Channel output gate.

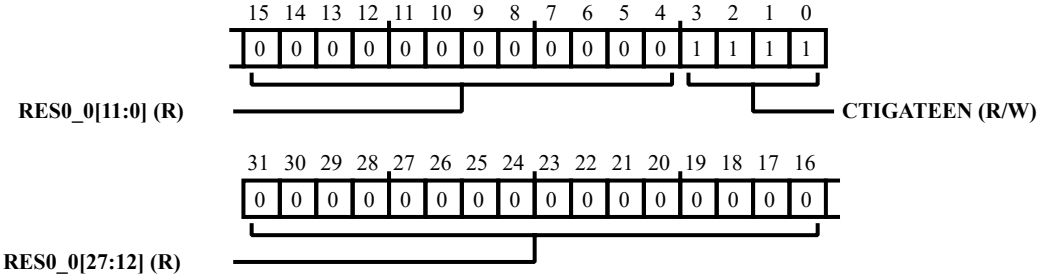


Figure 48-369: SOC600CTI_TRACE_CTIGATE Register Diagram

Table 48-368: SOC600CTI_TRACE_CTIGATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	CTIGATEEN	. Enables the propagation of channel events out of the CTI, one bit per channel.	
		0	Disable a channel from propagating.
		1	Enable channel propagation.

CTI Trigger 0 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

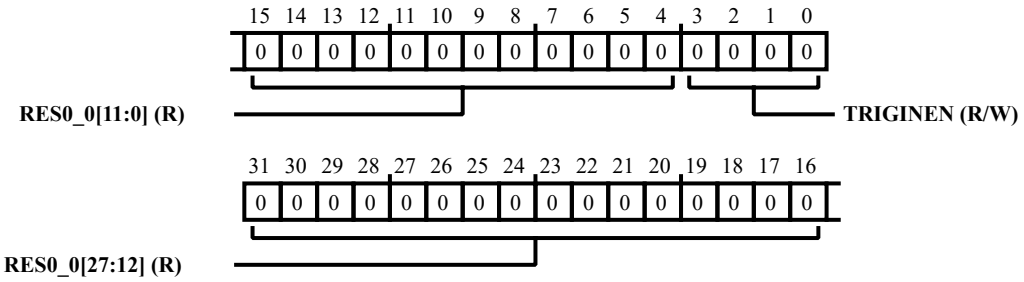


Figure 48-370: SOC600CTI_TRACE_CTIINEN0 Register Diagram

Table 48-369: SOC600CTI_TRACE_CTIINEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 0 events are ignored by the corresponding channel.
		1	When an event is received on event_in[0], generate an event on the channel corresponding to this bit.

CTI Trigger 1 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

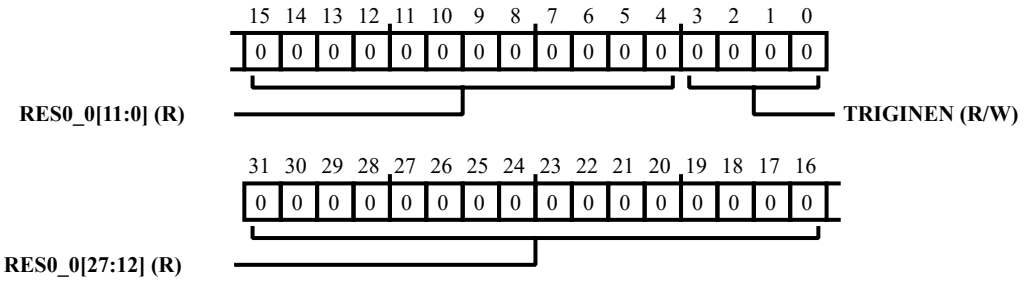


Figure 48-371: SOC600CTI_TRACE_CTIINEN1 Register Diagram

Table 48-370: SOC600CTI_TRACE_CTIINEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 1 events are ignored by the corresponding channel.
		1	When an event is received on event_in[1], generate an event on the channel corresponding to this bit.

CTI Trigger 2 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

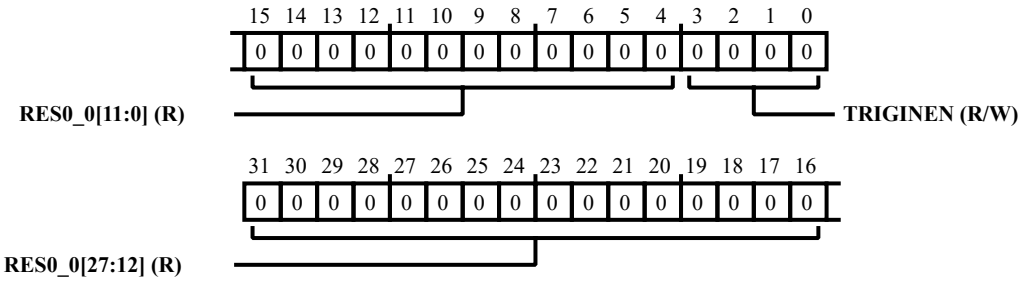


Figure 48-372: SOC600CTI_TRACE_CTIINEN2 Register Diagram

Table 48-371: SOC600CTI_TRACE_CTIINEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 2 events are ignored by the corresponding channel.
		1	When an event is received on event_in[2], generate an event on the channel corresponding to this bit.

CTI Trigger 3 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

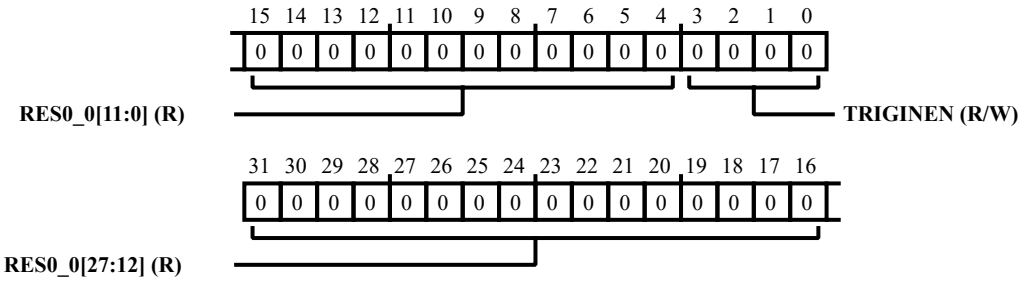


Figure 48-373: SOC600CTI_TRACE_CTIINEN3 Register Diagram

Table 48-372: SOC600CTI_TRACE_CTIINEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 3 events are ignored by the corresponding channel.
		1	When an event is received on event_in[3], generate an event on the channel corresponding to this bit.

CTI Trigger 4 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

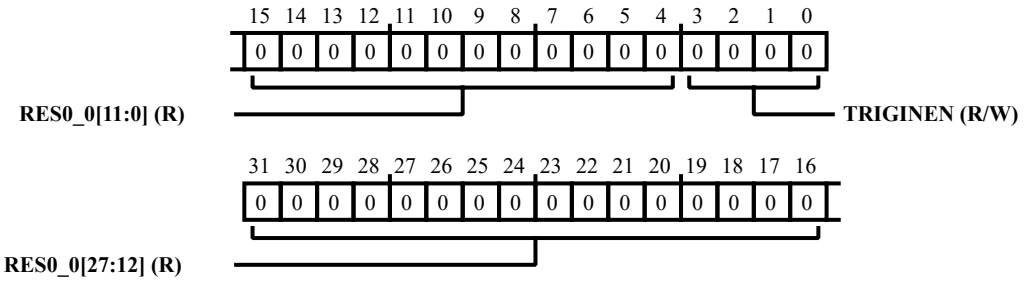


Figure 48-374: SOC600CTI_TRACE_CTIINEN4 Register Diagram

Table 48-373: SOC600CTI_TRACE_CTIINEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 4 events are ignored by the corresponding channel.
		1	When an event is received on event_in[4], generate an event on the channel corresponding to this bit.

CTI Trigger 5 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

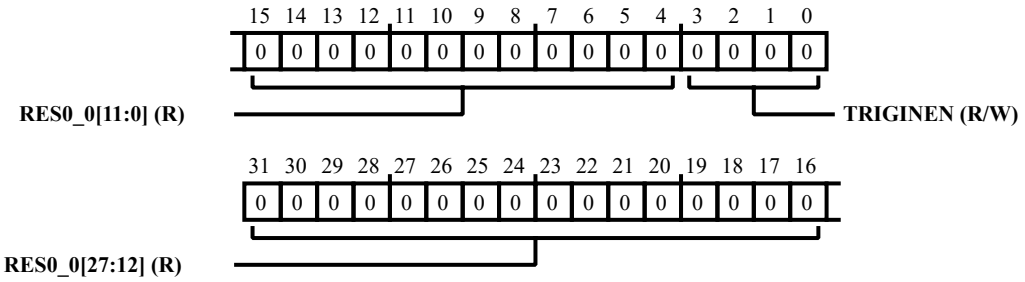


Figure 48-375: SOC600CTI_TRACE_CTIINEN5 Register Diagram

Table 48-374: SOC600CTI_TRACE_CTIINEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 5 events are ignored by the corresponding channel.
		1	When an event is received on event_in[5], generate an event on the channel corresponding to this bit.

CTI Trigger 6 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

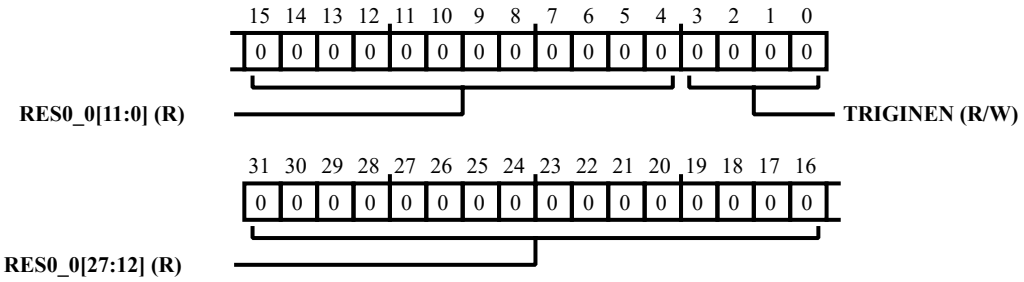


Figure 48-376: SOC600CTI_TRACE_CTIINEN6 Register Diagram

Table 48-375: SOC600CTI_TRACE_CTIINEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 6 events are ignored by the corresponding channel.
		1	When an event is received on event_in[6], generate an event on the channel corresponding to this bit.

CTI Interrupt Acknowledge register.

The CTIINTACK register acknowledges trigger outputs. It is a bit map that allows selective clearing of trigger output events. If the SW_HANDSHAKE parameter for a trigger output is set, indicating that the output latches HIGH when an event is sent to that output, then the output remains HIGH until the corresponding INTACK bit is written to a 1. A write of a bit to 0 has no effect. This allows different software threads to be responsible for clearing different trigger outputs without needing to perform a read-modify-write operation to find which bits are set.

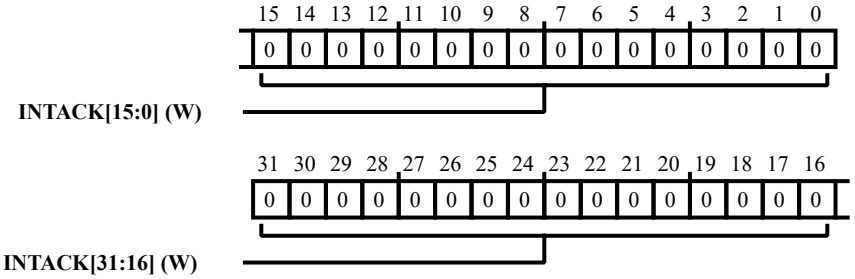


Figure 48-377: SOC600CTI_TRACE_CTIINTACK Register Diagram

Table 48-376: SOC600CTI_TRACE_CTIINTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	INTACK	. Acknowledges the corresponding event_out output.

CTI Channel to Trigger 0 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

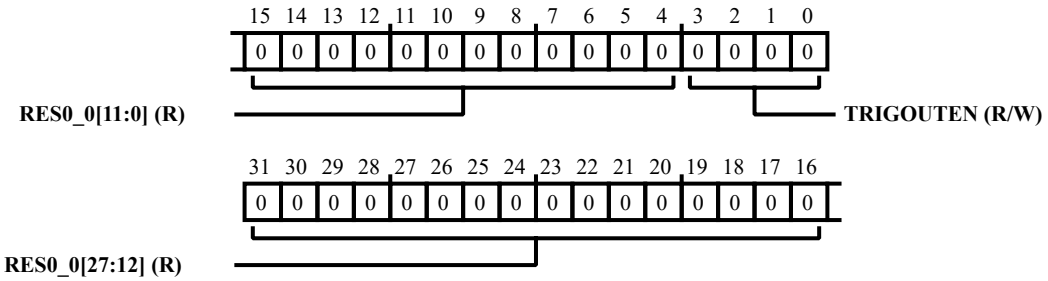


Figure 48-378: SOC600CTI_TRACE_CTIOUTEN0 Register Diagram

Table 48-377: SOC600CTI_TRACE_CTIOUTEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger0.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[0].

CTI Channel to Trigger 1 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

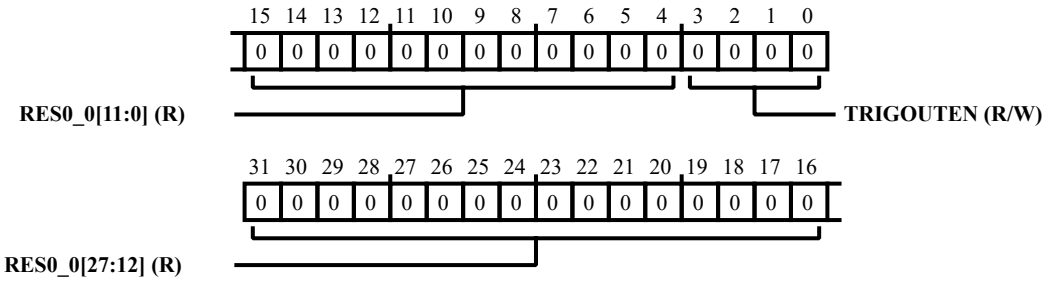


Figure 48-379: SOC600CTI_TRACE_CTIOUTEN1 Register Diagram

Table 48-378: SOC600CTI_TRACE_CTIOUTEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger1.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[1].

CTI Channel to Trigger 2 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

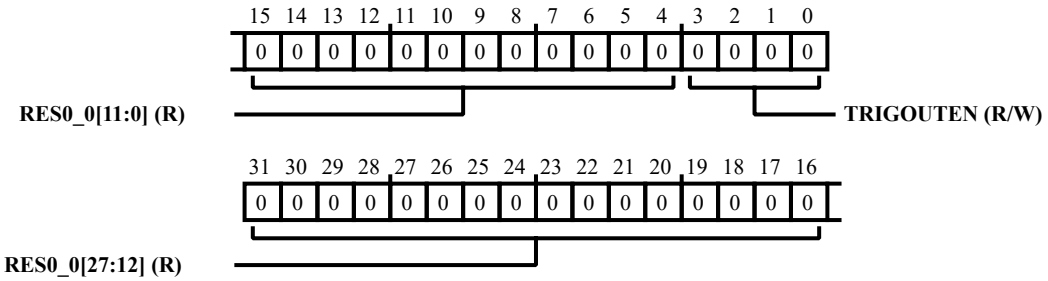


Figure 48-380: SOC600CTI_TRACE_CTIOUTEN2 Register Diagram

Table 48-379: SOC600CTI_TRACE_CTIOUTEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger2.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[2].

CTI Channel to Trigger 3 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

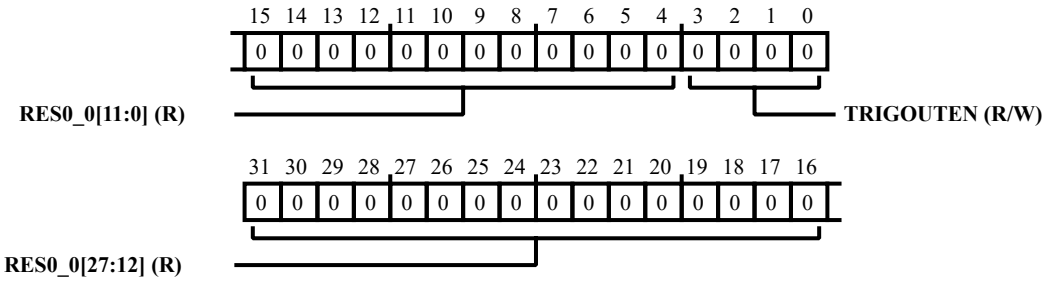


Figure 48-381: SOC600CTI_TRACE_CTIOUTEN3 Register Diagram

Table 48-380: SOC600CTI_TRACE_CTIOUTEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger3.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[3].

CTI Channel to Trigger 4 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

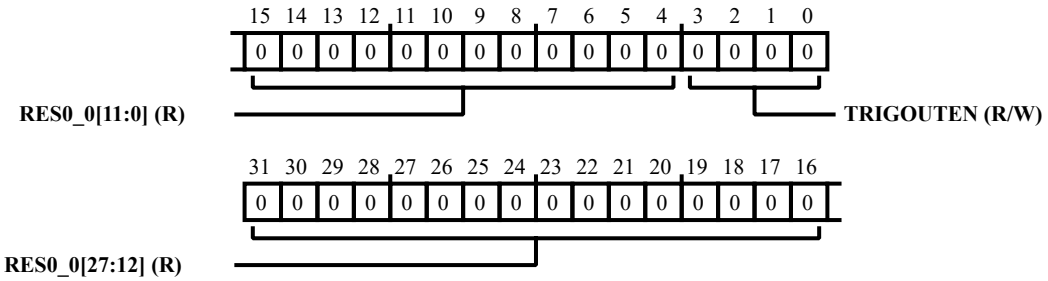


Figure 48-382: SOC600CTI_TRACE_CTIOUTEN4 Register Diagram

Table 48-381: SOC600CTI_TRACE_CTIOUTEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger4.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[4].

CTI Channel to Trigger 5 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

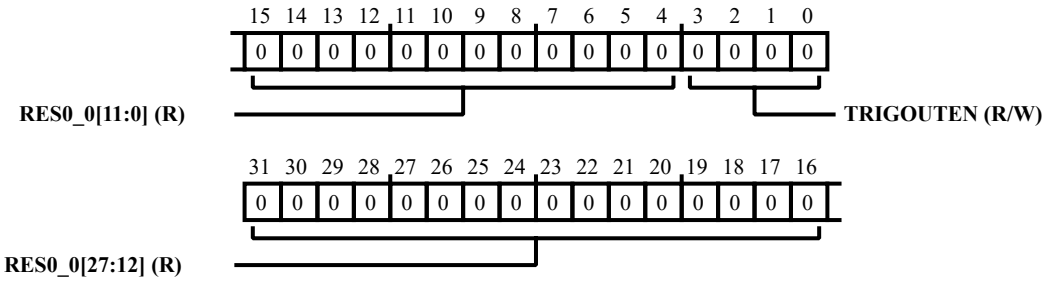


Figure 48-383: SOC600CTI_TRACE_CTIOUTEN5 Register Diagram

Table 48-382: SOC600CTI_TRACE_CTIOUTEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger5.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[5].

CTI Trigger Input Status register.

Trigger input status. If the event_in input is driven by a source that generates single cycle pulses, this register is generally read as 0.

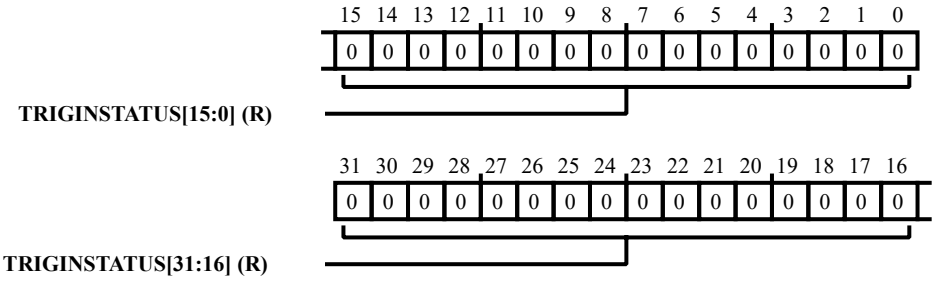


Figure 48-384: SOC600CTI_TRACE_CTITRIGINSTATUS Register Diagram

Table 48-383: SOC600CTI_TRACE_CTITRIGINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	TRIGINSTATUS	. Trigger input status.	
		0	One bit per trigger input. 0 means that the input is LOW.
		1	One bit per trigger input. 1 means that the input is HIGH.

CTI Trigger Output Status register.

Trigger output status. The register only has meaning if the trigger source drives static levels, rather than pulses.

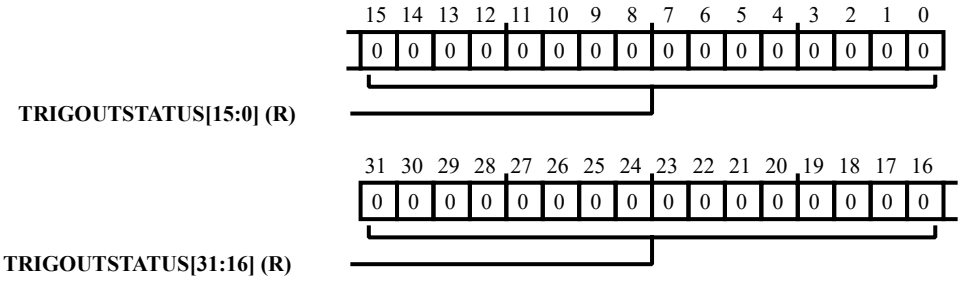


Figure 48-385: SOC600CTI_TRACE_CTITRIGOUTSTATUS Register Diagram

Table 48-384: SOC600CTI_TRACE_CTITRIGOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	TRIGOUTSTATUS	. Trigger output status.	
		0	One bit per trigger output. 0 means that the output is LOW.
		1	One bit per trigger output. 1 means that the output is HIGH.

Device Affinity register 0.

Enables a debugger to determine if two components have an affinity with each other.

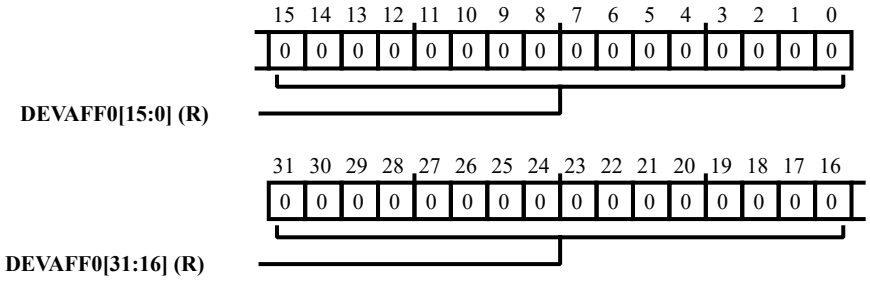


Figure 48-386: SOC600CTI_TRACE_DEVAFF0 Register Diagram

Table 48-385: SOC600CTI_TRACE_DEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVAFF0	. Lower 32-bits of DEVAFF. The value is set by the devaff[31:0] tie-off inputs.

Device Affinity register 1.

Enables a debugger to determine if two components have an affinity with each other.

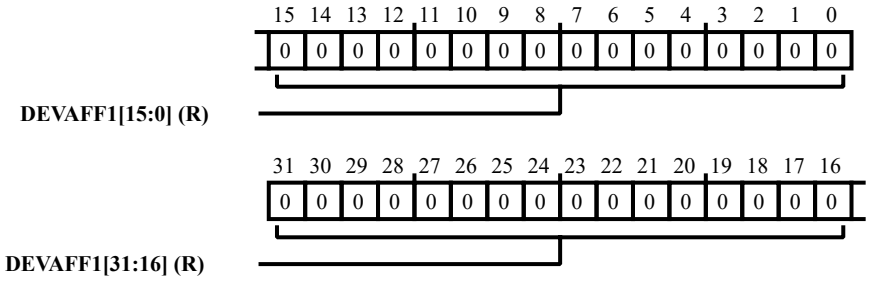


Figure 48-387: SOC600CTI_TRACE_DEVAFF1 Register Diagram

Table 48-386: SOC600CTI_TRACE_DEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVAFF1	. Upper 32-bits of DEVAFF. The value is set by the devaff[63:32] tie-off inputs.

Device Architecture Register.

Identifies the architect and architecture of a CoreSight component. The architect might differ from the designer of a component, for example Arm defines the architecture but another company designs and implements the component.

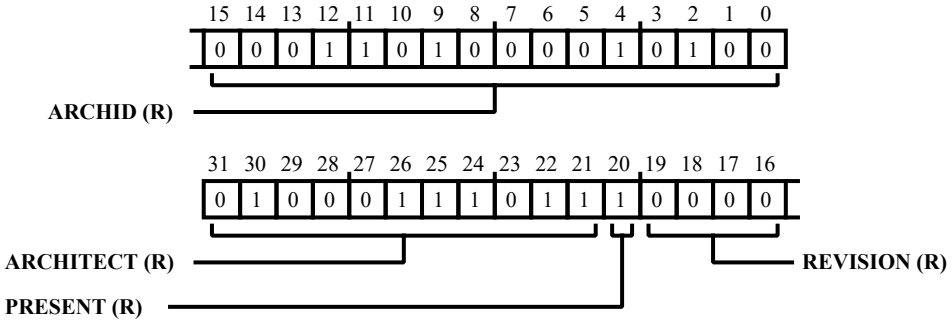


Figure 48-388: SOC600CTI_TRACE_DEVARCH Register Diagram

Table 48-387: SOC600CTI_TRACE_DEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:21 (R/NW)	ARCHITECT	. Defines the architect of the component.	
		571	Arm.
20 (R/NW)	PRESENT	. Indicates the presence of this register.	
		0	DEVARCH is not present.
		1	DEVARCH is present.
19:16 (R/NW)	REVISION	. Architecture revision. Returns the revision of the architecture that the ARCHID field specifies.	
15:0 (R/NW)	ARCHID	. Architecture ID. Returns a value that identifies the architecture of the component.	
		2583	Memory Access Port v2 architecture.
		2599	JTAG Access Port v2 architecture.
		2631	Unknown Access Port v2 architecture.
		2807	CoreSight ROM architecture.
		6676	CTI architecture.

Device Configuration Register.

This register is IMPLEMENTATION DEFINED for each Part Number and Designer. The register indicates the capabilities of the component.

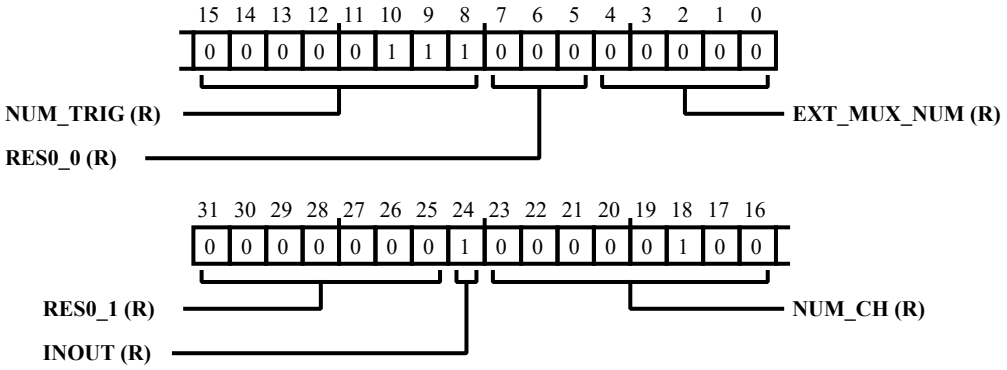


Figure 48-389: SOC600CTI_TRACE_DEVID Register Diagram

Table 48-388: SOC600CTI_TRACE_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:25 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
24 (R/NW)	INOUT	. Indicates channel inputs are also masked by the CTIGATE register. Always 1.
23:16 (R/NW)	NUM_CH	. The number of channels. Always either 4 or 16, set by CHANNEL_WIDTH.
15:8 (R/NW)	NUM_TRIG	. Indicates the maximum number of triggers - the maximum of the two parameters, NUM_EVENT_RECEIVERS and NUM_EVENT_TRANSMITTERS.
7:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4:0 (R/NW)	EXT_MUX_NUM	. Indicates the value of the EXT_MUX_NUM parameter, which determines if there is any external multiplexing on the trigger inputs and outputs. 0 indicates no multiplexing.

Device Type Identifier Register.

A debugger can use this register to get information about a component that has an unrecognized Part number.

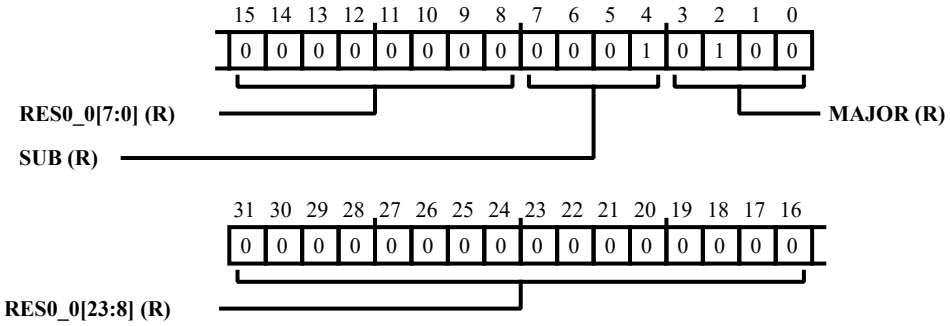


Figure 48-390: SOC600CTI_TRACE_DEVTYPE Register Diagram

Table 48-389: SOC600CTI_TRACE_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SUB	. Minor classification. Returns 0x1, indicating this component is a Trigger-Matrix.
3:0 (R/NW)	MAJOR	. Major classification. Returns 0x4, indicating this component performs Debug Control.

Integration Test Channel Input register.

Integration test mode register, used to view channel events. The integration test register includes a latch that is set when a pulse is received on a channel input. When read, a register bit reads as 1 if the channel has received a pulse since it was last read. The act of reading the register automatically clears the 1 to a 0. When performing integration testing it is therefore important to coordinate the setting of event latches and reading/clearing them.

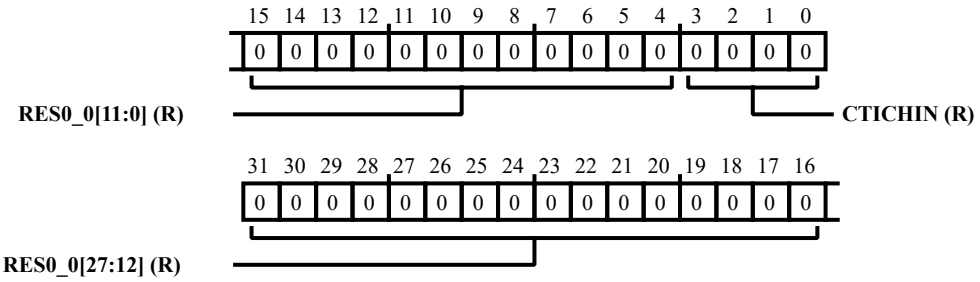


Figure 48-391: SOC600CTI_TRACE_ITCHIN Register Diagram

Table 48-390: SOC600CTI_TRACE_ITCHIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
3:0 (R/NW)	CTICHIN	. Reads the latched value of the channel inputs.

Integration Test Channel Output register.

Integration test mode register, used to generate channel events. Writing to the register creates a single pulse on the output. ITCHOUT is self-clearing.

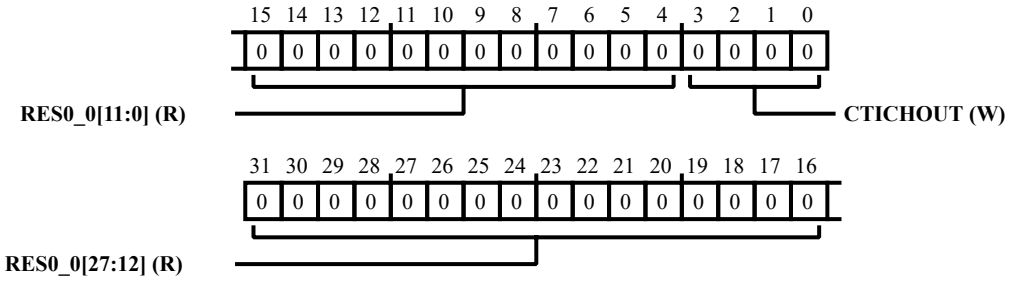


Figure 48-392: SOC600CTI_TRACE_ITCHOUT Register Diagram

Table 48-391: SOC600CTI_TRACE_ITCHOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (RX/W)	CTICHOUT	. Pulses the channel outputs.	
		0	No effect.
		1	Pulse channel event for one clk cycle.

Integration Mode Control Register.

The Integration Mode Control register is used to switch between functional mode and integration mode. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

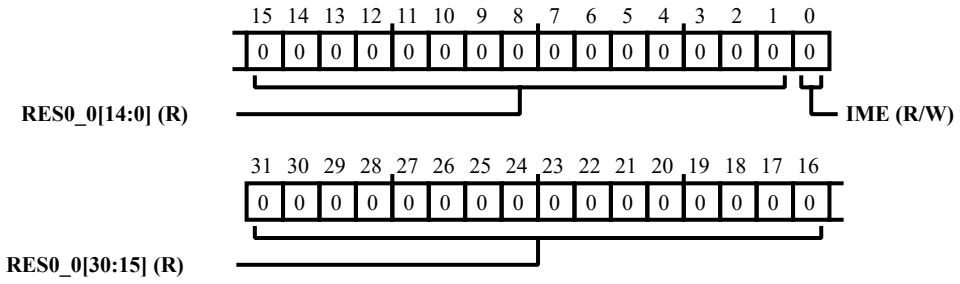


Figure 48-393: SOC600CTI_TRACE_ITCTRL Register Diagram

Table 48-392: SOC600CTI_TRACE_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	IME	. Integration Mode Enable.	
		0	The component must enter functional mode.
		1	The component must enter integration mode, and enable support for topology detection and integration testing.

Integration Test Trigger Input register.

Integration test mode register, used to view trigger events. The integration test register includes a latch that is set when a pulse is received on a trigger input. When read, a register bit reads as 1 if the trigger input has received a pulse since it was last read. The act of reading the register automatically clears the 1 to a 0. When performing integration testing it is therefore important to coordinate the setting of event latches and reading/clearing them.

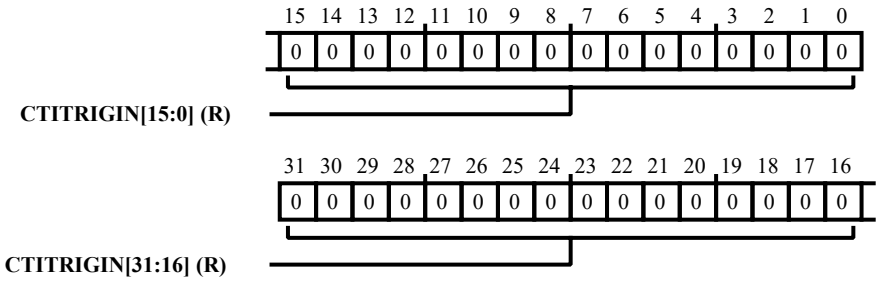


Figure 48-394: SOC600CTI_TRACE_ITTRIGIN Register Diagram

Table 48-393: SOC600CTI_TRACE_ITTRIGIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CTITRIGIN	. Reads the latched value of the trigger inputs.

Integration Test Trigger Output register.

Integration test mode register, used to generate trigger events.

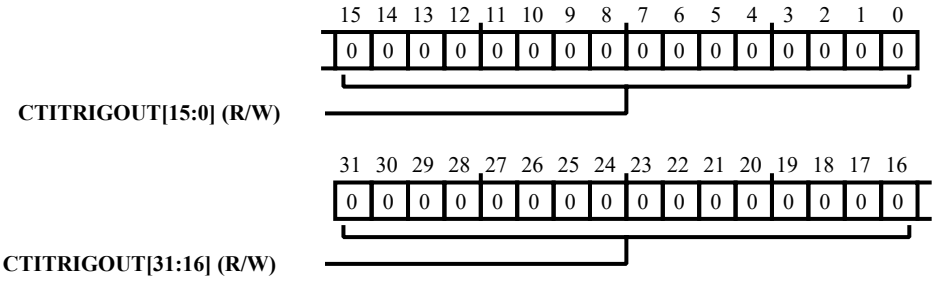


Figure 48-395: SOC600CTI_TRACE_ITTRIGOUT Register Diagram

Table 48-394: SOC600CTI_TRACE_ITTRIGOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	CTITRIGOUT	. Set/clear trigger output signal. Reads return the value in the register if SW_HANDSHAKE=1, otherwise 0 is returned if SW_HANDSHAKE=0. Writes:.	
		0	Clears the trigger output if SW_HANDSHAKE=1, no effect if SW_HANDSHAKE=0.
		1	Sets the trigger output if SW_HANDSHAKE=1, pulses trigger output if SW_HANDSHAKE=0.

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

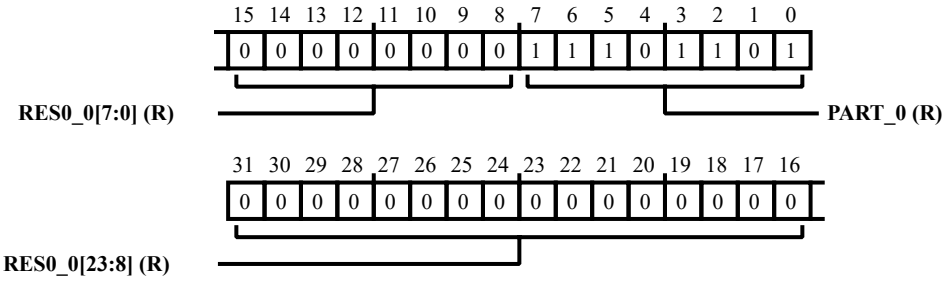


Figure 48-396: SOC600CTI_TRACE_PIDR0 Register Diagram

Table 48-395: SOC600CTI_TRACE_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PART_0	. Part number, bits[7:0]. Taken together with PIDR1.PART_1 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

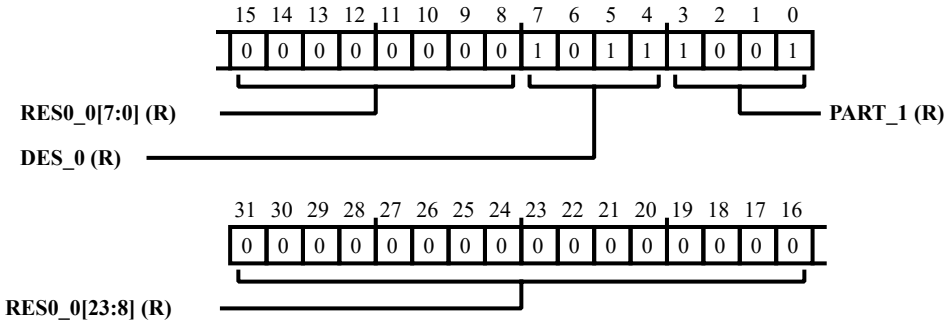


Figure 48-397: SOC600CTI_TRACE_PIDR1 Register Diagram

Table 48-396: SOC600CTI_TRACE_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

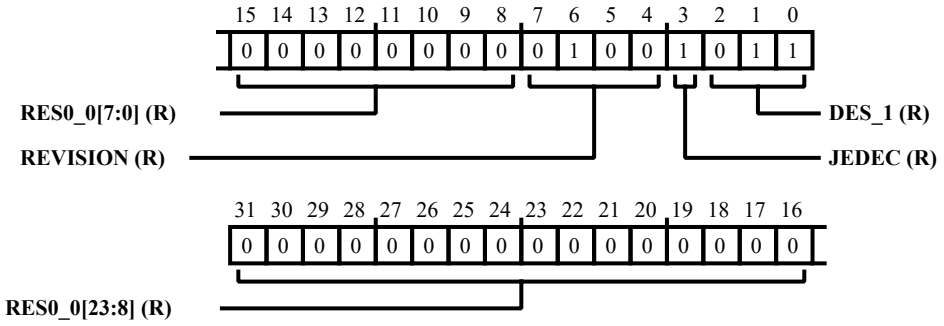


Figure 48-398: SOC600CTI_TRACE_PIDR2 Register Diagram

Table 48-397: SOC600CTI_TRACE_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

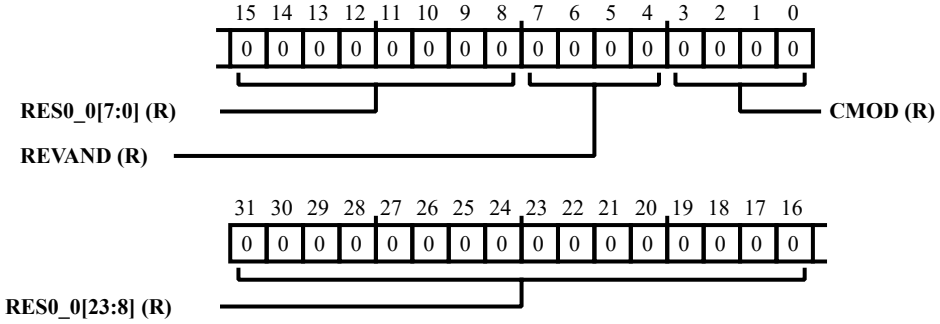


Figure 48-399: SOC600CTI_TRACE_PIDR3 Register Diagram

Table 48-398: SOC600CTI_TRACE_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

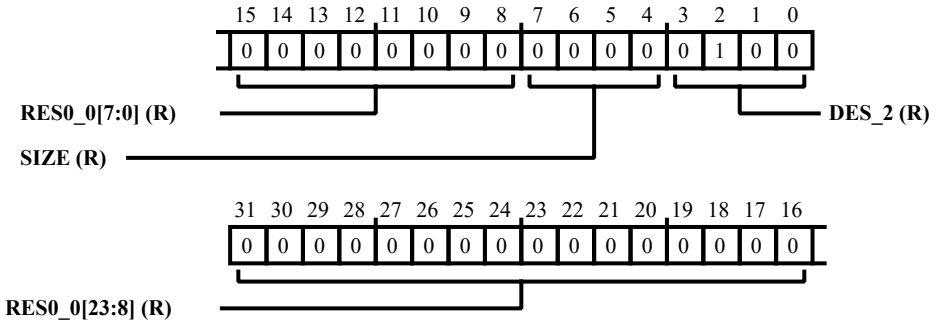


Figure 48-400: SOC600CTI_TRACE_PIDR4 Register Diagram

Table 48-399: SOC600CTI_TRACE_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

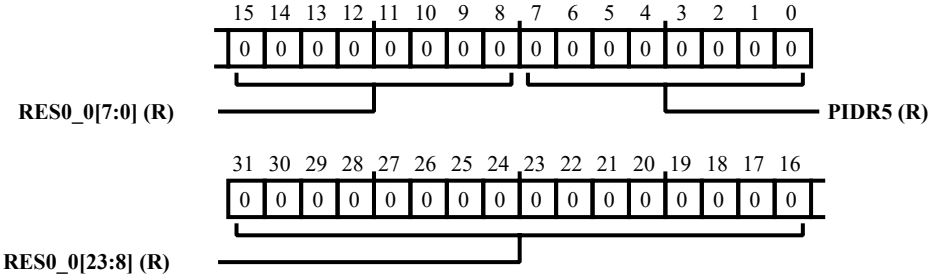


Figure 48-401: SOC600CTI_TRACE_PIDR5 Register Diagram

Table 48-400: SOC600CTI_TRACE_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

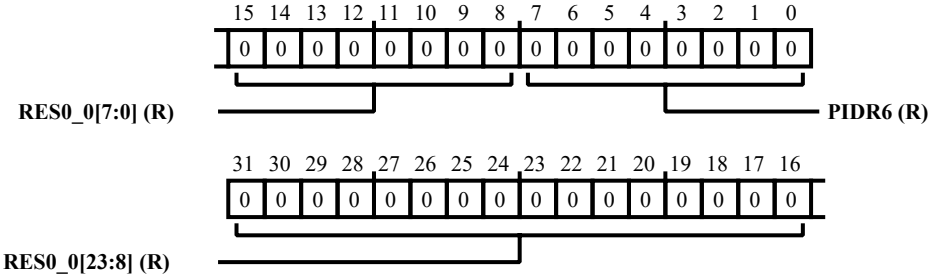


Figure 48-402: SOC600CTI_TRACE_PIDR6 Register Diagram

Table 48-401: SOC600CTI_TRACE_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

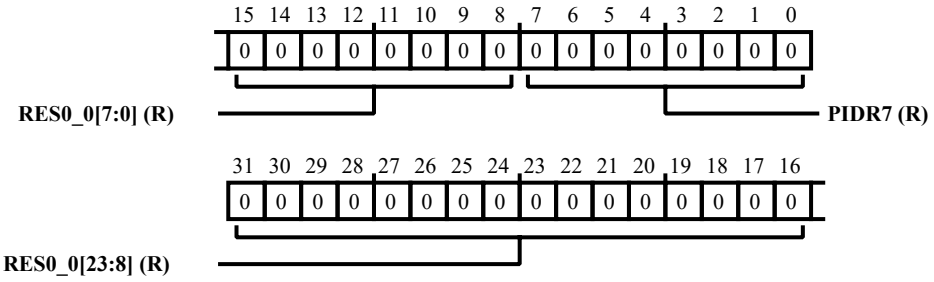


Figure 48-403: SOC600CTI_TRACE_PIDR7 Register Diagram

Table 48-402: SOC600CTI_TRACE_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

ADSP-2184x SYSCTI Register Descriptions

Your module description, here. (SYSCTI) contains the following registers.

Table 48-403: ADSP-2184x SYSCTI Register List

Name	Description
SYSCTI_ASICCTRL	External Multiplexer Control register.
SYSCTI_AUTHSTATUS	Authentication Status Register.
SYSCTI_CIDR0	Component Identification Register 0.
SYSCTI_CIDR1	Component Identification Register 1.
SYSCTI_CIDR2	Component Identification Register 2.
SYSCTI_CIDR3	Component Identification Register 3.
SYSCTI_CLAIMCLR	Claim Tag Clear Register.
SYSCTI_CLAIMSET	Claim Tag Set Register.
SYSCTI_CTIAPPCLEAR	CTI Application Channel Clear register.
SYSCTI_CTIAPPPULSE	CTI Application Channel Pulse register.

Table 48-403: ADSP-2184x SYSCTI Register List (Continued)

Name	Description
SYSCTI_CTIAPPSET	CTI Application Channel Set register.
SYSCTI_CTICHINSTATUS	CTI Channel Input Status register.
SYSCTI_CTICHOUTSTATUS	CTI Channel Output Status register.
SYSCTI_CTICONTROL	CTI Control register.
SYSCTI_CTIGATE	Enable CTI Channel Gate register.
SYSCTI_CTIINEN0	CTI Trigger 0 to Channel Enable register.
SYSCTI_CTIINEN1	CTI Trigger 1 to Channel Enable register.
SYSCTI_CTIINEN2	CTI Trigger 2 to Channel Enable register.
SYSCTI_CTIINEN3	CTI Trigger 3 to Channel Enable register.
SYSCTI_CTIINEN4	CTI Trigger 4 to Channel Enable register.
SYSCTI_CTIINEN5	CTI Trigger 5 to Channel Enable register.
SYSCTI_CTIINEN6	CTI Trigger 6 to Channel Enable register.
SYSCTI_CTIINEN7	CTI Trigger 7 to Channel Enable register.
SYSCTI_CTIINTACK	CTI Interrupt Acknowledge register.
SYSCTI_CTIOUTEN0	CTI Channel to Trigger 0 Enable register.
SYSCTI_CTIOUTEN1	CTI Channel to Trigger 1 Enable register.
SYSCTI_CTIOUTEN10	CTI Channel to Trigger 10 Enable register.
SYSCTI_CTIOUTEN11	CTI Channel to Trigger 11 Enable register.
SYSCTI_CTIOUTEN12	CTI Channel to Trigger 12 Enable register.
SYSCTI_CTIOUTEN13	CTI Channel to Trigger 13 Enable register.
SYSCTI_CTIOUTEN2	CTI Channel to Trigger 2 Enable register.
SYSCTI_CTIOUTEN3	CTI Channel to Trigger 3 Enable register.
SYSCTI_CTIOUTEN4	CTI Channel to Trigger 4 Enable register.
SYSCTI_CTIOUTEN5	CTI Channel to Trigger 5 Enable register.
SYSCTI_CTIOUTEN6	CTI Channel to Trigger 6 Enable register.
SYSCTI_CTIOUTEN7	CTI Channel to Trigger 7 Enable register.
SYSCTI_CTIOUTEN8	CTI Channel to Trigger 8 Enable register.
SYSCTI_CTIOUTEN9	CTI Channel to Trigger 9 Enable register.
SYSCTI_CTITRIGINSTATUS	CTI Trigger Input Status register.
SYSCTI_CTITRIGOUTSTATUS	CTI Trigger Output Status register.
SYSCTI_DEVAFF0	Device Affinity register 0.

Table 48-403: ADSP-2184x SYSCTI Register List (Continued)

Name	Description
SYSCTI_DEVAFF1	Device Affinity register 1.
SYSCTI_DEVARCH	Device Architecture Register.
SYSCTI_DEVID	Device Configuration Register.
SYSCTI_DEVTYPE	Device Type Identifier Register.
SYSCTI_ITCHIN	Integration Test Channel Input register.
SYSCTI_ITCHOUT	Integration Test Channel Output register.
SYSCTI_ITCTRL	Integration Mode Control Register.
SYSCTI_ITTRIGIN	Integration Test Trigger Input register.
SYSCTI_ITTRIGOUT	Integration Test Trigger Output register.
SYSCTI_PIDR0	Peripheral Identification Register 0.
SYSCTI_PIDR1	Peripheral Identification Register 1.
SYSCTI_PIDR2	Peripheral Identification Register 2.
SYSCTI_PIDR3	Peripheral Identification Register 3.
SYSCTI_PIDR4	Peripheral Identification Register 4.
SYSCTI_PIDR5	Peripheral Identification Register 5.
SYSCTI_PIDR6	Peripheral Identification Register 6.
SYSCTI_PIDR7	Peripheral Identification Register 7.

External Multiplexer Control register.

I/O port control.

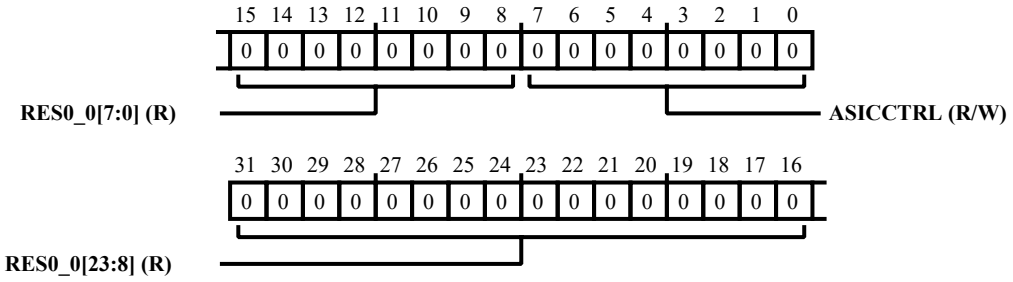


Figure 48-404: SYSCTI_ASICCTRL Register Diagram

Table 48-404: SYSCTI_ASICCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:0 (R/W)	ASICCTRL	. Set and clear external output signal.	
		0	Clear output bit to 0.
		1	Set output bit to 1.

Authentication Status Register.

Reports the current status of the authentication control signals.

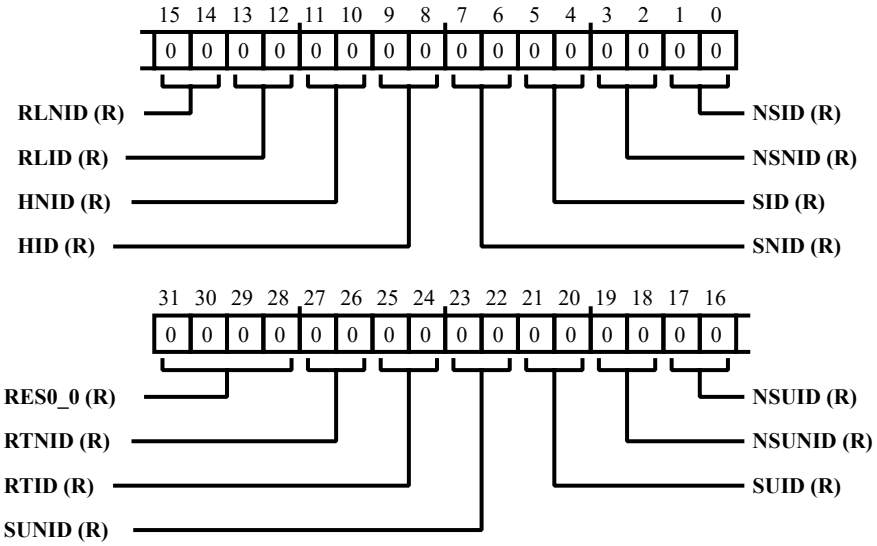


Figure 48-405: SYSCTI_AUTHSTATUS Register Diagram

Table 48-405: SYSCTI_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
27:26 (R/NW)	RTNID	. Root non-invasive debug.	
		0	Separate Root non-invasive debug enable not implemented or Root state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.

Table 48-405: SYSCTI_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	RTID	. Root invasive debug.	
		0	Separate Root invasive debug enable not implemented or Root state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
23:22 (R/NW)	SUNID	. Secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
21:20 (R/NW)	SUID	. Secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
19:18 (R/NW)	NSUNID	. Non-secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
17:16 (R/NW)	NSUID	. Non-secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-405: SYSCTI_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/NW)	RLNID	. Realm non-invasive debug.	
		0	Separate Realm non-invasive debug enable not implemented or Realm state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
13:12 (R/NW)	RLID	. Realm invasive debug.	
		0	Separate Realm invasive debug enable not implemented or Realm state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
11:10 (R/NW)	HNID	. Hypervisor non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
9:8 (R/NW)	HID	. Hypervisor invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-405: SYSCTI_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	. Secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
5:4 (R/NW)	SID	. Secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
3:2 (R/NW)	NSNID	. Non-secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
1:0 (R/NW)	NSID	. Non-secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

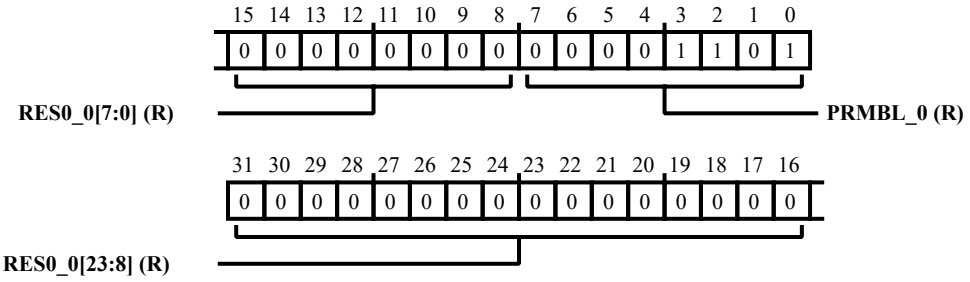


Figure 48-406: SYSCTI_CIDR0 Register Diagram

Table 48-406: SYSCTI_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

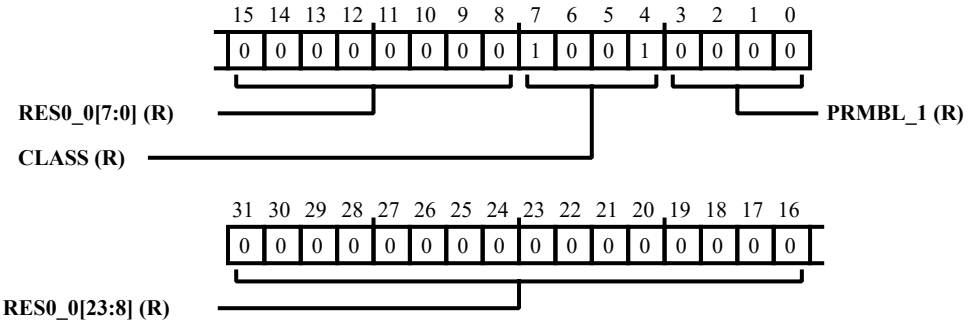


Figure 48-407: SYSCTI_CIDR1 Register Diagram

Table 48-407: SYSCTI_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

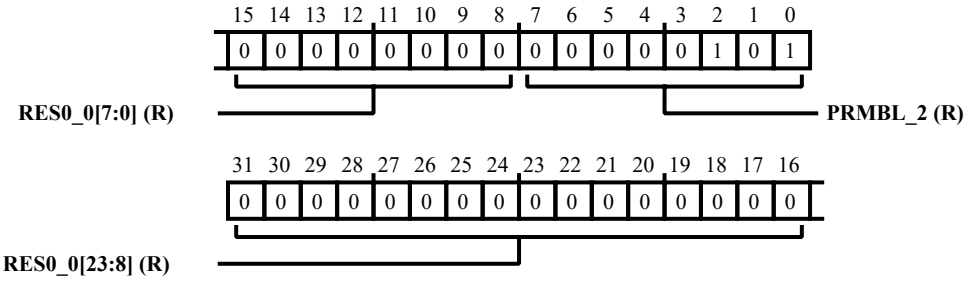


Figure 48-408: SYSCTI_CIDR2 Register Diagram

Table 48-408: SYSCTI_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

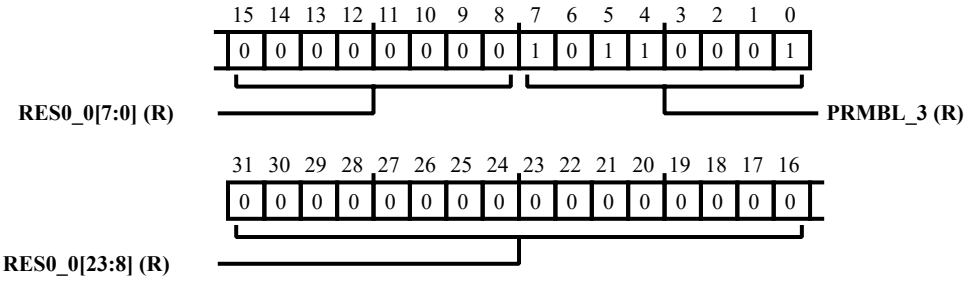


Figure 48-409: SYSCTI_CIDR3 Register Diagram

Table 48-409: SYSCTI_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Claim Tag Clear Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

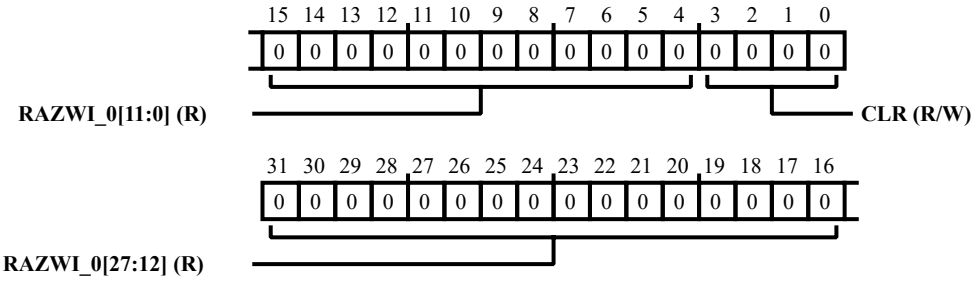


Figure 48-410: SYSCTI_CLAIMCLR Register Diagram

Table 48-410: SYSCTI_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	CLR	. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

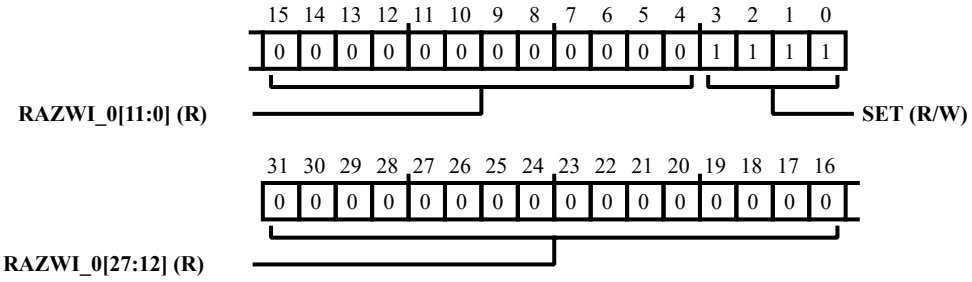


Figure 48-411: SYSCTI_CLAIMSET Register Diagram

Table 48-411: SYSCTI_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	SET	. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

CTI Application Channel Clear register.

The CTIAPPCLEAR register allows software to clear any channel output. Software can use this register to clear a channel event in place of a hardware source on a trigger input. This register must not be used in a system where all events are sent as single-cycle pulses. It is only retained for compatibility with older systems and software. The register is implemented before the CTIGATE register. Therefore, for the channel event to propagate outside the CTI, the corresponding CTIGATE bit must be set to 1.

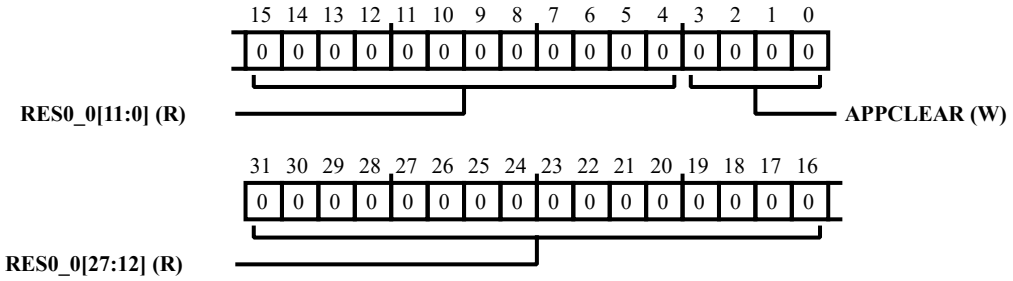


Figure 48-412: SYSCTI_CTIAPPCLEAR Register Diagram

Table 48-412: SYSCTI_CTIAPPCLEAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (RX/W)	APPCLEAR	. Clears the corresponding internal channel flag.	
		0	No effect.
		1	Clears the channel output.

CTI Application Channel Pulse register.

The application channel pulse register allows software to pulse any channel output. This register can be used by software to pulse a channel event in place of a hardware source on a trigger input. The register is implemented before the CTIGATE register so, for the channel event to propagate outside the CTI, it is necessary for the corresponding CTIGATE bit to be 1.

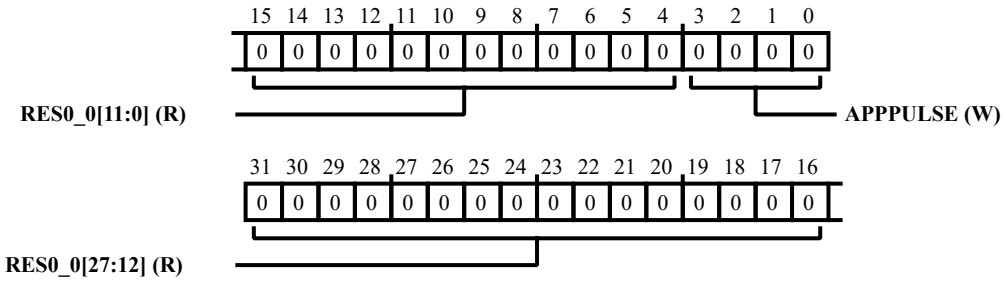


Figure 48-413: SYSCTI_CTIAPPPULSE Register Diagram

Table 48-413: SYSCTI_CTIAPPPULSE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (RX/W)	APPPULSE	. Pulses the channel outputs.	
		0	No effect.
		1	Pulse channel event for one clk cycle.

CTI Application Channel Set register.

The CTIAPPSET register allows software to set any channel output. Software can use this register to generate a channel event in place of a hardware source on a trigger input. This register must not be used in a system where all events are sent as single-cycle pulses. It is only retained for compatibility with older systems and software. The register is implemented before the CTIGATE register. Therefore, for the channel event to propagate outside the CTI, the corresponding CTIGATE bit must be set to 1.

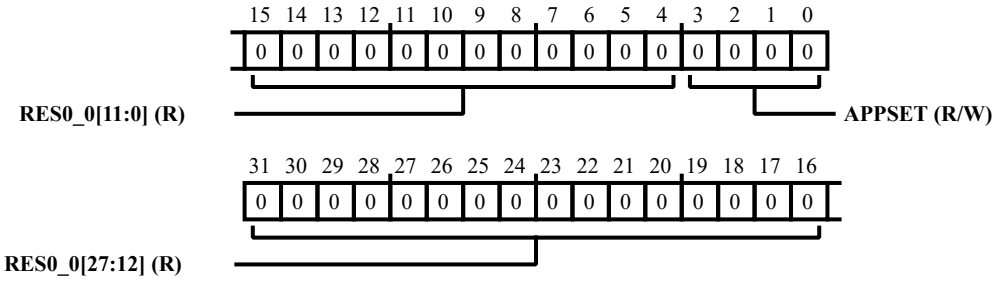


Figure 48-414: SYSCTI_CTIAPPSET Register Diagram

Table 48-414: SYSCTI_CTIAPPSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	APPSET	. Sets the corresponding internal channel flag:.	
		0	Read: application channel is inactive. Write: has no effect.
		1	Read: application channel is active. Write: sets the channel output.

CTI Channel Input Status register.

Channel input status. If the channel input is driven by a source that generates single cycle pulses, this register is generally read as 0.

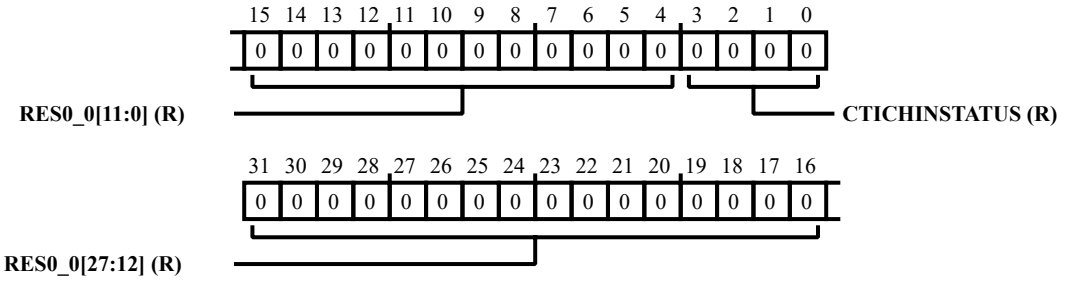


Figure 48-415: SYSCTI_CTICHINSTATUS Register Diagram

Table 48-415: SYSCTI_CTICHINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/NW)	CTICHINSTATUS	. Channel input status.	
		0	One bit per channel input. 0 means that the input is LOW.
		1	One bit per channel input. 1 means that the input is HIGH.

CTI Channel Output Status register.

Channel output status. The register only has meaning if the trigger source drives static levels, rather than pulses.

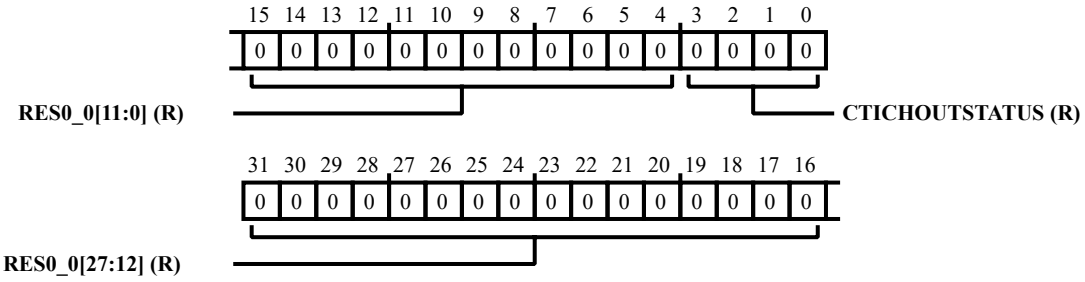


Figure 48-416: SYSCTI_CTICHOUTSTATUS Register Diagram

Table 48-416: SYSCTI_CTICHOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/NW)	CTICHOUTSTATUS	. Channel output status.	
		0	One bit per channel output. 0 means that the output is LOW.
		1	One bit per channel output. 1 means that the output is HIGH.

CTI Control register.

The CTICONTROL register enables and disables the CTI.

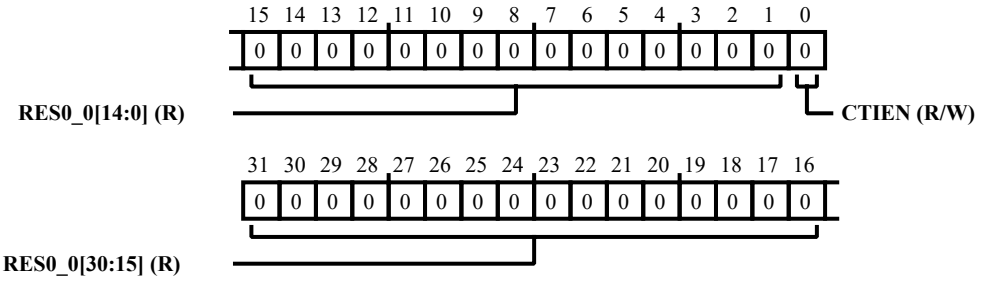


Figure 48-417: SYSCTI_CTICONTROL Register Diagram

Table 48-417: SYSCTI_CTICONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	CTIEN	. Enable control:.	
		0	CTI disabled.
		1	CTI enabled.

Enable CTI Channel Gate register.

Channel output gate.

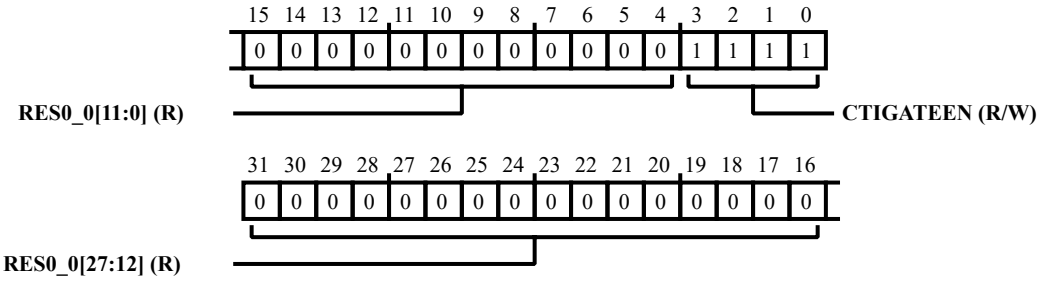


Figure 48-418: SYSCTI_CTIGATE Register Diagram

Table 48-418: SYSCTI_CTIGATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	CTIGATEEN	. Enables the propagation of channel events out of the CTI, one bit per channel.	
		0	Disable a channel from propagating.
		1	Enable channel propagation.

CTI Trigger 0 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

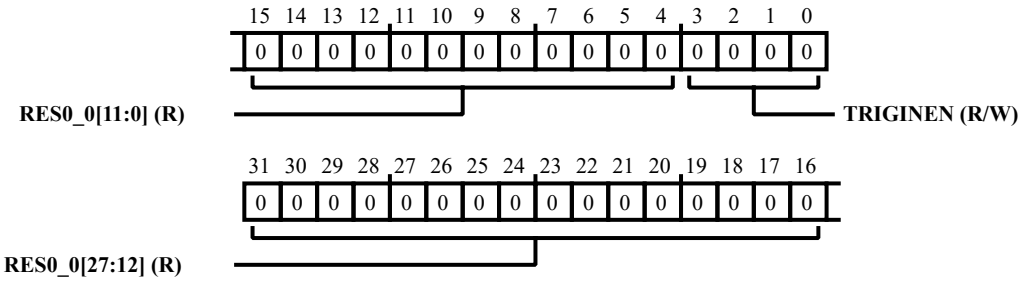


Figure 48-419: SYSCTI_CTIINEN0 Register Diagram

Table 48-419: SYSCTI_CTIINEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 0 events are ignored by the corresponding channel.
		1	When an event is received on event_in[0], generate an event on the channel corresponding to this bit.

CTI Trigger 1 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

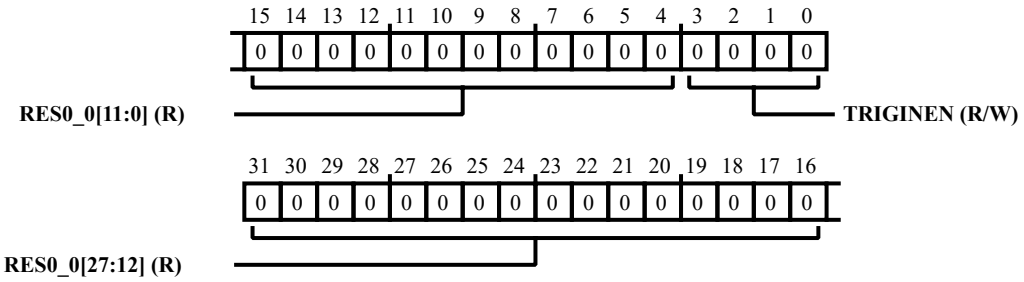


Figure 48-420: SYSCTI_CTIINEN1 Register Diagram

Table 48-420: SYSCTI_CTIINEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 1 events are ignored by the corresponding channel.
		1	When an event is received on event_in[1], generate an event on the channel corresponding to this bit.

CTI Trigger 2 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

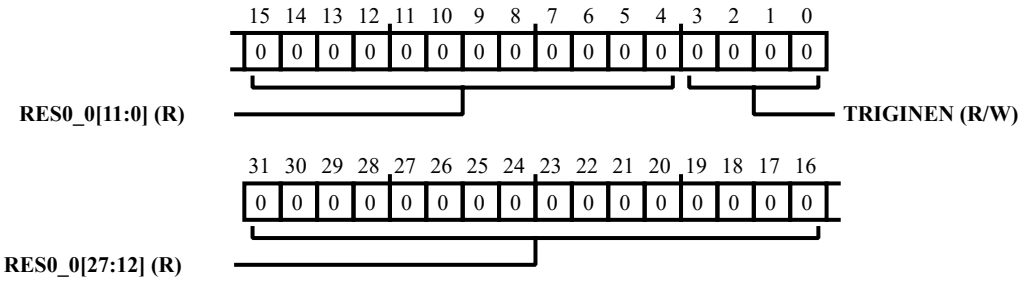


Figure 48-421: SYSCTI_CTIINEN2 Register Diagram

Table 48-421: SYSCTI_CTIINEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 2 events are ignored by the corresponding channel.
		1	When an event is received on event_in[2], generate an event on the channel corresponding to this bit.

CTI Trigger 3 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

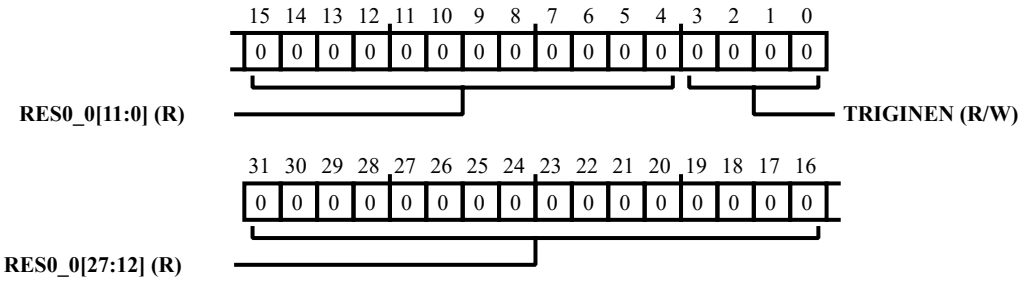


Figure 48-422: SYSCTI_CTIINEN3 Register Diagram

Table 48-422: SYSCTI_CTIINEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 3 events are ignored by the corresponding channel.
		1	When an event is received on event_in[3], generate an event on the channel corresponding to this bit.

CTI Trigger 4 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

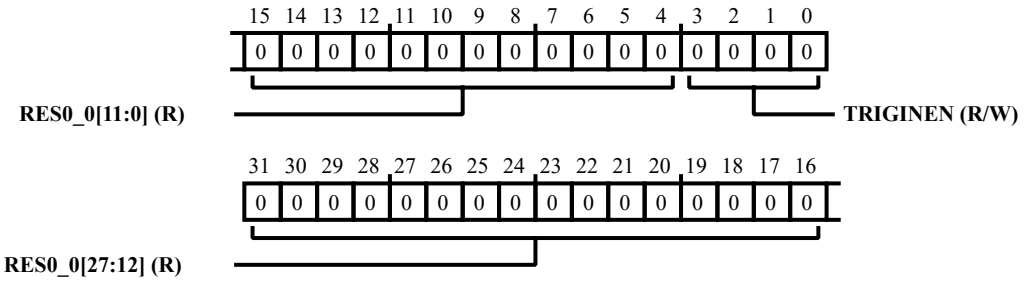


Figure 48-423: SYSCTI_CTIINEN4 Register Diagram

Table 48-423: SYSCTI_CTIINEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 4 events are ignored by the corresponding channel.
		1	When an event is received on event_in[4], generate an event on the channel corresponding to this bit.

CTI Trigger 5 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

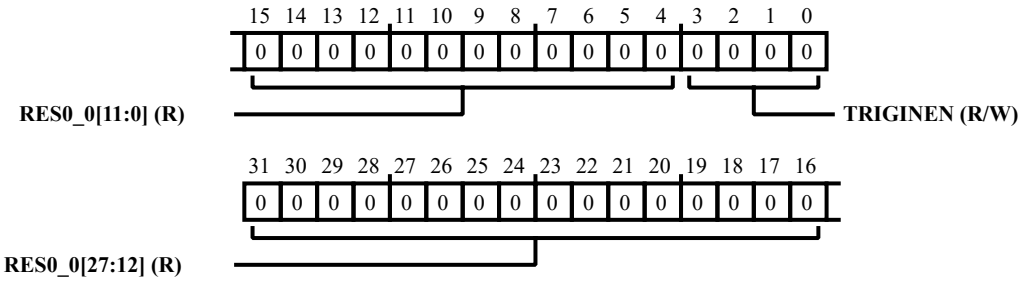


Figure 48-424: SYSCTI_CTIINEN5 Register Diagram

Table 48-424: SYSCTI_CTIINEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 5 events are ignored by the corresponding channel.
		1	When an event is received on event_in[5], generate an event on the channel corresponding to this bit.

CTI Trigger 6 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

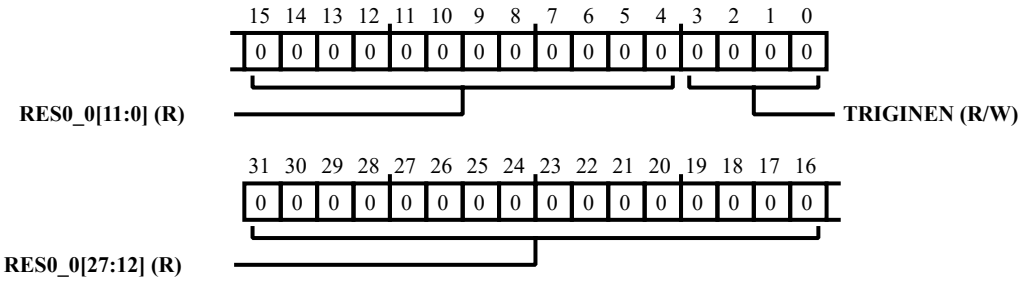


Figure 48-425: SYSCTI_CTIINEN6 Register Diagram

Table 48-425: SYSCTI_CTIINEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 6 events are ignored by the corresponding channel.
		1	When an event is received on event_in[6], generate an event on the channel corresponding to this bit.

CTI Trigger 7 to Channel Enable register.

This register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

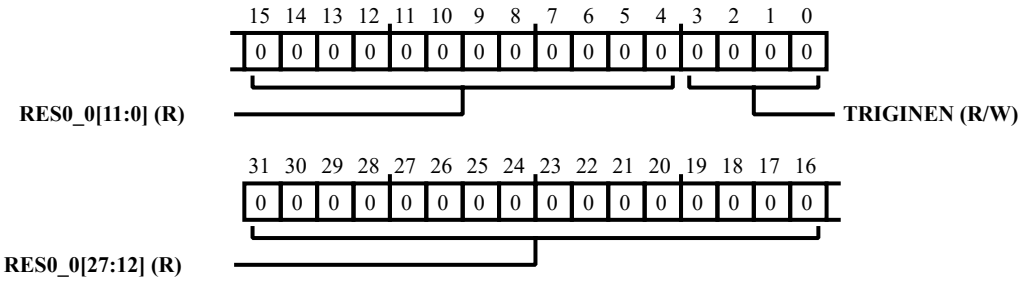


Figure 48-426: SYSCTI_CTIINEN7 Register Diagram

Table 48-426: SYSCTI_CTIINEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGINEN	. Trigger input to channel mapping.	
		0	Input trigger 7 events are ignored by the corresponding channel.
		1	When an event is received on event_in[7], generate an event on the channel corresponding to this bit.

CTI Interrupt Acknowledge register.

The CTIINTACK register acknowledges trigger outputs. It is a bit map that allows selective clearing of trigger output events. If the SW_HANDSHAKE parameter for a trigger output is set, indicating that the output latches HIGH when an event is sent to that output, then the output remains HIGH until the corresponding INTACK bit is written to a 1. A write of a bit to 0 has no effect. This allows different software threads to be responsible for clearing different trigger outputs without needing to perform a read-modify-write operation to find which bits are set.

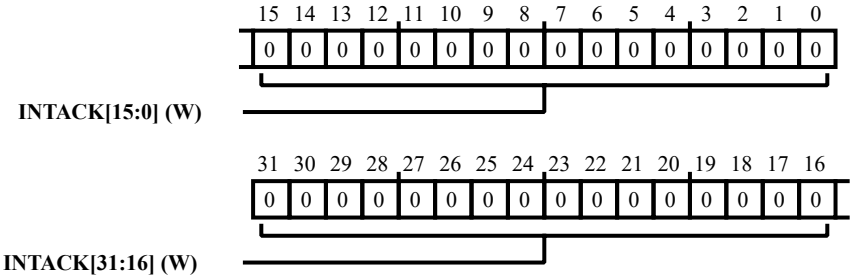


Figure 48-427: SYSCTI_CTIINTACK Register Diagram

Table 48-427: SYSCTI_CTIINTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	INTACK	. Acknowledges the corresponding event_out output.

CTI Channel to Trigger 0 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

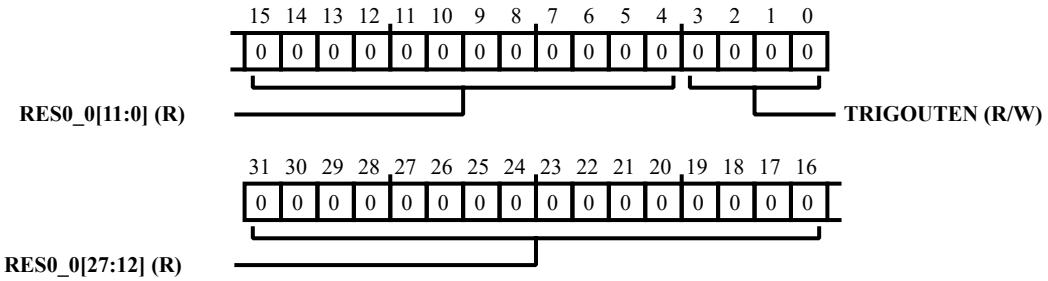


Figure 48-428: SYSCTI_CTIOUTEN0 Register Diagram

Table 48-428: SYSCTI_CTIOUTEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger0.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[0].

CTI Channel to Trigger 1 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

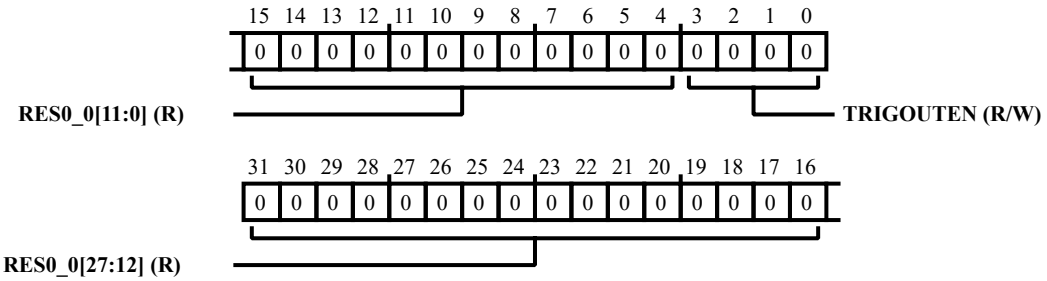


Figure 48-429: SYSCTI_CTIOUTEN1 Register Diagram

Table 48-429: SYSCTI_CTIOUTEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger1.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[1].

CTI Channel to Trigger 10 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

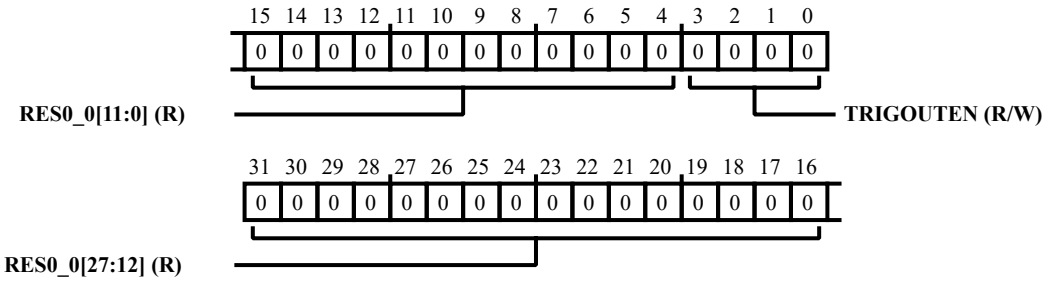


Figure 48-430: SYSCTI_CTIOUTEN10 Register Diagram

Table 48-430: SYSCTI_CTIOUTEN10 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger10.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[10].

CTI Channel to Trigger 11 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

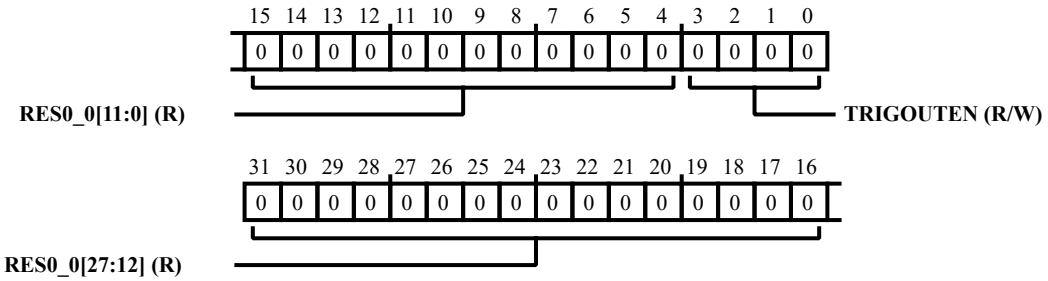


Figure 48-431: SYSCTI_CTIOUTEN11 Register Diagram

Table 48-431: SYSCTI_CTIOUTEN11 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger11.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[11].

CTI Channel to Trigger 12 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

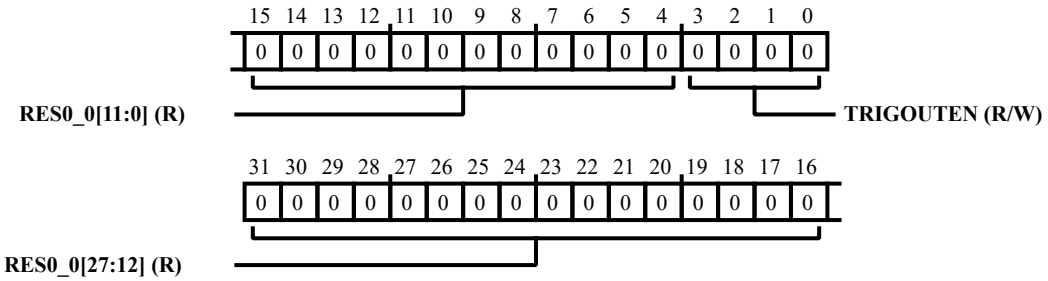


Figure 48-432: SYSCTI_CTIOUTEN12 Register Diagram

Table 48-432: SYSCTI_CTIOUTEN12 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger12.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[12].

CTI Channel to Trigger 13 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

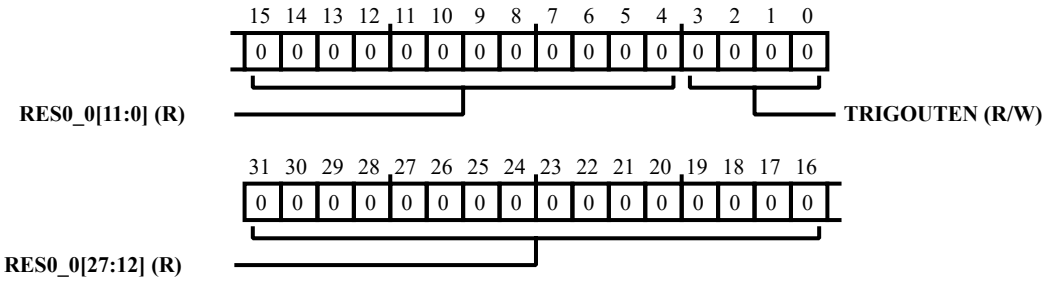


Figure 48-433: SYSCTI_CTIOUTEN13 Register Diagram

Table 48-433: SYSCTI_CTIOUTEN13 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger13.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[13].

CTI Channel to Trigger 2 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

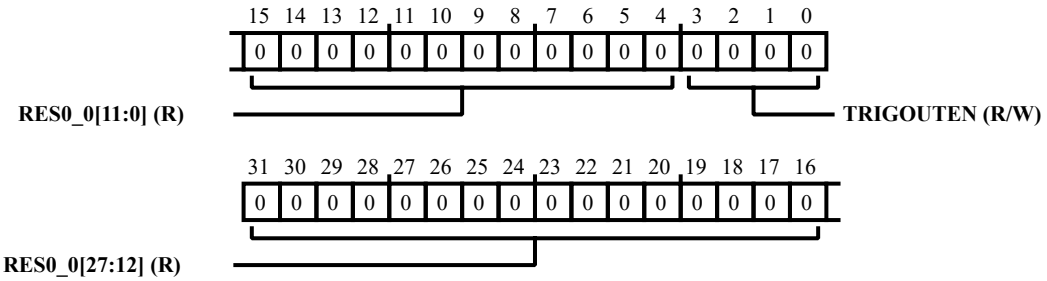


Figure 48-434: SYSCTI_CTIOUTEN2 Register Diagram

Table 48-434: SYSCTI_CTIOUTEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger2.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[2].

CTI Channel to Trigger 3 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

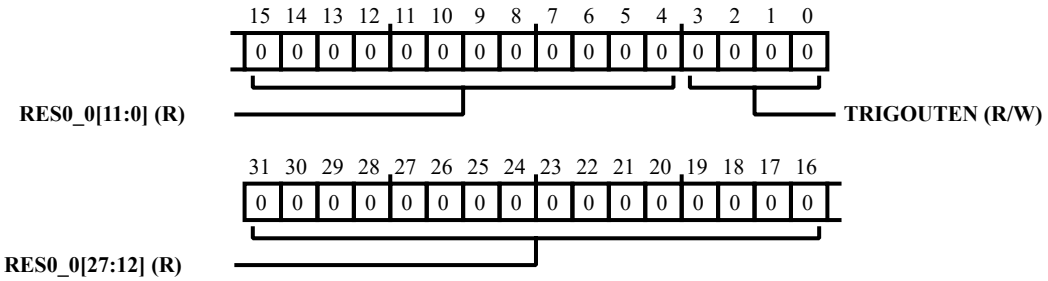


Figure 48-435: SYSCTI_CTIOUTEN3 Register Diagram

Table 48-435: SYSCTI_CTIOUTEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger3.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[3].

CTI Channel to Trigger 4 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

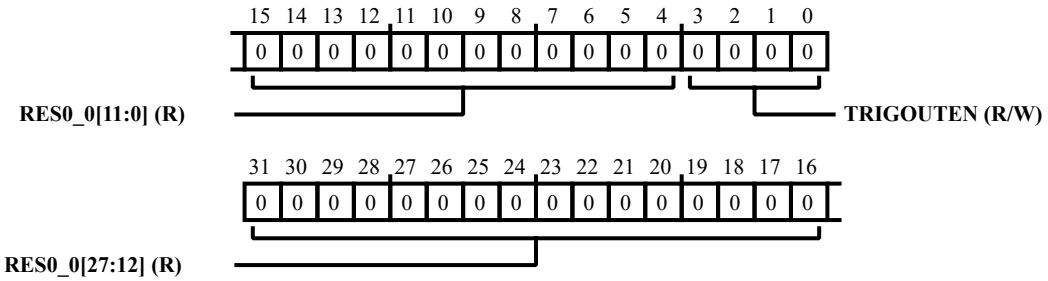


Figure 48-436: SYSCTI_CTIOUTEN4 Register Diagram

Table 48-436: SYSCTI_CTIOUTEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger4.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[4].

CTI Channel to Trigger 5 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

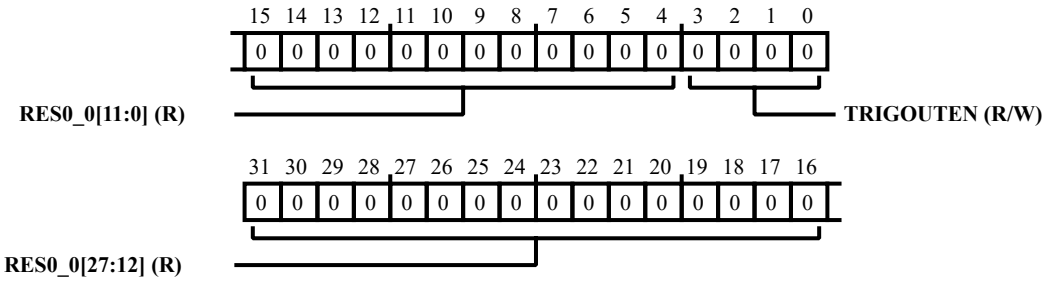


Figure 48-437: SYSCTI_CTIOUTEN5 Register Diagram

Table 48-437: SYSCTI_CTIOUTEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger5.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[5].

CTI Channel to Trigger 6 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

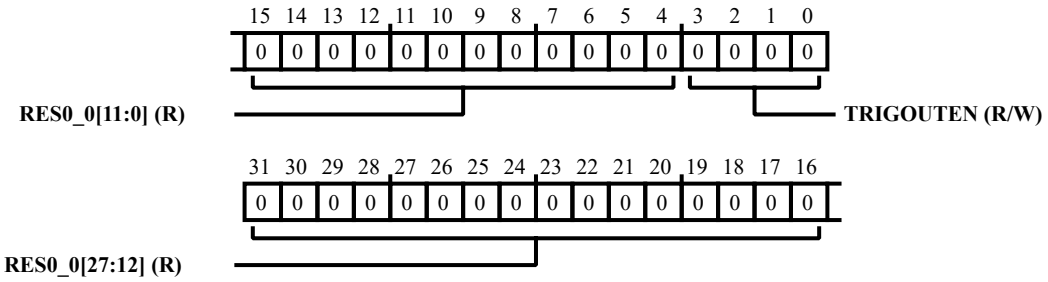


Figure 48-438: SYSCTI_CTIOUTEN6 Register Diagram

Table 48-438: SYSCTI_CTIOUTEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger6.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[6].

CTI Channel to Trigger 7 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

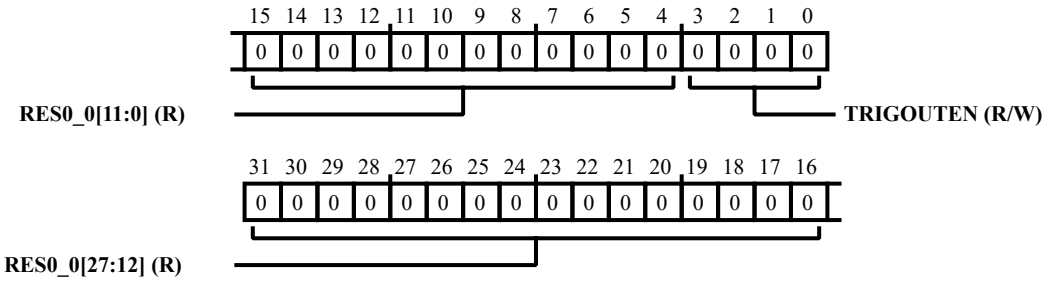


Figure 48-439: SYSCTI_CTIOUTEN7 Register Diagram

Table 48-439: SYSCTI_CTIOUTEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger7.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[7].

CTI Channel to Trigger 8 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

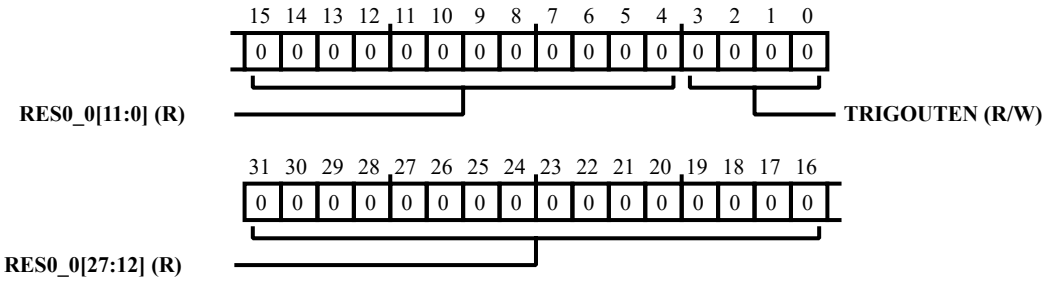


Figure 48-440: SYSCTI_CTIOUTEN8 Register Diagram

Table 48-440: SYSCTI_CTIOUTEN8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger8.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[8].

CTI Channel to Trigger 9 Enable register.

This register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (2^CHANNEL_WIDTH - 1). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

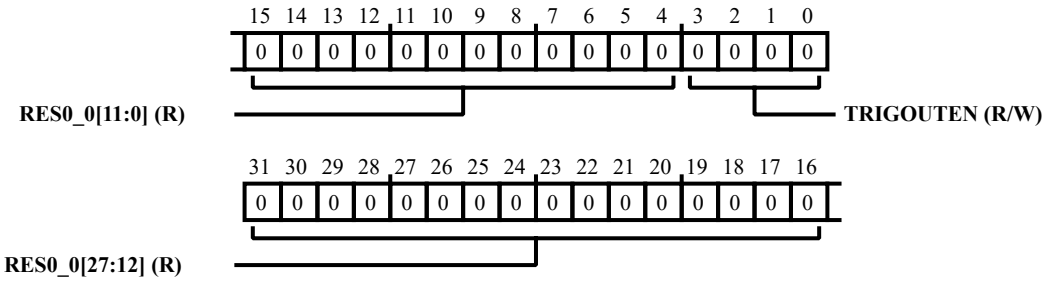


Figure 48-441: SYSCTI_CTIOUTEN9 Register Diagram

Table 48-441: SYSCTI_CTIOUTEN9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (R/W)	TRIGOUTEN	. Channel to trigger output mapping.	
		0	The corresponding channel is ignored by the output trigger9.
		1	When an event occurs on the channel corresponding to this bit, generate an event on event_out[9].

CTI Trigger Input Status register.

Trigger input status. If the event_in input is driven by a source that generates single cycle pulses, this register is generally read as 0.

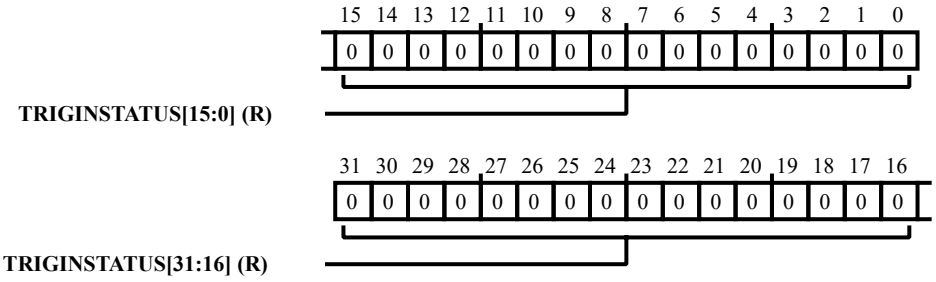


Figure 48-442: SYSCTI_CTITRIGINSTATUS Register Diagram

Table 48-442: SYSCTI_CTITRIGINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	TRIGINSTATUS	. Trigger input status.	
		0	One bit per trigger input. 0 means that the input is LOW.
		1	One bit per trigger input. 1 means that the input is HIGH.

CTI Trigger Output Status register.

Trigger output status. The register only has meaning if the trigger source drives static levels, rather than pulses.

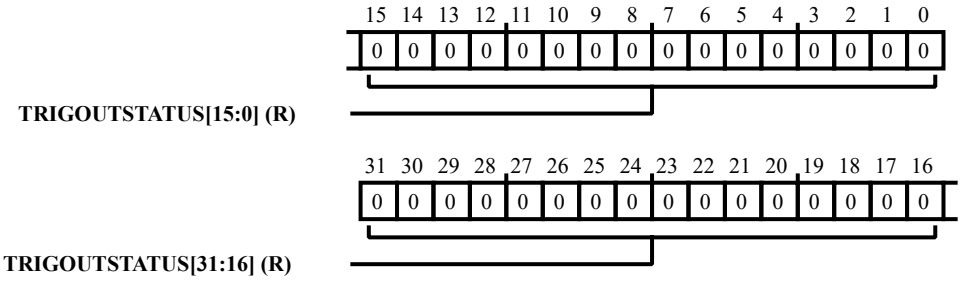


Figure 48-443: SYSCTI_CTITRIGOUTSTATUS Register Diagram

Table 48-443: SYSCTI_CTITRIGOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	TRIGOUTSTATUS	. Trigger output status.	
		0	One bit per trigger output. 0 means that the output is LOW.
		1	One bit per trigger output. 1 means that the output is HIGH.

Device Affinity register 0.

Enables a debugger to determine if two components have an affinity with each other.

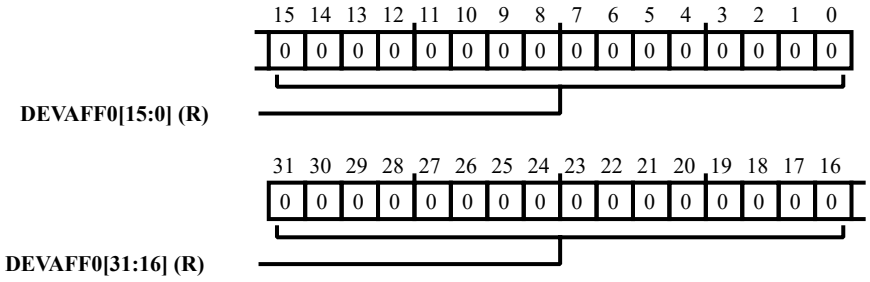


Figure 48-444: SYSCTI_DEVAFF0 Register Diagram

Table 48-444: SYSCTI_DEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVAFF0	. Lower 32-bits of DEVAFF. The value is set by the devaff[31:0] tie-off inputs.

Device Affinity register 1.

Enables a debugger to determine if two components have an affinity with each other.

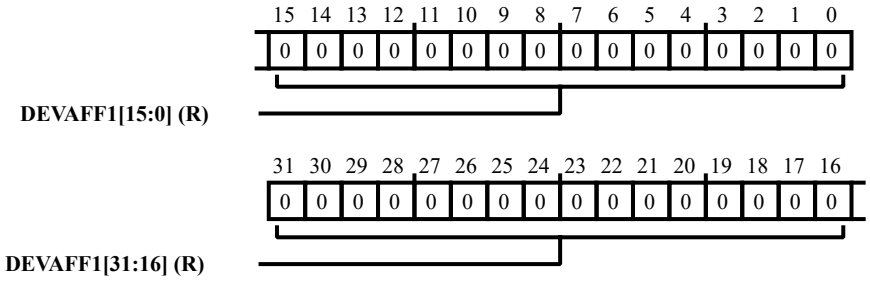


Figure 48-445: SYSCTI_DEVAFF1 Register Diagram

Table 48-445: SYSCTI_DEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVAFF1	. Upper 32-bits of DEVAFF. The value is set by the devaff[63:32] tie-off inputs.

Device Architecture Register.

Identifies the architect and architecture of a CoreSight component. The architect might differ from the designer of a component, for example Arm defines the architecture but another company designs and implements the component.

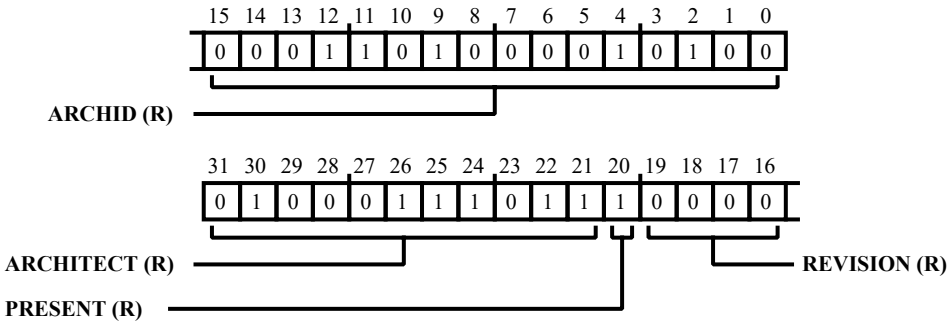


Figure 48-446: SYSCTI_DEVARCH Register Diagram

Table 48-446: SYSCTI_DEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:21 (R/NW)	ARCHITECT	. Defines the architect of the component.	
		571	Arm.
20 (R/NW)	PRESENT	. Indicates the presence of this register.	
		0	DEVARCH is not present.
		1	DEVARCH is present.
19:16 (R/NW)	REVISION	. Architecture revision. Returns the revision of the architecture that the ARCHID field specifies.	
15:0 (R/NW)	ARCHID	. Architecture ID. Returns a value that identifies the architecture of the component.	
		2583	Memory Access Port v2 architecture.
		2599	JTAG Access Port v2 architecture.
		2631	Unknown Access Port v2 architecture.
		2807	CoreSight ROM architecture.
		6676	CTI architecture.

Device Configuration Register.

This register is IMPLEMENTATION DEFINED for each Part Number and Designer. The register indicates the capabilities of the component.

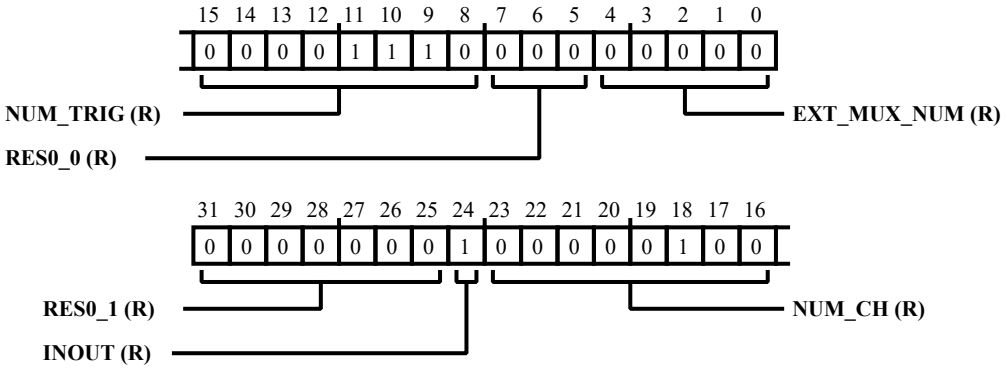


Figure 48-447: SYSCTI_DEVID Register Diagram

Table 48-447: SYSCTI_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:25 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
24 (R/NW)	INOUT	. Indicates channel inputs are also masked by the CTIGATE register. Always 1.
23:16 (R/NW)	NUM_CH	. The number of channels. Always either 4 or 16, set by CHANNEL_WIDTH.
15:8 (R/NW)	NUM_TRIG	. Indicates the maximum number of triggers - the maximum of the two parameters, NUM_EVENT_RECEIVERS and NUM_EVENT_TRANSMITTERS.
7:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4:0 (R/NW)	EXT_MUX_NUM	. Indicates the value of the EXT_MUX_NUM parameter, which determines if there is any external multiplexing on the trigger inputs and outputs. 0 indicates no multiplexing.

Device Type Identifier Register.

A debugger can use this register to get information about a component that has an unrecognized Part number.

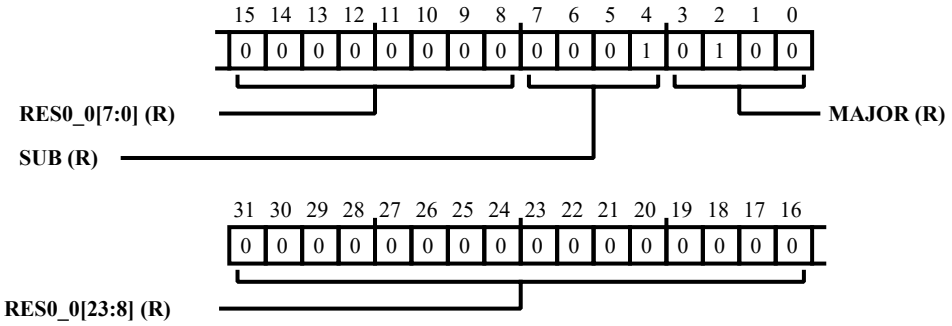


Figure 48-448: SYSCTI_DEVTYPE Register Diagram

Table 48-448: SYSCTI_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SUB	. Minor classification. Returns 0x1, indicating this component is a Trigger-Matrix.
3:0 (R/NW)	MAJOR	. Major classification. Returns 0x4, indicating this component performs Debug Control.

Integration Test Channel Input register.

Integration test mode register, used to view channel events. The integration test register includes a latch that is set when a pulse is received on a channel input. When read, a register bit reads as 1 if the channel has received a pulse since it was last read. The act of reading the register automatically clears the 1 to a 0. When performing integration testing it is therefore important to coordinate the setting of event latches and reading/clearing them.

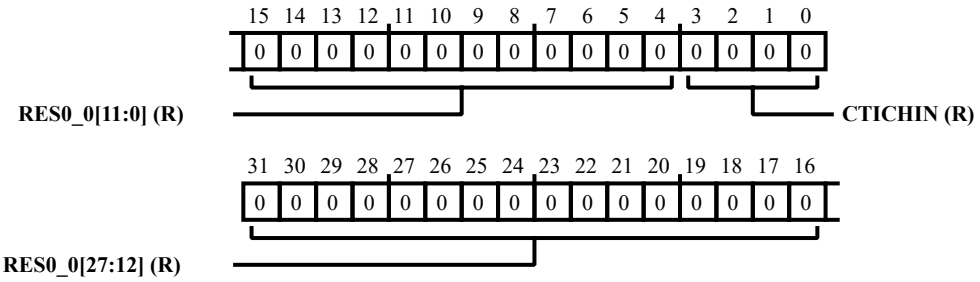


Figure 48-449: SYSCTI_ITCHIN Register Diagram

Table 48-449: SYSCTI_ITCHIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
3:0 (R/NW)	CTICHIN	. Reads the latched value of the channel inputs.

Integration Test Channel Output register.

Integration test mode register, used to generate channel events. Writing to the register creates a single pulse on the output. ITCHOUT is self-clearing.

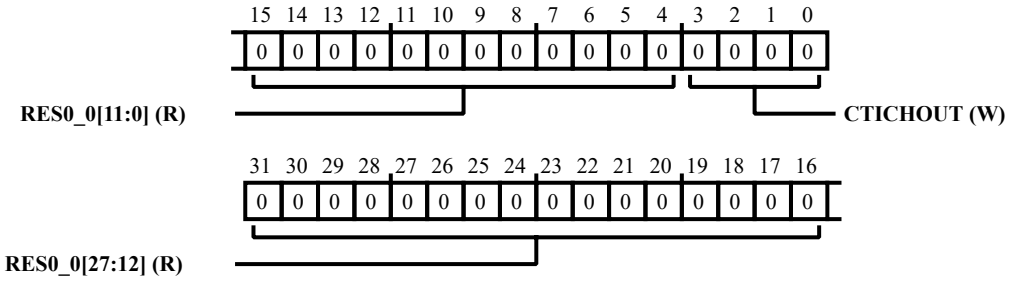


Figure 48-450: SYSCTI_ITCHOUT Register Diagram

Table 48-450: SYSCTI_ITCHOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3:0 (RX/W)	CTICHOUT	. Pulses the channel outputs.	
		0	No effect.
		1	Pulse channel event for one clk cycle.

Integration Mode Control Register.

The Integration Mode Control register is used to switch between functional mode and integration mode. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

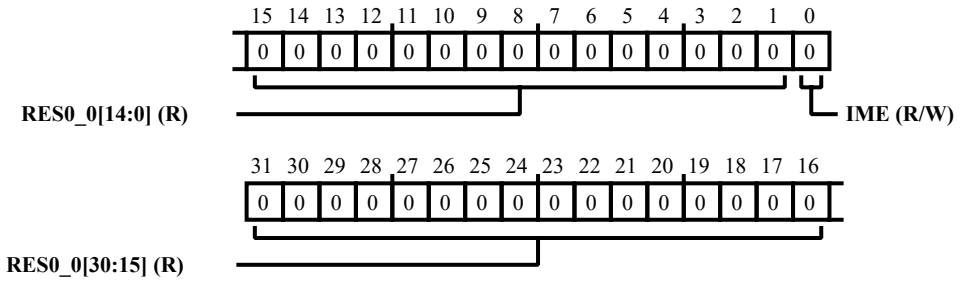


Figure 48-451: SYSCTI_ITCTRL Register Diagram

Table 48-451: SYSCTI_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	IME	. Integration Mode Enable.	
		0	The component must enter functional mode.
		1	The component must enter integration mode, and enable support for topology detection and integration testing.

Integration Test Trigger Input register.

Integration test mode register, used to view trigger events. The integration test register includes a latch that is set when a pulse is received on a trigger input. When read, a register bit reads as 1 if the trigger input has received a pulse since it was last read. The act of reading the register automatically clears the 1 to a 0. When performing integration testing it is therefore important to coordinate the setting of event latches and reading/clearing them.

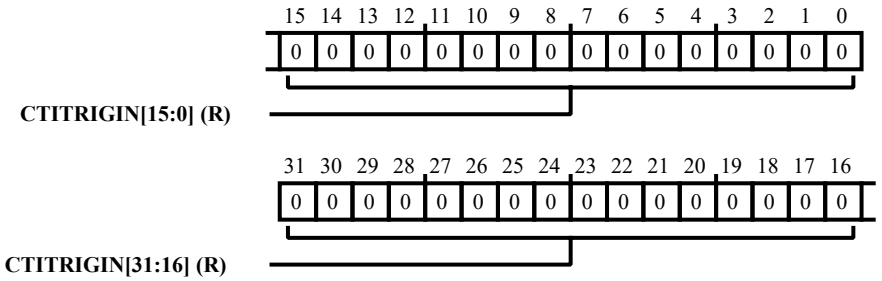


Figure 48-452: SYSCTI_ITTRIGIN Register Diagram

Table 48-452: SYSCTI_ITTRIGIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CTITRIGIN	. Reads the latched value of the trigger inputs.

Integration Test Trigger Output register.

Integration test mode register, used to generate trigger events.

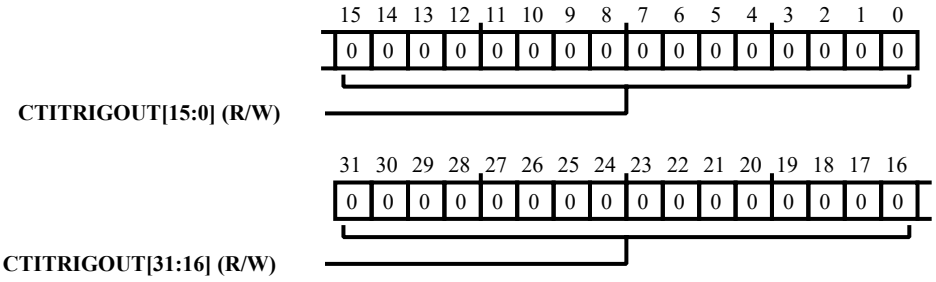


Figure 48-453: SYSCTI_ITTRIGOUT Register Diagram

Table 48-453: SYSCTI_ITTRIGOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	CTITRIGOUT	. Set/clear trigger output signal. Reads return the value in the register if SW_HANDSHAKE=1, otherwise 0 is returned if SW_HANDSHAKE=0. Writes:.	
		0	Clears the trigger output if SW_HANDSHAKE=1, no effect if SW_HANDSHAKE=0.
		1	Sets the trigger output if SW_HANDSHAKE=1, pulses trigger output if SW_HANDSHAKE=0.

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

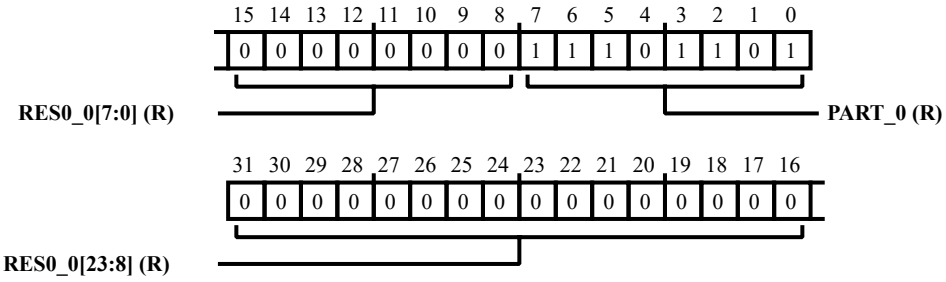


Figure 48-454: SYSCTI_PIDR0 Register Diagram

Table 48-454: SYSCTI_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PART_0	. Part number, bits[7:0]. Taken together with PIDR1.PART_1 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

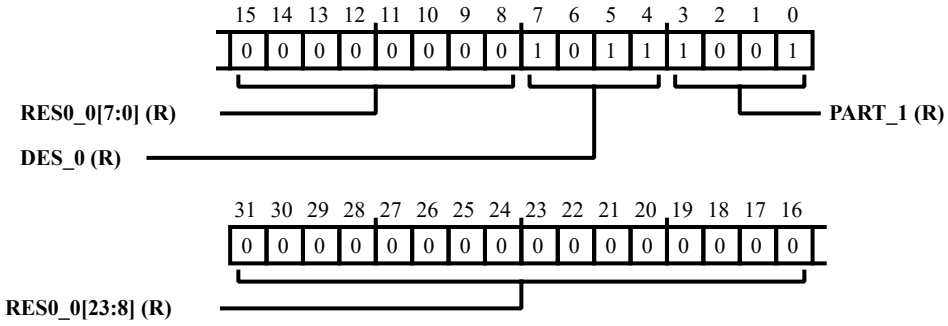


Figure 48-455: SYSCTI_PIDR1 Register Diagram

Table 48-455: SYSCTI_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

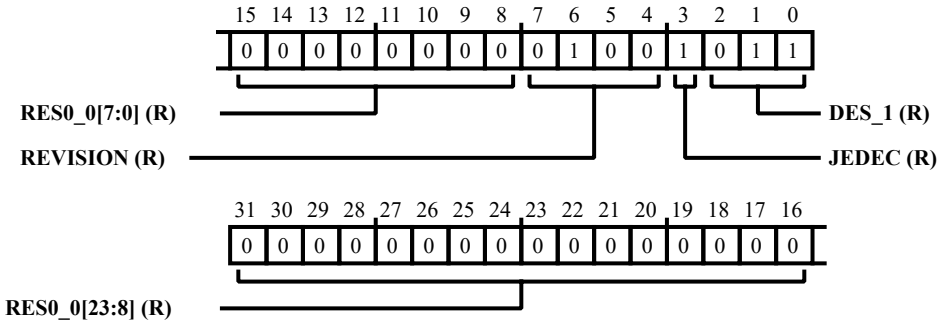


Figure 48-456: SYSCTI_PIDR2 Register Diagram

Table 48-456: SYSCTI_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

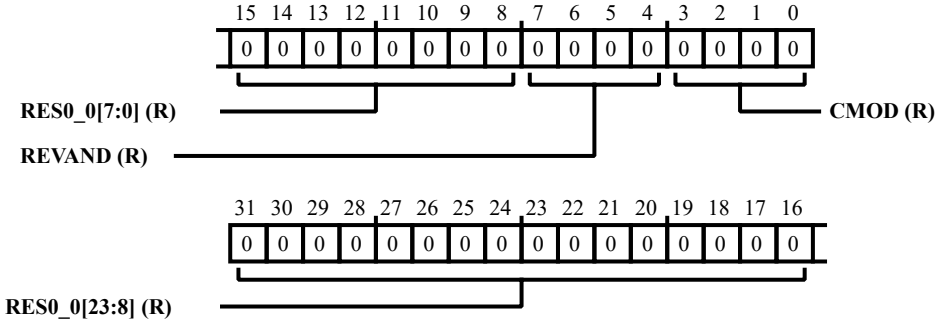


Figure 48-457: SYSCTI_PIDR3 Register Diagram

Table 48-457: SYSCTI_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

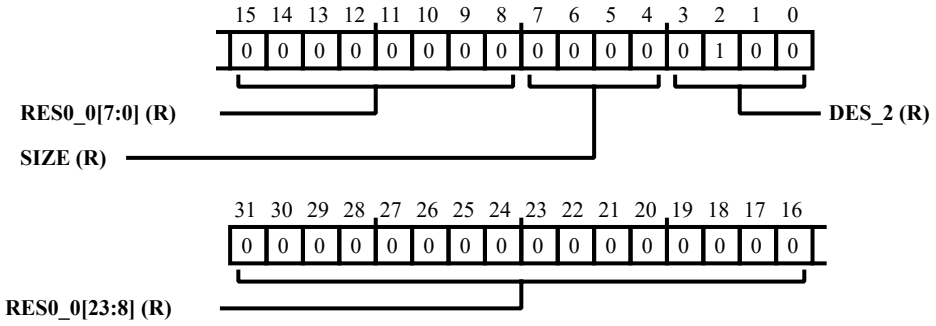


Figure 48-458: SYSCTI_PIDR4 Register Diagram

Table 48-458: SYSCTI_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

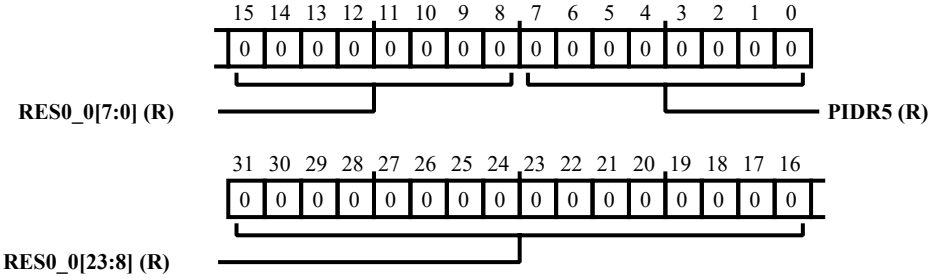


Figure 48-459: SYSCTI_PIDR5 Register Diagram

Table 48-459: SYSCTI_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

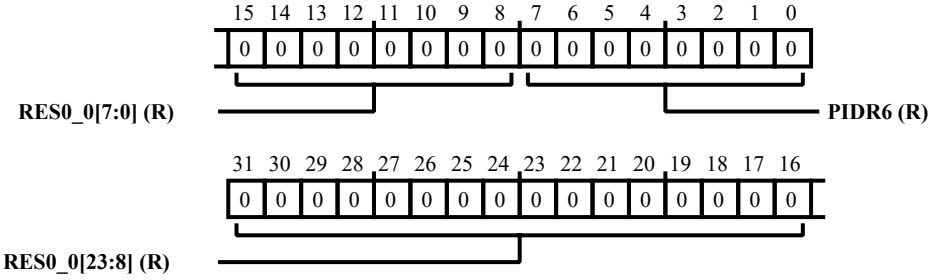


Figure 48-460: SYSCTI_PIDR6 Register Diagram

Table 48-460: SYSCTI_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

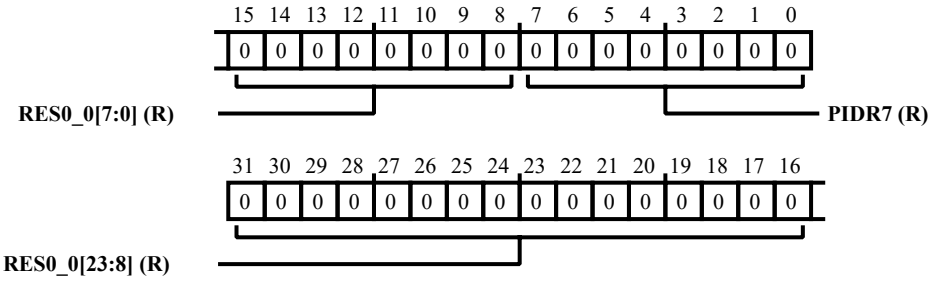


Figure 48-461: SYSCTI_PIDR7 Register Diagram

Table 48-461: SYSCTI_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

ADSP-2184x TAPC Register Descriptions

TAPC (TAPC) contains the following registers.

Table 48-462: ADSP-2184x TAPC Register List

Name	Description
TAPC_DBGCTL	Debug Control Register
TAPC_IDCODE	IDCODE Register
TAPC_SDBGKEY0	Secure Debug Key 0 Register
TAPC_SDBGKEY1	Secure Debug Key 1 Register
TAPC_SDBGKEY2	Secure Debug Key 2 Register
TAPC_SDBGKEY3	Secure Debug Key 3 Register
TAPC_SDBGKEY4	Secure Debug Key 4 Register
TAPC_SDBGKEY5	Secure Debug Key 5 Register
TAPC_SDBGKEY6	Secure Debug Key 6 Register
TAPC_SDBGKEY7	Secure Debug Key 7 Register

Table 48-462: ADSP-2184x TAPC Register List (Continued)

Name	Description
TAPC_SDBGKEY_CTL	Secure Debug Key Control Register
TAPC_SDBGKEY_STAT	Secure Debug Key Status Register
TAPC_USERCODE	USERCODE Register

Debug Control Register

The TAPC_DBGCTL register creates authentication signals to all CoreSight components in the system such as DBGEN, NIDEN, SPNIDEN and SPIDEN.

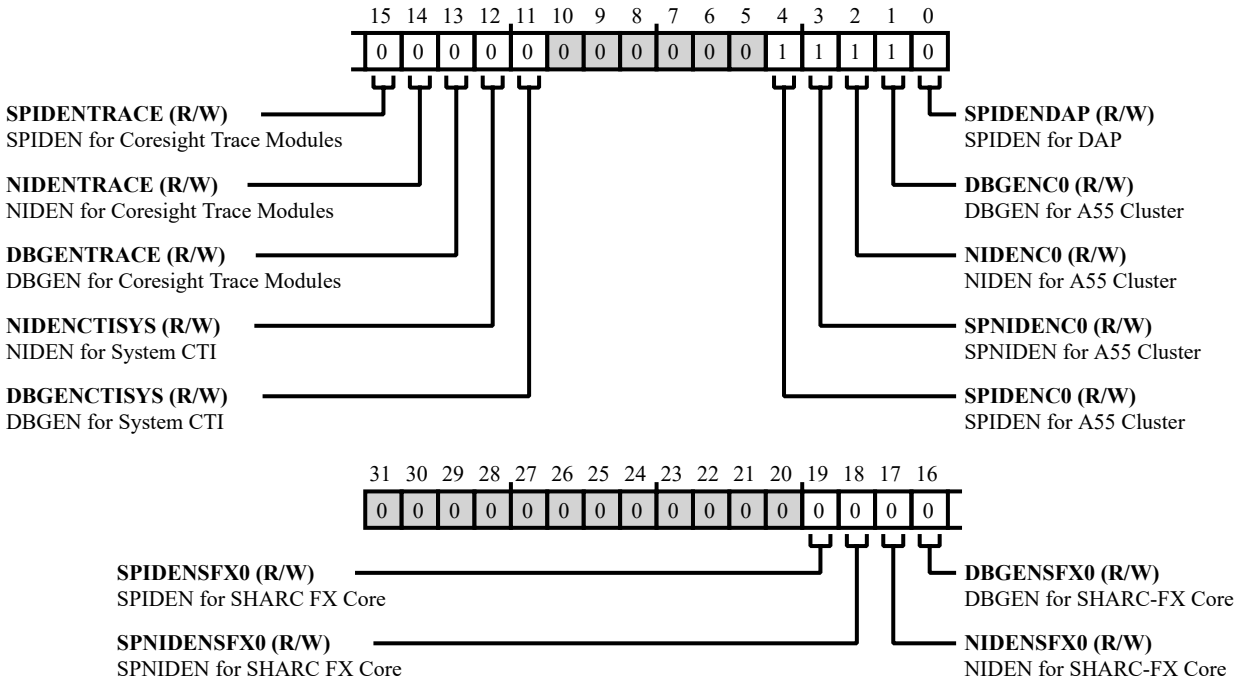


Figure 48-462: TAPC_DBGCTL Register Diagram

Table 48-463: TAPC_DBGCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	SPIDENSFX0	SPIDEN for SHARC FX Core. Secure Privileged Invasive Debug Enable for SHARC-FX Core
18 (R/W)	SPNIDENSFX0	SPNIDEN for SHARC FX Core. Secure Privileged Non-Invasive Debug Enable for SHARC-FX Core
17 (R/W)	NIDENSFX0	NIDEN for SHARC-FX Core. Non Invasive Debug Enable for SHARC FX Core
16 (R/W)	DBGENSFX0	DBGEN for SHARC-FX Core. Debug Enable for SHARC-FX Core
15 (R/W)	SPIDENTRACE	SPIDEN for Coresight Trace Modules.
14 (R/W)	NIDENTRACE	NIDEN for Coresight Trace Modules.

Table 48-463: TAPC_DBGCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	DBGENTRACE	DBGEN for Coresight Trace Modules.
12 (R/W)	NIDENCTISYS	NIDEN for System CTI.
11 (R/W)	DBGENCTISYS	DBGEN for System CTI.
4 (R/W)	SPIDENC0	SPIDEN for A55 Cluster.
3 (R/W)	SPNIDENC0	SPNIDEN for A55 Cluster.
2 (R/W)	NIDENC0	NIDEN for A55 Cluster.
1 (R/W)	DBGENC0	DBGEN for A55 Cluster.
0 (R/W)	SPIDENDAP	SPIDEN for DAP.

IDCODE Register

The TAPC_IDCODE register holds the IDCODE. The bit field is defined as follows.

IDCODE[31:28] = 0x1* – REVID

IDCODE[27:12] = 0x280B – JTAG ID

IDCODE[11:1] = 0x65 – Manufacturer ID

IDCODE[0] = 0x1

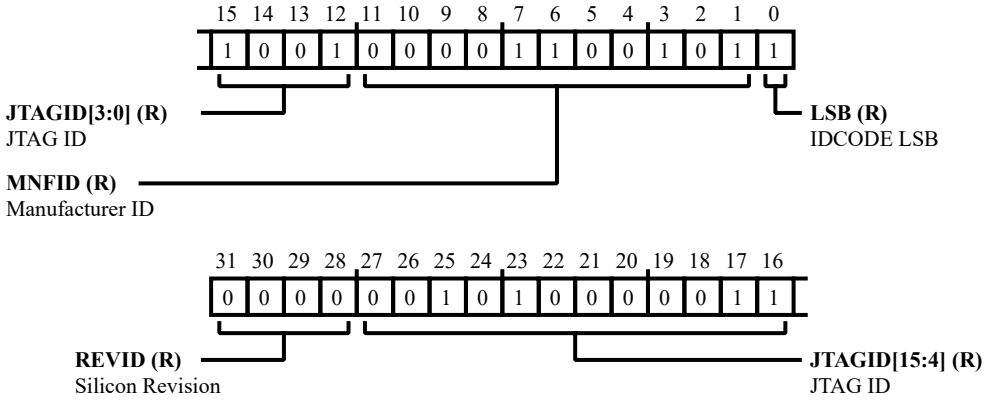


Figure 48-463: TAPC_IDCODE Register Diagram

Table 48-464: TAPC_IDCODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/NW)	REVID	Silicon Revision. The TAPC_IDCODE.REVID bit field holds the silicon revision. See the processor anomaly list for details.
27:12 (R/NW)	JTAGID	JTAG ID.
11:1 (R/NW)	MNFID	Manufacturer ID.
0 (R/NW)	LSB	IDCODE LSB.

Secure Debug Key 0 Register

The TAPC_SDBGKEY0 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. A debug key of 128 bits needs to be written into the Secure Debug Key registers (TAPC_SDBGKEY0, TAPC_SDBGKEY1, TAPC_SDBGKEY2, TAPC_SDBGKEY3) in the TAPC through the peripheral bus interface.

These registers hold the value of the key against which a matching key provided by the debug user is compared to enable a debug session. The task of writing these registers is performed (initially) by boot ROM code which copies a customer-selected key from the Flash memory info block to these registers.

An SDBGKEY value of all 0's is always an invalid key, a value of all 1's match the default value of the Secure Debug Key Compare registers and requires no entry in these registers. It is recommended programs have a significant number of 0's and 1's in a pseudo-random pattern throughout the 128-bit code for maximum protection.

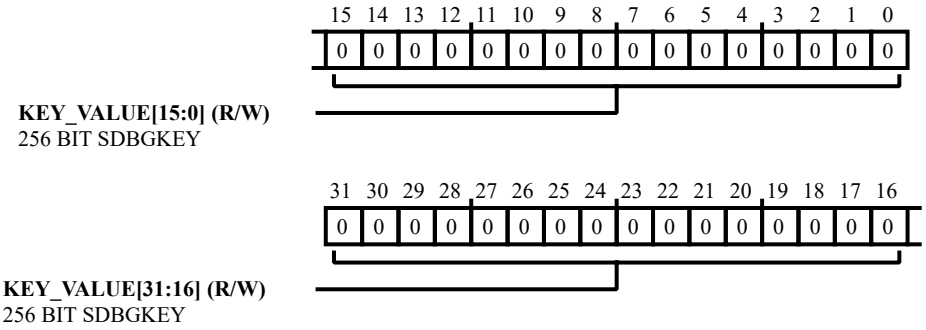


Figure 48-464: TAPC_SDBGKEY0 Register Diagram

Table 48-465: TAPC_SDBGKEY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The TAPC_SDBGKEY0.KEY_VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 1 Register

The TAPC_SDBGKEY1 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. See the TAPC_SDBGKEY0 register description for more information.

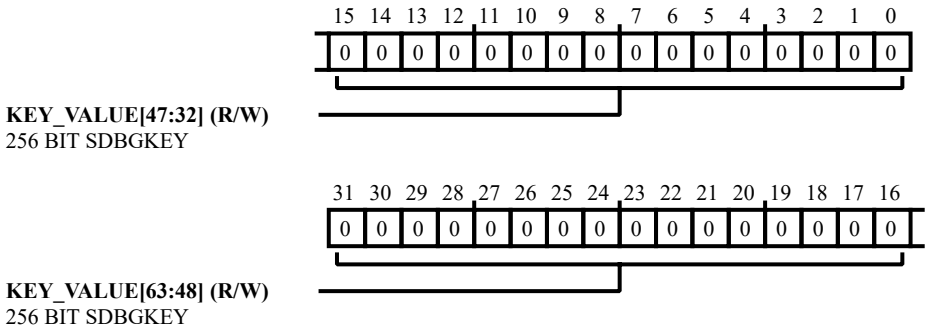


Figure 48-465: TAPC_SDBGKEY1 Register Diagram

Table 48-466: TAPC_SDBGKEY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The TAPC_SDBGKEY1.KEY_VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 2 Register

The TAPC_SDBGKEY2 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. See the TAPC_SDBGKEY0 register description for more information.

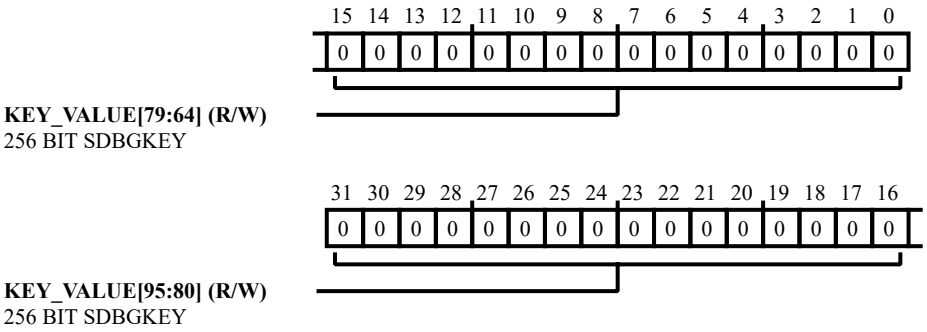


Figure 48-466: TAPC_SDBGKEY2 Register Diagram

Table 48-467: TAPC_SDBGKEY2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The TAPC_SDBGKEY2.KEY_VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 3 Register

The TAPC_SDBGKEY3 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. See the TAPC_SDBGKEY0 register description for more information.

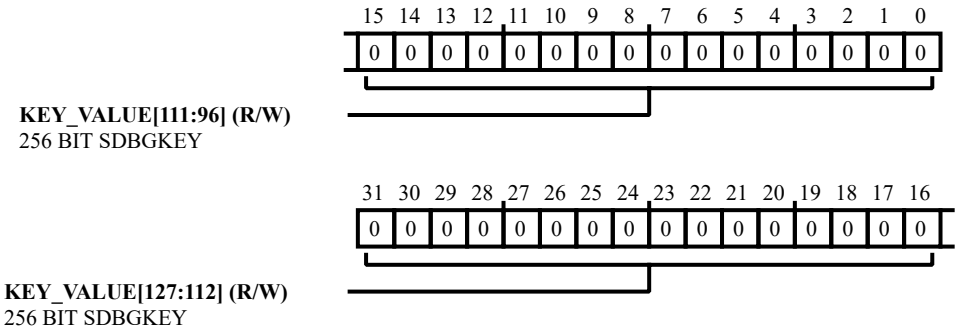


Figure 48-467: TAPC_SDBGKEY3 Register Diagram

Table 48-468: TAPC_SDBGKEY3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The TAPC_SDBGKEY3.KEY_VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 4 Register

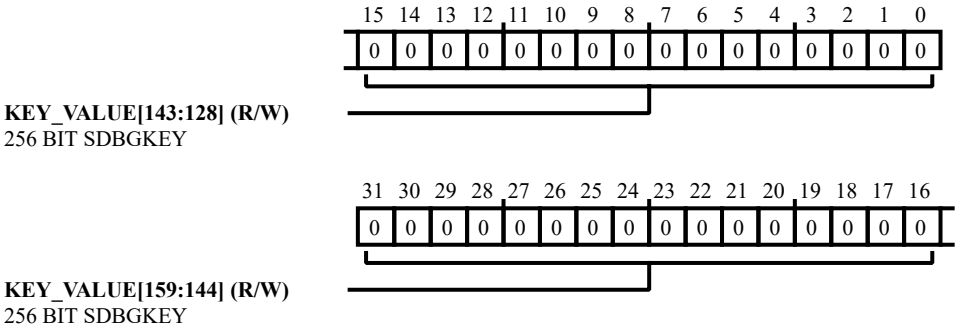


Figure 48-468: TAPC_SDBGKEY4 Register Diagram

Table 48-469: TAPC_SDBGKEY4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The `TAPC_SDBGKEY4.KEY_VALUE` bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 5 Register

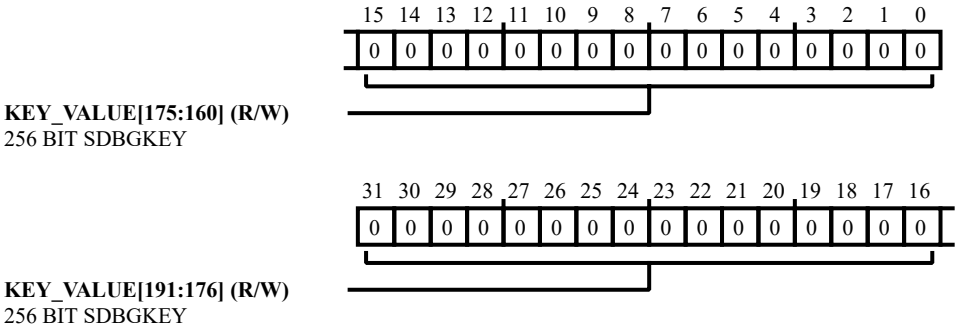


Figure 48-469: TAPC_SDBGKEY5 Register Diagram

Table 48-470: TAPC_SDBGKEY5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The `TAPC_SDBGKEY5.KEY_VALUE` bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 6 Register

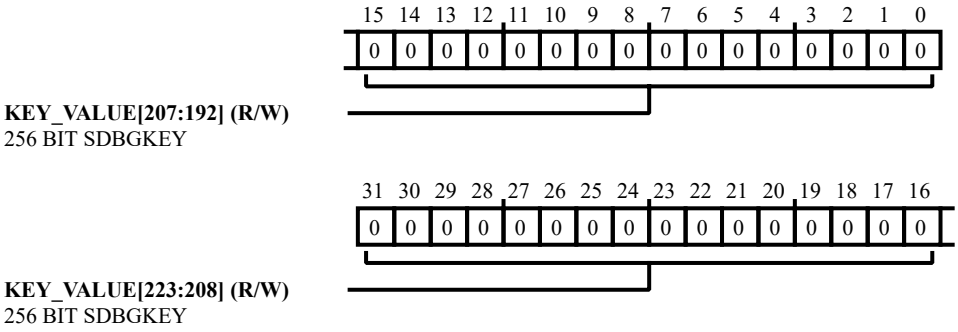


Figure 48-470: TAPC_SDBGKEY6 Register Diagram

Table 48-471: TAPC_SDBGKEY6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The `TAPC_SDBGKEY6.KEY_VALUE` bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 7 Register

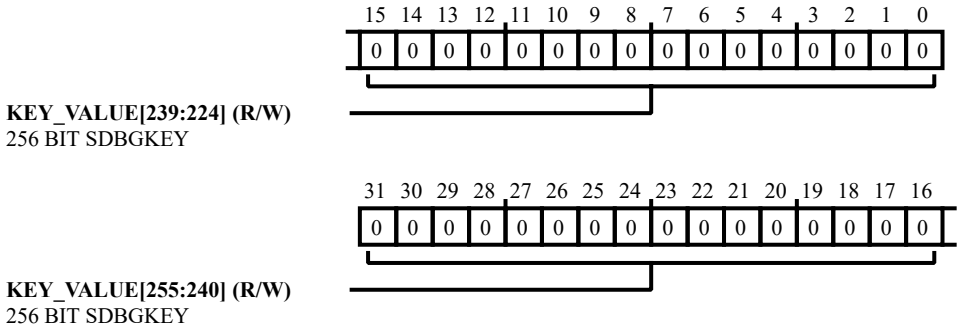


Figure 48-471: TAPC_SDBGKEY7 Register Diagram

Table 48-472: TAPC_SDBGKEY7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	KEY_VALUE	256 BIT SDBGKEY. The `TAPC_SDBGKEY7.KEY_VALUE` bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key Control Register

The TAPC_SDBGKEY_CTL register contains a bit to configure configure validity for the secure debug keys.

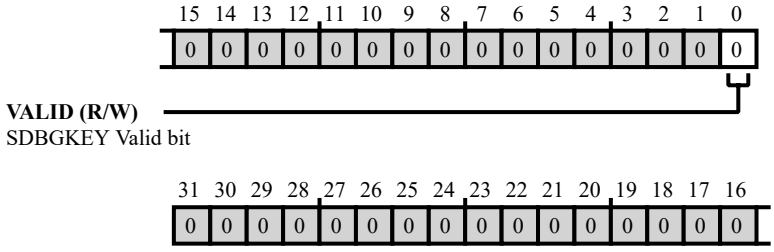


Figure 48-472: TAPC_SDBGKEY_CTL Register Diagram

Table 48-473: TAPC_SDBGKEY_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	VALID	SDBGKEY Valid bit.

Secure Debug Key Status Register

The TAPC_SDBGKEY_STAT register contains the PASS/FAIL status of the secure debug key match.

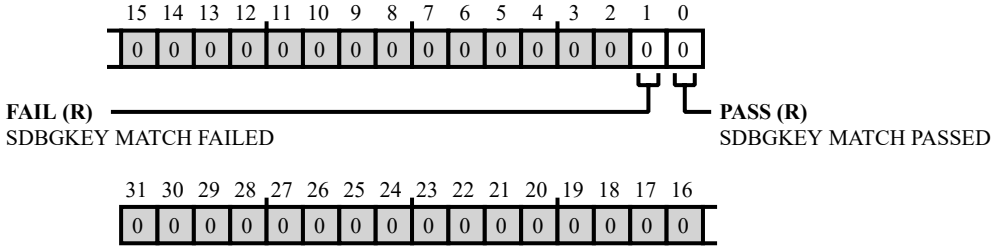


Figure 48-473: TAPC_SDBGKEY_STAT Register Diagram

Table 48-474: TAPC_SDBGKEY_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	FAIL	SDBGKEY MATCH FAILED.
0 (R/NW)	PASS	SDBGKEY MATCH PASSED.

USERCODE Register

The TAPC_USERCODE register stores the 32-bit USERCODE value for the SoC debug unit.

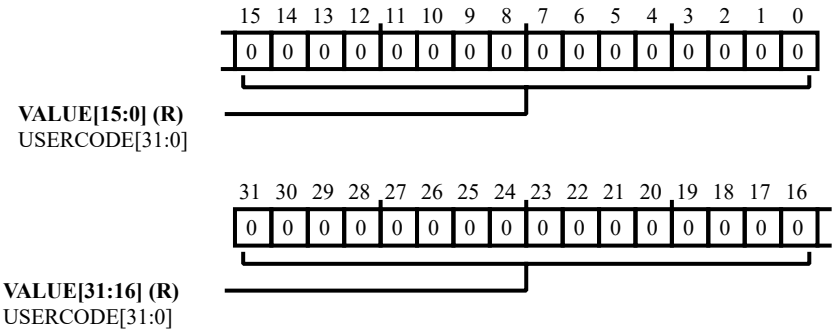


Figure 48-474: TAPC_USERCODE Register Diagram

Table 48-475: TAPC_USERCODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	USERCODE[31:0].

ADSP-2184x TRACE Register Descriptions

Your module description, here. (TRACE) contains the following registers.

Table 48-476: ADSP-2184x TRACE Register List

Name	Description
TRACE_AUTHSTATUS	Authentication Status Register.
TRACE_CIDR0	Component Identification Register 0.
TRACE_CIDR1	Component Identification Register 1.
TRACE_CIDR2	Component Identification Register 2.
TRACE_CIDR3	Component Identification Register 3.
TRACE_CLAIMCLR	Claim Tag Clear Register.
TRACE_CLAIMSET	Claim Tag Set Register.
TRACE_CSPSR	Current Port Size Register.
TRACE_CTPMR	Current Test Patterns/Modes Register.
TRACE_DEVARCH	Device Architecture Register.
TRACE_DEVID	Device Configuration Register.

Table 48-476: ADSP-2184x TRACE Register List (Continued)

Name	Description
TRACE_DEVTYPE	Device Type Identifier Register.
TRACE_EXTCTLIN	External Control Port In Register.
TRACE_EXTCTLOUT	External Control Port Out Register.
TRACE_FFCR	Formatter and Flush Control Register.
TRACE_FFSR	Formatter and Flush Status Register.
TRACE_FSCR	Formatter Synchronization Count Register.
TRACE_ITATBCTR0	Integration Test ATB Control Register 0.
TRACE_ITATBCTR1	Integration Test ATB Control Register 1.
TRACE_ITATBCTR2	Integration Test ATB Control Register 2.
TRACE_ITATBDATA0	Integration Test ATB Data Register 0.
TRACE_ITCTRL	Integration Mode Control Register.
TRACE_ITOUTCTR	Integration Test Output Control Register.
TRACE_ITTRFLIN	Integration Test Trigger In and Flush In Register.
TRACE_PIDR0	Peripheral Identification Register 0.
TRACE_PIDR1	Peripheral Identification Register 1.
TRACE_PIDR2	Peripheral Identification Register 2.
TRACE_PIDR3	Peripheral Identification Register 3.
TRACE_PIDR4	Peripheral Identification Register 4.
TRACE_PIDR5	Peripheral Identification Register 5.
TRACE_PIDR6	Peripheral Identification Register 6.
TRACE_PIDR7	Peripheral Identification Register 7.
TRACE_SSPSR	Supported Port Size Register.
TRACE_STMR	Supported Trigger Modes Register.
TRACE_STPMR	Supported Test Patterns/Modes Register.
TRACE_TCMR	Trigger Counter Multiplier Register.
TRACE_TCVR	Trigger Counter Value Register.
TRACE_TPRCR	Test Pattern Repeat Counter Register.

Authentication Status Register.

Reports the current status of the authentication control signals.

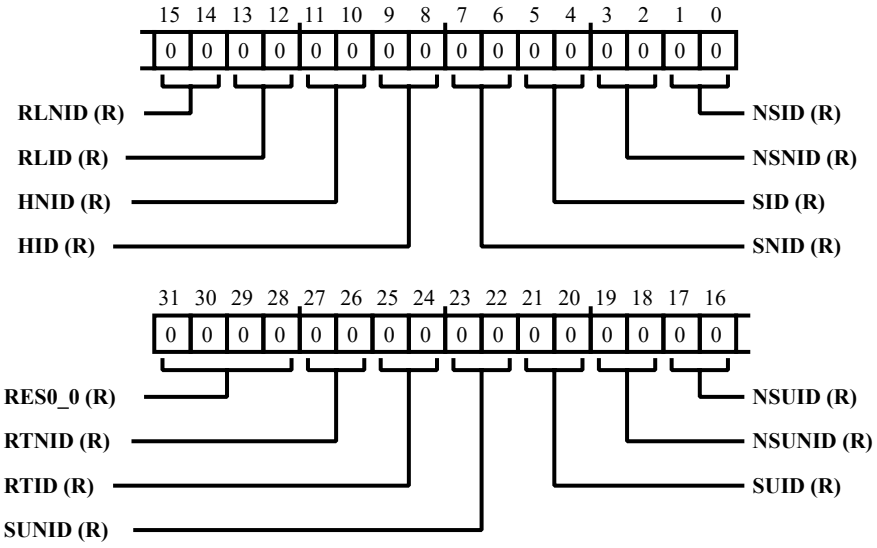


Figure 48-475: TRACE_AUTHSTATUS Register Diagram

Table 48-477: TRACE_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
27:26 (R/NW)	RTNID	. Root non-invasive debug.	
		0	Separate Root non-invasive debug enable not implemented or Root state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.

Table 48-477: TRACE_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	RTID	. Root invasive debug.	
		0	Separate Root invasive debug enable not implemented or Root state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
23:22 (R/NW)	SUNID	. Secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
21:20 (R/NW)	SUID	. Secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
19:18 (R/NW)	NSUNID	. Non-secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
17:16 (R/NW)	NSUID	. Non-secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-477: TRACE_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/NW)	RLNID	. Realm non-invasive debug.	
		0	Separate Realm non-invasive debug enable not implemented or Realm state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
13:12 (R/NW)	RLID	. Realm invasive debug.	
		0	Separate Realm invasive debug enable not implemented or Realm state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
11:10 (R/NW)	HNID	. Hypervisor non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
9:8 (R/NW)	HID	. Hypervisor invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-477: TRACE_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	. Secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
5:4 (R/NW)	SID	. Secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
3:2 (R/NW)	NSNID	. Non-secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
1:0 (R/NW)	NSID	. Non-secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

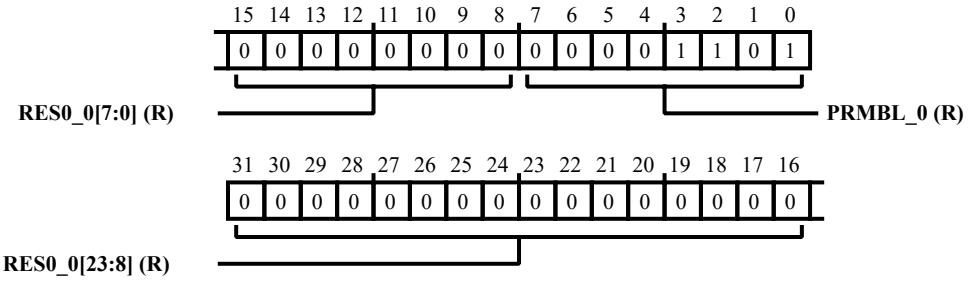


Figure 48-476: TRACE_CIDR0 Register Diagram

Table 48-478: TRACE_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

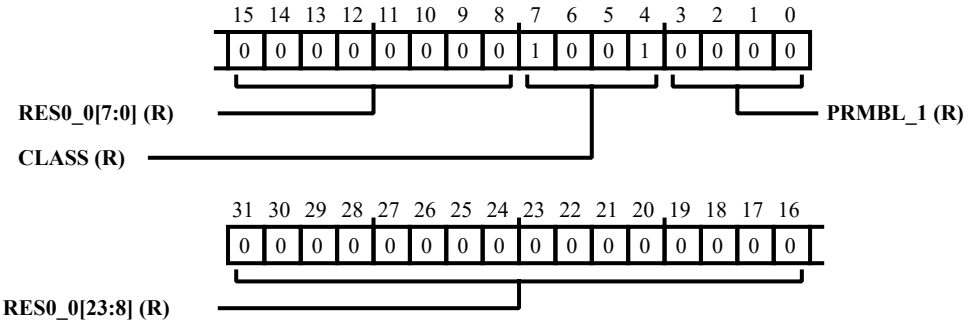


Figure 48-477: TRACE_CIDR1 Register Diagram

Table 48-479: TRACE_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

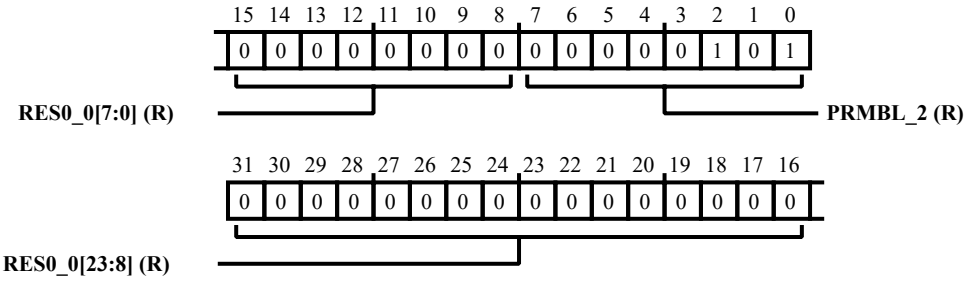


Figure 48-478: TRACE_CIDR2 Register Diagram

Table 48-480: TRACE_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

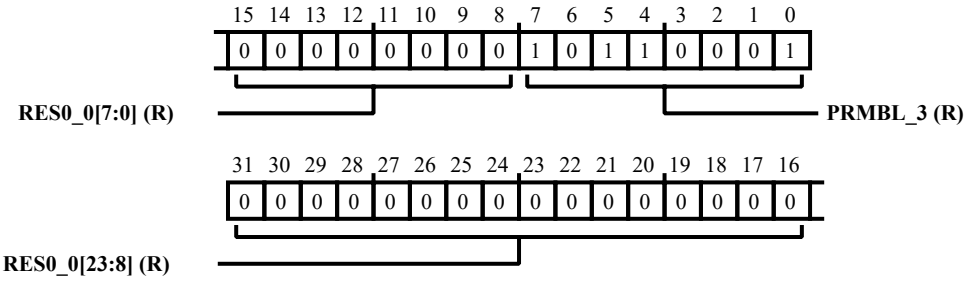


Figure 48-479: TRACE_CIDR3 Register Diagram

Table 48-481: TRACE_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Claim Tag Clear Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

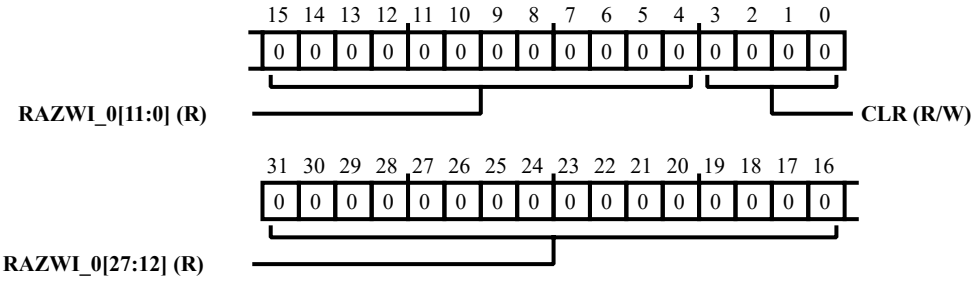


Figure 48-480: TRACE_CLAIMCLR Register Diagram

Table 48-482: TRACE_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	CLR	. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

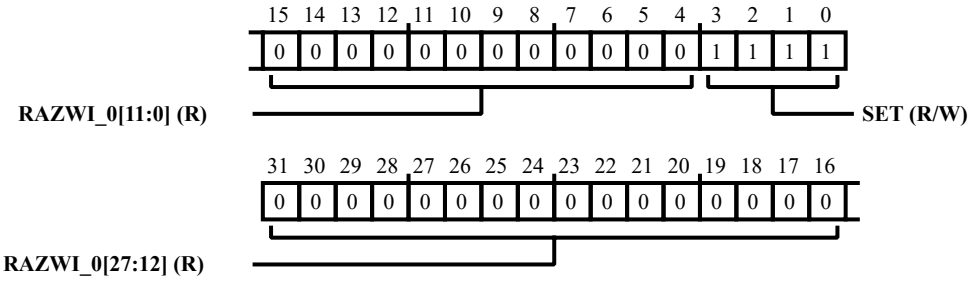


Figure 48-481: TRACE_CLAIMSET Register Diagram

Table 48-483: TRACE_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	SET	. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

Current Port Size Register.

The CSPSR register shows the currently selected size of the tracedata port. It has the same format as the Supported Port Size Register but only one bit is set to show the currently selected port size. If a bit that is indicated as not supported in the SSPSR is set in the CSPSR, it can corrupt the output trace stream, in trace capture mode, and the trace patterns in pattern generation mode. If more than one bit is set, this register indicates the programmed size. However, the port size is internally resolved to the highest order set bit. This register must not be modified while the trace port is still active, or without correctly stopping the formatter. If this happens, it can result in data not being aligned to the port width, for example, data on an 8-bit trace port might not be byte aligned. For the register access to complete on APB clocking on traceclkin is needed.

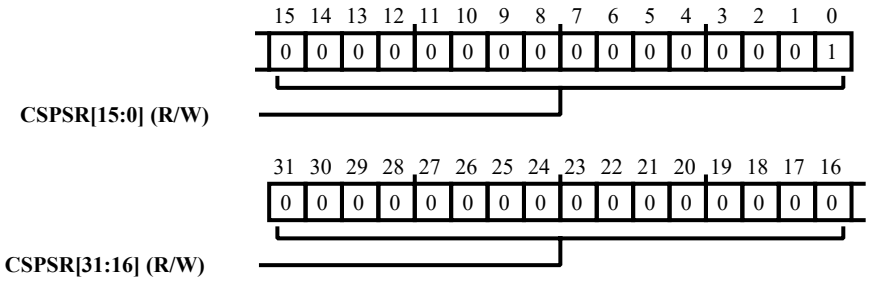


Figure 48-482: TRACE_CSPSR Register Diagram

Table 48-484: TRACE_CSPSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CSPSR	. Currently selected size of the tracedata port.

Current Test Patterns/Modes Register.

The CTPMR register indicates the current test pattern or mode selected. Only one of the two mode bits, bits[17:16], can be set at any one time, but a multiple number of bits for the patterns can be set using bits [3:0]. When timed mode is selected, after the allotted number of cycles is reached, the mode automatically switches to off mode. The pattern with higher bit index is output first when multiple patterns are selected. When no pattern is selected then a default pattern (00/00) is used instead. In continuous mode, the pattern generator continues to send patterns until CTPMR.PCONTEN bit is cleared by software. If multiple patterns are enabled, after sending out all enabled patterns, the pattern generator switches back to the first pattern type and continues to do so until stopped by software. When no pattern is selected then the default pattern (00/00) is used instead. Writing to this register when timed or continuous pattern mode is already enabled causes the current pattern generation to be abandoned and to be restarted with the new pattern mode and new pattern set. Writing to TPRCR or CSPSR when timed or continuous pattern mode is already enabled causes the current pattern generation to be abandoned and to be restarted. The reset value of this register is 0x00000000 which indicates off mode with no selected patterns. For the register access to complete on APB clocking on traceclkin is needed.

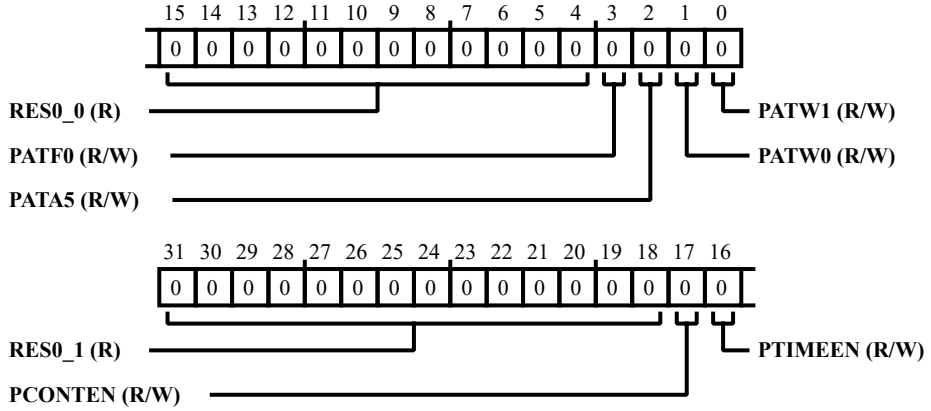


Figure 48-483: TRACE_CTPMR Register Diagram

Table 48-485: TRACE_CTPMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
17 (R/W)	PCONTEN	. Continuous Pattern Mode. Indicates whether continuous pattern mode is enabled:.	
		0	Mode disabled.
		1	Mode enabled.

Table 48-485: TRACE_CTPMR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	PTIMEEN	. Timed Pattern Mode. Indicates whether timed pattern mode is enabled:.	
		0	Mode disabled.
		1	Mode enabled.
15:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
3 (R/W)	PATF0	. FF/00 Pattern. Indicates whether the FF/00 pattern is enabled as output over the Trace Port. All pins toggle simultaneously as 1-0-1-0.	
		0	Pattern disabled.
		1	Pattern enabled.
2 (R/W)	PATA5	. 55/AA Pattern. Indicates whether the 55/AApattern is enabled as output over the Trace Port. The odd numbered pins toggle as 0-1-0-1, while the even numbered pins toggle as 1-0-1-0, simultaneously.	
		0	Pattern disabled.
		1	Pattern enabled.
1 (R/W)	PATW0	. Walking 0 Pattern. Indicates whether the walking 0s pattern is enabled as output over the Trace port. To start with, all pins are set to 1, except tracedata[0] which is driven LOW. In each subsequent cycle, the 0 bit shifts to its left by 1 position and eventually rotates around from its starting position, based on the CSPSR value, to tracedata[0], provided the pattern mode remains enabled for a sufficient number of cycles. When timed mode is selected, after the allotted number of cycles is reached (See TPRCR, 0x208), the wsmode automatically switches to off mode. The pattern with higher bit index is output first when multiple patterns are selected.	
		0	Pattern disabled.
		1	Pattern enabled.
0 (R/W)	PATW1	. Walking 1s Pattern. Indicates whether the walking 1s pattern is enabled as output over the Trace Port. It is similar to the walking 0s pattern except that tracedata[0] is set to 1 to start with and all other bits are 0. It is this set bit that rotates through all the selected pins of the tracedata port.	
		0	Pattern disabled.
		1	Pattern enabled.

Device Architecture Register.

Identifies the architect and architecture of a CoreSight component. The architect might differ from the designer of a component, for example Arm defines the architecture but another company designs and implements the component.

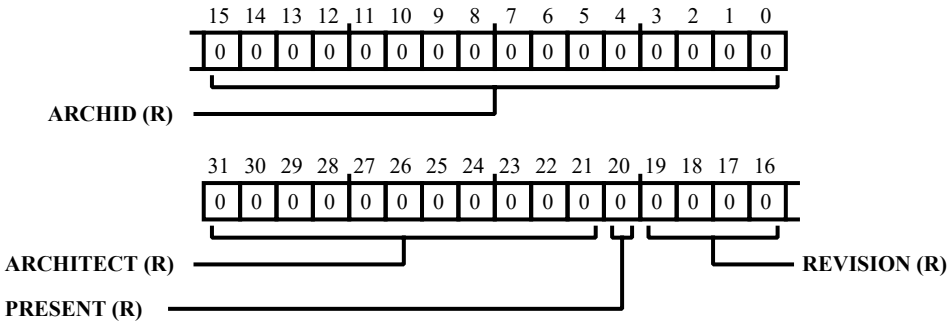


Figure 48-484: TRACE_DEVARCH Register Diagram

Table 48-486: TRACE_DEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:21 (R/NW)	ARCHITECT	. Defines the architect of the component.	
		571	Arm.
20 (R/NW)	PRESENT	. Indicates the presence of this register.	
		0	DEVARCH is not present.
		1	DEVARCH is present.
19:16 (R/NW)	REVISION	. Architecture revision. Returns the revision of the architecture that the ARCHID field specifies.	
15:0 (R/NW)	ARCHID	. Architecture ID. Returns a value that identifies the architecture of the component.	
		2583	Memory Access Port v2 architecture.
		2599	JTAG Access Port v2 architecture.
		2631	Unknown Access Port v2 architecture.
		2807	CoreSight ROM architecture.
		6676	CTI architecture.

Device Configuration Register.

This register is IMPLEMENTATION DEFINED for each Part Number and Designer. The register indicates the capabilities of the component.

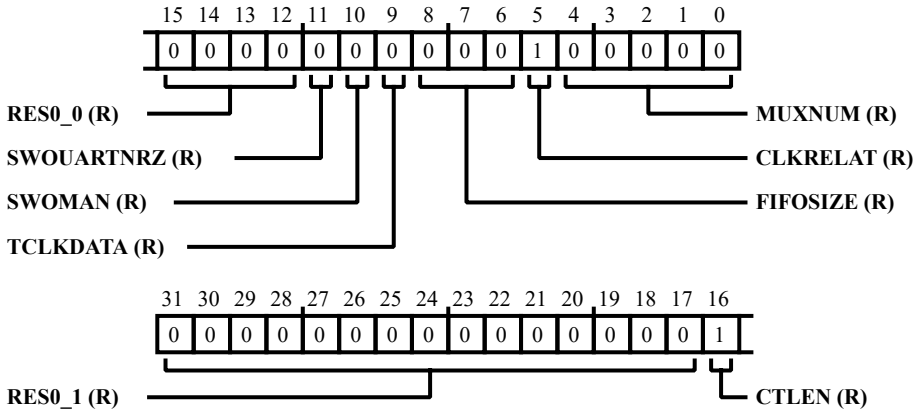


Figure 48-485: TRACE_DEVID Register Diagram

Table 48-487: TRACE_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
16 (R/NW)	CTLEN	. Trace Capture Enable support. Reads 0x1, which indicates that the CTL register is implemented and the software can enable and disable trace capture by programming the CTL register.
15:12 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
11 (R/NW)	SWOUARTNRZ	. Serial Wire Output, UART or NRZ support. Reads 0x0, which indicates that Serial Wire Output, UART or NRZ, is not supported.
10 (R/NW)	SWOMAN	. Serial Wire Output, Manchester-encoded format support. Reads 0x0, which indicates that Serial Wire Output, Manchester-encoded format, is not supported.
9 (R/NW)	TCLKDATA	. Trace Clock Plus Data support. Reads 0x0, which indicates that trace clock and data is supported.

Table 48-487: TRACE_DEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8:6 (R/NW)	FIFOSIZE	. FIFO size in powers of 2. Reads 0x0, indicating that the FIFO size is implementation-defined and is not visible in the programmers model.
5 (R/NW)	CLKRELAT	. Relationship between clk and traceclkin. Reads 0x1 which indicates that these two clocks are asynchronous.
4:0 (R/NW)	MUXNUM	. Indicates a hidden level of input multiplexing. When non-zero, this value indicates the type of multiplexing on the input to the ATB. Currently only 0x00 is supported, that is, no multiplexing is present. This value helps detect the ATB structure.

Device Type Identifier Register.

A debugger can use this register to get information about a component that has an unrecognized Part number.

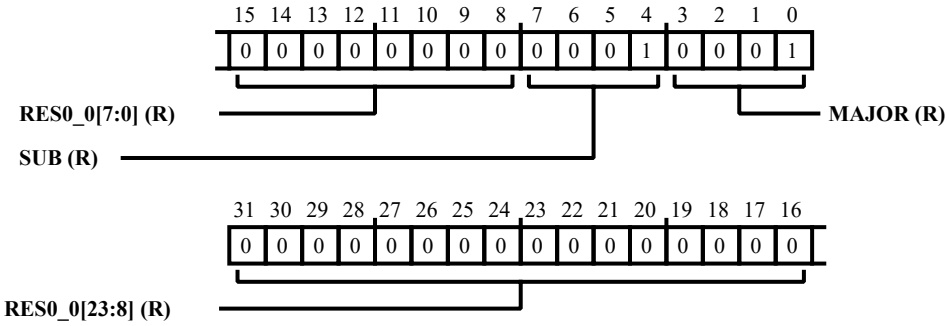


Figure 48-486: TRACE_DEVTYPE Register Diagram

Table 48-488: TRACE_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SUB	. Minor classification. Returns 0x1, indicating this component is a Trace Port.
3:0 (R/NW)	MAJOR	. Major classification. Returns 0x1, indicating this component is a Trace Sink.

External Control Port In Register.

Indicates the current status of external control input port extctl_in[7:0]. It can be used as a feedback mechanism for any serializers, pin sharing multiplexers, or other solutions that might be added to the trace output pins either for pin control or a high-speed trace port solution.

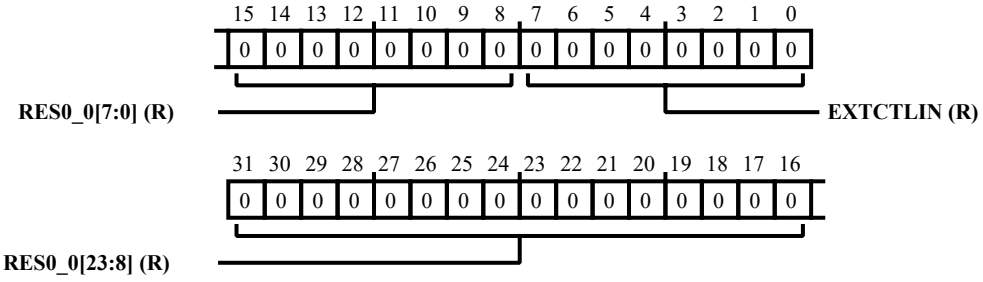


Figure 48-487: TRACE_EXTCTLIN Register Diagram

Table 48-489: TRACE_EXTCTLIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	EXTCTLIN	. This 8-bit field shows the current status of external control input port extctl_in[7:0]. The reset value depends on the external source driving this port.

External Control Port Out Register.

Value to be driven on external control output port extctl_out[7:0]. It can be used as a control mechanism for any serializers, pin sharing multiplexers, or other solutions that might be added to the trace output pins either for pin control or a high-speed trace port solution.

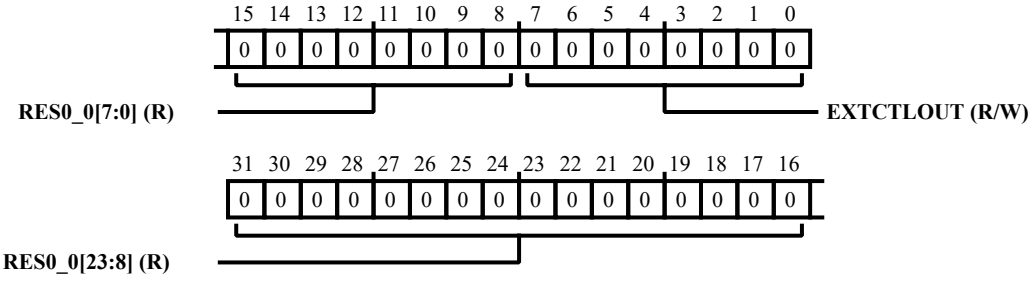


Figure 48-488: TRACE_EXTCTLOUT Register Diagram

Table 48-490: TRACE_EXTCTLOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/W)	EXTCTLOUT	. This 8-bit field holds the value to be driven on the external control output port extctl_out[7:0].

Formatter and Flush Control Register.

The FFCR controls the generation of stop, trigger and flush events. The insertion of a flush completion packet and the insertion of a trigger packet in the formatted trace is enabled here. In bypass mode formatting is disabled and triggers are indicated on tracectl pin, bits[1:0] must be 0b00. In normal mode formatting is enabled and triggers are indicated on tracectl pin, bits[1:0] must be 0b01. In continuous mode formatting is enabled and triggers are embedded in the trace stream with Triger Byte ID 0x7D with a single byte of data payload = 0x00, bits[1:0] must be 0b10. Setting both bits is the same as setting bit[1]. All three flush-generating conditions can be enabled together. However, if a second or third flush event is generated from another condition then the current flush completes before the next flush is serviced. Flush from flushin takes priority over flush from trigger, which in turn completes before a manuallyactivated flush. All trigger indication conditions can be enabled simultaneously although this can cause the appearance of multiple triggers if flush using trigger is also enabled. Both Stop On settings can be enabled although if Flush on Trigger is set up, none of the flushed data is stored. ARM recommends that you change the trace port width without enabling continuous mode. Enabling continuous mode causes data to be sent from the trace port and modifying the port size can result in data not being aligned.

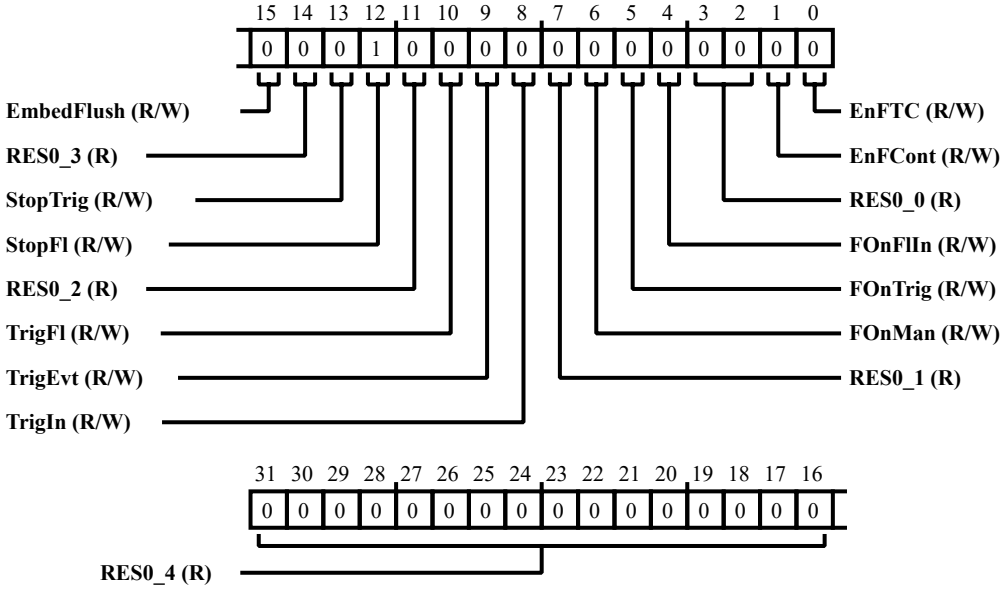


Figure 48-489: TRACE_FFCR Register Diagram

Table 48-491: TRACE_FFCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	RES0_4	. Reserved bit or field with SBZP behavior.

Table 48-491: TRACE_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EMBEDFLUSH	. Embed flush completion packet, Flush ID. Enables insertion of Flush ID 0x7B with a single byte of data payload = 0x00 in the output trace, after the last flush data byte, when a flush completes on the ATB receiver interface. This bit is effective only in Normal and Continuous formatter modes. In Bypass mode, the Flush ID insertion remains disabled and this bit is ignored.	
		0	Disable Flush ID insertion.
		1	Enable Flush ID insertion.
14 (R/NW)	RES0_3	. Reserved bit or field with SBZP behavior.	
13 (R/W)	STOPTRIG	. Stop on Trigger Event. Stops the formatter after a trigger event is observed.	
		0	Disable stopping the formatter after a trigger event is observed.
		1	Enable stopping the formatter after a trigger event is observed.
12 (R/W)	STOPFL	. Stop on Flush Completion. Forces the FIFO to drain off any partially completed packets after a flush completion and stops the formatter.	
		0	Disable stopping the formatter when afready_rx is received.
		1	Enable stopping the formatter when afready_rx is received.
11 (R/NW)	RES0_2	. Reserved bit or field with SBZP behavior.	
10 (R/W)	TRIGFL	. Trigger on Flush Completion.	
		0	Disable trigger indication on flush completion, that is, when afready_rx is high.
		1	Enable trigger indication on flush completion.
9 (R/W)	TRIGEVT	. Trigger on Trigger Event. Indicates a trigger when the trigger counter reaches 0 while decrementing. If FFCR.StopTrig is set, this bit is ignored.	
		0	Disable trigger indication on trigger event.
		1	Enable trigger indication on trigger event.

Table 48-491: TRACE_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	TRIGIN	. Trigger on trigin.	
		0	Disable trigger indication when trigin is asserted.
		1	Enable trigger indication when trigin is asserted.
7 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
6 (R/W)	FONMAN	. Flush Manual. Writing 1 to this bit generates a flush request on afvalid_rx pin, writing 0 has no effect. It is automatically cleared when the generated flush request completes and afready_rx is received by the TPIU. Reading this bit returns its current value.	
5 (R/W)	FONTRIG	. Flush on Trigger Event. Initiates a flush request when a trigger event occurs. A trigger event occurs when the trigger counter reaches 0 while decrementing, or, if the TCVR is 0x0 and trigin goes HIGH.	
		0	Disable generation of flush when a trigger event occurs.
		1	Enable generation of flush when a trigger event occurs.
4 (R/W)	FONFLIN	. Flush on flushin.	
		0	Disable generation of flush using flushin input.
		1	Enable generation of flush using flushin input.
3:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1 (R/W)	ENFCONT	. Enable Continuous Formatting Mode and Enable Formatter. The trigger packets are embedded in the trace stream. This bit can only be changed when FFSR.FtStopped is HIGH.	
0 (R/W)	ENFTC	. Enable Formatter. The trigger packets are not embedded in the trace stream and the trace disable cycles and triggers are indicated by tracectl pin where present. This bit can only be changed when FFSR.FtStopped is HIGH.	

Formatter and Flush Status Register.

The FFSR indicates the current status of formatter and flush features available in the TPIU.

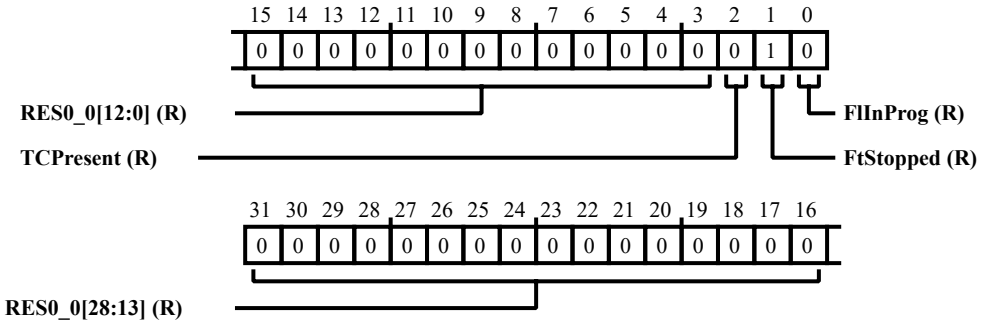


Figure 48-490: TRACE_FFSR Register Diagram

Table 48-492: TRACE_FFSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
2 (R/NW)	TCPRESENT	. Indicates whether the tracectl pin is available for use, based on the tie-off value of tpctl_valid.	
		0	tracectl pin not present. The data formatter must be used and in continuous mode.
		1	tracectl pin present.
1 (R/NW)	FTSTOPPED	. The formatter has received a stop request and all trace data and post-amble is sent. Any further trace on the ATB interface is dropped and atready_rx is asserted.	
		0	Formatter running.
		1	Formatter stopped.
0 (R/NW)	FLINPROG	. Indicates whether a flush is in progress. It is set when the TPIU sends a flush request on its ATB receiver interface. The bit remains set until the ATB flush is complete and the last byte of flush data, including the flush ID payload if FFCR.EmbedFlush is set, has been output on the trace port.	
		0	No ongoing flush.
		1	Flush in progress.

Formatter Synchronization Count Register.

The FSCR register indicates the maximum number of formatter frames sent to Trace Port after which a synchronization packet must be inserted. The register value indicates the programmed counter value and not the current state of the counter. The TPIU uses a frame sync counter that contains the number of formatter frames since the last frame synchronization packet. The counter is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes (4096 packets x 16 bytes per packet). On reset, the FSCR is set up for a synchronization packet every 1024 bytes, that is every 64 formatter frames. If the formatter is configured in continuous mode, full and half-word sync frames are inserted during normal operation. In this case, the counter value is the maximum number of complete frames between full synchronization packets. For the register access to complete on APB clocking on traceclkin is needed.

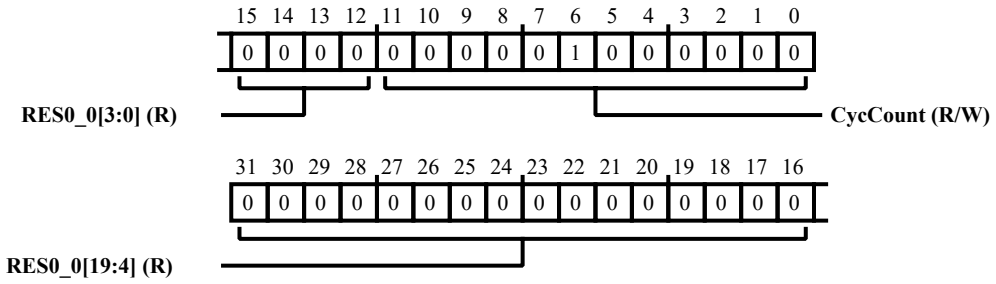


Figure 48-491: TRACE_FSCR Register Diagram

Table 48-493: TRACE_FSCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
11:0 (R/W)	CYCCOUNT	. 12-bit counter value to indicate the number of complete frames between full synchronization packets. It is also used to send periodic synchronization requests to the ATB transmitter using syncreq_rx output. If this field is programmed as 0x0, the synchronization counter is disabled. If this field is programmed with 0x1-0x7 the programmed value is 0x8. The reset value is 0x040, that is, 64 frames = 1024 bytes.

Integration Test ATB Control Register 0.

This register indicates the values of atvalid_rx, afready_rx, and atbytes_rx inputs in integration mode. Reads are allowed even in functional mode, but the register is disabled and does not get updated even if the inputs change. The reset value depends on an external source driving these inputs.

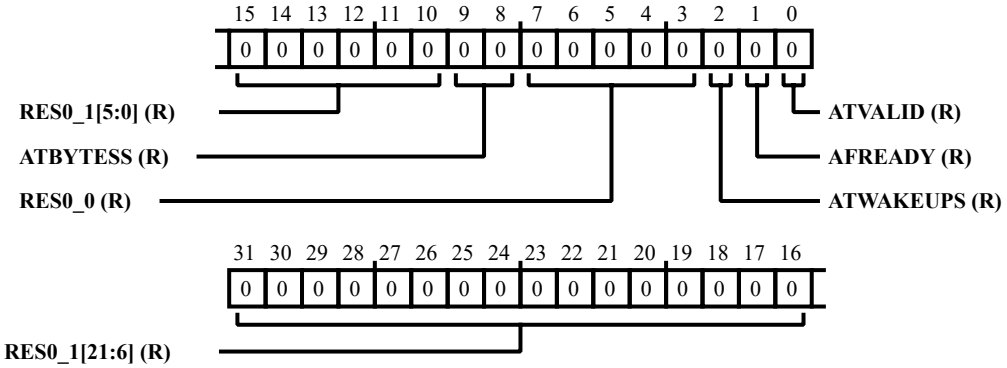


Figure 48-492: TRACE_ITATBCTR0 Register Diagram

Table 48-494: TRACE_ITATBCTR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:10 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
9:8 (R/NW)	ATBYTESS	. Reads the value of atbytes_rx[1:0] in integration mode.
7:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (R/NW)	ATWAKEUPS	. Reads the value of atwakeup_rx in integration mode.
1 (R/NW)	AFREADY	. Reads the value of afready_rx in integration mode.
0 (R/NW)	ATVALID	. Reads the value of atvalid_rx in integration mode.

Integration Test ATB Control Register 1.

This register indicates the value of the atid_rx input in integration mode. Reads are allowed even in functional mode, but the register is disabled and does not get updated even if the inputs change. The reset value depends on external source driving these inputs.

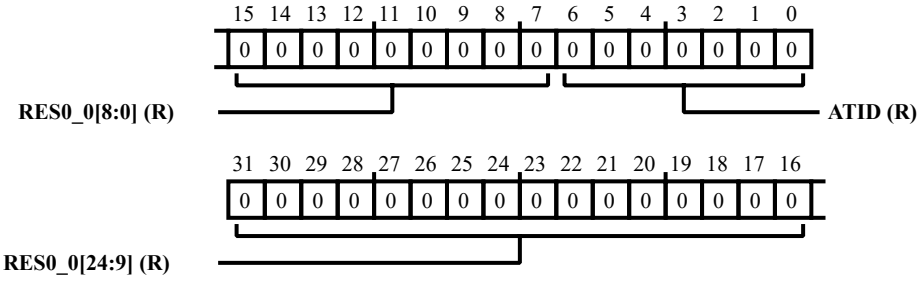


Figure 48-493: TRACE_ITATBCTR1 Register Diagram

Table 48-495: TRACE_ITATBCTR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
6:0 (R/NW)	ATID	. Reads the value of atid_rx[6:0] in integration mode.

Integration Test ATB Control Register 2.

This register enables control of the atready_rx, afvalid_rx, and syncreq_rx outputs in integration mode. Writes to this register are allowed in integration mode as well as functional mode. However, the programmed value is driven to the outputs only in integration mode.

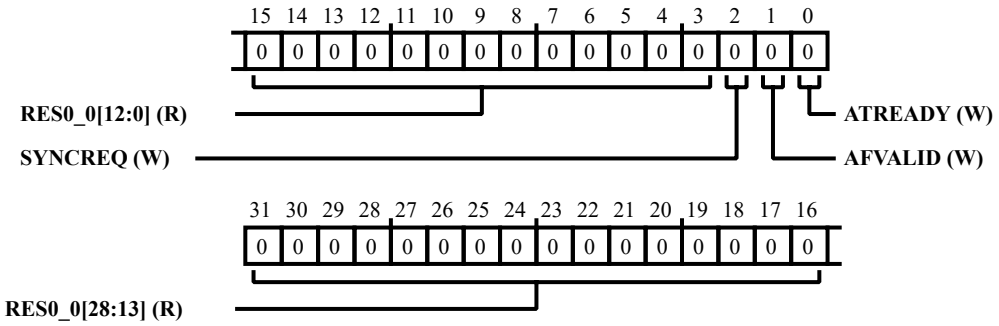


Figure 48-494: TRACE_ITATBCTR2 Register Diagram

Table 48-496: TRACE_ITATBCTR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (RX/W)	SYNCREQ	. Sets the value of syncreq_rx in integration mode.
1 (RX/W)	AFVALID	. Sets the value of afvalid_rx in integration mode.
0 (RX/W)	ATREADY	. Sets the value of atready_rx in integration mode.

Integration Test ATB Data Register 0.

This register indicates the value of the atdata_rx input in integration mode. Only 5 bits are readable through this register, the MSB of each of the four data bytes and the LSB. Reads are allowed even in functional mode, but the register is disabled and does not get updated even if the inputs change. The reset value depends on the external source driving the inputs.

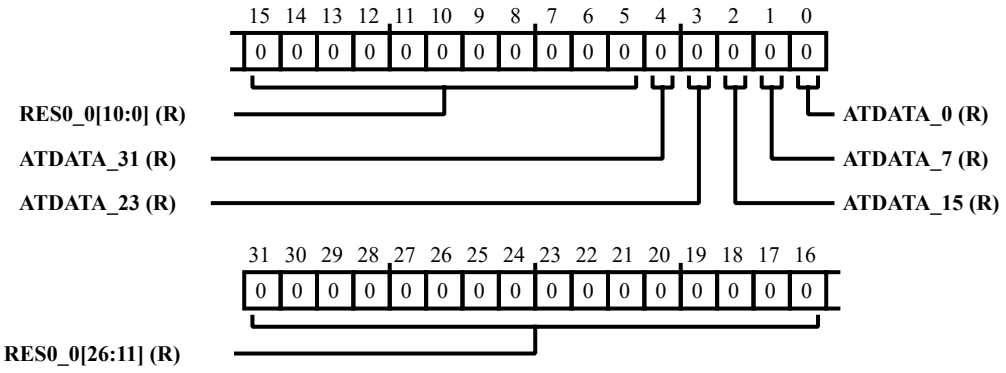


Figure 48-495: TRACE_ITATBDATA0 Register Diagram

Table 48-497: TRACE_ITATBDATA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4 (R/NW)	ATDATA_31	. Reads the value of atdata_rx[31] during integration mode.
3 (R/NW)	ATDATA_23	. Reads the value of atdata_rx[23] during integration mode.
2 (R/NW)	ATDATA_15	. Reads the value of atdata_rx[15] during integration mode.
1 (R/NW)	ATDATA_7	. Reads the value of atdata_rx[7] during integration mode.
0 (R/NW)	ATDATA_0	. Reads the value of atdata_rx[0] during integration mode.

Integration Mode Control Register.

The Integration Mode Control register is used to switch between functional mode and integration mode. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

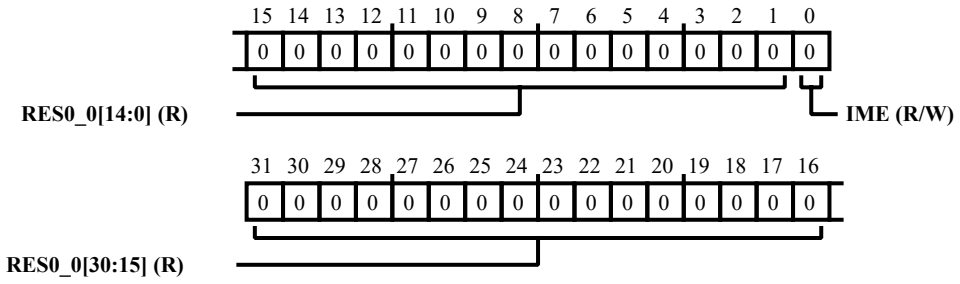


Figure 48-496: TRACE_ITCTRL Register Diagram

Table 48-498: TRACE_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	IME	. Integration Mode Enable.	
		0	The component must enter functional mode.
		1	The component must enter integration mode, and enable support for topology detection and integration testing.

Integration Test Output Control Register.

This register enables control of the flushcomp output in integration mode. Writes to this register are allowed in integration mode, as well as functional mode. However, the programmed value is driven to the output pin only in integration mode.

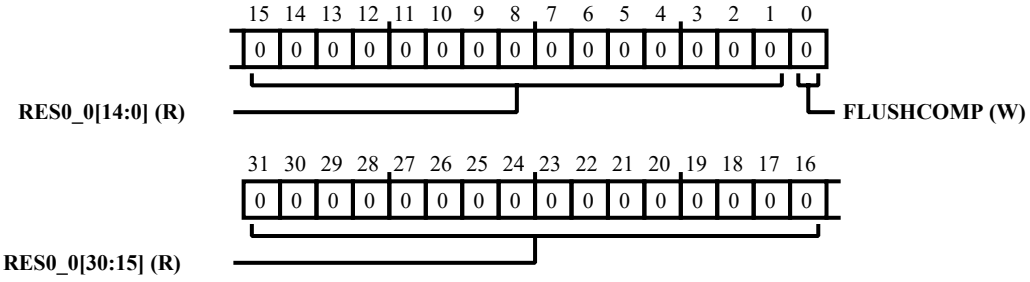


Figure 48-497: TRACE_ITOUTCTR Register Diagram

Table 48-499: TRACE_ITOUTCTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
0 (RX/W)	FLUSHCOMP	. Sets the value of flushcomp in integration mode.

Integration Test Trigger In and Flush In Register.

This register indicates the integration status of the flushin and trigin inputs in integration mode. Reads are allowed even in functional mode, but the register itself is disabled and does not get updated even if the inputs change. The reset value depends on the external source driving the inputs.

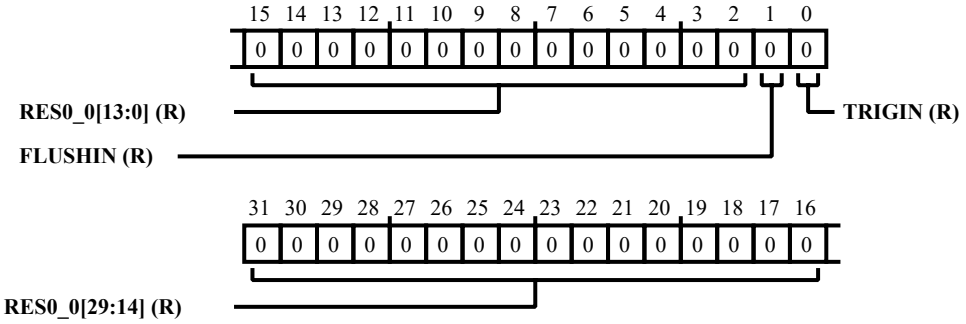


Figure 48-498: TRACE_ITTRFLIN Register Diagram

Table 48-500: TRACE_ITTRFLIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
1 (R/NW)	FLUSHIN	. In integration mode, this bit latches to 1 on a rising edge of the flushin input. It is cleared when this register is read, or when integration mode is disabled.
0 (R/NW)	TRIGIN	. In integration mode, this bit latches to 1 on a rising edge of the trigin input. It is cleared when this register is read or when integration mode is disabled.

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

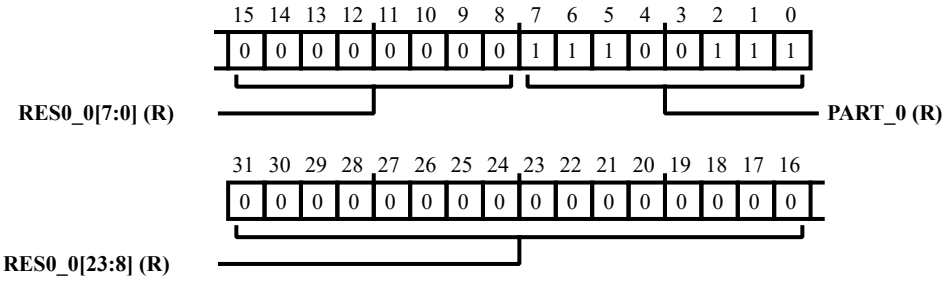


Figure 48-499: TRACE_PIDR0 Register Diagram

Table 48-501: TRACE_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PART_0	. Part number, bits[7:0]. Taken together with PIDR1.PART_1 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

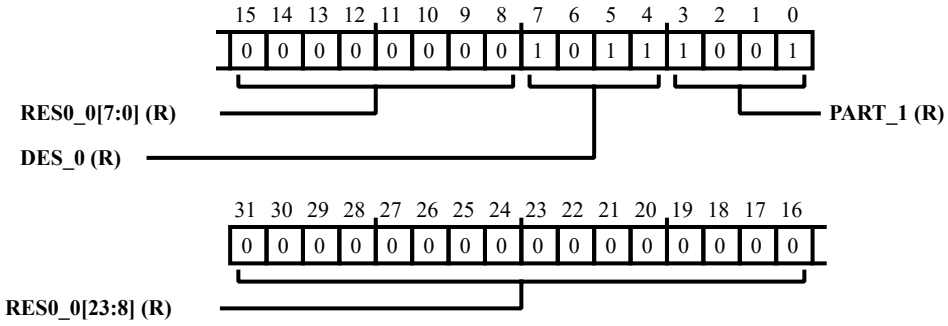


Figure 48-500: TRACE_PIDR1 Register Diagram

Table 48-502: TRACE_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

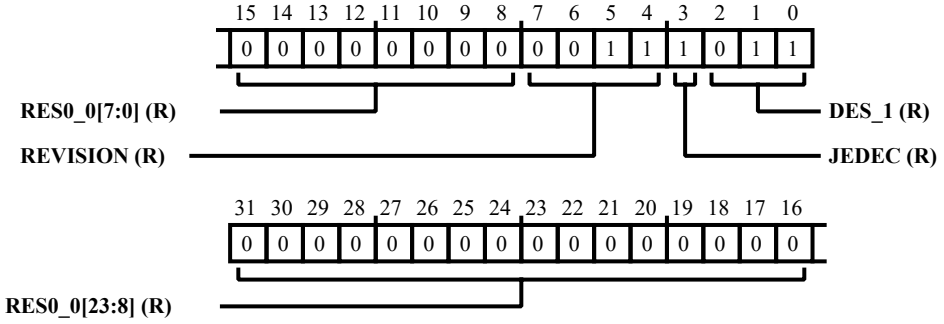


Figure 48-501: TRACE_PIDR2 Register Diagram

Table 48-503: TRACE_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

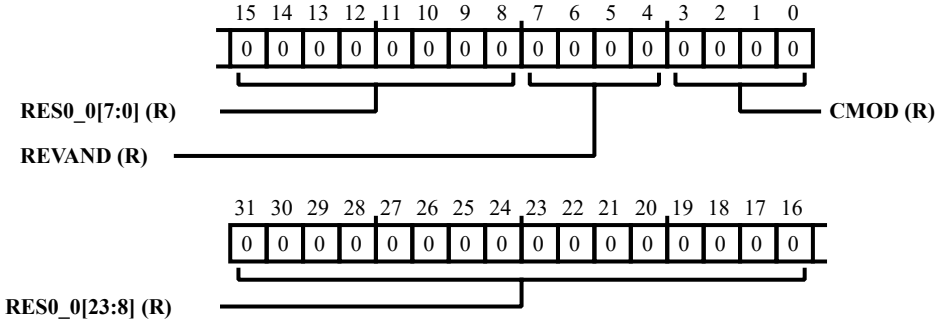


Figure 48-502: TRACE_PIDR3 Register Diagram

Table 48-504: TRACE_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

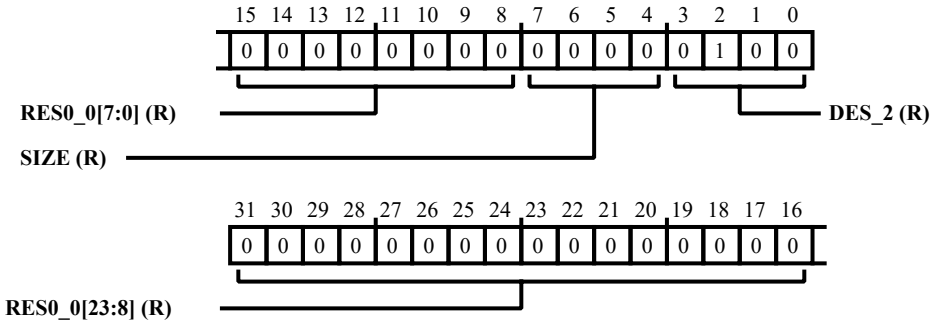


Figure 48-503: TRACE_PIDR4 Register Diagram

Table 48-505: TRACE_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

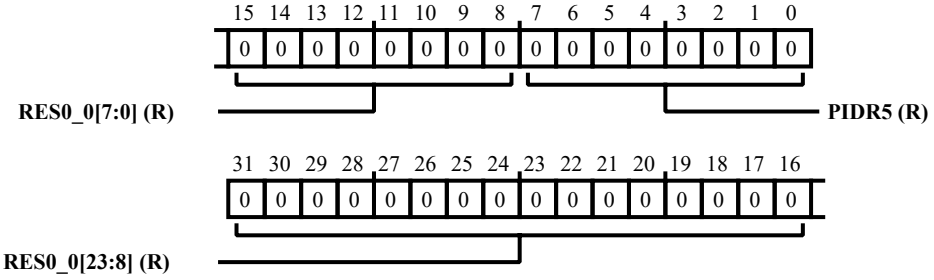


Figure 48-504: TRACE_PIDR5 Register Diagram

Table 48-506: TRACE_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

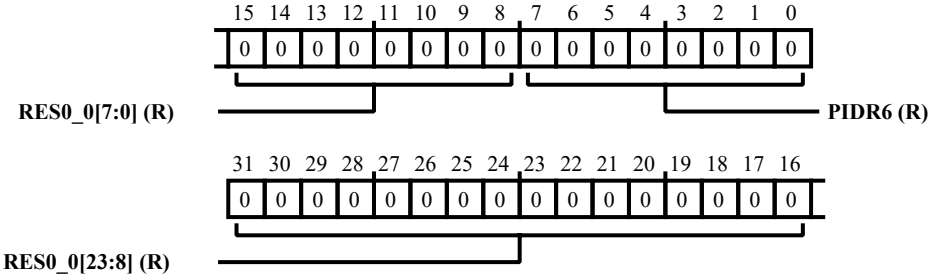


Figure 48-505: TRACE_PIDR6 Register Diagram

Table 48-507: TRACE_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

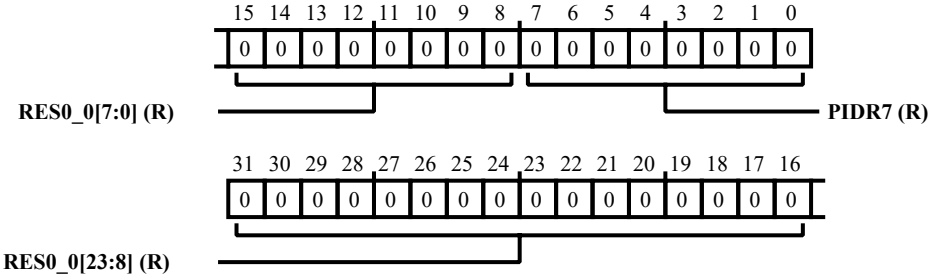


Figure 48-506: TRACE_PIDR7 Register Diagram

Table 48-508: TRACE_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

Supported Port Size Register.

The SSPSR register shows supported width configurations of the tracedata port. Each bit location represents a single port size that is supported, that is sizes 32 bits down to 1 bit, in bit locations [31:0]. If a bit is set, then that port size is supported. By default, the RTL is designed to support all port sizes. Port sizes, other than 1-bit, are configuration-dependent on the tie-off value of tp_maxdatasize. Bit[0] is always 1.

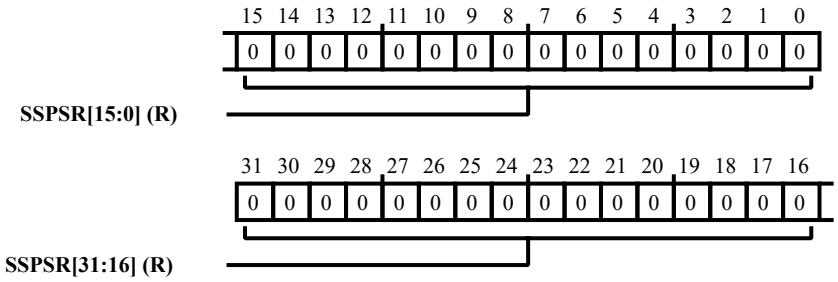


Figure 48-507: TRACE_SSPSR Register Diagram

Table 48-509: TRACE_SSPSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	SSPSR	. Supported tracedata port sizes. Bit[0] is always 1.

Supported Trigger Modes Register.

The STMR register indicates the implemented Trigger Counter multipliers and other supported features of the trigger system.

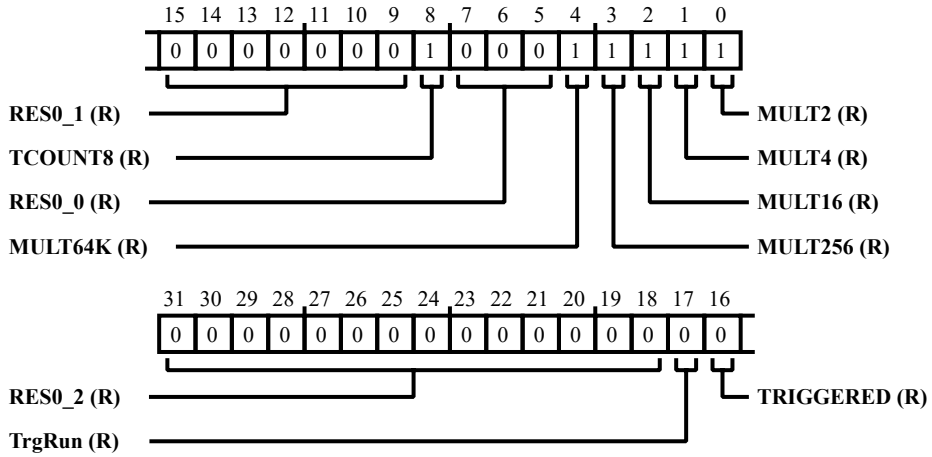


Figure 48-508: TRACE_STMR Register Diagram

Table 48-510: TRACE_STMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/NW)	RES0_2	. Reserved bit or field with SBZP behavior.	
17 (R/NW)	TRGRUN	. A trigger has occurred after a rising edge of trigin input, or a trigger packet (atid_rx = 0x7D) is received in the input trace:.	
		0	Either a trigger has not occurred or the counter is at 0.
		1	A trigger has occurred but the counter is not at 0.
16 (R/NW)	TRIGGERED	. A trigger has occurred after a rising edge of trigin input, or a trigger packet (atid_rx = 0x7D) is received in the input trace and the counter has reached 0. This bit is cleared when the TCVR or TCMR register is written.	
		0	Trigger has not occurred.
		1	Trigger has occurred.
15:9 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
8 (R/NW)	TCOUNT8	. Returns 1 indicating that an 8-bit wide counter register is implemented for trigger insertion.	

Table 48-510: TRACE_STMR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4 (R/NW)	MULT64K	. Returns 1, indicating that multiplying the trigger counter by 65536 is supported.
3 (R/NW)	MULT256	. Returns 1, indicating that multiplying the trigger counter by 256 is supported.
2 (R/NW)	MULT16	. Returns 1, indicating that multiplying the trigger counter by 16 is supported.
1 (R/NW)	MULT4	. Returns 1, indicating that multiplying the trigger counter by 4 is supported.
0 (R/NW)	MULT2	. Returns 1, indicating that multiplying the trigger counter by 2 is supported.

Supported Test Patterns/Modes Register.

The STPMR Register provides a set of known bit sequences or patterns that can be output over the trace port and can be detected by the TPA or TCD.

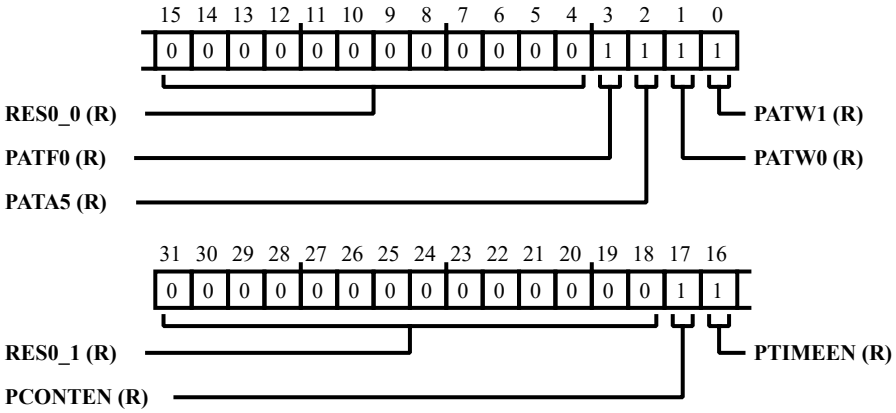


Figure 48-509: TRACE_STPMR Register Diagram

Table 48-511: TRACE_STPMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:18 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
17 (R/NW)	PCONTEN	. Continuous Pattern Mode, returns 1 indicating that continuous pattern mode is supported.
16 (R/NW)	PTIMEEN	. Timed Pattern Mode, returns 1 indicating that timed pattern mode is supported.
15:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
3 (R/NW)	PATF0	. FF/00 Pattern, returns 1 indicating that the FF/00 pattern is supported over the trace port.
2 (R/NW)	PATA5	. 55/AA Pattern, returns 1 indicating that the 55/AA pattern is supported over the trace port.
1 (R/NW)	PATW0	. Walking 0 Pattern, returns 1 indicating that the walking 0s pattern is supported over the trace port.

Table 48-511: TRACE_STPMR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	PATW1	. Walking 1s Pattern, returns 1 indicating that the walking 1s pattern is supported over the trace port.

Trigger Counter Multiplier Register.

The TCMR register contains the selectors for the trigger counter multiplier. Several multipliers can be selected to create the required multiplier value between 1 (TCMR[4:0] = 0x0) and 2^31 (TCMR[4:0] = 0x1F). When more than one bit it set, the effective multiplier value is the product of selected multipliers. Writing to this register causes the trigger counter (the actual counter) to be reloaded and the state in the multipliers to be reset.

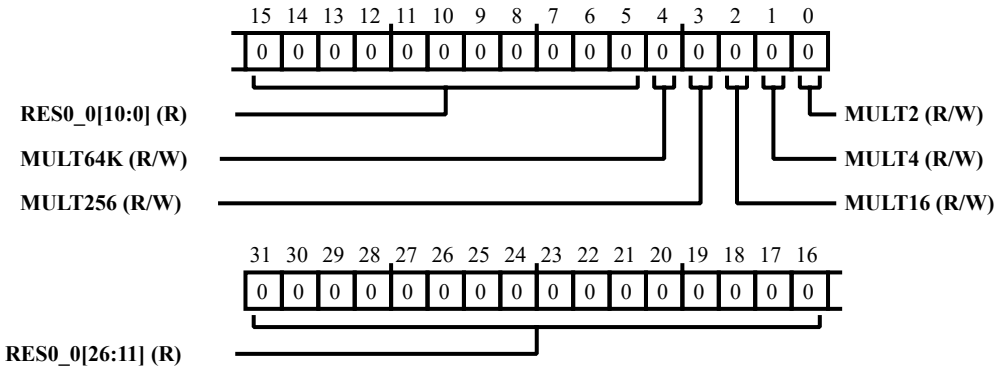


Figure 48-510: TRACE_TCMR Register Diagram

Table 48-512: TRACE_TCMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
4 (R/W)	MULT64K	. Multiply the Trigger Counter by 65536:.	
		0	Multiplier disabled.
		1	Multiplier enabled.
3 (R/W)	MULT256	. Multiply the Trigger Counter by 256:.	
		0	Multiplier disabled.
		1	Multiplier enabled.
2 (R/W)	MULT16	. Multiply the Trigger Counter by 16:.	
		0	Multiplier disabled.
		1	Multiplier enabled.

Table 48-512: TRACE_TCMR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	MULT4	. Multiply the Trigger Counter by 4:.	
		0	Multiplier disabled.
		1	Multiplier enabled.
0 (R/W)	MULT2	. Multiply the Trigger Counter by 2:.	
		0	Multiplier disabled.
		1	Multiplier enabled.

Trigger Counter Value Register.

The TCVR register indicates the programmed trigger counter value. The TPIU implements a trigger counter that enables delaying the indication of triggers to any external connected trace capture devices. The counter is 8 bits wide and is intended only to be used with the counter multipliers within the Trigger Multiplier Register. When a trigger occurs (observed trigin=1 or atid_rx=0x7D), the TCVR.TrigCount value, with the TCMR, determines the number of words to be output from the formatter before the trigger is indicated on the Trace Port. When the trigger counter reaches zero, the value from the TCVR register is reloaded into the trigger counter. Writing to this register causes the trigger counter (the actual counter) to be reloaded. Reading this register returns the programmed count value and not the current count.

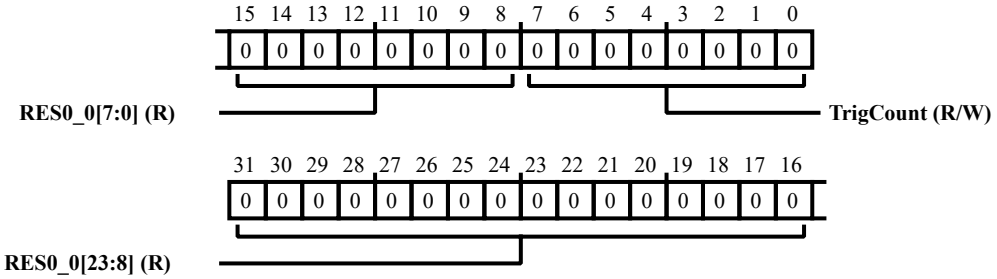


Figure 48-511: TRACE_TCVR Register Diagram

Table 48-513: TRACE_TCVR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/W)	TRIGCOUNT	. Programmed trigger counter value.

Test Pattern Repeat Counter Register.

This register indicates the number of times each test pattern is output on the Trace Port before switching to next pattern. For the register access to complete on APB clocking on traceclkin is needed.

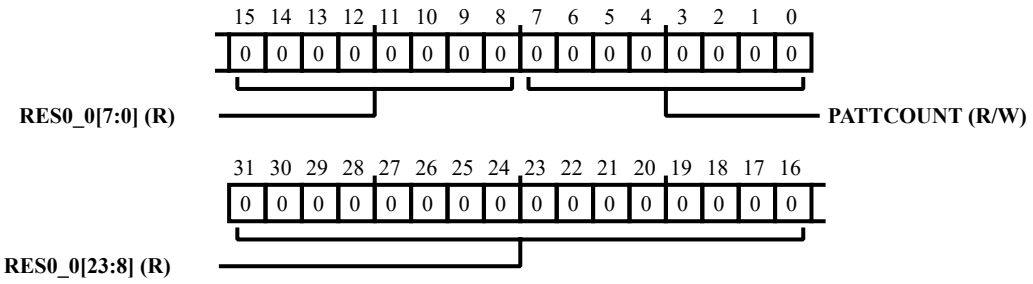


Figure 48-512: TRACE_TPRCR Register Diagram

Table 48-514: TRACE_TPRCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/W)	PATTCOUNT	. An 8-bit counter value that indicates the number of traceclk cycles for which a pattern runs before it switches to the next enabled pattern. A write sets the initial counter value, and a read returns the programmed value. The pattern length is PATTCOUNT+1. The reset value is 0x0.

ADSP-2184x CSETF Register Descriptions

Your module description, here. (CSETF) contains the following registers.

Table 48-515: ADSP-2184x CSETF Register List

Name	Description
CSETF_AUTHSTATUS	Authentication Status Register.
CSETF_BUFWM	Buffer Level Water Mark.
CSETF_CBUFLEVEL	Current Buffer Fill Level.
CSETF_CIDR0	Component Identification Register 0.
CSETF_CIDR1	Component Identification Register 1.
CSETF_CIDR2	Component Identification Register 2.
CSETF_CIDR3	Component Identification Register 3.
CSETF_CLAIMCLR	Claim Tag Clear Register.

Table 48-515: ADSP-2184x CSETF Register List (Continued)

Name	Description
CSETF_CLAIMSET	Claim Tag Set Register.
CSETF_CTL	Control Register.
CSETF_DEVID	Device Configuration Register.
CSETF_DEVID1	Device Configuration Register 1.
CSETF_DEVTYPE	Device Type Identifier Register.
CSETF_FFCR	Formatter and Flush Control Register.
CSETF_FFSR	Formatter and Flush Status Register.
CSETF_ITATBCTR0	Integration Test ATB Control 0 Register.
CSETF_ITATBCTR1	Integration Test ATB Control 1 Register.
CSETF_ITATBCTR2	Integration Test ATB Control 2 Register.
CSETF_ITATBDATA0	Integration Test ATB Data 0 Register.
CSETF_ITATBMCTR0	Integration Test ATB Transmitter Control 0 register.
CSETF_ITATBMCTR1	Integration Test ATB Transmitter Control 1 register.
CSETF_ITATBMCTR2	Integration Test ATB Transmitter Control 2 register.
CSETF_ITATBMDATA0	Integration Test ATB Transmitter Data 0 register.
CSETF_ITCTRL	Integration Mode Control Register.
CSETF_ITEVTINTR	Integration Test Event and Interrupt Control Register.
CSETF_ITTRFLIN	Integration Test Trigger In and Flush In register.
CSETF_LBUFLEVEL	Latched Buffer Fill Level.
CSETF_MODE	Mode register.
CSETF_PIDR0	Peripheral Identification Register 0.
CSETF_PIDR1	Peripheral Identification Register 1.
CSETF_PIDR2	Peripheral Identification Register 2.
CSETF_PIDR3	Peripheral Identification Register 3.
CSETF_PIDR4	Peripheral Identification Register 4.
CSETF_PIDR5	Peripheral Identification Register 5.
CSETF_PIDR6	Peripheral Identification Register 6.
CSETF_PIDR7	Peripheral Identification Register 7.
CSETF_PSCR	Periodic Synchronization Counter Register.
CSETF_RRD	RAM Read Data register.
CSETF_RRP	RAM Read Pointer register.

Table 48-515: ADSP-2184x CSETF Register List (Continued)

Name	Description
CSETF_RSZ	RAM Size register.
CSETF_RWD	RAM Write Data register.
CSETF_RWP	RAM Write Pointer register.
CSETF_STS	Status register.
CSETF_TRG	Trigger Counter register.

Authentication Status Register.

Reports the current status of the authentication control signals.

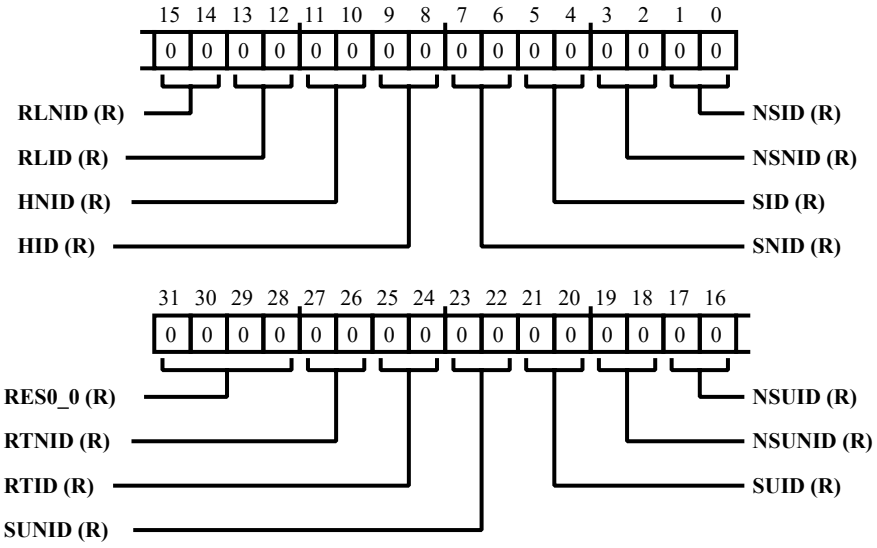


Figure 48-513: CSETF_AUTHSTATUS Register Diagram

Table 48-516: CSETF_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
27:26 (R/NW)	RTNID	. Root non-invasive debug.	
		0	Separate Root non-invasive debug enable not implemented or Root state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.

Table 48-516: CSETF_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	RTID	. Root invasive debug.	
		0	Separate Root invasive debug enable not implemented or Root state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
23:22 (R/NW)	SUNID	. Secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
21:20 (R/NW)	SUID	. Secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
19:18 (R/NW)	NSUNID	. Non-secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
17:16 (R/NW)	NSUID	. Non-secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-516: CSETF_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/NW)	RLNID	. Realm non-invasive debug.	
		0	Separate Realm non-invasive debug enable not implemented or Realm state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
13:12 (R/NW)	RLID	. Realm invasive debug.	
		0	Separate Realm invasive debug enable not implemented or Realm state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
11:10 (R/NW)	HNID	. Hypervisor non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
9:8 (R/NW)	HID	. Hypervisor invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-516: CSETF_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	. Secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
5:4 (R/NW)	SID	. Secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
3:2 (R/NW)	NSNID	. Non-secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
1:0 (R/NW)	NSID	. Non-secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Buffer Level Water Mark.

The value that is programmed into the BUFWM register indicates the required threshold vacancy level in 32-bit words in the trace memory. When the available space in the FIFO is less than or equal to this value, that is, fill level >= (MEM_SIZE - BUFWM), the full output is asserted and the STS.Full bit is set. This register is used only in the FIFO modes, that is, SWF1, SWF2, and HWF modes. In CB mode, the same functionality is obtained by programming the RWP to the required vacancy trigger level, so that when the pointer wraps around, the full output gets asserted indicating that the vacancy level has fallen below the required level. Reading this register returns the programmed value. The maximum value that can be written into this register is MEM_SIZE - 1, in which case the full output is asserted after the first 32-bit word is written to trace memory. Writing to this register other than when in Disabled state results in UNPREDICTABLE behavior. Any software using it must program it with an initial value before setting the CTL.TraceCaptEn bit to 1.

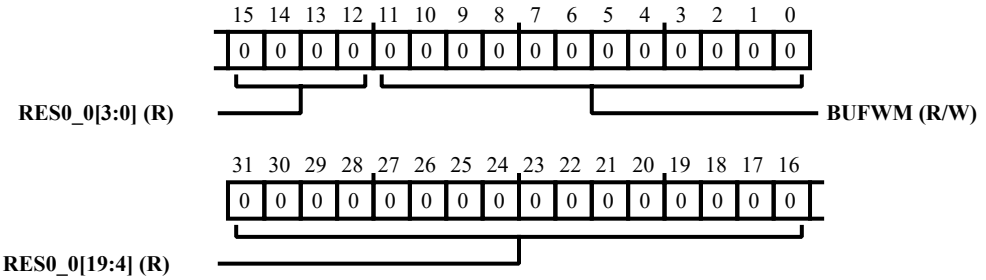


Figure 48-514: CSETF_BUFWM Register Diagram

Table 48-517: CSETF_BUFWM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
11:0 (R/W)	BUFWM	. Buffer Level Watermark. Indicates the desired threshold vacancy level in 32-bit words in the trace memory. The width of the register is log2(MEM_SIZE).

Current Buffer Fill Level.

The CBUFLEVEL register indicates the current fill level of the trace memory in units of 32-bit words. When the TMC leaves Disabled state, this register dynamically indicates the current fill level of trace memory. It retains its value on entering Disabled state. It is not affected by the reprogramming of pointer registers in Disabled state with the exception of RRD reads and RWD writes. Before leaving the Disabled state software must program RRP with the same value as RWP. Without doing this results in UNPREDICTABLE behavior.

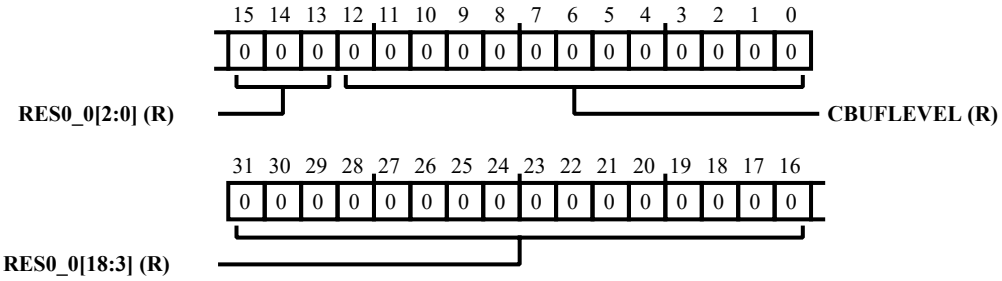


Figure 48-515: CSETF_CBUFLEVEL Register Diagram

Table 48-518: CSETF_CBUFLEVEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:13 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
12:0 (R/NW)	CBUFLEVEL	. Current Buffer Fill Level. Indicates the current fill level of the trace memory in 32-bit words. The width of the register is 1 + log2(MEM_SIZE).

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

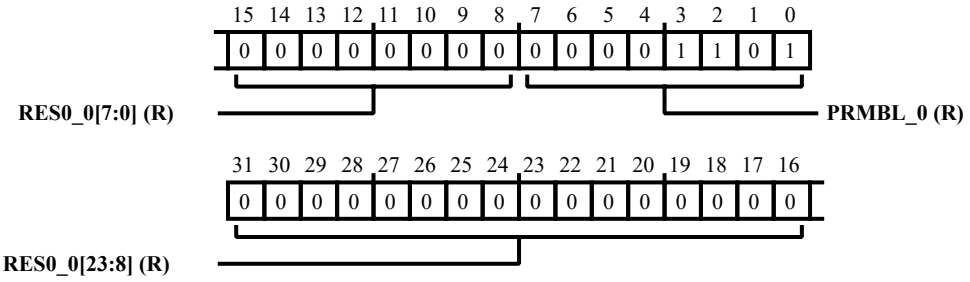


Figure 48-516: CSETF_CIDR0 Register Diagram

Table 48-519: CSETF_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

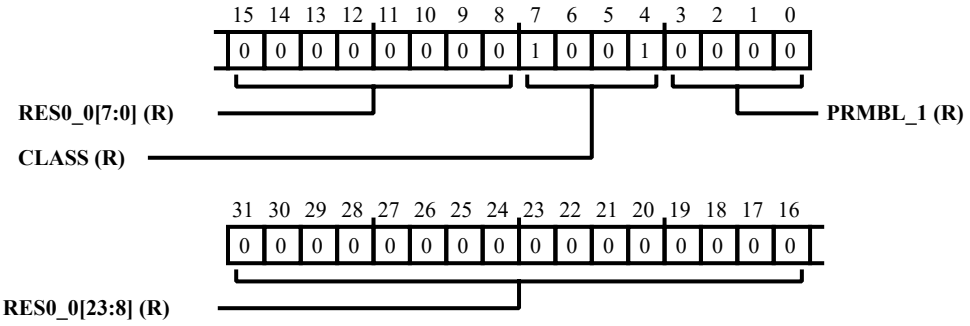


Figure 48-517: CSETF_CIDR1 Register Diagram

Table 48-520: CSETF_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

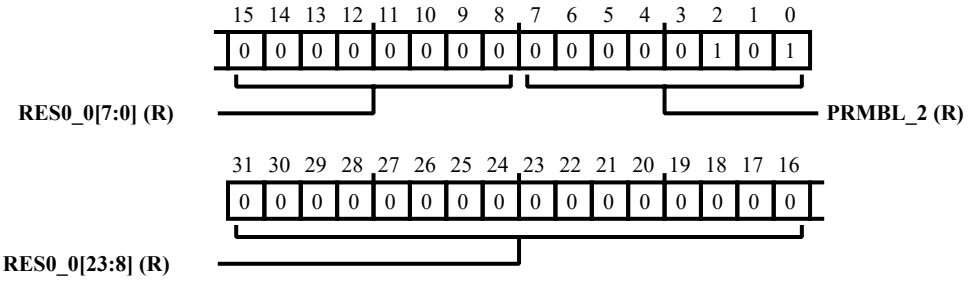


Figure 48-518: CSETF_CIDR2 Register Diagram

Table 48-521: CSETF_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

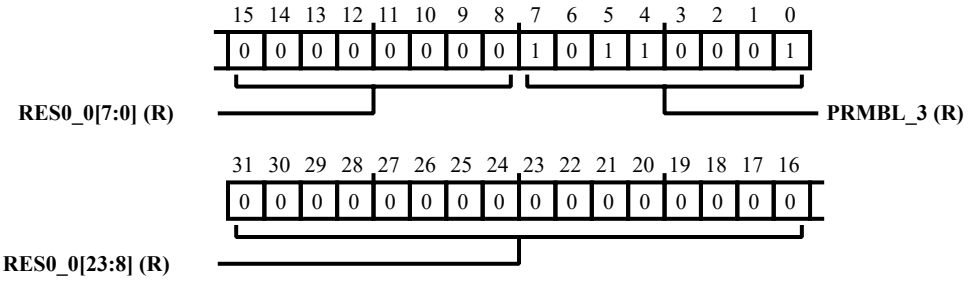


Figure 48-519: CSETF_CIDR3 Register Diagram

Table 48-522: CSETF_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Claim Tag Clear Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

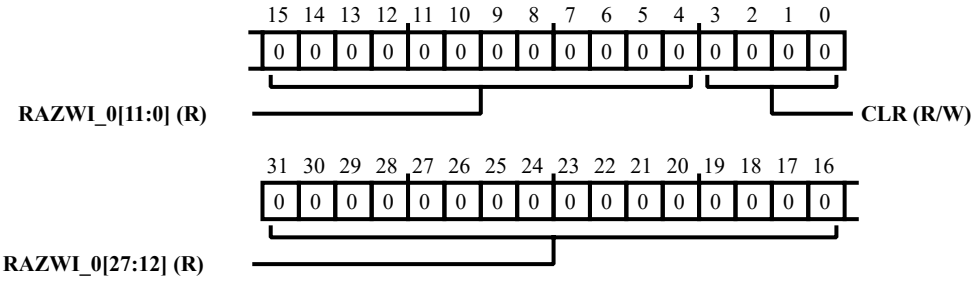


Figure 48-520: CSETF_CLAIMCLR Register Diagram

Table 48-523: CSETF_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	CLR	. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

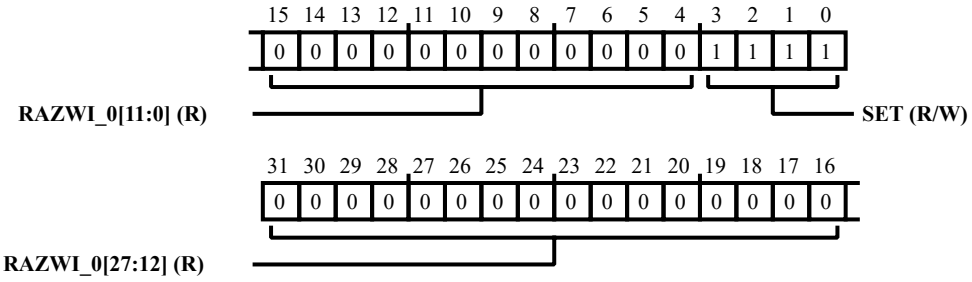


Figure 48-521: CSETF_CLAIMSET Register Diagram

Table 48-524: CSETF_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	SET	. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

Control Register.

The CTL register controls trace stream capture. Setting the CTL.TraceCaptEn bit to 1 enables the TMC to capture the trace data. When trace capture is enabled, the formatter behavior is controlled by the FFCR register.

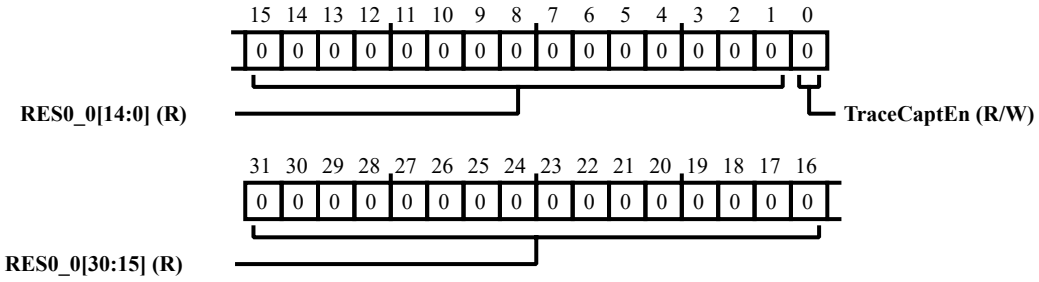


Figure 48-522: CSETF_CTL Register Diagram

Table 48-525: CSETF_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	TRACECAPTEN	. Trace capture enable:.	
		0	Disable trace capture.
		1	Enable trace capture.

Device Configuration Register.

This register is IMPLEMENTATION DEFINED for each Part Number and Designer. The register indicates the capabilities of the component.

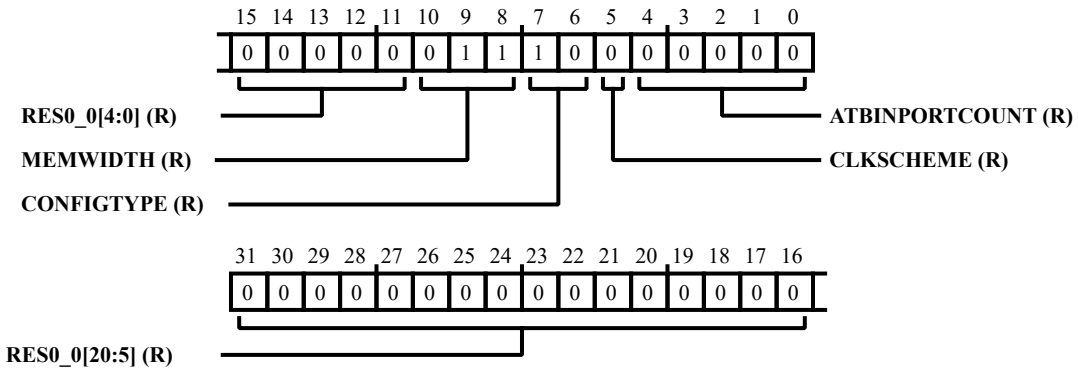


Figure 48-523: CSETF_DEVID Register Diagram

Table 48-526: CSETF_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:11 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
10:8 (R/NW)	MEMWIDTH	. Indicates the width of the internal memory data bus. For the ETF this value is twice ATB_DATA_WIDTH.	
		3	Memory interface databus is 64 bits wide. (ATB_DATA_WIDTH = 32bit).
		4	Memory interface databus is 128 bits wide. (ATB_DATA_WIDTH = 64bit).
		5	Memory interface databus is 256 bits wide. (ATB_DATA_WIDTH = 128bit).
7:6 (R/NW)	CONFIGTYPE	. Indicates the TMC configuration.	
		0	ETB - Embedded Trace Buffer.
		1	ETR - Embedded Trace Router.
		2	ETF - Embedded Trace FIFO.
		3	ETS - Embedded Trace Streamer.

Table 48-526: CSETF_DEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/NW)	CLKSCHEME	. RAM Clocking Scheme. This value indicates the TMC RAM clocking scheme used, that is, whether the TMC RAM operates synchronously or asynchronously to the TMC clock. Fixed to 0 indicating that TMC RAM clock is synchronous to the clk input.
4:0 (R/NW)	ATBINPORTCOUNT	. Hidden Level of ATB input multiplexing. This value indicates the type/number of ATB multiplexing present on the input ATB. Fixed to 0x00 indicating that no multiplexing is present.

Device Configuration Register 1.

Contains an IMPLEMENTATION DEFINED value.

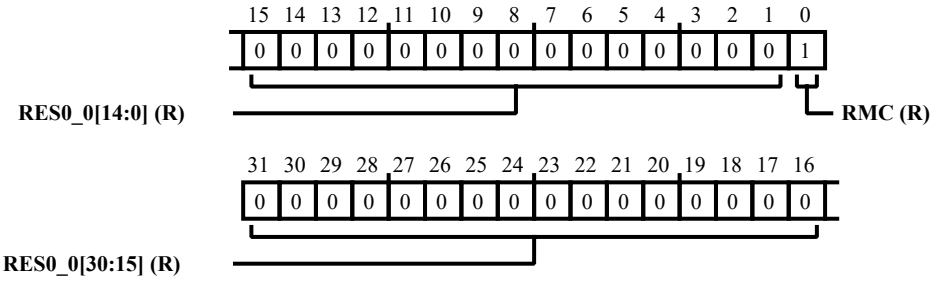


Figure 48-524: CSETF_DEVID1 Register Diagram

Table 48-527: CSETF_DEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
0 (R/NW)	RMC	. Register management mode. TMC implements register management mode 1.

Device Type Identifier Register.

A debugger can use this register to get information about a component that has an unrecognized Part number.

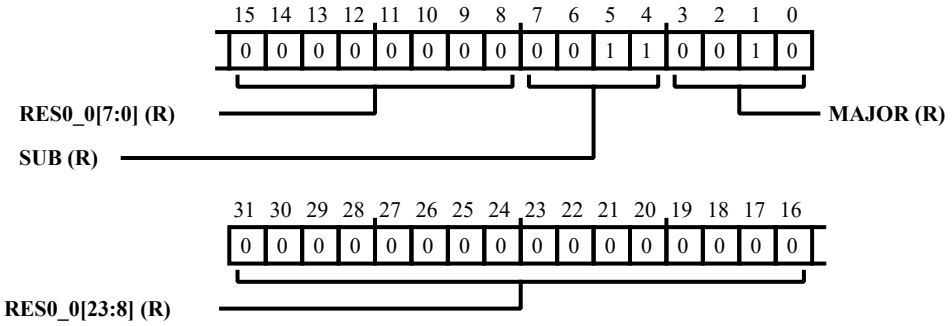


Figure 48-525: CSETF_DEVTYPE Register Diagram

Table 48-528: CSETF_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SUB	. Minor classification. Returns 0x3, indicating this component is a FIFO.
3:0 (R/NW)	MAJOR	. Major classification. Returns 0x2, indicating this component is a Trace Link.

Formatter and Flush Control Register.

The FFCR controls the generation of stop, trigger and flush events. The insertion of a flush completion packet and the insertion of a trigger packet in the formatted trace is enabled here. Also one of the 2 formatter modes for bypass mode and normal mode can be changed here when the formatter has stopped.

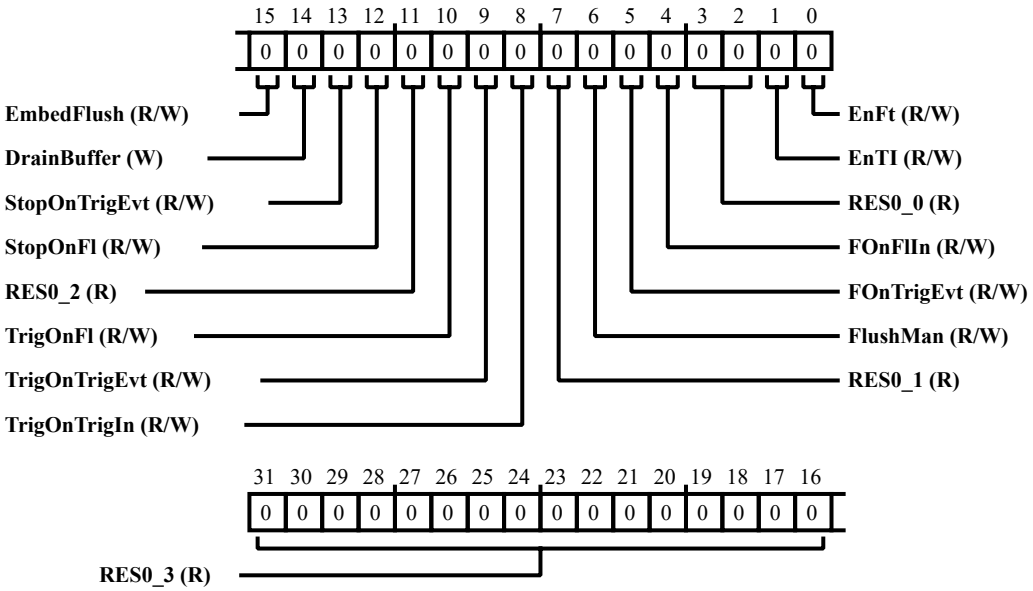


Figure 48-526: CSETF_FFCR Register Diagram

Table 48-529: CSETF_FFCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/NW)	RES0_3	. Reserved bit or field with SBZP behavior.	
15 (R/W)	EMBEDFLUSH	. Embed Flush ID (flush completion packet). Enables insertion of Flush ID 0x7B with a single byte of data payload = 0x00 in the output trace, immediately after the last flush data byte, when a flush completes on the ATB receiver interface. This bit is effective only in Normal formatting modes. In Bypass mode, the Flush ID insertion remains disabled and this bit is ignored.	
		0	Disable Flush ID insertion.
		1	Enable Flush ID insertion.

Table 48-529: CSETF_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (RX/W)	DRAINBUFFER	. Drain Buffer. This bit is used to enable draining of the trace data through the ATB transmitter interface after the formatter has stopped. It is useful in CB mode to capture trace data into trace memory and then to drain the captured trace through the ATB transmitter interface. Writing a 1 to this bit when in Stopped state starts the drain of the contents of trace buffer. This bit always reads as 0. The STS.TMCReady bit goes LOW while the drain is in progress. This bit is functional only when the TMC is in CB mode and formatting is enabled, that is, FFCR.EnFt=1. Setting this bit when the TMC is in any other mode, or when not in Stopped state, results in UNPREDICTABLE behavior. When trace capture is complete in CB mode, all of the captured trace must be retrieved from the trace memory, either by reading all trace data out through RRD reads, or draining all trace data by setting the FFCR.DrainBuffer bit. Setting this bit after some of the captured trace has been read out through RRD results in UNPREDICTABLE behavior.
13 (R/W)	STOPONTRIGEVT	. Stop On Trigger Event. If this bit is set, the formatter is stopped when a Trigger Event has been observed. This bit must be used only in CB mode. If trace capture is enabled in SWF1, or HWF mode with this bit set, it results in UNPREDICTABLE behavior.
12 (R/W)	STOPONFL	. Stop On Flush. If this bit is set, the formatter is stopped on completion of a flush operation. The initiation of a flush operation is controlled by programming the register bits FFCR.FlushMan, FFCR.FOnTrigEvt, and FFCR.FOnFlIn. When a flush-initiation condition occurs, afvalid_rx is asserted, and when the flush completion is received, that is, afready_rx=1, trace capture is stopped. Any remaining data in the formatter is appended with a post-amble and written to trace memory. The flush operation is then complete. If a flush is initiated by the ATB Transmitter interface, its completion does not lead to a formatter stop regardless of the value that is programmed in this bit.
11 (R/NW)	RES0_2	. Reserved bit or field with SBZP behavior.
10 (R/W)	TRIGONFL	. Indicate on trace stream the completion of flush. If this bit is set, a trigger is indicated on the trace stream when afready_rx is received for a flush in progress. If this bit is clear, no triggers are embedded in the trace stream on flush completion. If Trigger Insertion is disabled, that is, FFCR.EnTI=0, then trigger indication on the trace stream is blocked regardless of the value that is programmed in this bit. If a flush is initiated by the ATB Transmitter interface, its completion does not lead to a trigger indication on the trace stream regardless of the value that is programmed in this bit.

Table 48-529: CSETF_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	TRIGONTRIGEVT	. Indicate on trace stream the occurrence of a Trigger Event. If this bit is set, a trigger is indicated on the output trace stream when a Trigger Event occurs. If Trigger Insertion is disabled, that is, FFCR.EnTI=0, then trigger indication on the trace stream is blocked regardless of the value that is programmed in this bit. This bit must be used only in CB mode. If trace capture is enabled in SWF1, or HWF mode with this bit set, it results in UNPREDICTABLE behavior.
8 (R/W)	TRIGONTRIGIN	. Indicate on trace stream the occurrence of a rising edge on trigin. If this bit is set, a trigger is indicated on the trace stream when a rising edge is detected on the trigin input. If Trigger Insertion is disabled, that is, FFCR.EnTI=0, then trigger indication on the trace stream is blocked regardless of the value that is programmed in this bit.
7 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
6 (R/W)	FLUSHMAN	. Manually generate a flush of the system. Writing 1 to this bit causes a flush to be generated. This bit is cleared automatically when, in formatter bypass mode, afready_rx was sampled high, or, in normal formatting mode, afready_rx was sampled high and all flush data was output to the trace memory. If CTL.TraceCaptEn=0, writes to this bit are ignored.
5 (R/W)	FONTRIGEVT	. Flush on Trigger Event. If FFCR.StopOnTrigEvt is set, this bit is ignored. Setting this bit generates a flush when a Trigger Event occurs. If FFCR.StopOnTrigEvt is set, this bit is ignored. This bit must be used only in CB mode. If trace capture is enabled in SWF1, or HWF mode with this bit set, it results in UNPREDICTABLE behavior.
4 (R/W)	FONFLIN	. Setting this bit enables the detection of transitions on the flushin input by the TMC. If this bit is set and the formatter has not already stopped, a rising edge on flushin initiates a flush request.
3:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
1 (R/W)	ENTI	. Enable Trigger Insertion. Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 0x00 with atid_rx=0x7D in the trace stream. Trigger indication on the trace stream is also controlled by the register bits FFCR.TrigOnFl, FFCR.TrigOnTrigEvt, and FFCR.TrigOnTrigIn. This bit can only be changed when the TMC is in Disabled state. If FFCR.EnTI bit is set formatting is enabled.

Table 48-529: CSETF_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	ENFT	. Enable Formatter. If this bit is set, formatting is enabled. When EnTi is set, formatting is enabled. When CB mode is not used, formatting is also enabled. For backwards-compatibility with earlier versions of the ETB disabling of formatting is supported only in CB mode. This bit can only be changed when TMC is in Disabled state.

Formatter and Flush Status Register.

This register indicates the status of the Formatter, and the status of Flush request.

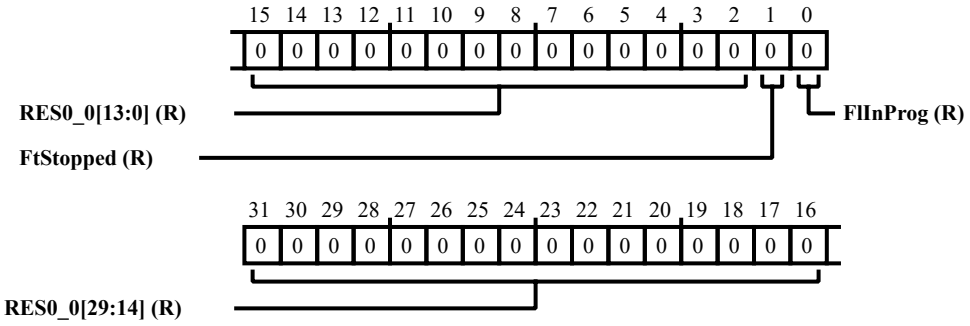


Figure 48-527: CSETF_FFSR Register Diagram

Table 48-530: CSETF_FFSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1 (R/NW)	FTSTOPPED	. Formatter Stopped. This bit behaves the same way as STS.FtEmpty.	
0 (R/NW)	FLINPROG	. Flush In Progress. This bit indicates whether the TMC is currently processing a flush request. The flush initiation is controlled by the flush control bits in the FFCR register. The flush request could additionally be from the ATB transmitter port. This bit is cleared to 0 when leaving the Disabled state and retains its value when entering the Disabled state. When in Disabled state, this bit is not updated.	
		0	No flush activity in progress.
		1	Flush in progress on the ATB receiver interface or the TMC internal pipeline.

Integration Test ATB Control 0 Register.

This register captures the values of ATB receiver inputs atvalid_rx, afready_rx, atwakeup_rx, and atbytes_rx in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, writes to this register are ignored and the reads return the value of corresponding input pins. The width of this register is given by: 8+log2(ATB DATA WIDTH/8).

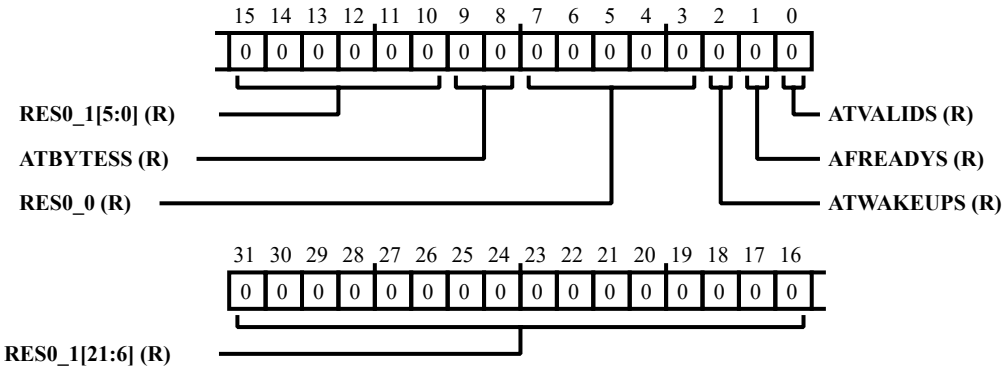


Figure 48-528: CSETF_ITATBCTR0 Register Diagram

Table 48-531: CSETF_ITATBCTR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:10 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
9:8 (R/NW)	ATBYTESS	. Returns the value of atbytes_rx input in integration mode. N=8+log2(ATB DATA WIDTH/8).
7:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (R/NW)	ATWAKEUPS	. Returns the value of atwakeup_rx input in integration mode.
1 (R/NW)	AFREADYS	. Returns the value of afready_rx input in integration mode.
0 (R/NW)	ATVALIDS	. Returns the value of atvalid_rx input in integration mode.

Integration Test ATB Control 1 Register.

This register captures the value of the atid_rx[6:0] input in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, writes to this register are ignored and the reads return the value of atid_rx input.

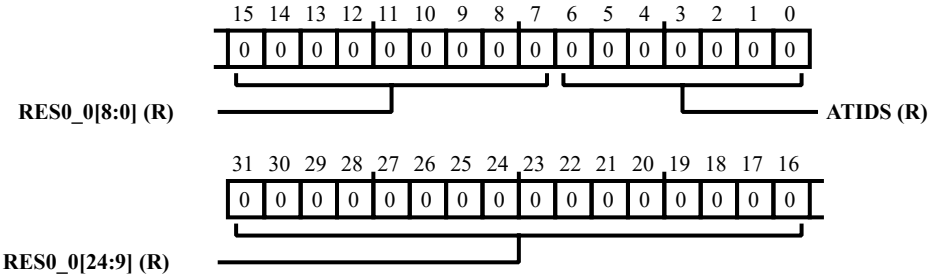


Figure 48-529: CSETF_ITATBCTR1 Register Diagram

Table 48-532: CSETF_ITATBCTR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
6:0 (R/NW)	ATIDS	. Returns the value of atid_rx[6:0] input in integration mode.

Integration Test ATB Control 2 Register.

This register enables control of ATB receiver outputs atready_rx, afvalid_rx, and syncreq_rx in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to any bit of this register is driven on the output pin that is controlled by that bit and the reads return 0x0.

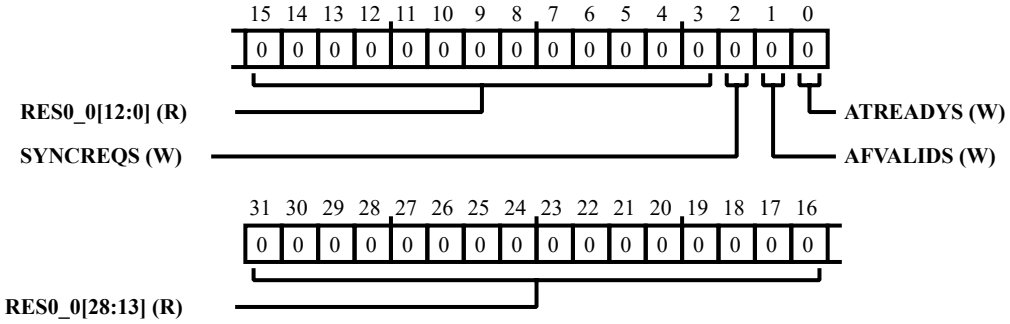


Figure 48-530: CSETF_ITATBCTR2 Register Diagram

Table 48-533: CSETF_ITATBCTR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (RX/W)	SYNCREQS	. Controls the value of syncreq_rx output in integration mode.
1 (RX/W)	AFVALIDS	. Controls the value of afvalid_rx output in integration mode.
0 (RX/W)	ATREADYS	. Controls the value of atready_rx output in integration mode.

Integration Test ATB Data 0 Register.

This register captures the value of atdata_rx input in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, writes to this register are ignored and the reads return the value of corresponding atdata_rx bits. The width of this register is given by: 1+(ATB DATA WIDTH)/8.

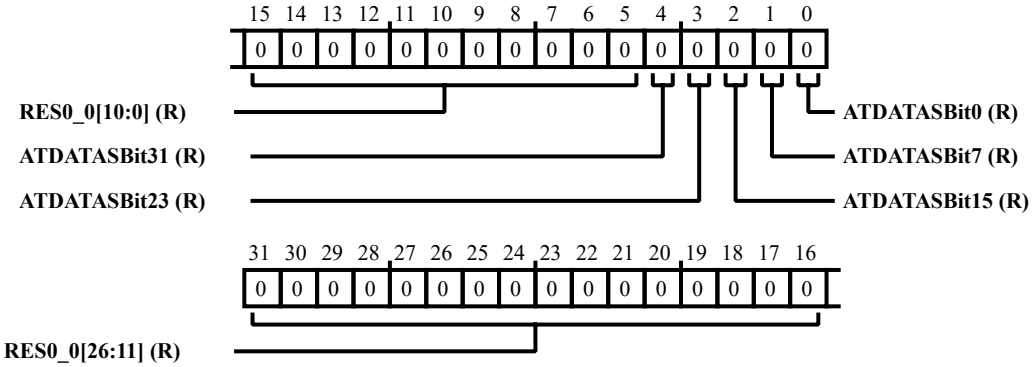


Figure 48-531: CSETF_ITATBDATA0 Register Diagram

Table 48-534: CSETF_ITATBDATA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4 (R/NW)	ATDATASBIT31	. Returns the value of atdata_rx[31] input in integration mode.
3 (R/NW)	ATDATASBIT23	. Returns the value of atdata_rx[23] input in integration mode.
2 (R/NW)	ATDATASBIT15	. Returns the value of atdata_rx[15] input in integration mode.
1 (R/NW)	ATDATASBIT7	. Returns the value of atdata_rx[7] input in integration mode.
0 (R/NW)	ATDATASBIT0	. Returns the value of atdata_rx[0] input in integration mode.

Integration Test ATB Transmitter Control 0 register.

This register enables control of the ATB transmitter outputs atbytes_tx, atwakeup_tx, afready_tx, and atvalid_tx in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to any bit of this register is driven on the output pin that is controlled by that bit, and the reads return 0x0.

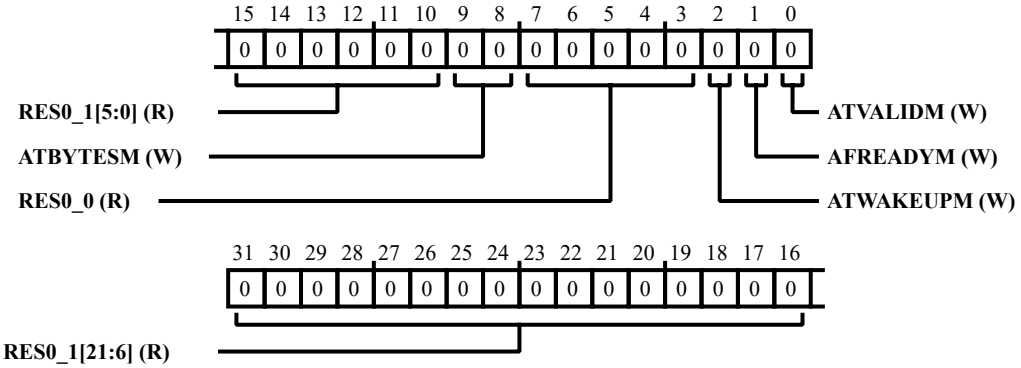


Figure 48-532: CSETF_ITATBMCTR0 Register Diagram

Table 48-535: CSETF_ITATBMCTR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:10 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
9:8 (RX/W)	ATBYTESM	. Controls the value of atbytes_tx output in integration mode. This width of this field is log2(ATB DATA WIDTH/8).
7:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (RX/W)	ATWAKEUPM	. Controls the value of atwakeup_tx output in integration mode.
1 (RX/W)	AFREADYM	. Controls the value of afready_tx output in integration mode.
0 (RX/W)	ATVALIDM	. Controls the value of atvalid_tx output in integration mode.

Integration Test ATB Transmitter Control 1 register.

This register enables control of the atid_tx output in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to this register is driven on the atid_tx output and the reads return 0x0.

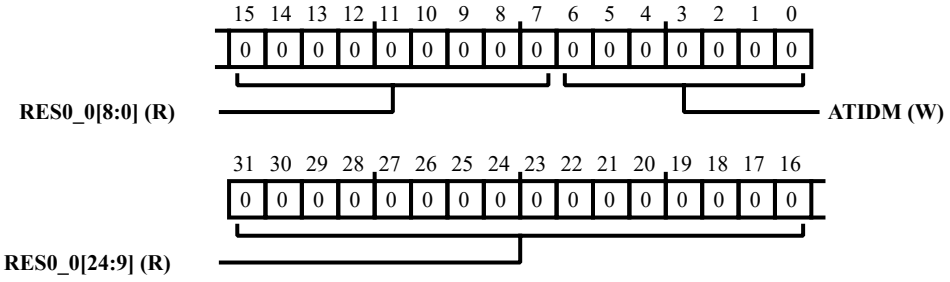


Figure 48-533: CSETF_ITATBMCTR1 Register Diagram

Table 48-536: CSETF_ITATBMCTR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
6:0 (RX/W)	ATIDM	. Controls the value of the atid_tx[6:0] output in integration mode.

Integration Test ATB Transmitter Control 2 register.

This register captures the values of ATB transmitter inputs atready_tx, afvalid_tx, and syncreq_tx in integration mode. In functional mode, this register behaves as RAZ/WI.

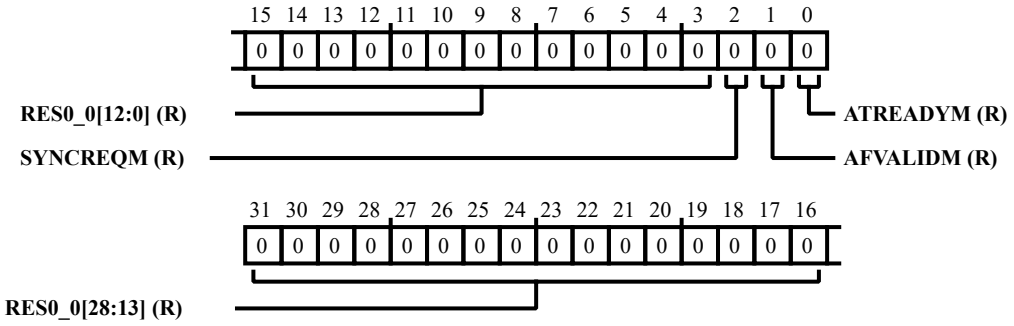


Figure 48-534: CSETF_ITATBMCTR2 Register Diagram

Table 48-537: CSETF_ITATBMCTR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (R/NW)	SYNCREQM	. Integration status of syncreq_tx input. In integration mode, this bit latches to 1 on a rising edge of syncreq_tx input, which is cleared when this register is read or when integration mode is disabled.
1 (R/NW)	AFVALIDM	. Integration status of afvalid_tx input. In integration mode, writes are ignored and reads return the value of afvalid_tx input.
0 (R/NW)	ATREADYM	. Integration status of atready_tx input. In integration mode, writes are ignored and reads return the value of the atready_tx input.

Integration Test ATB Transmitter Data 0 register.

This register enables control of the atdata_tx output in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to any given bit is driven on the output pin that is controlled by that bit and the reads return 0x0. The width of this register is given by 1+(ATB DATA WIDTH)/8.

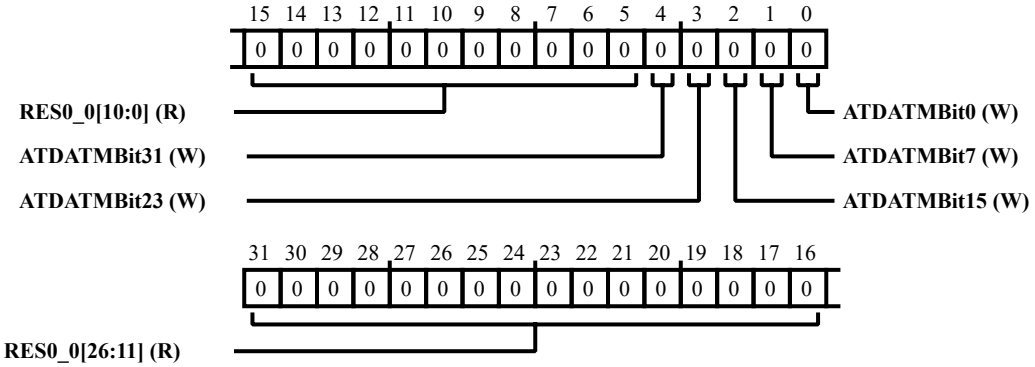


Figure 48-535: CSETF_ITATBMDATA0 Register Diagram

Table 48-538: CSETF_ITATBMDATA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4 (RX/W)	ATDATMBIT31	. Controls the value of atdata_tx[31] output in integration mode.
3 (RX/W)	ATDATMBIT23	. Controls the value of atdata_tx[23] output in integration mode.
2 (RX/W)	ATDATMBIT15	. Controls the value of atdata_tx[15] output in integration mode.
1 (RX/W)	ATDATMBIT7	. Controls the value of atdata_tx[7] output in integration mode.
0 (RX/W)	ATDATMBIT0	. Controls the value of atdata_tx[0] output in integration mode.

Integration Mode Control Register.

The Integration Mode Control register is used to switch between functional mode and integration mode. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

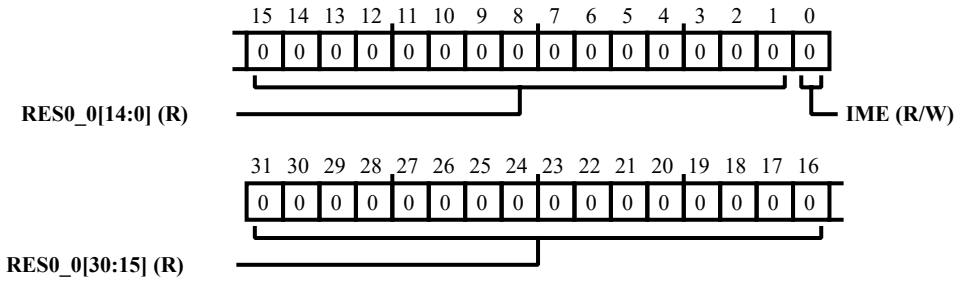


Figure 48-536: CSETF_ITCTRL Register Diagram

Table 48-539: CSETF_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	IME	. Integration Mode Enable.	
		0	The component must enter functional mode.
		1	The component must enter integration mode, and enable support for topology detection and integration testing.

Integration Test Event and Interrupt Control Register.

This register controls the values of event and interrupt outputs in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to any bit of this register is driven on the output pin that is controlled by that bit and the reads return 0x0.

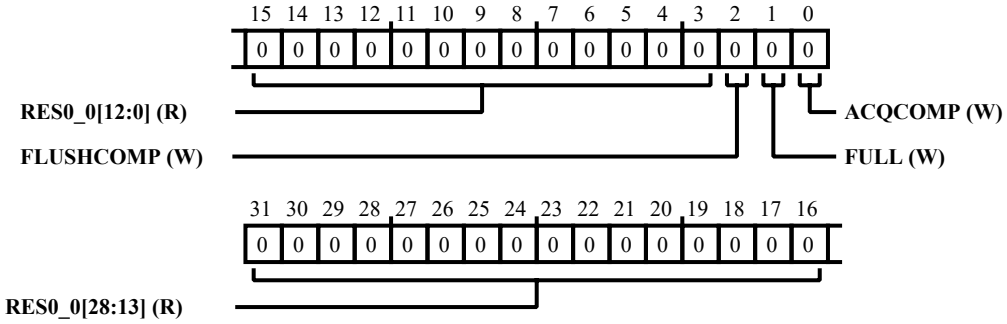


Figure 48-537: CSETF_ITEVTINTR Register Diagram

Table 48-540: CSETF_ITEVTINTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (RX/W)	FLUSHCOMP	. Controls the value of flushcomp output in integration mode.
1 (RX/W)	FULL	. Controls the value of full output in integration mode.
0 (RX/W)	ACQCOMP	. Controls the value of acqcomp output in integration mode.

Integration Test Trigger In and Flush In register.

This register captures the values of the flushin and trigin inputs in integration mode. In functional mode, this register behaves as RAZ/WI.

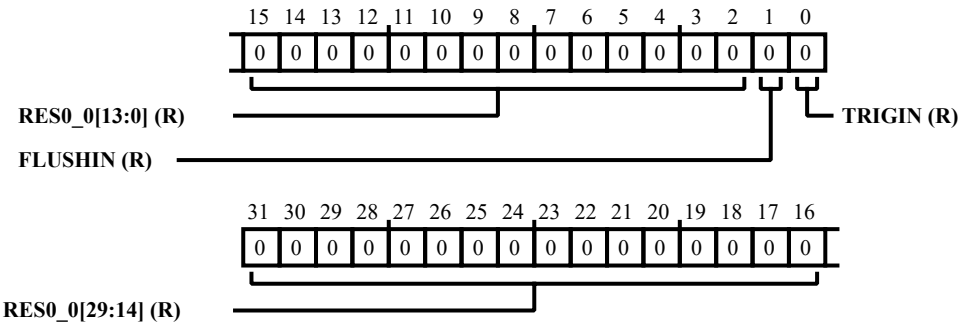


Figure 48-538: CSETF_ITTRFLIN Register Diagram

Table 48-541: CSETF_ITTRFLIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
1 (R/NW)	FLUSHIN	. Integration status of flushin input. In integration mode, this bit latches to 1 on a rising edge of the flushin input. It is cleared when the register is read or when integration mode is disabled.
0 (R/NW)	TRIGIN	. Integration status of trigin input. In integration mode, this bit latches to 1 on a rising edge of the trigin input. It is cleared when the register is read or when integration mode is disabled.

Latched Buffer Fill Level.

Reading the LBUFLEVEL register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level. When entering Disabled state, it retains its last value. While in Disabled state, reads from this register do not affect its value. When exiting Disabled state, the LBUFLEVEL register is updated to the current fill level.

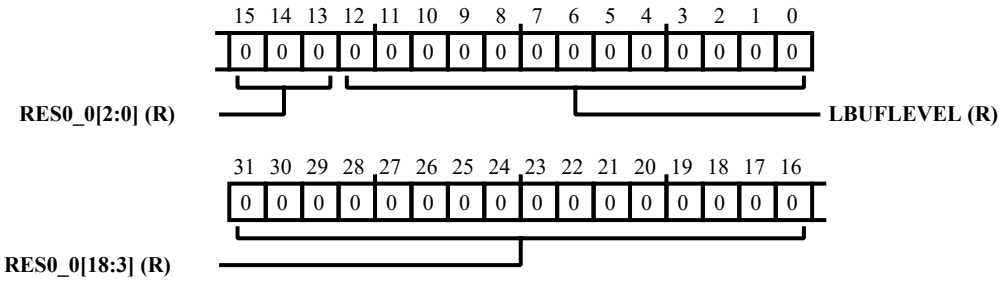


Figure 48-539: CSETF_LBUFLEVEL Register Diagram

Table 48-542: CSETF_LBUFLEVEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:13 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
12:0 (R/NW)	LBUFLEVEL	. Latched Buffer Fill Level. Indicates the maximum fill level of the trace memory in 32-bit words since this register was last read. The width of the register is 1 + log2(MEM_SIZE).

Mode register.

The MODE register controls the TMC operating mode. The operating mode can only be changed when the TMC is in Disabled state. Attempting to write to this register in any other state results in UNPREDICTABLE behavior. The operating mode is ignored when in Disabled state.

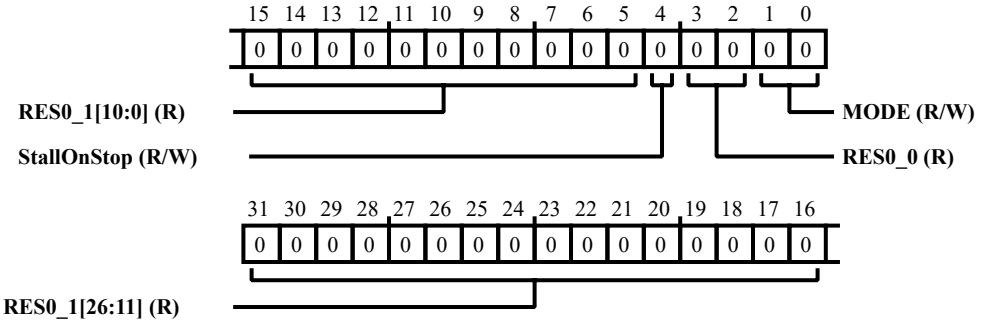


Figure 48-540: CSETF_MODE Register Diagram

Table 48-543: CSETF_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
4 (R/W)	STALLONSTOP	. Stall On Stop. If this bit is set and the formatter stops as a result of a stop event, the output atready_rx is de-asserted to stall the ATB interface and avoid loss of trace. If this bit is clear and the formatter stops as a result of a stop event, signal atready_rx remains asserted but the TMC discards further incoming trace.	
3:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1:0 (R/W)	MODE	. Selects the operating mode. If a reserved MODE value is programmed and trace capture is enabled, the TMC starts to operate in SWF1 mode. However, reading the MODE.MODE field returns the programmed value.	
		0	CB, Circular Buffer mode.
		1	SWF1, Software Read FIFO mode 1.
		2	HWF, Hardware Read FIFO mode.
		3	Reserved. (SWF1).

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

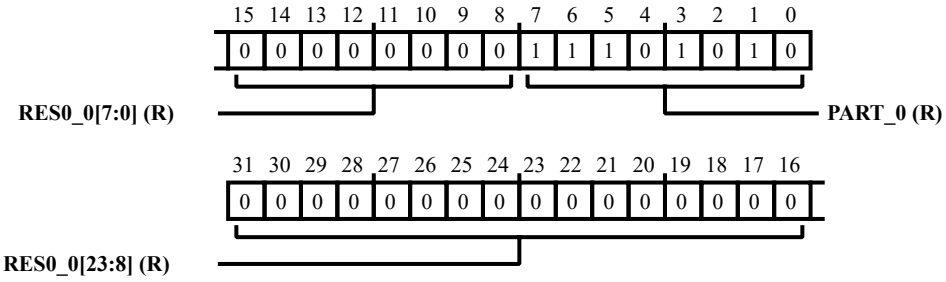


Figure 48-541: CSETF_PIDR0 Register Diagram

Table 48-544: CSETF_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:0 (R/NW)	PART_0	. Part number (lower 8 bits).	
		232	ETR or ETS - Embedded Trace Router or Streamer.
		233	ETB - Embedded Trace Buffer.
		234	ETF - Embedded Trace FIFO.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

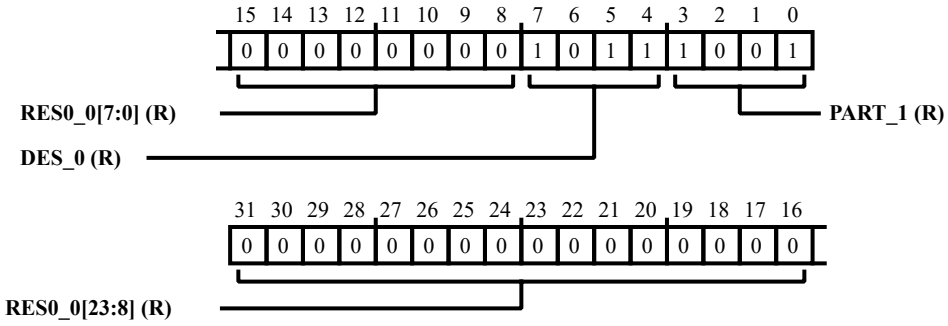


Figure 48-542: CSETF_PIDR1 Register Diagram

Table 48-545: CSETF_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

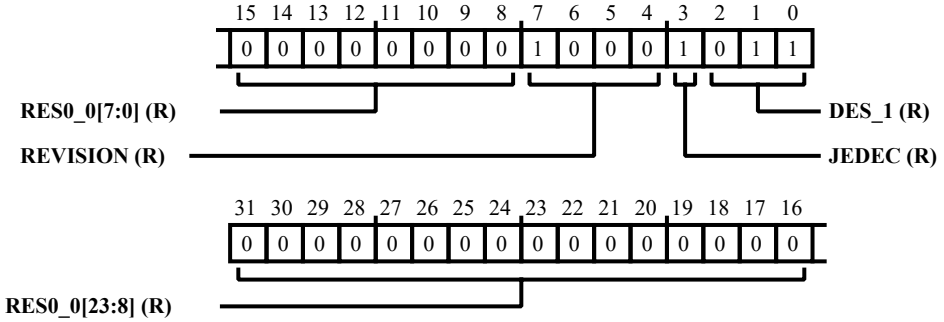


Figure 48-543: CSETF_PIDR2 Register Diagram

Table 48-546: CSETF_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

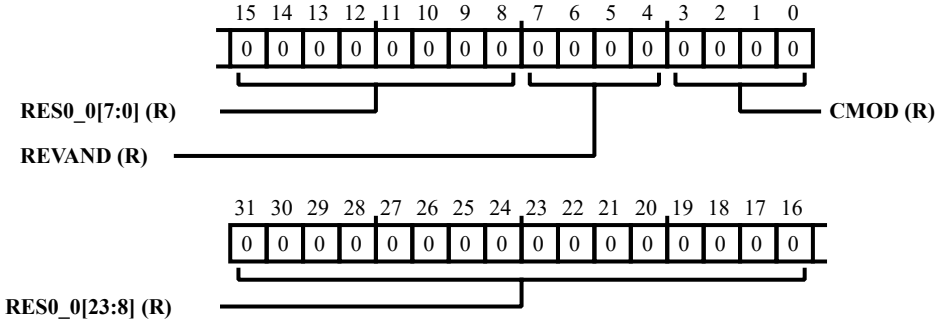


Figure 48-544: CSETF_PIDR3 Register Diagram

Table 48-547: CSETF_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

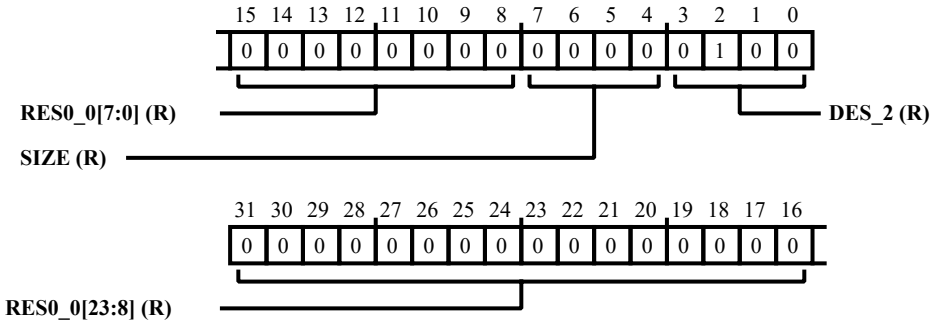


Figure 48-545: CSETF_PIDR4 Register Diagram

Table 48-548: CSETF_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

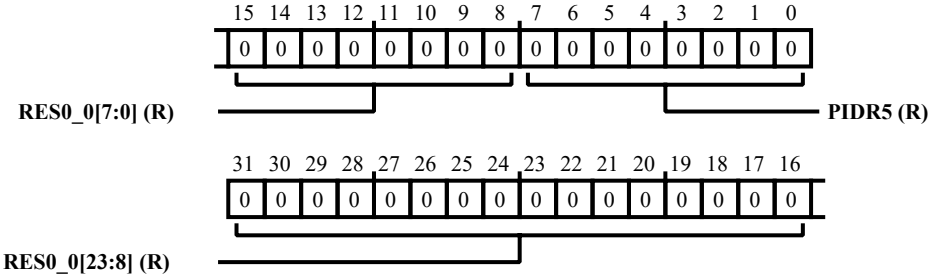


Figure 48-546: CSETF_PIDR5 Register Diagram

Table 48-549: CSETF_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

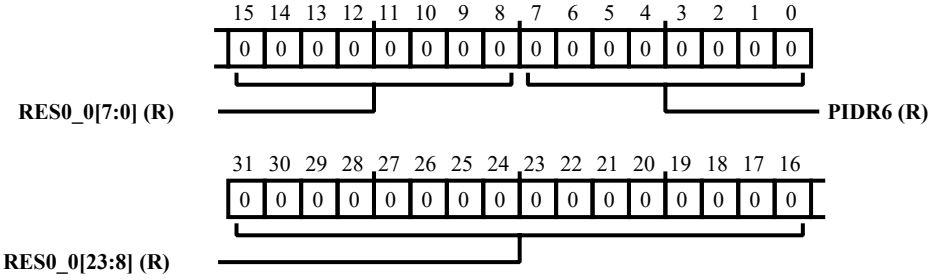


Figure 48-547: CSETF_PIDR6 Register Diagram

Table 48-550: CSETF_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

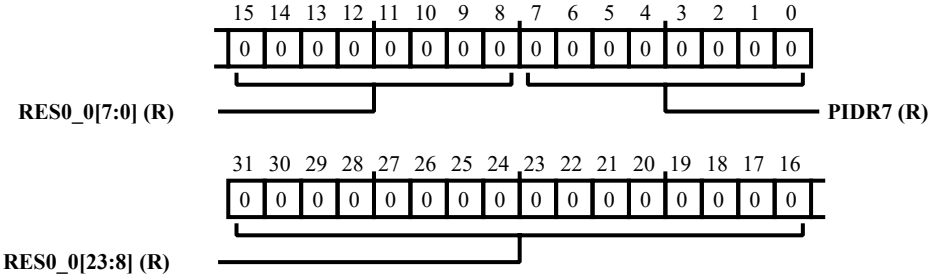


Figure 48-548: CSETF_PIDR7 Register Diagram

Table 48-551: CSETF_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

Periodic Synchronization Counter Register.

This register determines the reload value of the Periodic Synchronization Counter. This counter enables the frequency of sync packets to be optimized to the trace capture buffer size. The default behavior of the counter is to generate periodic synchronization requests, syncreq_rx, on the ATB receiver interface.

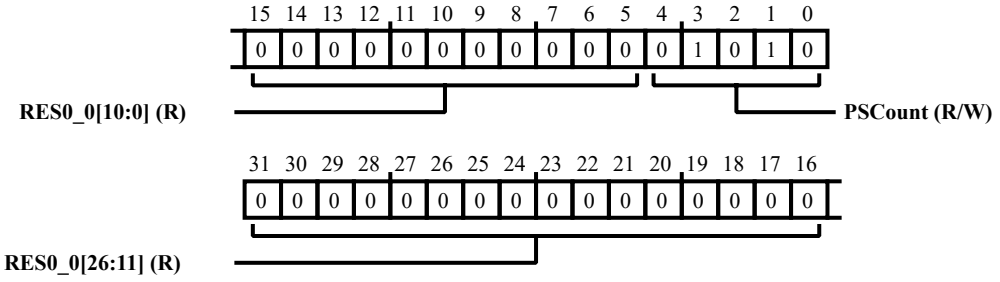


Figure 48-549: CSETF_PSCR Register Diagram

Table 48-552: CSETF_PSCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4:0 (R/W)	PSCOUNT	. Periodic Synchronization Count. Determines the reload value of the Synchronization Counter. The reload value takes effect the next time the counter reaches zero. When trace capture is enabled, the Synchronization Counter counts the number of bytes of trace data that is stored into the trace memory, regardless of whether the trace data has been formatted by the TMC or not, since the occurrence of the last sync request on the ATB receiver interface. When the counter reaches 0, a sync request is sent on the ATB receiver interface. Reads from this register return the reload value that is programmed in this register. This field resets to 0x0A, that is, the default sync period is 2^10 bytes. If a reserved value is programmed in this register field, the value 0x1B is used instead, and subsequent reads from this register also return 0x1B. The following constraints apply to the values written to the PSCount field: 0x0 - synchronization is disabled, 0x1-0x6 - reserved, 0x7-0x1B - synchronization period is 2^PSCount bytes. The smallest value 0x7 gives a sync period of 128 bytes. The maximum allowed value 0x1B gives a sync period of 2^27 bytes, 0x1C-0x1F - reserved.

RAM Read Data register.

Reading this register allows data to be read from the trace memory at the location pointed to by the RRP register when either in the Disabled state or operating in CB or SWF1 mode. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the memory width is twice as wide and a memory word holds 8, 16 or 32 bytes. Multiple RRD reads must be performed to read a full memory word. When a full memory width of data has been read via the RRD register, the RRP register is incremented to the next memory word. When the TMC left the Disabled state and the trace memory is empty, this register returns 0xFFFFFFFF. When the TMC left the Disabled state and the trace memory is empty, this register returns 0xFFFFFFFF. When operating in CB mode and the TMC left the Disabled state, this register returns 0xFFFFFFFF in all other states except the Stopped state. When operating in HWF mode, this register also returns 0xFFFFFFFF.

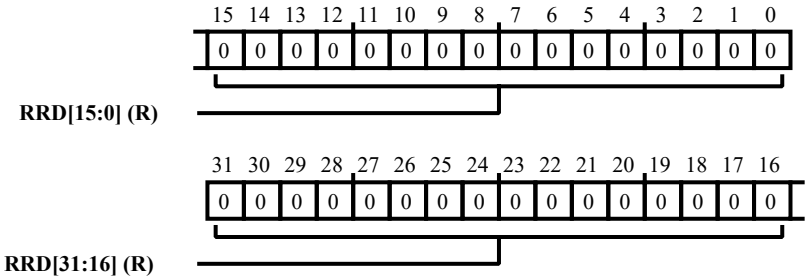


Figure 48-550: CSETF_RRD Register Diagram

Table 48-553: CSETF_RRD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RRD	. Returns the data read from trace memory.

RAM Read Pointer register.

The RRP register contains the value of the read pointer that is used to read entries from trace memory over the APB interface. Software must program it with the same value as RWP before enabling trace capture.

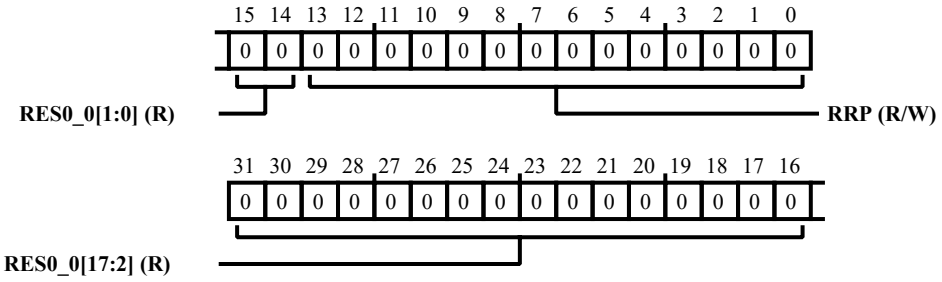


Figure 48-551: CSETF_RRP Register Diagram

Table 48-554: CSETF_RRP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:14 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
13:0 (R/W)	RRP	. The RRP width depends on the size of trace memory and is given by log2(MEM_SIZE x 4). The remaining MSBs of the 32-bit register are of type RAZ/WI. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the memory width is twice as wide and a memory word holds 8, 16 or 32 bytes. When a full memory width of data has been read via the RRD register, the RRP register is incremented to the next memory word. The lowest 4 bits have access type RAZ/WI when ATB_DATA_WIDTH is 32 or 64 bits. The lowest 5 bits have access type RAZ/WI when ATB_DATA_WIDTH is 128 bits.

RAM Size register.

The RSZ register defines the size of trace memory in units of 32-bit words.

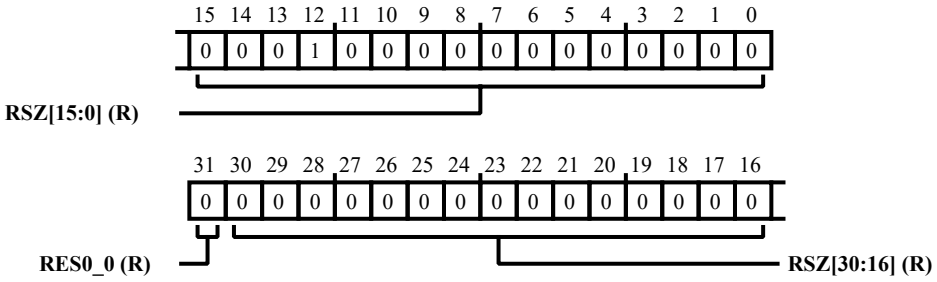


Figure 48-552: CSETF_RSZ Register Diagram

Table 48-555: CSETF_RSZ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
30:0 (R/NW)	RSZ	. RAM size. Indicates the size of the RAM in 32-bit words. For example: Returns 0x00000100 if trace memory size is 1KB, 0x40000000 if trace memory size is 4GB. This field has the same value as the MEM_SIZE parameter.

RAM Write Data register.

The RWD register enables testing of trace memory connectivity to the TMC. Writing this register allows data to be written to the trace memory at the location pointed to by the RWP register when in the Disabled state. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the memory width is twice as wide and a memory word holds 8, 16 or 32 bytes. Multiple RWD writes must be performed to write a full memory word. When a full memory width of data has been written via the RWD register, the data is written to the trace memory and the RWP register is incremented to the next memory word.

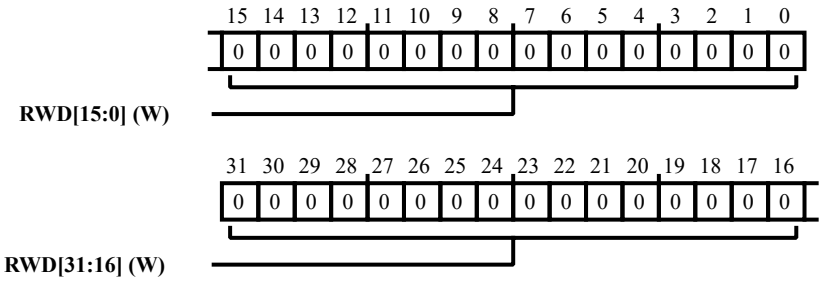


Figure 48-553: CSETF_RWD Register Diagram

Table 48-556: CSETF_RWD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	RWD	. Data written to this register is placed in the trace memory.

RAM Write Pointer register.

The RWP register sets the write pointer that writes entries into the trace memory. Software must program it before enabling trace capture.

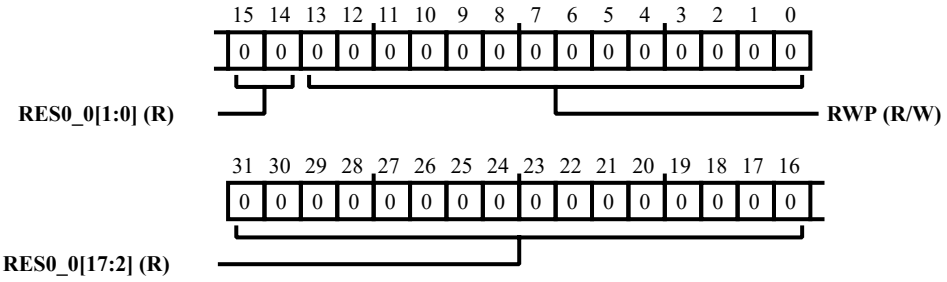


Figure 48-554: CSETF_RWP Register Diagram

Table 48-557: CSETF_RWP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:14 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
13:0 (R/W)	RWP	. The RWP width depends on the size of trace memory and is given by log2(MEM_SIZE x 4). The remaining MSBs of the 32-bit register are of type RAZ/WI. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the memory width is twice as wide and a memory word holds 8, 16 or 32 bytes. When a full memory width of data has been written to the RWD register, the RWP register is incremented to the next memory word. The lowest 4 bits have access type RAZ/WI when ATB_DATA_WIDTH is 32 or 64 bits. The lowest 5 bits have access type RAZ/WI when ATB_DATA_WIDTH is 128 bits.

Status register.

The STS register indicates the status of the Trace Memory Controller. After a reset, software must ignore all the fields of this register except STS.TMCReady. The other fields only have meaning when the TMC has left the Disabled state. Writes to all RO fields of this register are ignored.

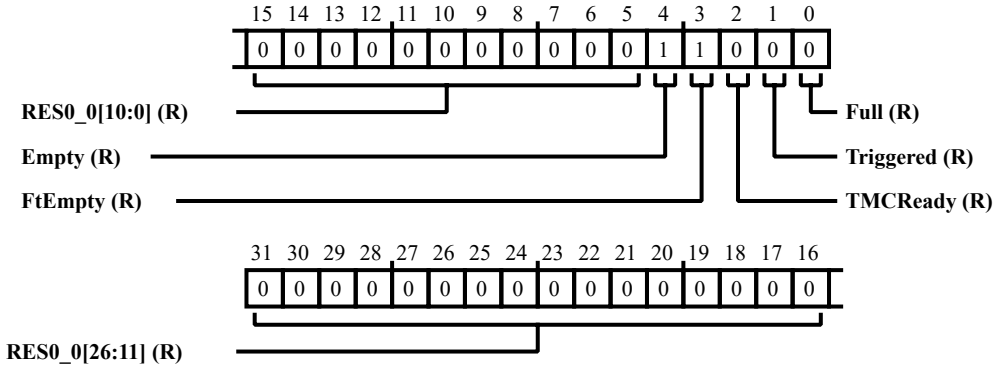


Figure 48-555: CSETF_STS Register Diagram

Table 48-558: CSETF_STS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4 (R/NW)	EMPTY	. Trace buffer empty. If set, this bit indicates that the trace memory does not contain any valid trace data. The reset value of this bit is 1. On leaving Disabled state, this bit dynamically indicates the empty status of trace memory, CBUFLEVEL == 0. However, this does not mean that the pipeline stages within the TMC are empty. To determine whether the internal pipeline stages are empty, the software must read the STS.TMCReady bit. In Circular Buffer mode the Empty bit and the Full bit in this register can be 1 at the same time. This happens because the Full bit in this mode, when set, does not clear until TraceCaptEn is set.
3 (R/NW)	FTEMPTY	. Trace capture has been completed and all captured trace data has been written to the trace memory, set in Stopped or Disabled state. Otherwise, it is cleared. The reset value is 1.
2 (R/NW)	TMCREADY	. Trace capture has been completed, all captured trace data has been written to the trace memory, and reading from trace memory completed as a result of FFCR.DrainBuffer bit being set.

Table 48-558: CSETF_STS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	TRIGGERED	. TMC triggered. This bit is set when trace capture is in progress and the TMC has detected a trigger event. This bit is cleared to 0 when leaving the Disabled state and retains its value when entering the Disabled state. A trigger event is when the TMC has written a set number of data words, as programmed in the TRG register, into the trace memory after a rising edge of trigin input, or a trigger packet (atid_rx = 0x7D) is received in the input trace.
0 (R/NW)	FULL	. Trace memory full. This bit helps determine the amount of valid data present in the trace memory. It is not affected by the reprogramming of pointer registers in Disabled state. It is cleared when the TMC leaves Disabled state. In Circular Buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer. It remains set until the TraceCaptEn bit is cleared and set. In Software FIFO and Hardware FIFO mode, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the BUFWM Register, that is, Fill level >= MEM_SIZE - BUFWM. The FULL output from the TMC reflects the value of this register bit, except when the Integration Mode bit in the ITCTRL Register, 0xF00, is set.

Trigger Counter register.

The TRG register, in Circular Buffer mode, specifies the number of 32-bit words to capture in the trace memory, after detecting either a rising edge on the trigin input or a trigger packet in the incoming trace stream, that is, where atid_rx = 0x7D. The value programmed must be aligned to the frame length of 128 bits. Software must program this register before leaving Disabled state.

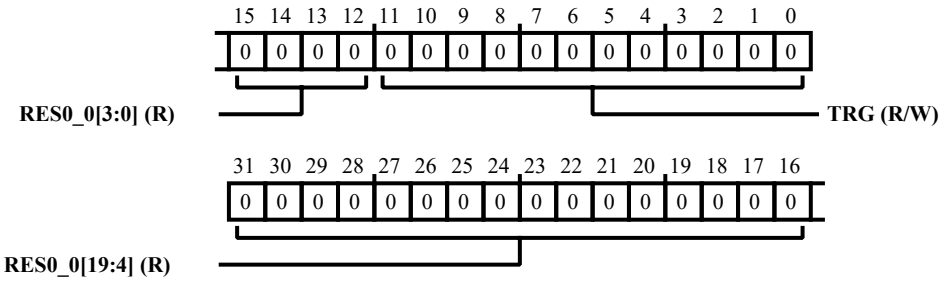


Figure 48-556: CSETF_TRG Register Diagram

Table 48-559: CSETF_TRG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
11:0 (R/W)	TRG	. Trigger count. This count represents the number of 32-bit words of trace that are captured between a trigger packet and a trigger event. The lowest two bits have access type RAZ/WI.

ADSP-2184x CSETR Register Descriptions

Your module description, here. (CSETR) contains the following registers.

Table 48-560: ADSP-2184x CSETR Register List

Name	Description
CSETR_AUTHSTATUS	Authentication Status Register.
CSETR_AXICTL	AXI Control Register.
CSETR_AXICTL1	AXI Control Register 1.
CSETR_BUFWM	Buffer Level Water Mark.
CSETR_CBUFLEVEL	Current Buffer Fill Level.
CSETR_CIDR0	Component Identification Register 0.
CSETR_CIDR1	Component Identification Register 1.

Table 48-560: ADSP-2184x CSETR Register List (Continued)

Name	Description
CSETR_CIDR2	Component Identification Register 2.
CSETR_CIDR3	Component Identification Register 3.
CSETR_CLAIMCLR	Claim Tag Clear Register.
CSETR_CLAIMSET	Claim Tag Set Register.
CSETR_CTL	Control Register.
CSETR_DBAHI	Data Buffer Address HIGH register.
CSETR_DBALO	Data Buffer Address Low register.
CSETR_DEVID	Device Configuration Register.
CSETR_DEVID1	Device Configuration Register 1.
CSETR_DEVTYPE	Device Type Identifier Register.
CSETR_FFCR	Formatter and Flush Control Register.
CSETR_FFSR	Formatter and Flush Status Register.
CSETR_ITATBCTR0	Integration Test ATB Control 0 Register.
CSETR_ITATBCTR1	Integration Test ATB Control 1 Register.
CSETR_ITATBCTR2	Integration Test ATB Control 2 Register.
CSETR_ITATBDATA0	Integration Test ATB Data 0 Register.
CSETR_ITCTRL	Integration Mode Control Register.
CSETR_ITEVTINTR	Integration Test Event and Interrupt Control Register.
CSETR_ITTRFLIN	Integration Test Trigger In and Flush In register.
CSETR_LBUFLEVEL	Latched Buffer Fill Level.
CSETR_MODE	Mode register.
CSETR_PIDR0	Peripheral Identification Register 0.
CSETR_PIDR1	Peripheral Identification Register 1.
CSETR_PIDR2	Peripheral Identification Register 2.
CSETR_PIDR3	Peripheral Identification Register 3.
CSETR_PIDR4	Peripheral Identification Register 4.
CSETR_PIDR5	Peripheral Identification Register 5.
CSETR_PIDR6	Peripheral Identification Register 6.
CSETR_PIDR7	Peripheral Identification Register 7.
CSETR_PSCR	Periodic Synchronization Counter Register.
CSETR_RRD	RAM Read Data register.

Table 48-560: ADSP-2184x CSETR Register List (Continued)

Name	Description
CSETR_RRP	RAM Read Pointer register.
CSETR_RRPHI	RAM Read Pointer High register.
CSETR_RSZ	RAM Size register.
CSETR_RURP	RAM Update Read Pointer register.
CSETR_RWD	RAM Write Data register.
CSETR_RWP	RAM Write Pointer register.
CSETR_RWPHI	RAM Write Pointer High register.
CSETR_STS	Status register.
CSETR_TRG	Trigger Counter register.

Authentication Status Register.

Reports the current status of the authentication control signals.

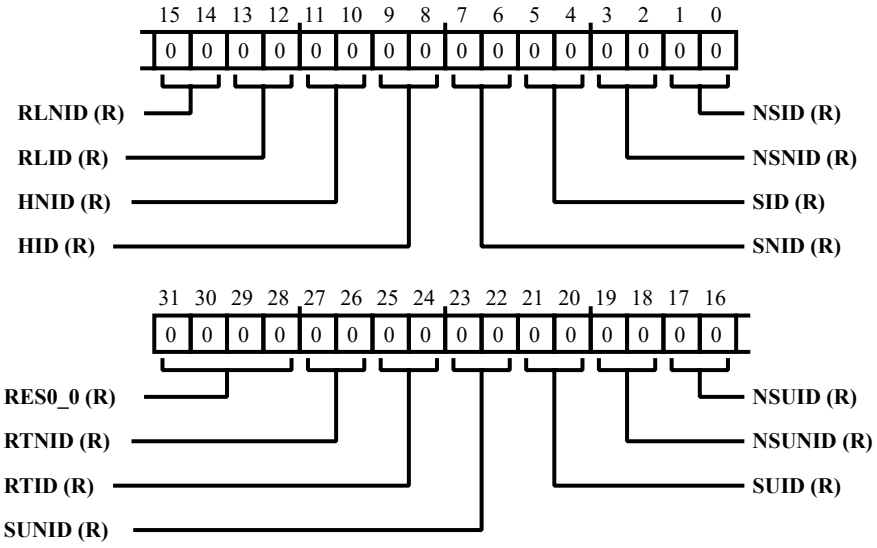


Figure 48-557: CSETR_AUTHSTATUS Register Diagram

Table 48-561: CSETR_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
27:26 (R/NW)	RTNID	. Root non-invasive debug.	
		0	Separate Root non-invasive debug enable not implemented or Root state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.

Table 48-561: CSETR_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	RTID	. Root invasive debug.	
		0	Separate Root invasive debug enable not implemented or Root state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
23:22 (R/NW)	SUNID	. Secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
21:20 (R/NW)	SUID	. Secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
19:18 (R/NW)	NSUNID	. Non-secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
17:16 (R/NW)	NSUID	. Non-secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-561: CSETR_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/NW)	RLNID	. Realm non-invasive debug.	
		0	Separate Realm non-invasive debug enable not implemented or Realm state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
13:12 (R/NW)	RLID	. Realm invasive debug.	
		0	Separate Realm invasive debug enable not implemented or Realm state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
11:10 (R/NW)	HNID	. Hypervisor non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
9:8 (R/NW)	HID	. Hypervisor invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-561: CSETR_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	. Secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
5:4 (R/NW)	SID	. Secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
3:2 (R/NW)	NSNID	. Non-secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
1:0 (R/NW)	NSID	. Non-secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

AXI Control Register.

This register controls TMC accesses to system memory through the AXI interface. The TMC only performs data accesses, so the arprot_m[2] and awprot_m[2] outputs are LOW for all AXI transfers.

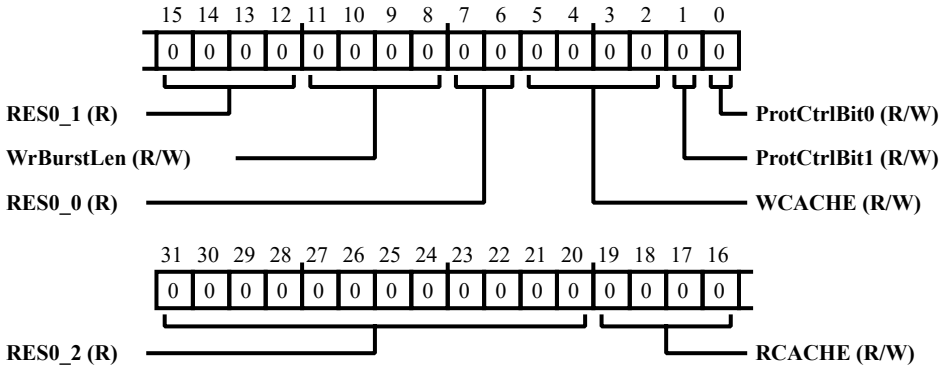


Figure 48-558: CSETR_AXICTL Register Diagram

Table 48-562: CSETR_AXICTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:20 (R/NW)	RES0_2	. Reserved bit or field with SBZP behavior.
19:16 (R/W)	RCACHE	. This field controls the AXI cache encoding for read transfers, that is, the value to be driven on the arcache_m[3:0] bus. Software must only program a valid AXI cache encoding value in this field. These values are defined in the AMBA AXI Protocol Specification. If software attempts to program an invalid value, 0x0 is written to this field instead.
15:12 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
11:8 (R/W)	WRBURSTLEN	. Write Burst Length. This field indicates the maximum number of data transfers that can occur within each burst that is initiated by the TMC on the AXI interface. The write burst that is initiated on the AXI can be shorter than the programmed value in a case when the formatter has stopped due to a stop condition having occurred.
7:6 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.

Table 48-562: CSETR_AXICTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5:2 (R/W)	WCACHE	. This field controls the AXI cache encoding for write transfers, that is, the value to be driven on the awcache_m[3:0] bus. Software must only program a valid AXI cache encoding value in this field. These values are defined in the AMBA AXI Protocol Specification. If software attempts to program an invalid value, 0x0 is written to this field instead.
1 (R/W)	PROTCTRLBIT1	. Secure Access (AXI definition). This bit controls the value that is driven on arprot_m[1] or awprot_m[1] on the AXI manager interface when performing AXI transfers.
0 (R/W)	PROTCTRLBIT0	. Privileged Access (AXI definition). This bit controls the value that is driven on arprot_m[0] or awprot_m[0] on the AXI manager interface when performing AXI transfers.

AXI Control Register 1.

This register controls address translation stashing requests made to a system MMU through the AXI interface.When the ETR is integrated with stash_disable == 1, this register behaves as RAZ/WI.

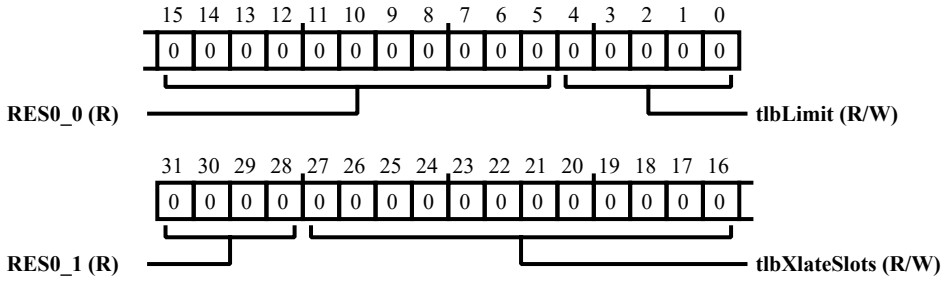


Figure 48-559: CSETR_AXICTL1 Register Diagram

Table 48-563: CSETR_AXICTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
27:16 (R/W)	TLBXLATESLOTS	. Writes: are ignored when stash_disabled tie-off is high. Otherwise, in Disabled state, stores APB write data [27:16] (The reset default is 0x0). Otherwise, in all other states, writing this field is disabled. Reads: Returns 0x0 when stash_disable tie-off is high. Otherwise this field returns the result of the stash_xlate_limit tie-off value exored with the stored APB write data. However, results less than 0x2 are limited to 0x2. 0x02 - 0xFFF ETR issues a maximum of 2 - 4095 outstanding stash translation requests.
15:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4:0 (R/W)	TLBLIMIT	. Writes: are ignored when stash_disabled tie-off is high. Otherwise, in Disabled state, stores APB write data [5:0] (The reset default is 0x0). Otherwise, in all other states, writing this field is disabled.Reads: Returns 0x0 when stash_disable tie-off is high. Otherwise this field returns the result of tlb_limit tie-off value exored with the stored APB write data. However, results larger than 0x10 are limited to 0x10. 0x00 : Translation stashing is disabled 0x01 - 0x10 : The ETR issues stash translation requests for up to 2^tlbLimit * 4k pages ahead of RWP page.

Buffer Level Water Mark.

The value that is programmed into the BUFWM register indicates the required threshold vacancy level in 32-bit words in the trace memory. When the available space in the FIFO is less than or equal to this value, that is, fill level >= (MEM_SIZE - BUFWM), the full output is asserted and the STS.Full bit is set. This register is used only in the FIFO modes, that is, SWF1, SWF2, and HWF modes. In CB mode, the same functionality is obtained by programming the RWP to the required vacancy trigger level, so that when the pointer wraps around, the full output gets asserted indicating that the vacancy level has fallen below the required level. Reading this register returns the programmed value. The maximum value that can be written into this register is MEM_SIZE - 1, in which case the full output is asserted after the first 32-bit word is written to trace memory. Writing to this register other than when in Disabled state results in UNPREDICTABLE behavior. Any software using it must program it with an initial value before setting the CTL.TraceCaptEn bit to 1.

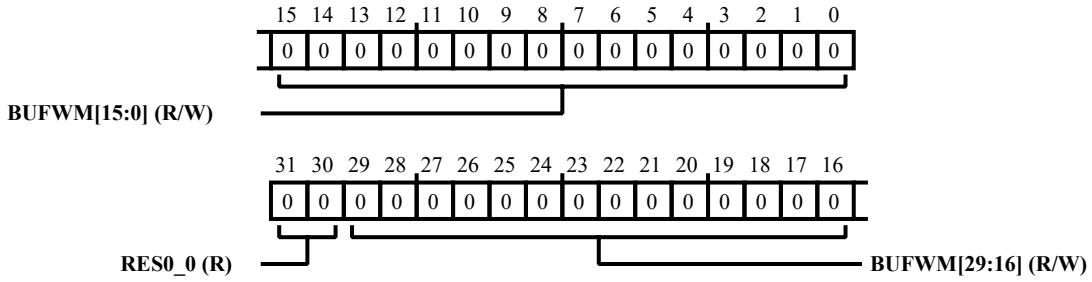


Figure 48-560: CSETR_BUFWM Register Diagram

Table 48-564: CSETR_BUFWM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
29:0 (R/W)	BUFWM	. Buffer Level Watermark. Indicates the required threshold vacancy level in 32-bit words in the trace memory.

Current Buffer Fill Level.

The CBUFLEVEL register indicates the current fill level of the trace memory in units of 32-bit words. When the TMC leaves Disabled state, this register dynamically indicates the current fill level of trace memory. It retains its value on entering Disabled state. It is not affected by the reprogramming of pointer registers in Disabled state with the exception of RRD reads and RWD writes.

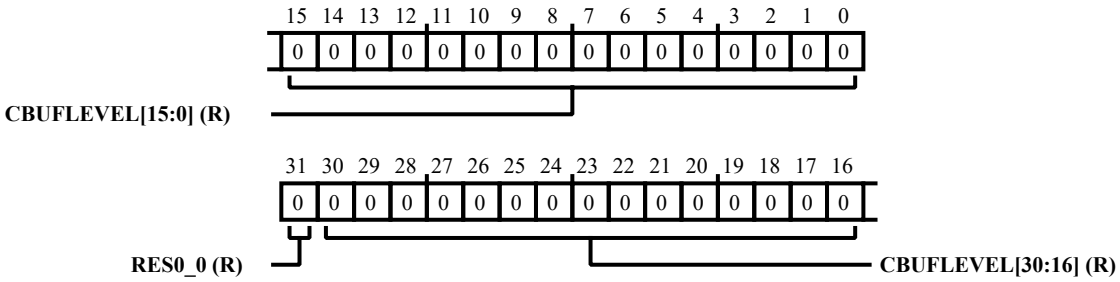


Figure 48-561: CSETR_CBUFLEVEL Register Diagram

Table 48-565: CSETR_CBUFLEVEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
30:0 (R/NW)	CBUFLEVEL	. Current Buffer Fill Level. Indicates the current fill level of the trace memory in 32-bit words.

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

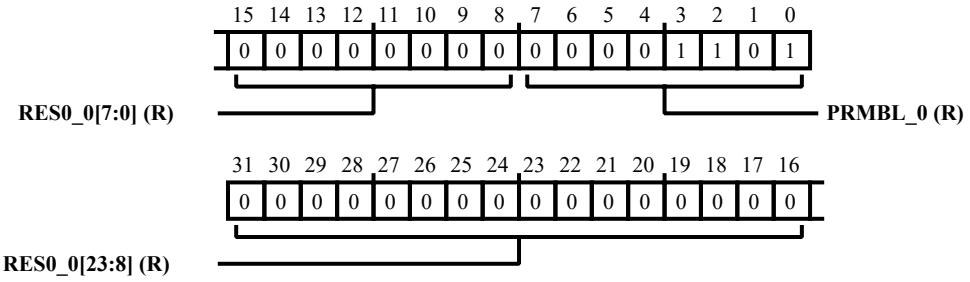


Figure 48-562: CSETR_CIDR0 Register Diagram

Table 48-566: CSETR_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

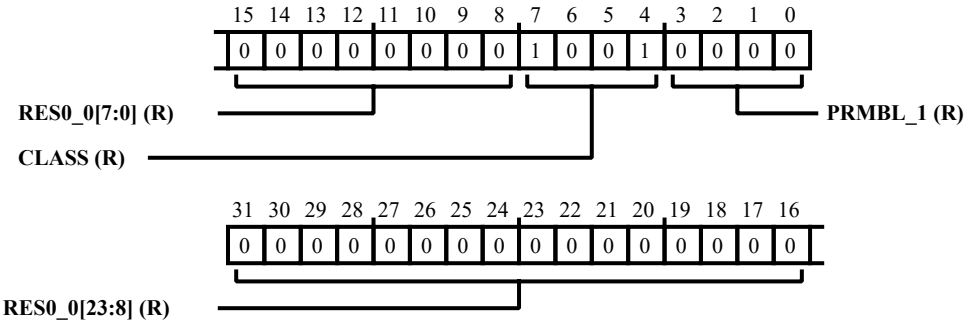


Figure 48-563: CSETR_CIDR1 Register Diagram

Table 48-567: CSETR_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

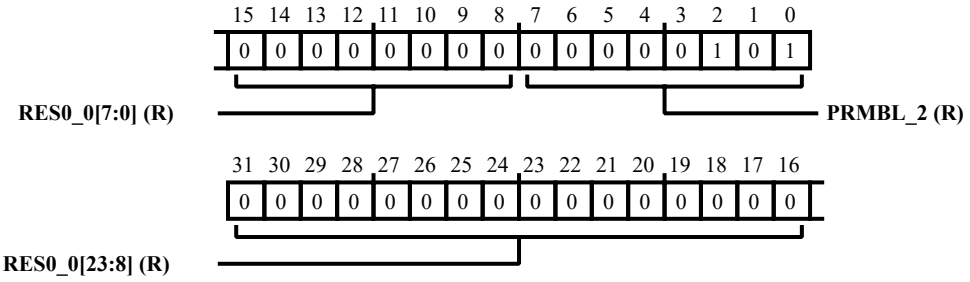


Figure 48-564: CSETR_CIDR2 Register Diagram

Table 48-568: CSETR_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

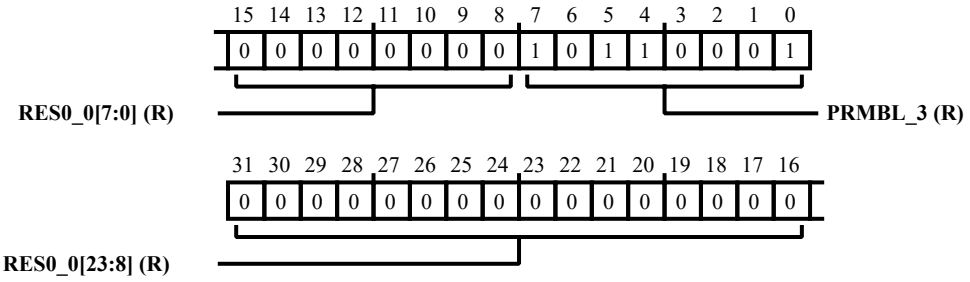


Figure 48-565: CSETR_CIDR3 Register Diagram

Table 48-569: CSETR_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Claim Tag Clear Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

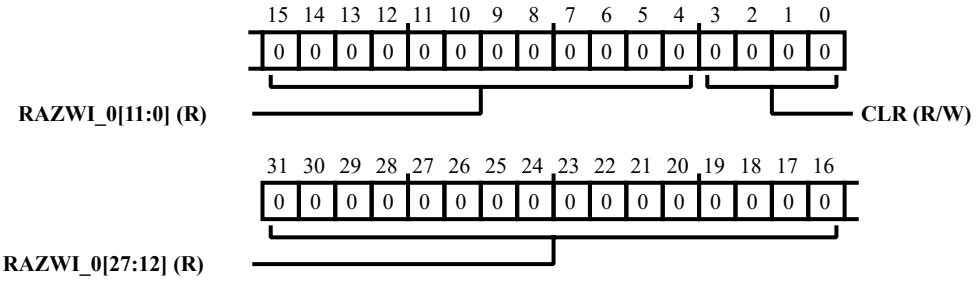


Figure 48-566: CSETR_CLAIMCLR Register Diagram

Table 48-570: CSETR_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	CLR	. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

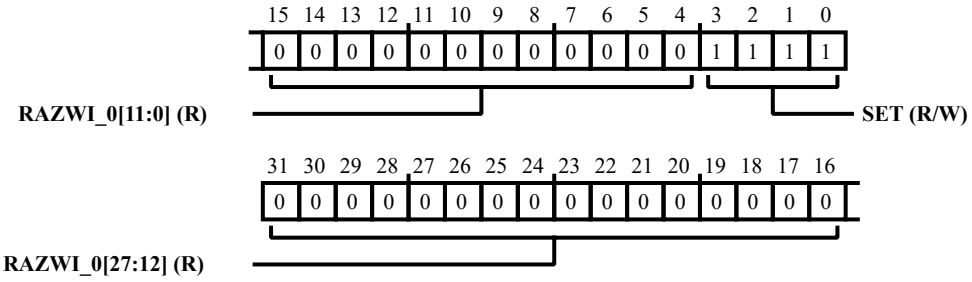


Figure 48-567: CSETR_CLAIMSET Register Diagram

Table 48-571: CSETR_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	SET	. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

Control Register.

The CTL register controls trace stream capture. Setting the CTL.TraceCaptEn bit to 1 enables the TMC to capture the trace data. When trace capture is enabled, the formatter behavior is controlled by the FFCR register.

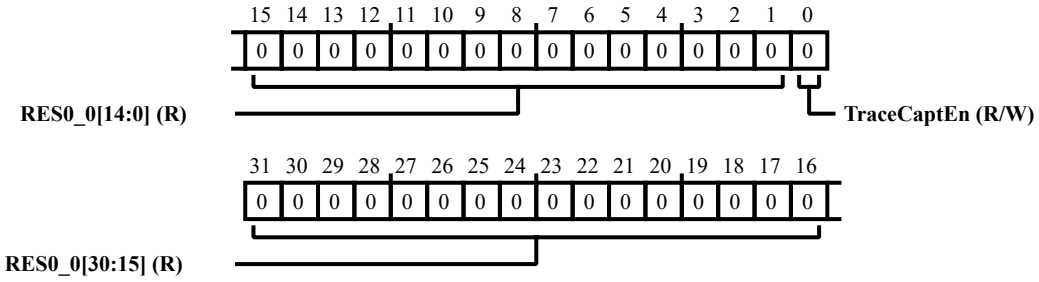


Figure 48-568: CSETR_CTL Register Diagram

Table 48-572: CSETR_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	TRACECAPTEN	. Trace capture enable:.	
		0	Disable trace capture.
		1	Enable trace capture.

Data Buffer Address HIGH register.

This register, together with the DBALO register, enables the TMC to locate the trace buffer in system memory. It contains bits >= bit[32] of the start address of the trace buffer in system memory. The width of this register is given by: (AXI_ADDR_WIDTH - 32), however, if AXI_ADDR_WIDTH is <= 32 bits, this register is reserved and access type is RAZ/WI. Modifying this register, other than when in Disabled state, results in UNPREDICTABLE behavior. Software must program it with an initial value before setting CTL.TraceCaptEn bit to 1.

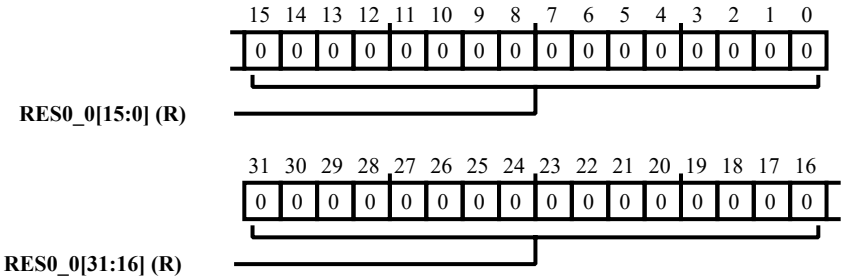


Figure 48-569: CSETR_DBAHI Register Diagram

Table 48-573: CSETR_DBAHI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.

Data Buffer Address Low register.

This register, together with the DBAHI register, enables the TMC to locate the trace buffer in system memory. This register contains bits [31:0] of the start address of the trace buffer in system memory. This register is 32 bits wide if AXI_ADDR_WIDTH is >= 32 bits. Software must program it before enabling trace capture, and the programmed value must be aligned to the Trace Memory Data Width and the Frame Width. Programming an unaligned value results in UNPREDICTABLE behavior. Modifying this register other than when in Disabled state results in UNPREDICTABLE behavior.

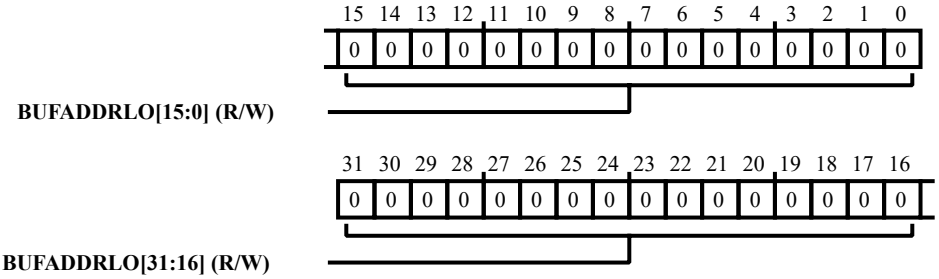


Figure 48-570: CSETR_DBALO Register Diagram

Table 48-574: CSETR_DBALO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	BUFADDRLO	. Data Buffer Low Address. Holds the lower 32 bits of the AXI address that is used to locate the trace buffer in system memory.The lowest four bits have access type RAZ/WI.

Device Configuration Register.

This register is IMPLEMENTATION DEFINED for each Part Number and Designer. The register indicates the capabilities of the component.

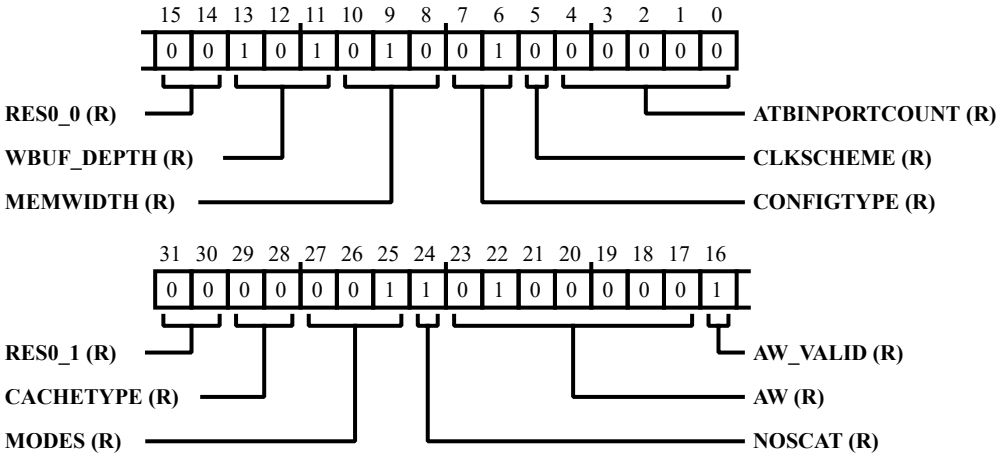


Figure 48-571: CSETR_DEVID Register Diagram

Table 48-575: CSETR_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
29:28 (R/NW)	CACHETYPE	. Indicates the format of BUSCTL register bus control fields. Reads as 2'b00 indicating that AXICTL bus attribute bits [19:16] and [5:2] follow an implementation-defined non-generic format. See AXICTL register description.	
27:25 (R/NW)	MODES	. Indicates the supported modes of operation. Reads as 3'b001 indicating that ETR supports CB, SWF1, and SWF2 modes.	
24 (R/NW)	NOSCAT	. Indicates whether the scatter-gather mode is implemented. Fixed at 1 indicating that scatter-gather mode is not implemented.	
23:17 (R/NW)	AW	. This field indicates the width of AXI address bus in ETR configuration. This field is valid only when DEVID.AW_VALID is set. Possible values are:.	
		32	32-bit AXI address bus.
		40	40-bit AXI address bus.
		44	44-bit AXI address bus.

Table 48-575: CSETR_DEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		48	48-bit AXI address bus.
		52	52-bit AXI address bus.
		64	64-bit AXI address bus.
16 (R/NW)	AW_VALID	. Indicates whether field DEVID.AW is valid. The value of this field is fixed at 1.	
15:14 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
13:11 (R/NW)	WBUF_DEPTH	. Log2 of the number of write buffer entries. This value is set by the parameter WBUFFER_DEPTH. Each entry is of size ATB_DATA_WIDTH.	
		2	Depth of Write buffer is 4 entries.
		3	Depth of Write buffer is 8 entries.
		4	Depth of Write buffer is 16 entries.
		5	Depth of Write buffer is 32 entries.
		6	Depth of Write buffer is 64 entries.
		7	Depth of Write buffer is 128 entries.
10:8 (R/NW)	MEMWIDTH	. Indicates the width of the internal memory data bus. For the ETR this value is equal to ATB_DATA_WIDTH.	
		2	Memory interface databus is 32 bits wide. (ATB_DATA_WIDTH = 32bit).
		3	Memory interface databus is 64 bits wide. (ATB_DATA_WIDTH = 64bit).
		4	Memory interface databus is 128 bits wide. (ATB_DATA_WIDTH = 128bit).
7:6 (R/NW)	CONFIGTYPE	. Indicates the TMC configuration.	
		0	ETB - Embedded Trace Buffer.
		1	ETR - Embedded Trace Router.
		2	ETF - Embedded Trace FIFO.
		3	ETS - Embedded Trace Streamer.

Table 48-575: CSETR_DEVID Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/NW)	CLKSCHEME	. RAM Clocking Scheme. This value indicates the TMC RAM clocking scheme used, that is, whether the TMC RAM operates synchronously or asynchronously to the TMC clock. Fixed to 0 indicating that TMC RAM clock is synchronous to the clk input.
4:0 (R/NW)	ATBINPORTCOUNT	. Hidden Level of ATB input multiplexing. This value indicates the type/number of ATB multiplexing present on the input ATB. Fixed to 0x00 indicating that no multiplexing is present.

Device Configuration Register 1.

Contains an IMPLEMENTATION DEFINED value.

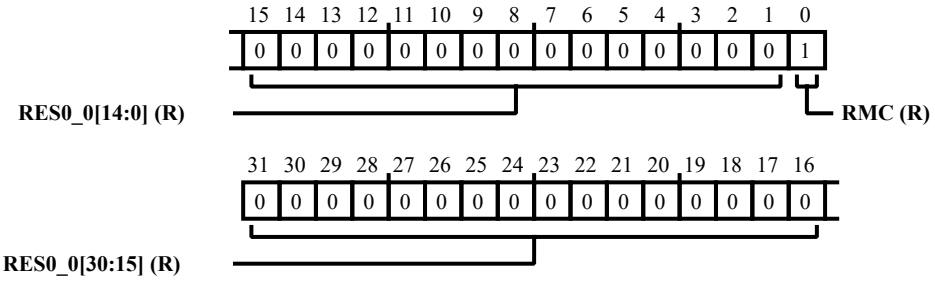


Figure 48-572: CSETR_DEVID1 Register Diagram

Table 48-576: CSETR_DEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
0 (R/NW)	RMC	. Register management mode. TMC implements register management mode 1.

Device Type Identifier Register.

A debugger can use this register to get information about a component that has an unrecognized Part number.

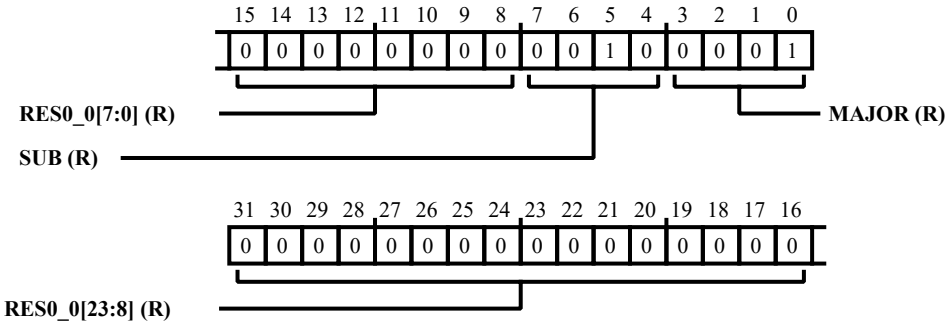


Figure 48-573: CSETR_DEVTYPE Register Diagram

Table 48-577: CSETR_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SUB	. Minor classification. Returns 0x2, indicating this component is a Buffer.
3:0 (R/NW)	MAJOR	. Major classification. Returns 0x1, indicating this component is a Trace Sink.

Formatter and Flush Control Register.

The FFCR controls the generation of stop, trigger and flush events. The insertion of a flush completion packet and the insertion of a trigger packet in the formatted trace is enabled here. Also one of the 2 formatter modes for bypass mode and normal mode can be changed here when the formatter has stopped.

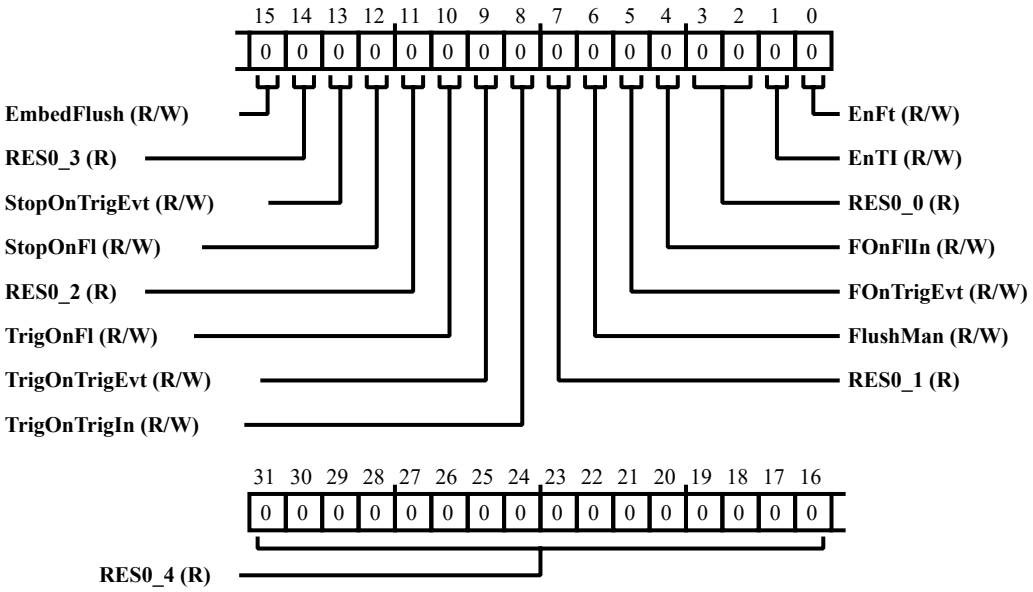


Figure 48-574: CSETR_FFCR Register Diagram

Table 48-578: CSETR_FFCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/NW)	RES0_4	. Reserved bit or field with SBZP behavior.	
15 (R/W)	EMBEDFLUSH	. Embed Flush ID (flush completion packet). Enables insertion of Flush ID 0x7B with a single byte of data payload = 0x00 in the output trace, immediately after the last flush data byte, when a flush completes on the ATB receiver interface. This bit is effective only in Normal formatting modes. In Bypass mode, the Flush ID insertion remains disabled and this bit is ignored.	
		0	Disable Flush ID insertion.
		1	Enable Flush ID insertion.
14 (R/NW)	RES0_3	. Reserved bit or field with SBZP behavior.	

Table 48-578: CSETR_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	STOPONTRIGEVT	. Stop On Trigger Event. If this bit is set, the formatter is stopped when a Trigger Event has been observed. This bit must be used only in CB mode. If trace capture is enabled in SWF1, or SWF2 mode with this bit set, it results in UNPREDICTABLE behavior.
12 (R/W)	STOPONFL	. Stop On Flush. If this bit is set, the formatter is stopped on completion of a flush operation. The initiation of a flush operation is controlled by programming the register bits FFCR.FlushMan, FFCR.FOnTrigEvt, and FFCR.FOnFlIn. When a flush-initiation condition occurs, afvalid_rx is asserted, and when the flush completion is received, that is, afready_rx=1, trace capture is stopped. Any remaining data in the formatter is appended with a post-amble and written to trace memory. The flush operation is then complete.
11 (R/NW)	RES0_2	. Reserved bit or field with SBZP behavior.
10 (R/W)	TRIGONFL	. Indicate on trace stream the completion of flush. If this bit is set, a trigger is indicated on the trace stream when afready_rx is received for a flush in progress. If this bit is clear, no triggers are embedded in the trace stream on flush completion. If Trigger Insertion is disabled, that is, FFCR.EnTI=0, then trigger indication on the trace stream is blocked regardless of the value that is programmed in this bit.
9 (R/W)	TRIGONTRIGEVT	. Indicate on trace stream the occurrence of a Trigger Event. If this bit is set, a trigger is indicated on the output trace stream when a Trigger Event occurs. If Trigger Insertion is disabled, that is, FFCR.EnTI=0, then trigger indication on the trace stream is blocked regardless of the value that is programmed in this bit. This bit must be used only in CB mode. If trace capture is enabled in SWF1, or SWF2 mode with this bit set, it results in UNPREDICTABLE behavior.
8 (R/W)	TRIGONTRIGIN	. Indicate on trace stream the occurrence of a rising edge on trigin. If this bit is set, a trigger is indicated on the trace stream when a rising edge is detected on the trigin input. If Trigger Insertion is disabled, that is, FFCR.EnTI=0, then trigger indication on the trace stream is blocked regardless of the value that is programmed in this bit.
7 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.

Table 48-578: CSETR_FFCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W)	FLUSHMAN	. Manually generate a flush of the system. Writing 1 to this bit causes a flush to be generated. This bit is cleared automatically when, in formatter bypass mode, afready_rx was sampled high, or, in normal formatting mode, afready_rx was sampled high and all flush data was output to the trace memory. If CTL.TraceCaptEn=0, writes to this bit are ignored.
5 (R/W)	FONTRIGEVT	. Flush on Trigger Event. If FFCR.StopOnTrigEvt is set, this bit is ignored. Setting this bit generates a flush when a Trigger Event occurs. If FFCR.StopOnTrigEvt is set, this bit is ignored. This bit must be used only in CB mode. If trace capture is enabled in SWF1, or SWF2 mode with this bit set, it results in UNPREDICTABLE behavior.
4 (R/W)	FONFLIN	. Setting this bit enables the detection of transitions on the flushin input by the TMC. If this bit is set and the formatter has not already stopped, a rising edge on flushin initiates a flush request.
3:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
1 (R/W)	ENTI	. Enable Trigger Insertion. Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 0x00 with atid_rx=0x7D in the trace stream. Trigger indication on the trace stream is also controlled by the register bits FFCR.TrigOnFl, FFCR.TrigOnTrigEvt, and FFCR.TrigOnTrigIn. This bit can only be changed when the TMC is in Disabled state. If FFCR.EnTI bit is set formatting is enabled.
0 (R/W)	ENFT	. Enable Formatter. If this bit is set, formatting is enabled. When EnTi is set, formatting is enabled. When CB mode is not used, formatting is also enabled. For backwards-compatibility with earlier versions of the ETB disabling of formatting is supported only in CB mode. This bit can only be changed when TMC is in Disabled state.

Formatter and Flush Status Register.

This register indicates the status of the Formatter, and the status of Flush request.

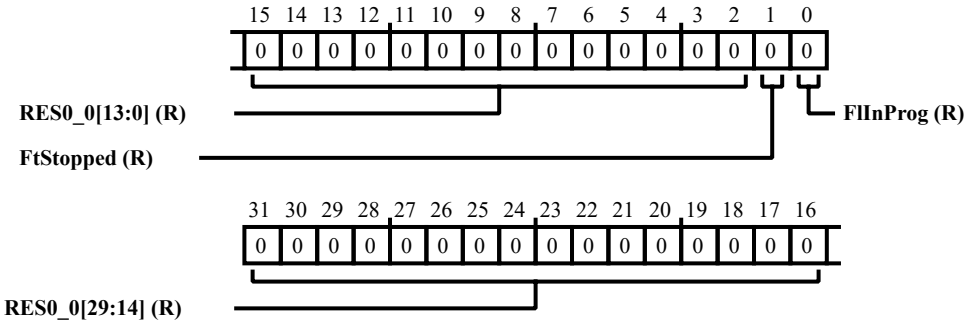


Figure 48-575: CSETR_FFSR Register Diagram

Table 48-579: CSETR_FFSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1 (R/NW)	FTSTOPPED	. Formatter Stopped. This bit behaves the same way as STS.FtEmpty.	
0 (R/NW)	FLINPROG	. Flush In Progress. This bit indicates whether the TMC is currently processing a flush request. The flush initiation is controlled by the flush control bits in the FFCR register. This bit is cleared to 0 when leaving the Disabled state and retains its value when entering the Disabled state. When in Disabled state, this bit is not updated.	
		0	No flush activity in progress.
		1	Flush in progress on the ATB receiver interface or the TMC internal pipeline.

Integration Test ATB Control 0 Register.

This register captures the values of ATB receiver inputs atvalid_rx, afready_rx, atwakeup_rx, and atbytes_rx in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, writes to this register are ignored and the reads return the value of corresponding input pins. The width of this register is given by: 8+log2(ATB DATA WIDTH/8).

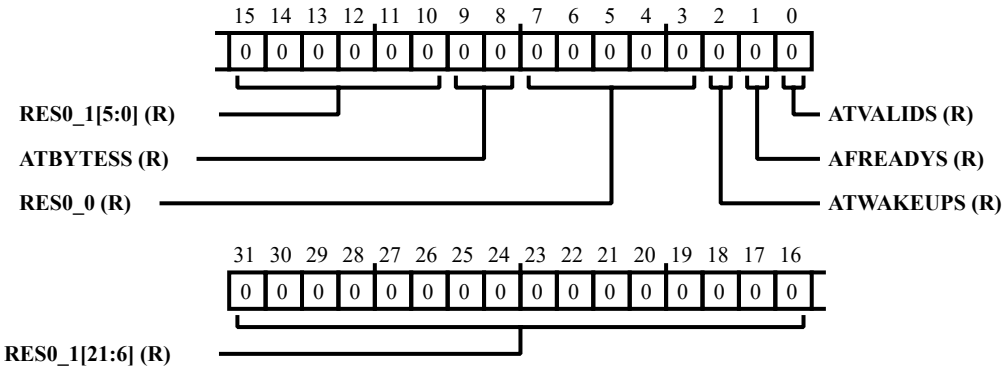


Figure 48-576: CSETR_ITATBCTR0 Register Diagram

Table 48-580: CSETR_ITATBCTR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:10 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.
9:8 (R/NW)	ATBYTESS	. Returns the value of atbytes_rx input in integration mode. N=8+log2(ATB DATA WIDTH/8).
7:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (R/NW)	ATWAKEUPS	. Returns the value of atwakeup_rx input in integration mode.
1 (R/NW)	AFREADYS	. Returns the value of afready_rx input in integration mode.
0 (R/NW)	ATVALIDS	. Returns the value of atvalid_rx input in integration mode.

Integration Test ATB Control 1 Register.

This register captures the value of the atid_rx[6:0] input in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, writes to this register are ignored and the reads return the value of atid_rx input.

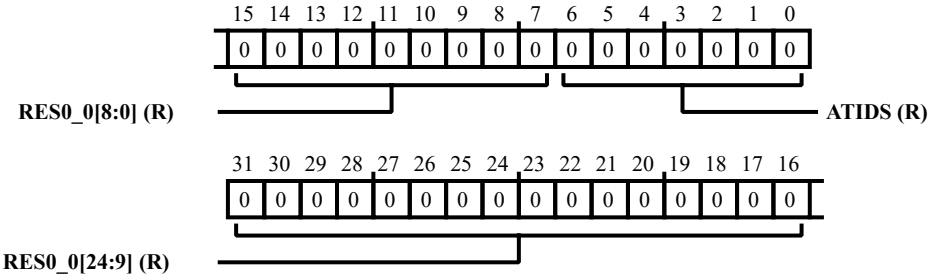


Figure 48-577: CSETR_ITATBCTR1 Register Diagram

Table 48-581: CSETR_ITATBCTR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
6:0 (R/NW)	ATIDS	. Returns the value of atid_rx[6:0] input in integration mode.

Integration Test ATB Control 2 Register.

This register enables control of ATB receiver outputs atready_rx, afvalid_rx, and syncreq_rx in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to any bit of this register is driven on the output pin that is controlled by that bit and the reads return 0x0.

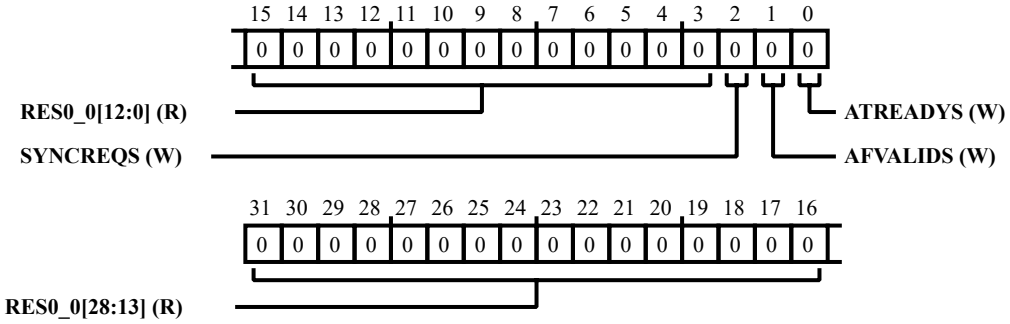


Figure 48-578: CSETR_ITATBCTR2 Register Diagram

Table 48-582: CSETR_ITATBCTR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
2 (RX/W)	SYNCREQS	. Controls the value of syncreq_rx output in integration mode.
1 (RX/W)	AFVALIDS	. Controls the value of afvalid_rx output in integration mode.
0 (RX/W)	ATREADYS	. Controls the value of atready_rx output in integration mode.

Integration Test ATB Data 0 Register.

This register captures the value of atdata_rx input in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, writes to this register are ignored and the reads return the value of corresponding atdata_rx bits. The width of this register is given by: 1+(ATB DATA WIDTH)/8.

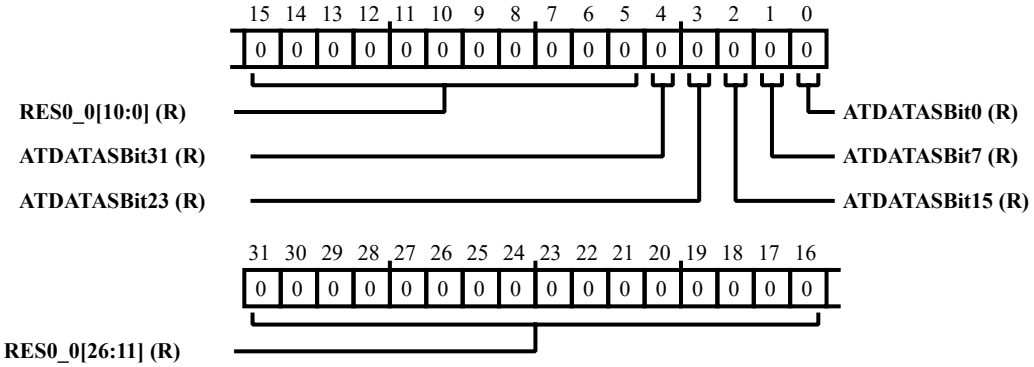


Figure 48-579: CSETR_ITATBDATA0 Register Diagram

Table 48-583: CSETR_ITATBDATA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
4 (R/NW)	ATDATASBIT31	. Returns the value of atdata_rx[31] input in integration mode.
3 (R/NW)	ATDATASBIT23	. Returns the value of atdata_rx[23] input in integration mode.
2 (R/NW)	ATDATASBIT15	. Returns the value of atdata_rx[15] input in integration mode.
1 (R/NW)	ATDATASBIT7	. Returns the value of atdata_rx[7] input in integration mode.
0 (R/NW)	ATDATASBIT0	. Returns the value of atdata_rx[0] input in integration mode.

Integration Mode Control Register.

The Integration Mode Control register is used to switch between functional mode and integration mode. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

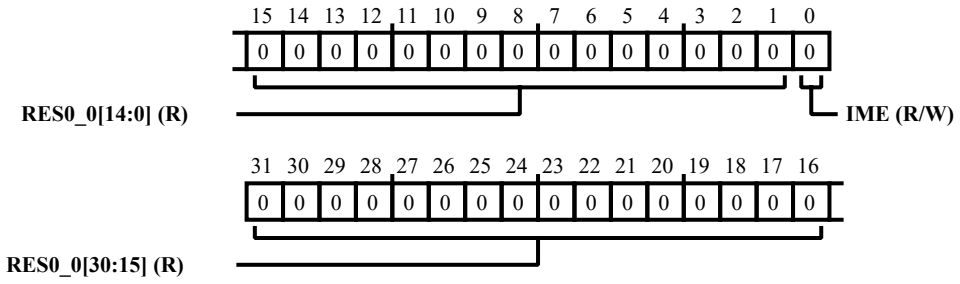


Figure 48-580: CSETR_ITCTRL Register Diagram

Table 48-584: CSETR_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	IME	. Integration Mode Enable.	
		0	The component must enter functional mode.
		1	The component must enter integration mode, and enable support for topology detection and integration testing.

Integration Test Event and Interrupt Control Register.

This register controls the values of event and interrupt outputs in integration mode. In functional mode, this register behaves as RAZ/WI. In integration mode, the value that is written to any bit of this register is driven on the output pin that is controlled by that bit and the reads return 0x0.

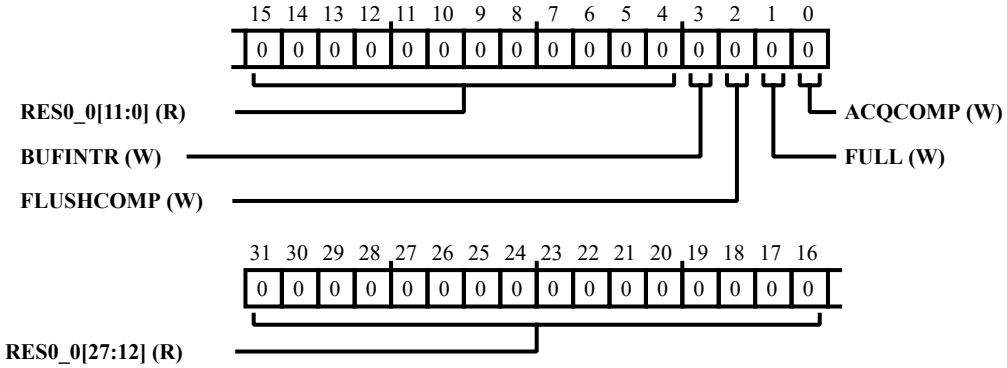


Figure 48-581: CSETR_ITEVTINTR Register Diagram

Table 48-585: CSETR_ITEVTINTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
3 (RX/W)	BUFINTR	. Controls the value of bufintr output in integration mode.
2 (RX/W)	FLUSHCOMP	. Controls the value of flushcomp output in integration mode.
1 (RX/W)	FULL	. Controls the value of full output in integration mode.
0 (RX/W)	ACQCOMP	. Controls the value of acqcomp output in integration mode.

Integration Test Trigger In and Flush In register.

This register captures the values of the flushin and trigin inputs in integration mode. In functional mode, this register behaves as RAZ/WI.

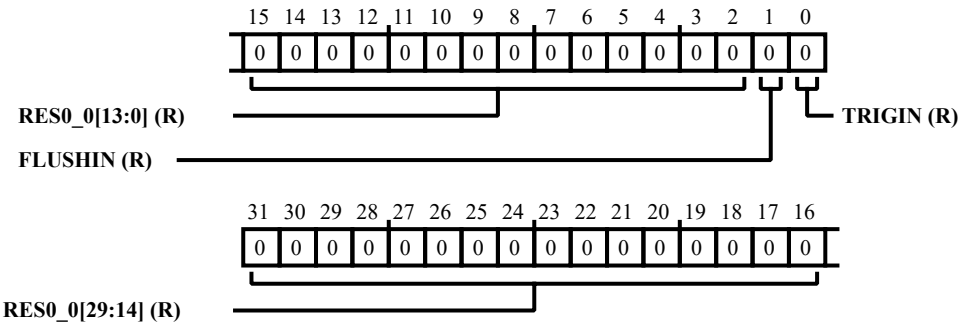


Figure 48-582: CSETR_ITTRFLIN Register Diagram

Table 48-586: CSETR_ITTRFLIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
1 (R/NW)	FLUSHIN	. Integration status of flushin input. In integration mode, this bit latches to 1 on a rising edge of the flushin input. It is cleared when the register is read or when integration mode is disabled.
0 (R/NW)	TRIGIN	. Integration status of trigin input. In integration mode, this bit latches to 1 on a rising edge of the trigin input. It is cleared when the register is read or when integration mode is disabled.

Latched Buffer Fill Level.

Reading the LBUFLEVEL register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level. When entering Disabled state, it retains its last value. While in Disabled state, reads from this register do not affect its value. When exiting Disabled state, the LBUFLEVEL register is updated to the current fill level.

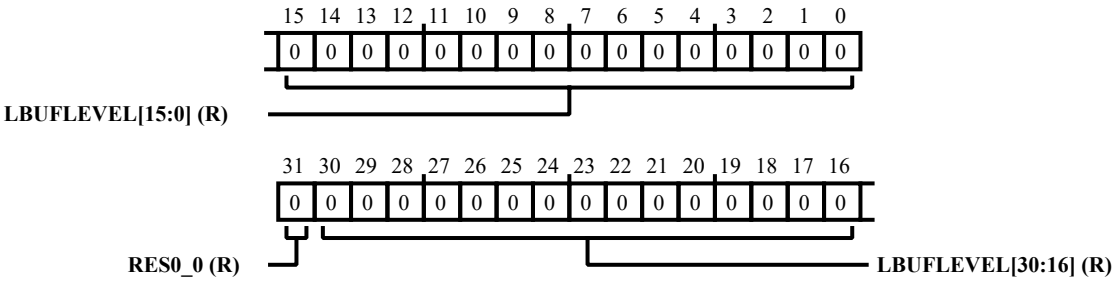


Figure 48-583: CSETR_LBUFLEVEL Register Diagram

Table 48-587: CSETR_LBUFLEVEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
30:0 (R/NW)	LBUFLEVEL	. Latched Buffer Fill Level. Indicates the maximum fill level of the trace memory in 32-bit words since this register was last read.

Mode register.

The MODE register controls the TMC operating mode. The operating mode can only be changed when the TMC is in Disabled state. Attempting to write to this register in any other state results in UNPREDICTABLE behavior. The operating mode is ignored when in Disabled state.

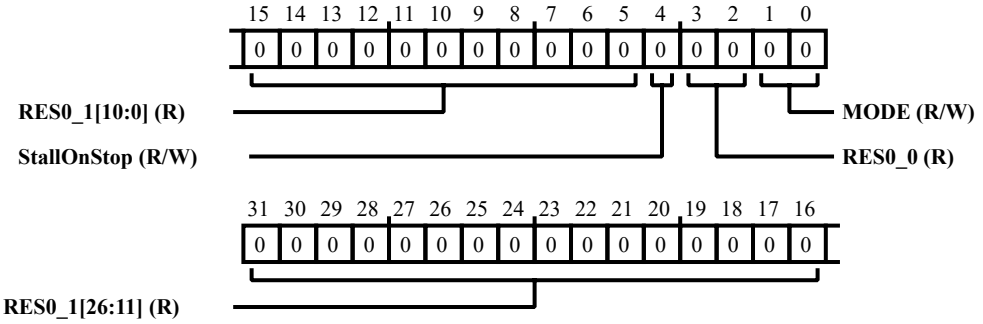


Figure 48-584: CSETR_MODE Register Diagram

Table 48-588: CSETR_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
4 (R/W)	STALLONSTOP	. Stall On Stop. If this bit is set and the formatter stops as a result of a stop event, the output atready_rx is de-asserted to stall the ATB interface and avoid loss of trace. If this bit is clear and the formatter stops as a result of a stop event, signal atready_rx remains asserted but the TMC discards further incoming trace.	
3:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1:0 (R/W)	MODE	. Selects the operating mode. If a reserved MODE value is programmed and trace capture is enabled, the TMC starts to operate in SWF1 mode. However, reading the MODE.MODE field returns the programmed value.	
		0	CB, Circular Buffer mode.
		1	SWF1, Software Read FIFO mode 1.
		2	Reserved. (SWF1).
		3	SWF2, Software Read FIFO mode 2.

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

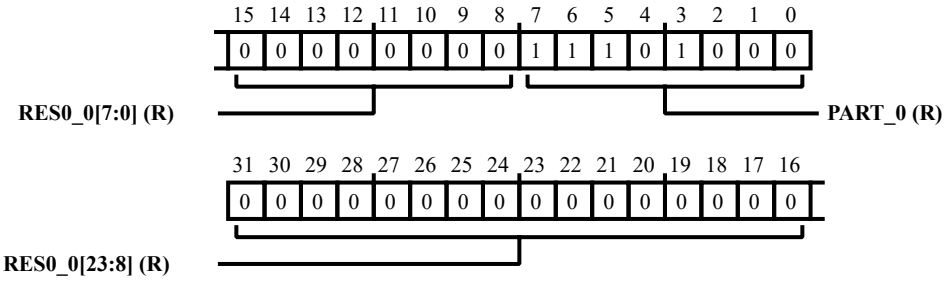


Figure 48-585: CSETR_PIDR0 Register Diagram

Table 48-589: CSETR_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:0 (R/NW)	PART_0	. Part number (lower 8 bits).	
		232	ETR or ETS - Embedded Trace Router or Streamer.
		233	ETB - Embedded Trace Buffer.
		234	ETF - Embedded Trace FIFO.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

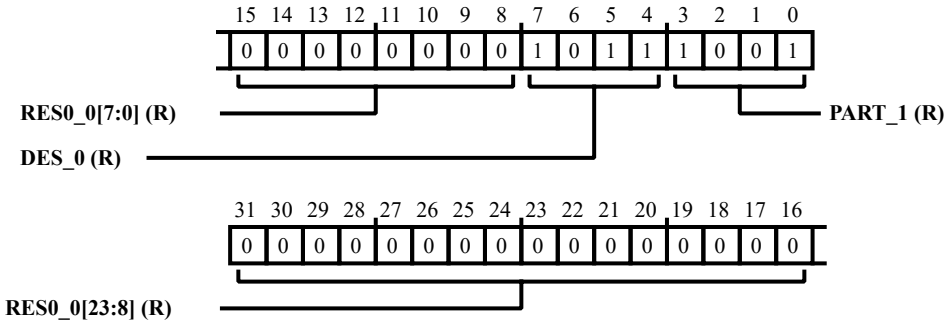


Figure 48-586: CSETR_PIDR1 Register Diagram

Table 48-590: CSETR_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

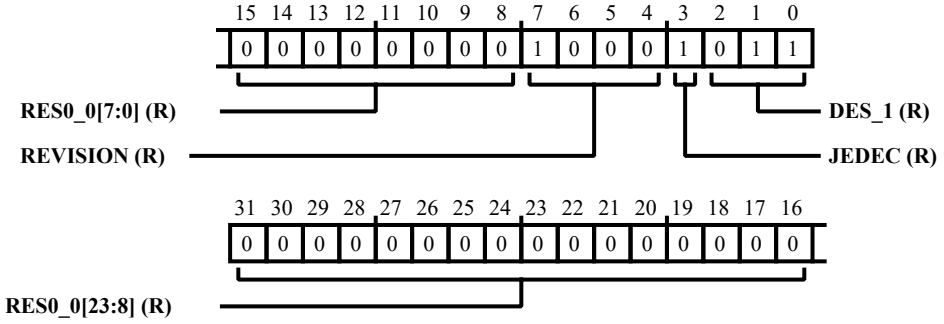


Figure 48-587: CSETR_PIDR2 Register Diagram

Table 48-591: CSETR_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

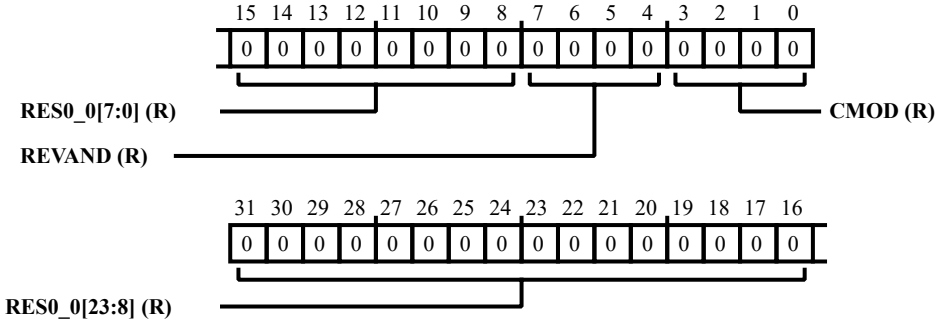


Figure 48-588: CSETR_PIDR3 Register Diagram

Table 48-592: CSETR_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

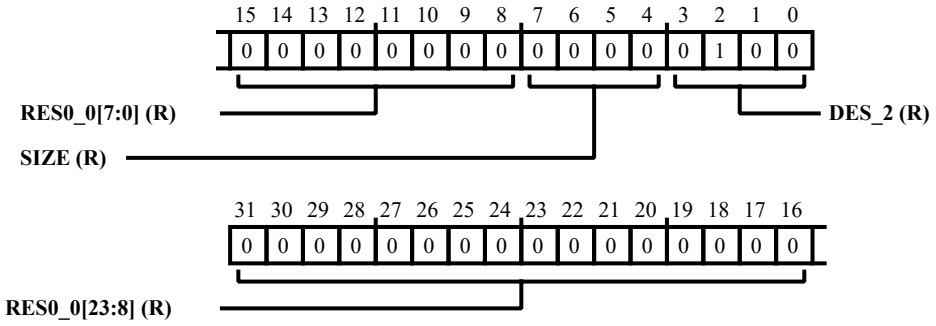


Figure 48-589: CSETR_PIDR4 Register Diagram

Table 48-593: CSETR_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

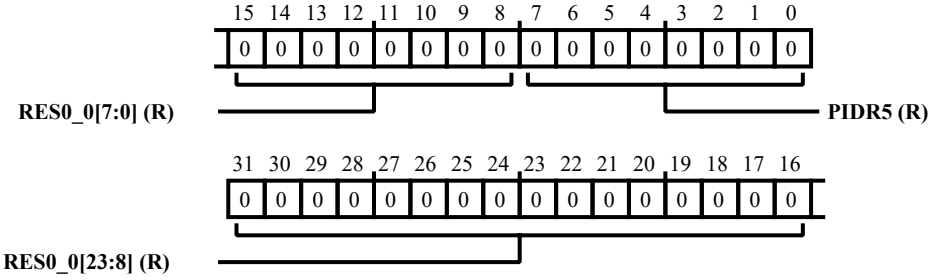


Figure 48-590: CSETR_PIDR5 Register Diagram

Table 48-594: CSETR_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

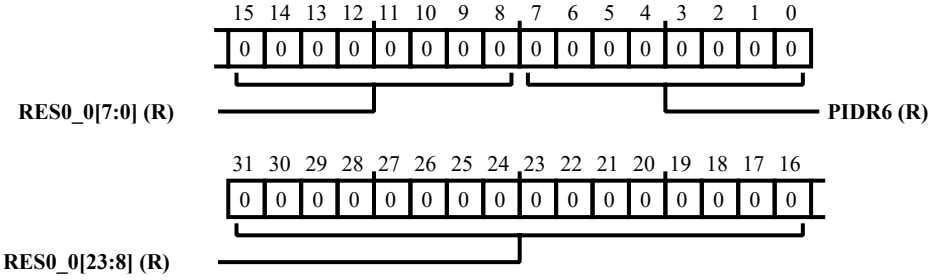


Figure 48-591: CSETR_PIDR6 Register Diagram

Table 48-595: CSETR_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

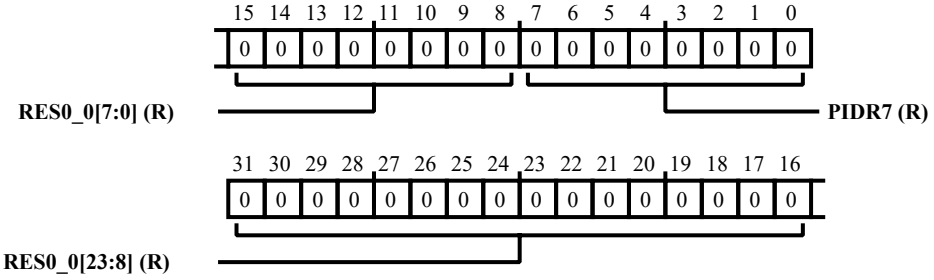


Figure 48-592: CSETR_PIDR7 Register Diagram

Table 48-596: CSETR_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

Periodic Synchronization Counter Register.

This register determines the reload value of the Periodic Synchronization Counter. This counter enables the frequency of sync packets to be optimized to the trace capture buffer size. The default behavior of the counter is to generate periodic synchronization requests, syncreq_rx, on the ATB receiver interface.

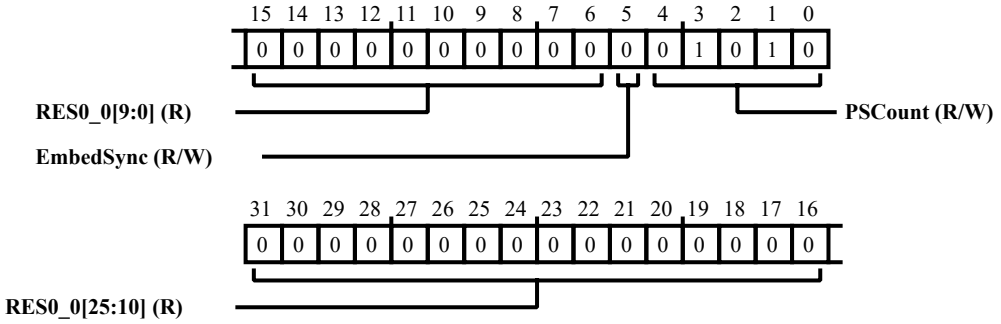


Figure 48-593: CSETR_PSCR Register Diagram

Table 48-597: CSETR_PSCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:6 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
5 (R/W)	EMBEDSYNC	. Embed Frame Sync Packet in the trace stream. Setting this bit to 1 enables the formatter to insert frame sync packets in the trace stream at periodic intervals. If this bit is set and the Synchronization Counter is enabled, the formatter inserts a 32-bit frame sync packet in the trace stream when the counter reaches 0. This bit is effective only when formatting is enabled, that is when FFCR.EnTI=1 or FFCR.EnFt=1, and it is ignored when the formatter is in bypass mode.
4:0 (R/W)	PSCOUNT	. Periodic Synchronization Count. Determines the reload value of the Synchronization Counter. The reload value takes effect the next time the counter reaches zero. When trace capture is enabled, the Synchronization Counter counts the number of bytes of trace data that is stored into the trace memory, regardless of whether the trace data has been formatted by the TMC or not, since the occurrence of the last sync request on the ATB receiver interface. When the counter reaches 0, a sync request is sent on the ATB receiver interface. Reads from this register return the reload value that is programmed in this register. This field resets to 0x0A, that is, the default sync period is 2^10 bytes. If a reserved value is programmed in this register field, the value 0x1B is used instead, and subsequent reads from this register also return 0x1B. The following constraints apply to the values written to the PSCount field: 0x0 - synchronization is disabled, 0x1-0x6 - reserved, 0x7-0x1B - synchronization period is 2^PSCount bytes. The smallest value 0x7 gives a sync period of 128 bytes. The maximum allowed value 0x1B gives a sync period of 2^27 bytes, 0x1C-0x1F - reserved.

RAM Read Data register.

Reading this register allows data to be read from the trace memory at the location pointed to by the RRP/RRPHI registers when either in the Disabled state or operating in CB or SWF1 mode. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the AXI data width is the same and a memory word holds 4, 8, or 16 bytes. When ATB_DATA_WIDTH is 64 or 128 bit wide multiple RRD reads must be performed to read a full memory word. When a full memory width of data has been read via the RRD register, the RRP register is incremented to the next memory word. When the TMC left the Disabled state and the trace memory is empty, this register returns 0xFFFFFFFF. When the TMC left the Disabled state and the trace memory is empty, this register returns 0xFFFFFFFF. When operating in CB mode and the TMC left the Disabled state, this register returns 0xFFFFFFFF in all other states except the Stopped state. When operating in SWF2 mode, this register also returns 0xFFFFFFFF. When the MemErr bit in the STS Register is set, reading this register returns an error response on the APB completer interface.

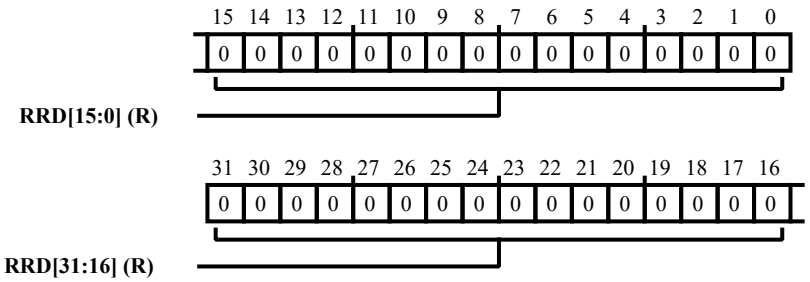


Figure 48-594: CSETR_RRD Register Diagram

Table 48-598: CSETR_RRD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RRD	. Returns the data read from trace memory.

RAM Read Pointer register.

The RRP register contains the value of the read pointer that is used to read entries from trace memory over the APB interface. Software must program it before enabling trace capture.

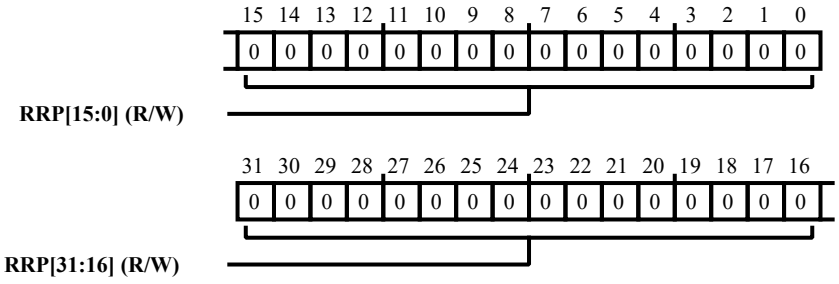


Figure 48-595: CSETR_RRP Register Diagram

Table 48-599: CSETR_RRP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RRP	. This register represents the lower 32 bits of the AXI address that is used to access the trace memory with RRD read acesses. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the AXI data width is the same and a memory word holds 4, 8, or 16 bytes. When a full memory width of data has been read via the RRD register, the RRP register is incremented to the next memory word. The lowest 2 bits have access type RAZ/WI when ATB_DATA_WIDTH is 32 bits. The lowest 3 bits have access type RAZ/WI when ATB_DATA_WIDTH is 64 bits. The lowest 4 bits have access type RAZ/WI when ATB_DATA_WIDTH is 128 bits.

RAM Read Pointer High register.

The RAM Read Pointer High register contains address bits >= bit[32] of the read pointer that is used to read entries from trace memory over the APB interface. Software must program it before enabling trace capture.

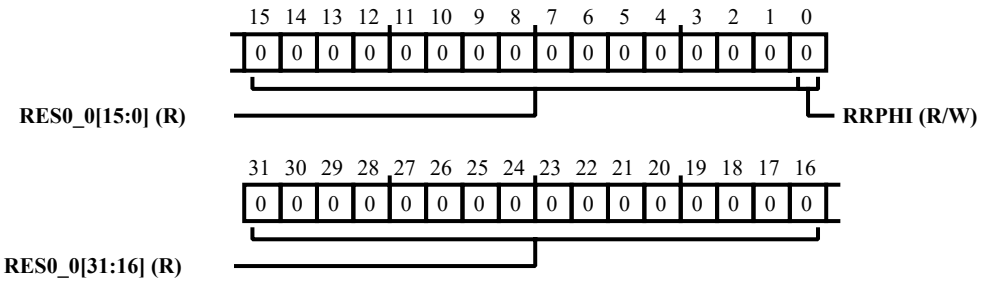


Figure 48-596: CSETR_RRPHI Register Diagram

Table 48-600: CSETR_RRPHI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
-1:0 (R/W)	RRPHI	. RAM Read Pointer High. Bits [32] and above of the RAM read pointer.
31:0 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.

RAM Size register.

The RSZ register defines the size of trace memory in units of 32-bit words.

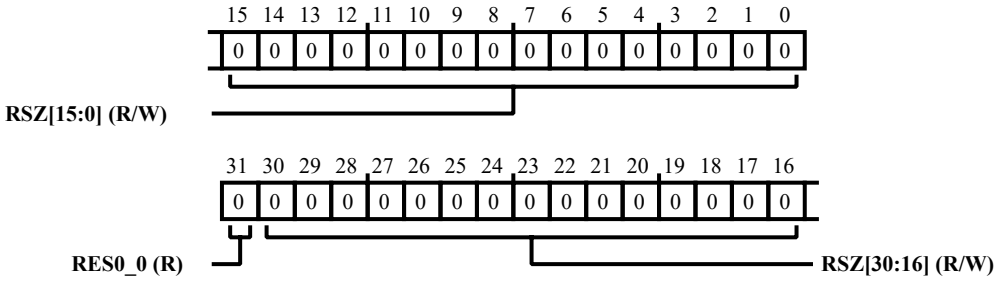


Figure 48-597: CSETR_RSZ Register Diagram

Table 48-601: CSETR_RSZ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
30:0 (R/W)	RSZ	. RAM size. Indicates the size of trace memory in 32-bit words. The size of the trace buffer must be a multiple of the AXI data width. This can be found by looking at the MEMWIDTH field in the DEVID Register. The maximum trace buffer size permitted is 4GB. The minimum trace buffer size enabled in Software FIFO mode and Hardware FIFO mode is 512 bytes. The minimum trace buffer size enabled in Circular Buffer mode is one AXI dataword. Returns the programmed value on reading. The burst length programmed in the AXICTL Register, WrBurstLen, must be compatible with the trace buffer size and the AXI data width so that the total number of bytes of data transferred in a burst is not greater than the trace buffer size. Programming an incompatible burst length results in UNPREDICTABLE behavior. Modifying this register when the TMCReady bit, STS Register, 0x00C is clear, or the TraceCaptEn bit, CTL Register, 0x020 is set, results in UNPREDICTABLE behavior. Arm recommends that DBA+RSZ*4<2^AXI_ADDR_WIDTH.

RAM Update Read Pointer register.

The RURP register enables software to inform the TMC of the amount of trace data that is extracted directly from system memory in SWF2 mode. Writes to this register are ignored when the TMC is in Disabled state or when not in SWF2 mode.

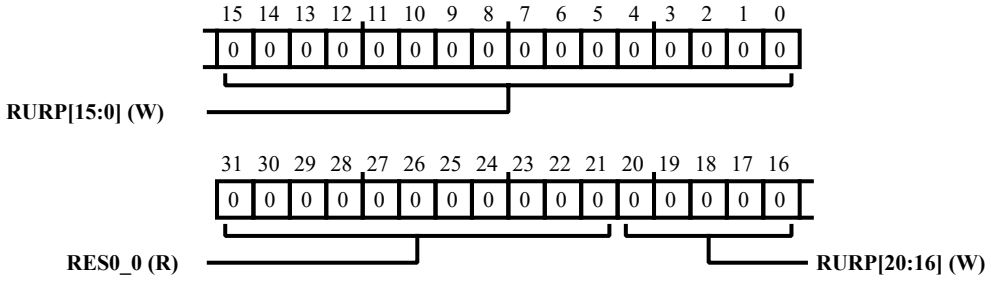


Figure 48-598: CSETR_RURP Register Diagram

Table 48-602: CSETR_RURP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:21 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
20:0 (RX/W)	RURP	. RAM Update Read Pointer. A write to the RURP register causes the TMC to update the RAM Read Pointer, both the RRP and RRPHI registers, based on the value that is written to it. RURP allows up to 1MB of data to be extracted in a single chunk. Reads always return 0x0. The following constraints apply to the write values: 0x000000 - no effect, 0x000010-0x100000 - increment RRP by this value, 0x100010-0x1FFFFF - reserved. The programmed value must also be aligned to the Trace Memory Data Width and the Frame Width. Programming an unaligned or reserved value results in UNPREDICTABLE behavior.

RAM Write Data register.

The RWD register enables testing of trace memory connectivity to the TMC. Writing this register allows data to be written to the trace memory at the location pointed to by the RWP/RWPHI registers when in the Disabled state. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the AXI data width is the same and a memory word holds 4, 8, or 16 bytes. When ATB_DATA_WIDTH is 64 or 128 bit wide multiple RWD writes must be performed to write a full memory word. When a full memory width of data has been written via the RWD register, the data is written to the trace memory and the RWP register is incremented to the next memory word. When the STS.MemErr bit is set, writing to this register returns an error response on the APB interface and the write data is discarded.

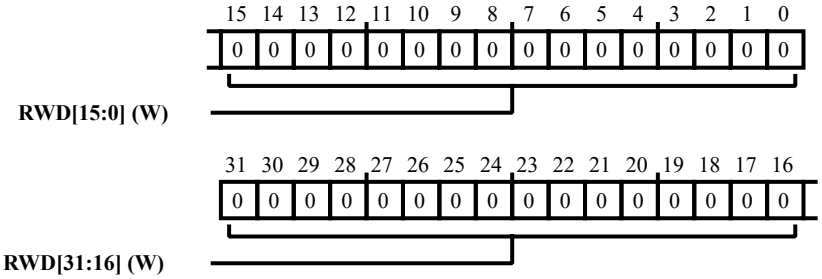


Figure 48-599: CSETR_RWD Register Diagram

Table 48-603: CSETR_RWD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	RWD	. Data written to this register is placed in the trace memory.

RAM Write Pointer register.

The RWP register sets the write pointer that writes entries into the trace memory. Software must program it before enabling trace capture.

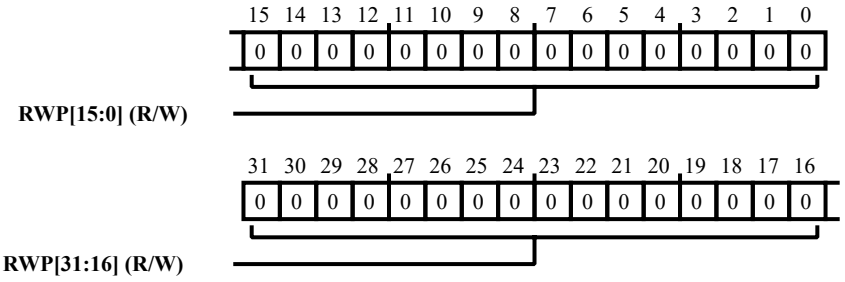


Figure 48-600: CSETR_RWP Register Diagram

Table 48-604: CSETR_RWP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RWP	. This register represents the lower 32 bits of the AXI address that is used to access the trace memory with RRD read acesses. When ATB_DATA_WIDTH is 32, 64 or 128 bit wide the AXI data width is the same and a memory word holds 4, 8, or 16 bytes. When a full memory width of data has been written via the RWD register, the RWP register is incremented to the next memory word. The lowest 2 bits have access type RAZ/WI when ATB_DATA_WIDTH is 32 bits. The lowest 3 bits have access type RAZ/WI when ATB_DATA_WIDTH is 64 bits. The lowest 4 bits have access type RAZ/WI when ATB_DATA_WIDTH is 128 bits. When leaving the Disabled state and RWP does not point inside the buffer space (starting with DBA and ending before DBA+RSZ*4) then STS.MemErr is set.

RAM Write Pointer High register.

The RAM Write Pointer High register sets bits >= bit[32] of the write pointer that is used to write entries into the trace memory. Software must program it before enabling trace capture.

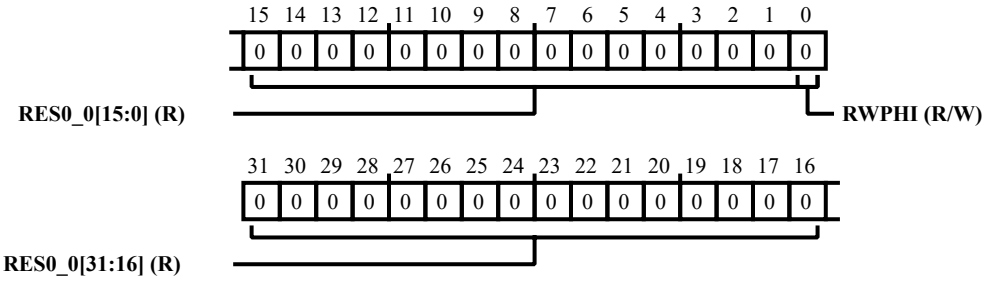


Figure 48-601: CSETR_RWPHI Register Diagram

Table 48-605: CSETR_RWPHI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
-1:0 (R/W)	RWPHI	. RAM Write Pointer High. Bits [32] and above of the RAM write pointer.
31:0 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.

Status register.

The STS register indicates the status of the Trace Memory Controller. After a reset, software must ignore all the fields of this register except STS.TMCReady. The other fields only have meaning when the TMC has left the Disabled state. Writes to all RO fields of this register are ignored.

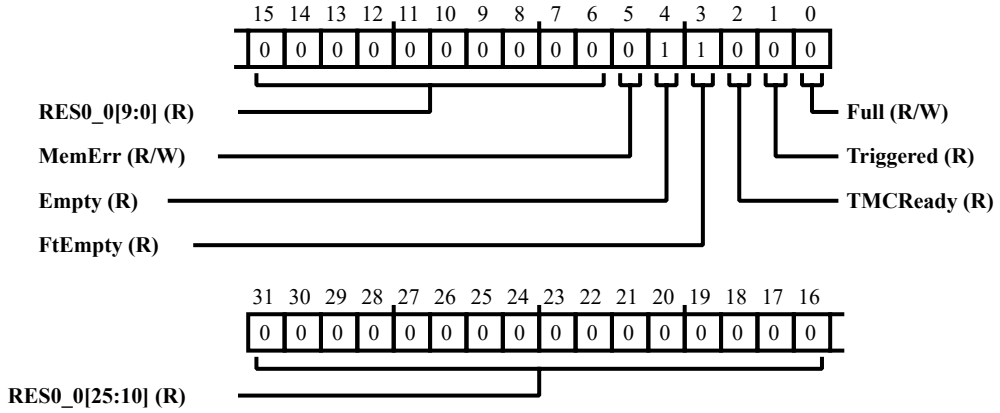


Figure 48-602: CSETR_STS Register Diagram

Table 48-606: CSETR_STS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:6 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
5 (R/W)	MEMERR	. Memory error status. This bit is set when an error has occurred on the AXI manager interface. The error can be caused by either an error response received from the connected AXI subordinate or attempted AXI transfers without proper authentication and Leaving the Disbled state and RWP does not point inside the buffer space (starting with DBA and ending before DBA+RSZ*4). Writes to this bit are allowed in Disabled or Stopped state. Writing 0 has no affect. Writing 1 clears this bit.
4 (R/NW)	EMPTY	. Trace buffer empty. If set, this bit indicates that the trace memory does not contain any valid trace data. The reset value of this bit is 1. On leaving Disabled state, this bit dynamically indicates the empty status of trace memory, CBUFLEVEL == 0. However, this does not mean that the pipeline stages within the TMC are empty. To determine whether the internal pipeline stages are empty, the software must read the STS.TMCReady bit. In Circular Buffer mode the Empty bit and the Full bit in this register can be 1 at the same time. This happens because the Full bit in this mode, when set, does not clear until TraceCaptEn is set.

Table 48-606: CSETR_STS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/NW)	FTEMPTY	. Trace capture has been completed and all captured trace data has been written to the trace memory, set in Stopped, Disabled or Draining state. Otherwise, it is cleared. The reset value is 1.
2 (R/NW)	TMCREADY	. Trace capture has been completed, all captured trace data has been written to the trace memory, and reading from trace memory completed as a result of final AXI write completing.
1 (R/NW)	TRIGGERED	. TMC triggered. This bit is set when trace capture is in progress and the TMC has detected a trigger event. This bit is cleared to 0 when leaving the Disabled state and retains its value when entering the Disabled state. A trigger event is when the TMC has written a set number of data words, as programmed in the TRG register, into the trace memory after a rising edge of trigin input, or a trigger packet (atid_rx = 0x7D) is received in the input trace.
0 (R/W)	FULL	. Trace memory full. This bit helps determine the amount of valid data present in the trace memory. In Circular Buffer mode writes to this bit are allowed in Disabled state, for example for clearing this bit. However, it is not affected by the reprogramming of pointer registers in this state. In Circular Buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer. It remains set until this bit is written 0 in Disabled state. In Software FIFO mode, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the BUFWM Register, that is, Fill level >= MEM_SIZE - BUFWM. The FULL output from the TMC reflects the value of this register bit, except when the Integration Mode bit in the ITCTRL Register, 0xF00, is set.

Trigger Counter register.

The TRG register, in Circular Buffer mode, specifies the number of 32-bit words to capture in the trace memory, after detecting either a rising edge on the trigin input or a trigger packet in the incoming trace stream, that is, where atid_rx = 0x7D. The value programmed must be aligned to the frame length of 128 bits. Software must program this register before leaving Disabled state.

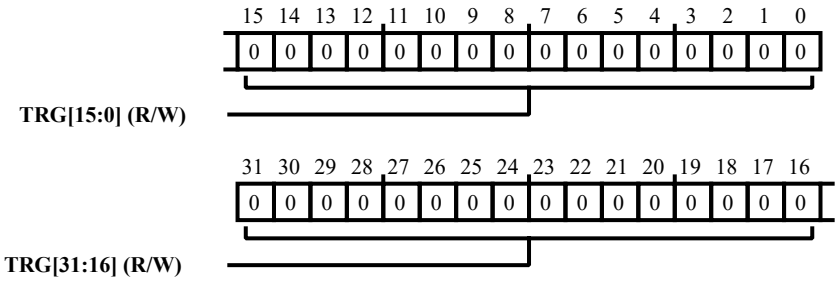


Figure 48-603: CSETR_TRG Register Diagram

Table 48-607: CSETR_TRG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TRG	. Trigger count. This count represents the number of 32-bit words of trace that are captured between a trigger packet and a trigger event. The lowest two bits have access type RAZ/WI.

ADSP-2184x CSTF Register Descriptions

Your module description, here. (CSTF) contains the following registers.

Table 48-608: ADSP-2184x CSTF Register List

Name	Description
CSTF_AUTHSTATUS	Authentication Status Register.
CSTF_CIDR0	Component Identification Register 0.
CSTF_CIDR1	Component Identification Register 1.
CSTF_CIDR2	Component Identification Register 2.
CSTF_CIDR3	Component Identification Register 3.
CSTF_CLAIMCLR	Claim Tag Clear Register.
CSTF_CLAIMSET	Claim Tag Set Register.
CSTF_DEVAFF0	Device Affinity register 0.
CSTF_DEVAFF1	Device Affinity register 1.

Table 48-608: ADSP-2184x CSTF Register List (Continued)

Name	Description
CSTF_DEVARCH	Device Architecture Register.
CSTF_DEVID	Device Configuration Register.
CSTF_DEVID1	Device Configuration Register 1.
CSTF_DEVID2	Device Configuration Register 2.
CSTF_DEVTYPE	Device Type Identifier Register.
CSTF_FUNNELCONTROL	Funnel Control register.
CSTF_ITATBCTR0	Integration test control register 0.
CSTF_ITATBCTR1	Integration test control register 1.
CSTF_ITATBCTR2	Integration test control register 2.
CSTF_ITATBCTR3	Integration test control register 3.
CSTF_ITATBDATA0	Integration test data register.
CSTF_ITCTRL	Integration Mode Control Register.
CSTF_PIDR0	Peripheral Identification Register 0.
CSTF_PIDR1	Peripheral Identification Register 1.
CSTF_PIDR2	Peripheral Identification Register 2.
CSTF_PIDR3	Peripheral Identification Register 3.
CSTF_PIDR4	Peripheral Identification Register 4.
CSTF_PIDR5	Peripheral Identification Register 5.
CSTF_PIDR6	Peripheral Identification Register 6.
CSTF_PIDR7	Peripheral Identification Register 7.
CSTF_PRIORITYCONTROL	Priority Control register.

Authentication Status Register.

Reports the current status of the authentication control signals.

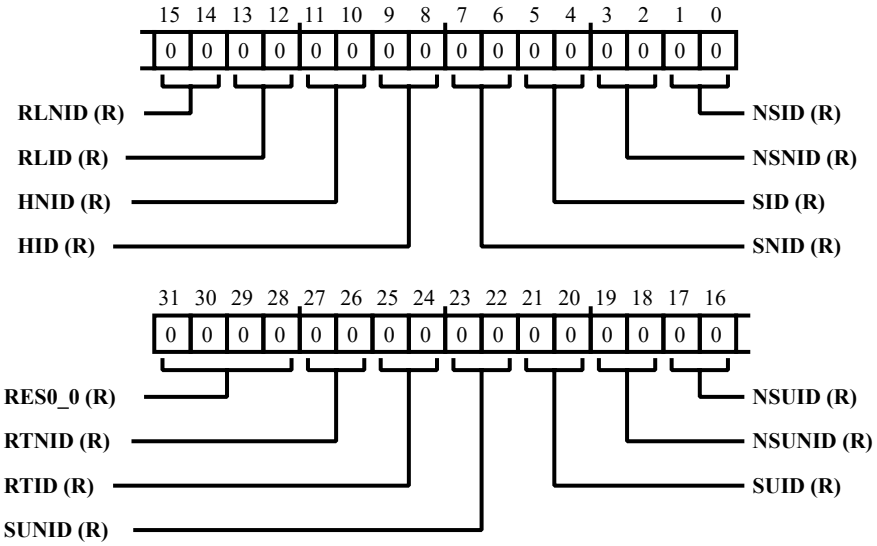


Figure 48-604: CSTF_AUTHSTATUS Register Diagram

Table 48-609: CSTF_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
27:26 (R/NW)	RTNID	. Root non-invasive debug.	
		0	Separate Root non-invasive debug enable not implemented or Root state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.

Table 48-609: CSTF_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/NW)	RTID	. Root invasive debug.	
		0	Separate Root invasive debug enable not implemented or Root state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
23:22 (R/NW)	SUNID	. Secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
21:20 (R/NW)	SUID	. Secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
19:18 (R/NW)	NSUNID	. Non-secure unprivileged non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
17:16 (R/NW)	NSUID	. Non-secure unprivileged invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-609: CSTF_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/NW)	RLNID	. Realm non-invasive debug.	
		0	Separate Realm non-invasive debug enable not implemented or Realm state non-invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
13:12 (R/NW)	RLID	. Realm invasive debug.	
		0	Separate Realm invasive debug enable not implemented or Realm state invasive debug features not implemented.
		1	Reserved.
		2	Implemented and disabled.
		3	Implemented and enabled.
11:10 (R/NW)	HNID	. Hypervisor non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
9:8 (R/NW)	HID	. Hypervisor invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Table 48-609: CSTF_AUTHSTATUS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	SNID	. Secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
5:4 (R/NW)	SID	. Secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
3:2 (R/NW)	NSNID	. Non-secure non-invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.
1:0 (R/NW)	NSID	. Non-secure invasive debug.	
		0	Debug level is not supported.
		1	Reserved.
		2	Supported and disabled.
		3	Supported and enabled.

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

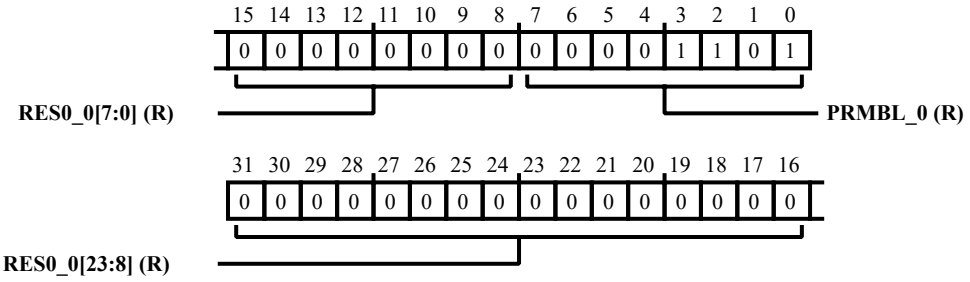


Figure 48-605: CSTF_CIDR0 Register Diagram

Table 48-610: CSTF_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

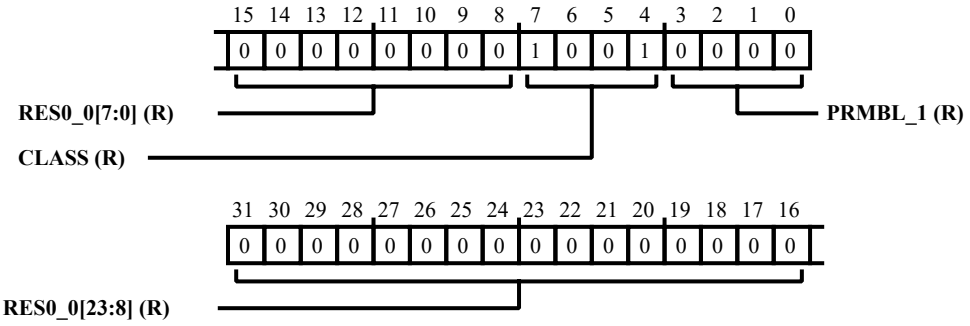


Figure 48-606: CSTF_CIDR1 Register Diagram

Table 48-611: CSTF_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

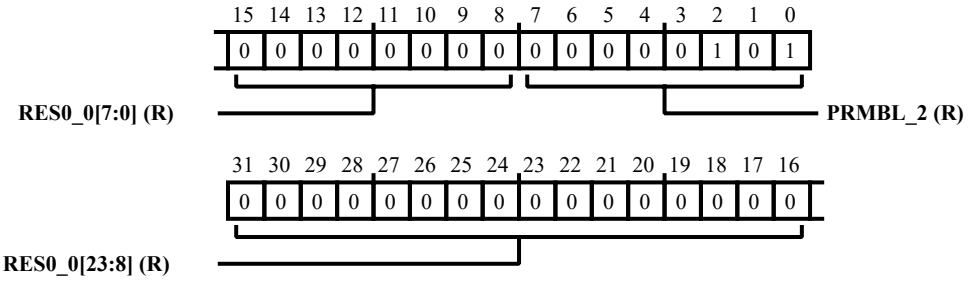


Figure 48-607: CSTF_CIDR2 Register Diagram

Table 48-612: CSTF_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

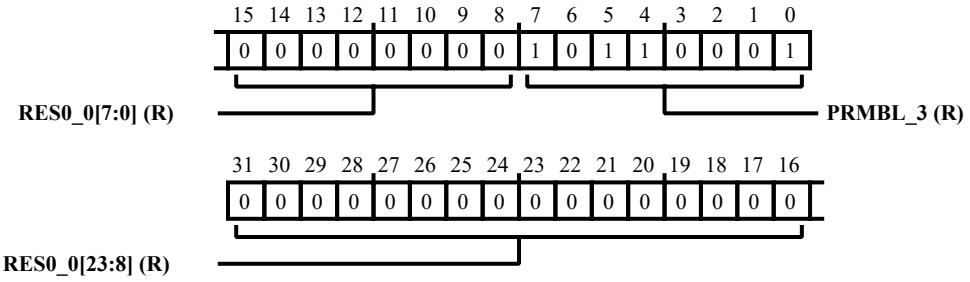


Figure 48-608: CSTF_CIDR3 Register Diagram

Table 48-613: CSTF_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Claim Tag Clear Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

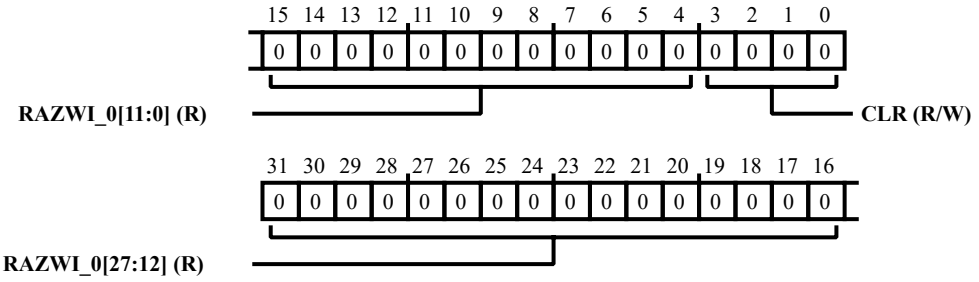


Figure 48-609: CSTF_CLAIMCLR Register Diagram

Table 48-614: CSTF_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	CLR	. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register.

This register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

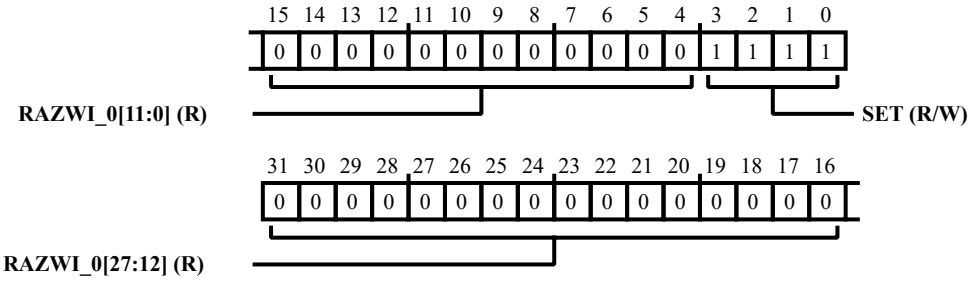


Figure 48-610: CSTF_CLAIMSET Register Diagram

Table 48-615: CSTF_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/NW)	RAZWI_0	. RAZ/WI.
3:0 (R/W)	SET	. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

Device Affinity register 0.

Enables a debugger to determine if two components have an affinity with each other.

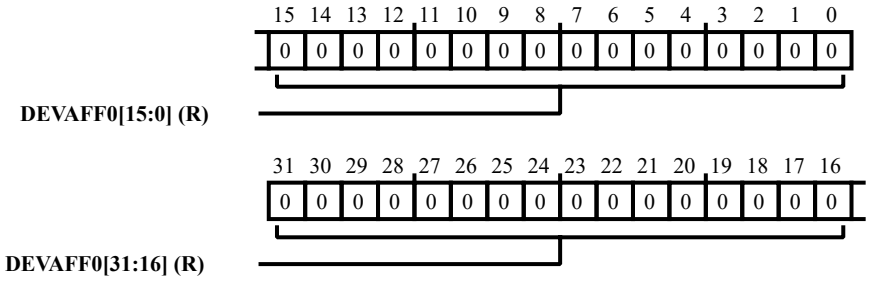


Figure 48-611: CSTF_DEVAFF0 Register Diagram

Table 48-616: CSTF_DEVAFF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVAFF0	. This field is RAZ.

Device Affinity register 1.

Enables a debugger to determine if two components have an affinity with each other.

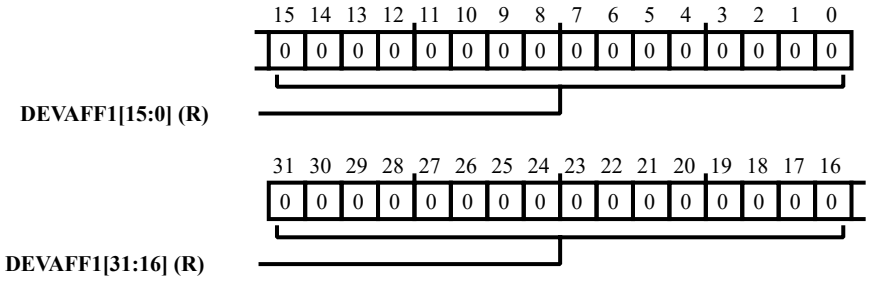


Figure 48-612: CSTF_DEVAFF1 Register Diagram

Table 48-617: CSTF_DEVAFF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVAFF1	. This field is RAZ.

Device Architecture Register.

Identifies the architect and architecture of a CoreSight component. The architect might differ from the designer of a component, for example Arm defines the architecture but another company designs and implements the component.

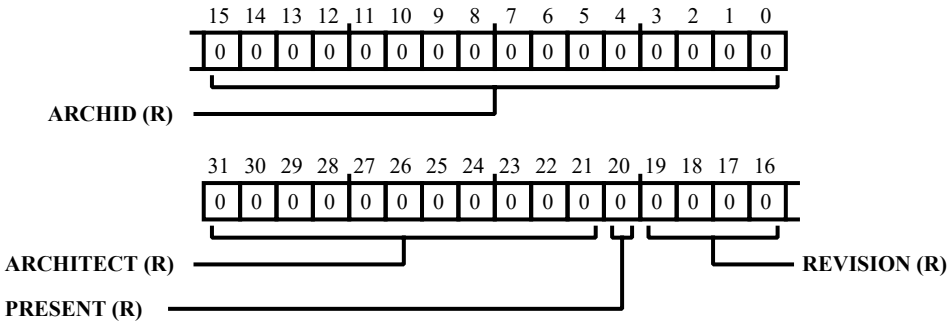


Figure 48-613: CSTF_DEVARCH Register Diagram

Table 48-618: CSTF_DEVARCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:21 (R/NW)	ARCHITECT	. Defines the architect of the component.	
		571	Arm.
20 (R/NW)	PRESENT	. Indicates the presence of this register.	
		0	DEVARCH is not present.
		1	DEVARCH is present.
19:16 (R/NW)	REVISION	. Architecture revision. Returns the revision of the architecture that the ARCHID field specifies.	
15:0 (R/NW)	ARCHID	. Architecture ID. Returns a value that identifies the architecture of the component.	
		2583	Memory Access Port v2 architecture.
		2599	JTAG Access Port v2 architecture.
		2631	Unknown Access Port v2 architecture.
		2807	CoreSight ROM architecture.
		6676	CTI architecture.

Device Configuration Register.

This register is IMPLEMENTATION DEFINED for each Part Number and Designer. The register indicates the capabilities of the component.

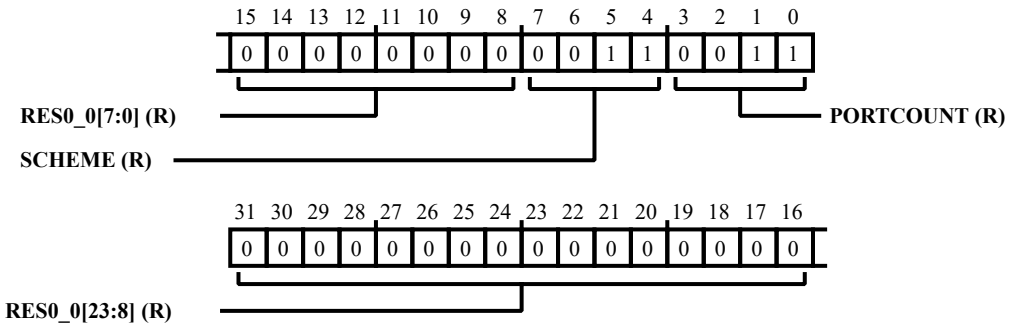


Figure 48-614: CSTF_DEVID Register Diagram

Table 48-619: CSTF_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SCHEME	. Indicates priority scheme implemented. Input priority is controlled by the PRIORITYCONTROL register.
3:0 (R/NW)	PORTCOUNT	. Indicates the number of input ports connected.

Device Configuration Register 1.

Contains an IMPLEMENTATION DEFINED value.

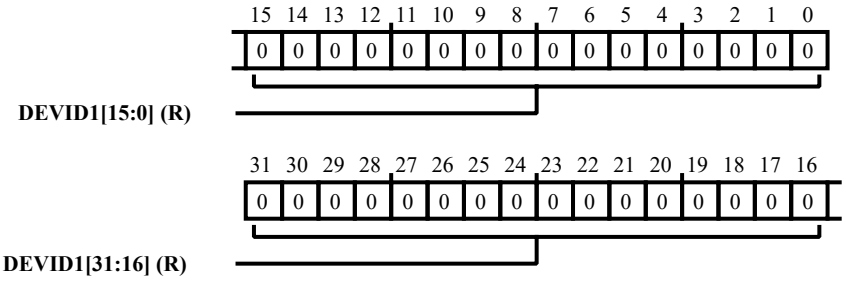


Figure 48-615: CSTF_DEVID1 Register Diagram

Table 48-620: CSTF_DEVID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVID1	. This field is RAZ.

Device Configuration Register 2.

Contains an IMPLEMENTATION DEFINED value.

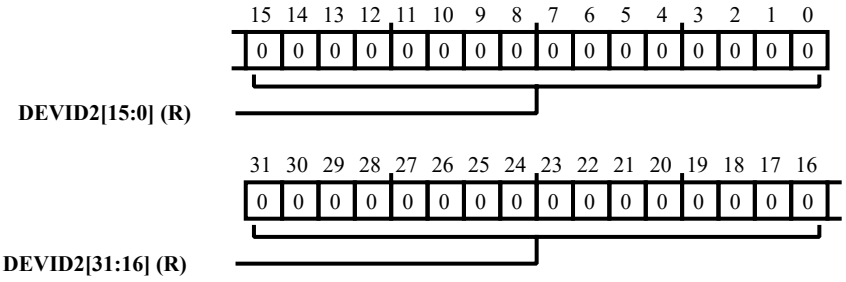


Figure 48-616: CSTF_DEVID2 Register Diagram

Table 48-621: CSTF_DEVID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DEVID2	. This field is RAZ.

Device Type Identifier Register.

A debugger can use this register to get information about a component that has an unrecognized Part number.

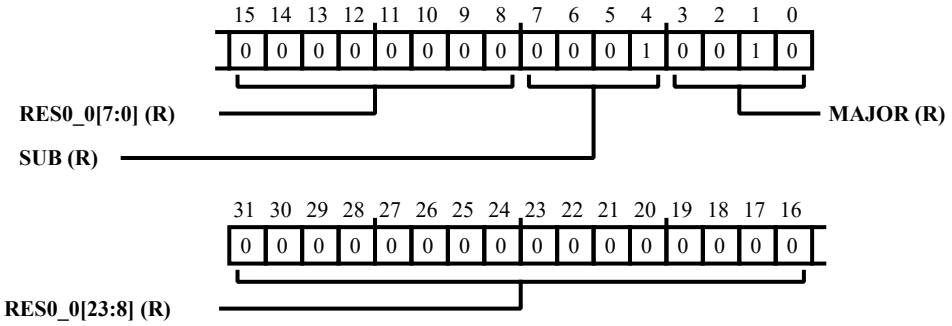


Figure 48-617: CSTF_DEVTYPE Register Diagram

Table 48-622: CSTF_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SUB	. Minor classification. Returns 0x1, indicating this component is a Funnel/Router.
3:0 (R/NW)	MAJOR	. Major classification. Returns 0x2, indicating this component is a Trace Link.

Funnel Control register.

The FUNNELCONTROL register enables each of the trace sources and controls the hold time for switching between them.

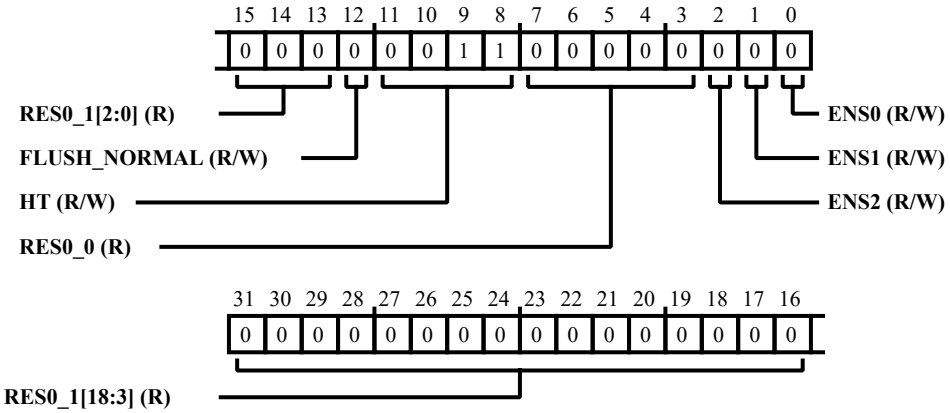


Figure 48-618: CSTF_FUNNELCONTROL Register Diagram

Table 48-623: CSTF_FUNNELCONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:13 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
12 (R/W)	FLUSH_NORMAL	. This bit, when clear, allows receiver ports that are already flushed to receive further data even if there are other ports that have not completed flush. If set, a port that has completed flush is not be allowed to receive further data until all ports have completed flush.	
11:8 (R/W)	HT	. Hold time. Value sets the minimum hold time before switching trace sources (funnel inputs) based on the ID. Value used is programmed value + 1.	
		0	1 transaction hold time.
		1	2 transactions hold time.
		2	3 transactions hold time.
		3	4 transactions hold time.
		4	5 transactions hold time.
		5	6 transactions hold time.
		6	7 transactions hold time.
		7	8 transactions hold time.

Table 48-623: CSTF_FUNNELCONTROL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		8	9 transactions hold time.
		9	10 transactions hold time.
		10	11 transactions hold time.
		11	12 transactions hold time.
		12	13 transactions hold time.
		13	14 transactions hold time.
		14	15 transactions hold time.
		15	Reserved.
7:3 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
2 (R/W)	ENS2	. Enable receiver interface 2. Field is RES0 if receiver interface 2 is not implemented.	
		0	Receiver interface disabled.
		1	Receiver interface enabled.
1 (R/W)	ENS1	. Enable receiver interface 1.	
		0	Receiver interface disabled.
		1	Receiver interface enabled.
0 (R/W)	ENS0	. Enable receiver interface 0.	
		0	Receiver interface disabled.
		1	Receiver interface enabled.

Integration test control register 0.

The ITATBCTR0 register enables you to observabe and control the ATBYTES buses, and the AFREADY and ATVALID signals into and out of the funnel. For receiver signals coming into the funnel, the register views the ports that are selected through the FUNNELCONTROL register. Only one port must be selected for integration test.

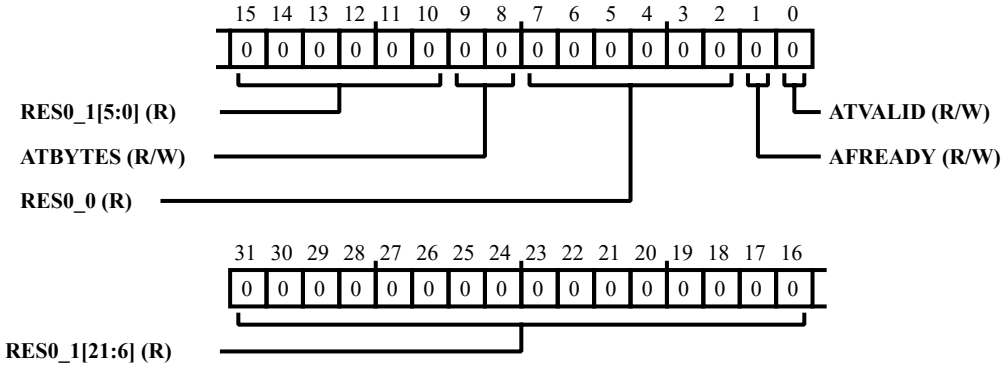


Figure 48-619: CSTF_ITATBCTR0 Register Diagram

Table 48-624: CSTF_ITATBCTR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:10 (R/NW)	RES0_1	. Reserved bit or field with SBZP behavior.	
9:8 (R/W)	ATBYTES	. Reads the value on atbytes_rx[1:0] and writes the values on atbytes_tx[1:0].	
7:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1 (R/W)	AFREADY	. Reads and controls the afready signals into, and out of, the funnel:.	
		0	On reads: afready_rx is LOW. On writes: sets afready_tx LOW.
		1	On reads: afready_rx is HIGH. On writes: sets afready_tx HIGH.
0 (R/W)	ATVALID	. Reads and controls the atvalid signals into, and out of, the funnel:.	
		0	On reads: atvalid_rx is LOW. On writes: sets atvalid_tx LOW.
		1	On reads: atvalid_rx is HIGH. On writes: sets atvalid_tx HIGH.

Integration test control register 1.

The ITATBCTR1 register enables you to observe and control the ATID buses into and out of the funnel. For receiver signals coming into the funnel, the register views the ports that are selected through the FUNNELCONTROL register. Only one port must be selected for integration test.

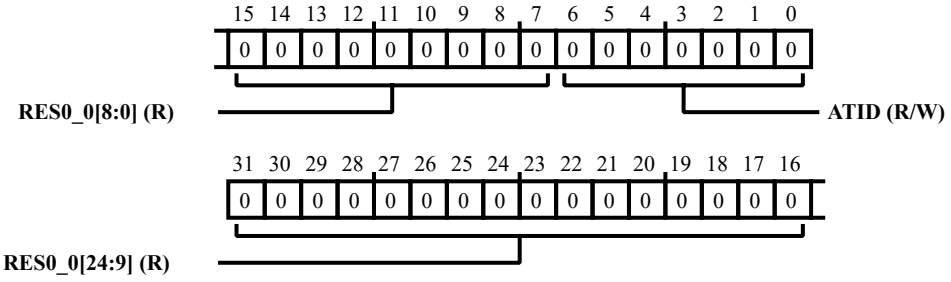


Figure 48-620: CSTF_ITATBCTR1 Register Diagram

Table 48-625: CSTF_ITATBCTR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
6:0 (R/W)	ATID	. When read returns the value on atid_rx, when written drives the value on atid_tx.

Integration test control register 2.

The ITATBCTR2 register enables you to observe and control the afvalid and atready signals into and out of the funnel. For receiver signals coming into the funnel, the register views the ports that are selected through the FUNNELCONTROL register. Only one port must be selected for integration test.

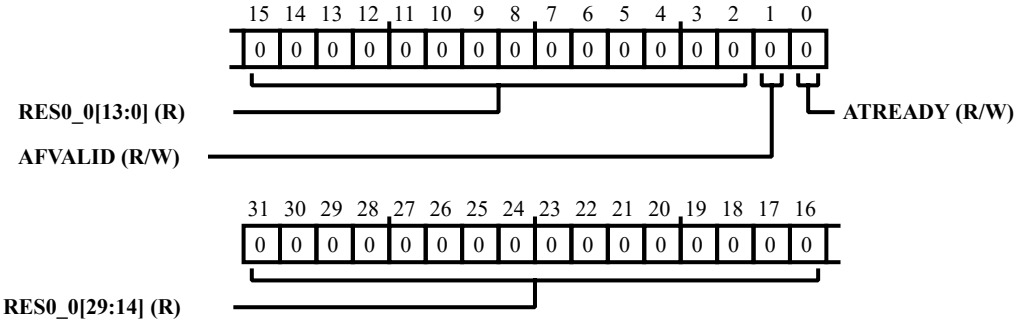


Figure 48-621: CSTF_ITATBCTR2 Register Diagram

Table 48-626: CSTF_ITATBCTR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
1 (R/W)	AFVALID	. Reads and controls the afvalid signals into, and out of, the funnel:.	
		0	On reads: afvalid_tx is LOW. On writes: sets afvalid_rx LOW.
		1	On reads: afvalid_tx is HIGH. On writes: sets afvalid_rx HIGH.
0 (R/W)	ATREADY	. Reads and controls the atready signal into, and out of, the funnel:.	
		0	On reads: atready_tx is LOW. On writes: sets atready_rx LOW.
		1	On reads: atready_tx is HIGH. On writes: sets atready_rx HIGH.

Integration test control register 3.

The ITATBCTR3 register enables you to observe and control the SYNCREQ signals into and out of the funnel. Only one receiver interface must be selected for integration test. The syncreq receiver on the transmitter interface has a latching function to capture a pulse arriving on that input. The arrival of a pulse sets the latch so that the value can be read. Reading the register clears the latch. Reading a 1 indicates that a syncreq_tx pulse arrived since the last read. Reading a 0 indicates that no syncreq_tx pulse has arrived. Writing a 1 to the register causes a syncreq_rx pulse to be generated to the upstream component.

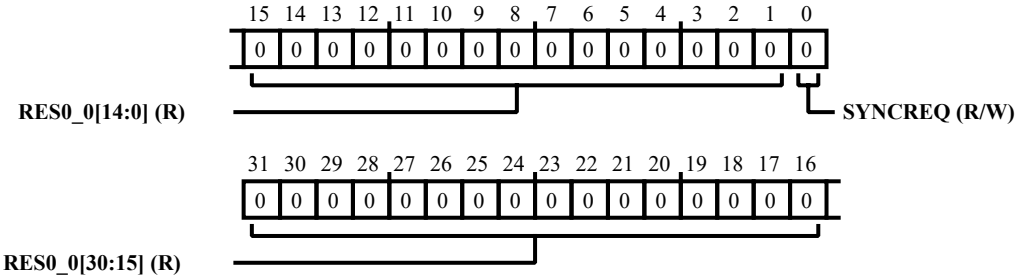


Figure 48-622: CSTF_ITATBCTR3 Register Diagram

Table 48-627: CSTF_ITATBCTR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	SYNCREQ	. Reads and controls the SYNCREQ signals into, and out of, the funnel. Reading clears the latch.	
		0	On reads: no syncreq_tx pulse has arrived. On writes: no effect.
		1	On reads: a syncreq_tx pulse arrived since the last read. On writes: generates a syncreq_rx pulse to the upstream component.

Integration test data register.

The ITATBDATA0 register allows observability and controllability of the ATDATA buses into and out of the funnel. For receiver signals coming into the funnel, the register views the ports that are selected through the funnel control register. Only one port must be selected for integration test.

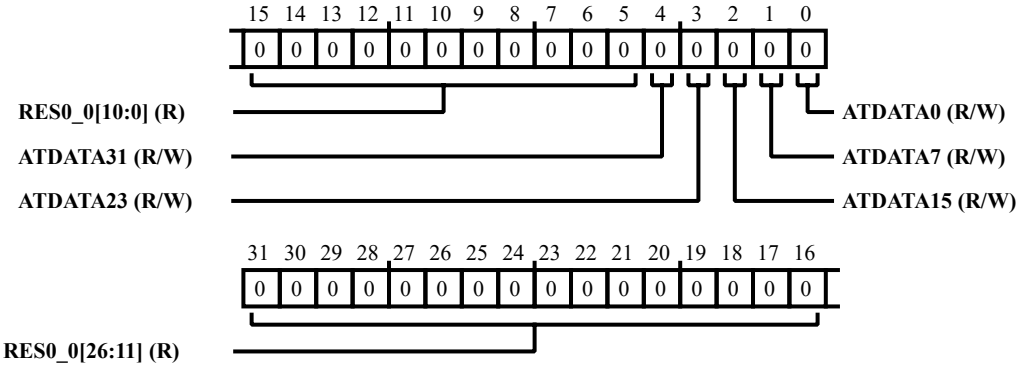


Figure 48-623: CSTF_ITATBDATA0 Register Diagram

Table 48-628: CSTF_ITATBDATA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
4 (R/W)	ATDATA31	. Reads atdata_rx[31] and writes atdata_tx[31]. RES0 if ATB_DATA_WIDTH < 32.	
		0	On reads, the value of atdata_rx[31] is 0. On writes, sets atdata_tx[31] to 0.
		1	On reads, the value of atdata_rx[31] is 1. On writes, sets atdata_tx[31] to 1.
3 (R/W)	ATDATA23	. Reads atdata_rx[23] and writes atdata_tx[23]. RES0 if ATB_DATA_WIDTH < 32.	
		0	On reads, the value of atdata_rx[23] is 0. On writes, sets atdata_tx[23] to 0.
		1	On reads, the value of atdata_rx[23] is 1. On writes, sets atdata_tx[23] to 1.

Table 48-628: CSTF_ITATBDATA0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	ATDATA15	. Reads atdata_rx[15] and writes atdata_tx[15]. RES0 if ATB_DATA_WIDTH < 16.	
		0	On reads, the value of atdata_rx[15] is 0. On writes, sets atdata_tx[15] to 0.
		1	On reads, the value of atdata_rx[15] is 1. On writes, sets atdata_tx[15] to 1.
1 (R/W)	ATDATA7	. Reads atdata_rx[7] and writes atdata_tx[7].	
		0	On reads, the value of atdata_rx[7] is 0. On writes, sets atdata_tx[7] to 0.
		1	On reads, the value of atdata_rx[7] is 1. On writes, sets atdata_tx[7] to 1.
0 (R/W)	ATDATA0	. Reads atdata_rx[0] and writes atdata_tx[0].	
		0	On reads, the value of atdata_rx[0] is 0. On writes, sets atdata_tx[0] to 0.
		1	On reads, the value of atdata_rx[0] is 1. On writes, sets atdata_tx[0] to 1.

Integration Mode Control Register.

The Integration Mode Control register is used to switch between functional mode and integration mode. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

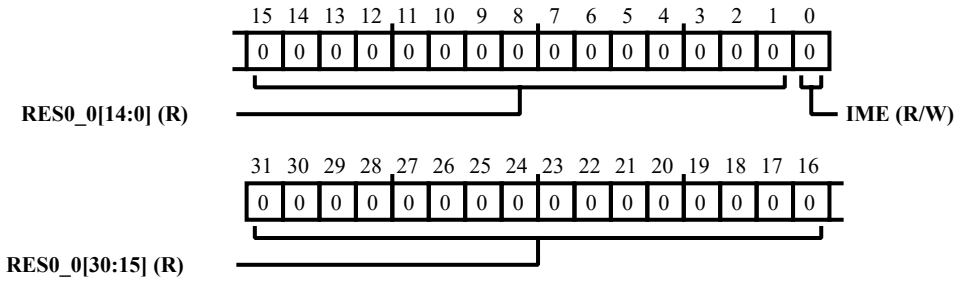


Figure 48-624: CSTF_ITCTRL Register Diagram

Table 48-629: CSTF_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
0 (R/W)	IME	. Integration Mode Enable.	
		0	The component must enter functional mode.
		1	The component must enter integration mode, and enable support for topology detection and integration testing.

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

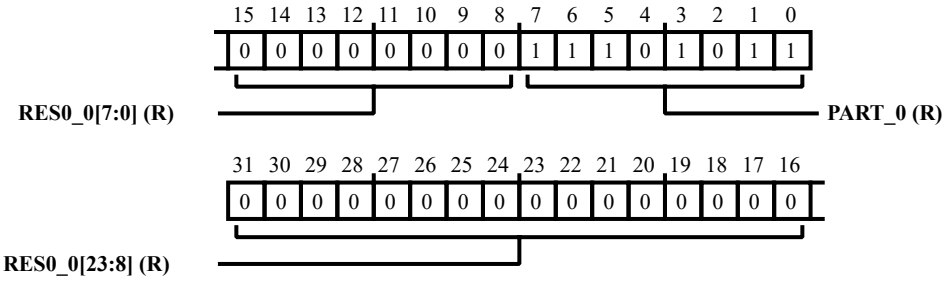


Figure 48-625: CSTF_PIDR0 Register Diagram

Table 48-630: CSTF_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PART_0	. Part number, bits[7:0]. Taken together with PIDR1.PART_1 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

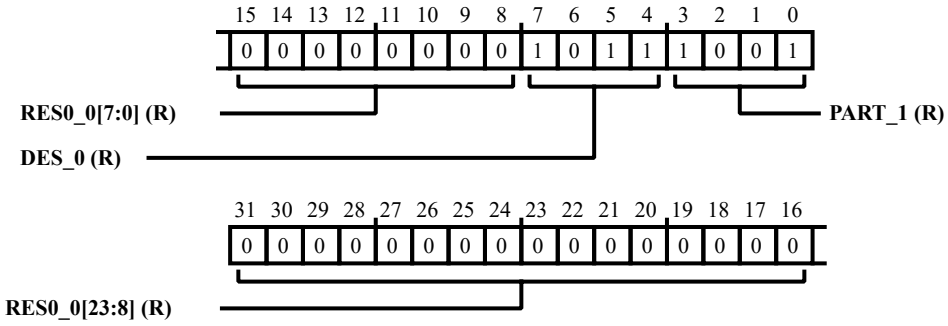


Figure 48-626: CSTF_PIDR1 Register Diagram

Table 48-631: CSTF_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

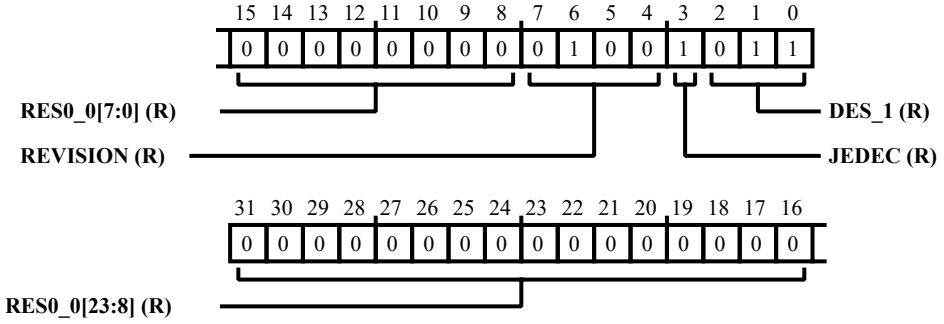


Figure 48-627: CSTF_PIDR2 Register Diagram

Table 48-632: CSTF_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

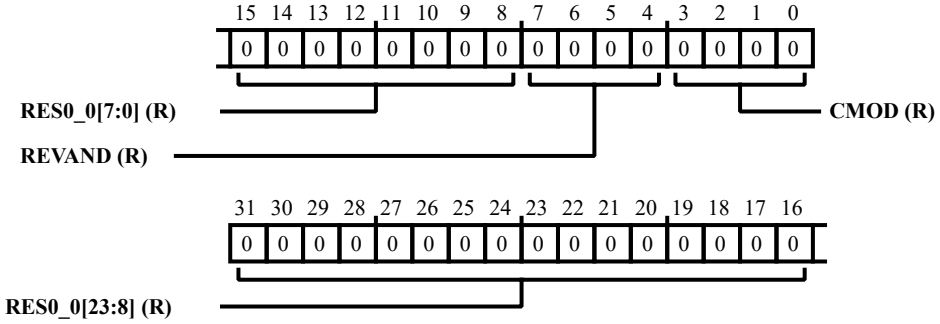


Figure 48-628: CSTF_PIDR3 Register Diagram

Table 48-633: CSTF_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

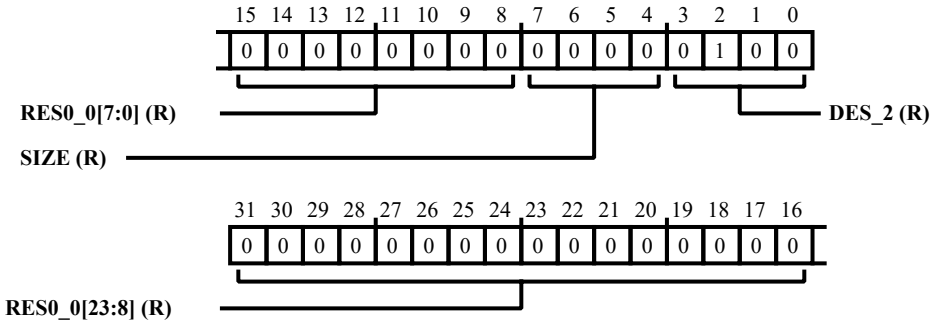


Figure 48-629: CSTF_PIDR4 Register Diagram

Table 48-634: CSTF_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

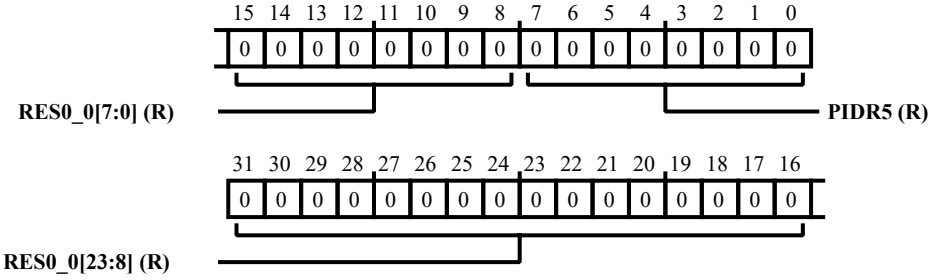


Figure 48-630: CSTF_PIDR5 Register Diagram

Table 48-635: CSTF_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

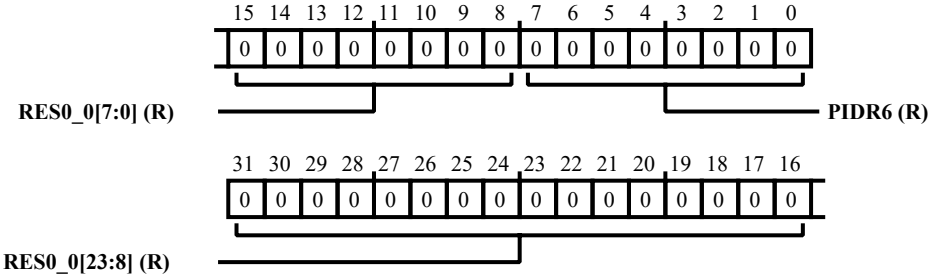


Figure 48-631: CSTF_PIDR6 Register Diagram

Table 48-636: CSTF_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

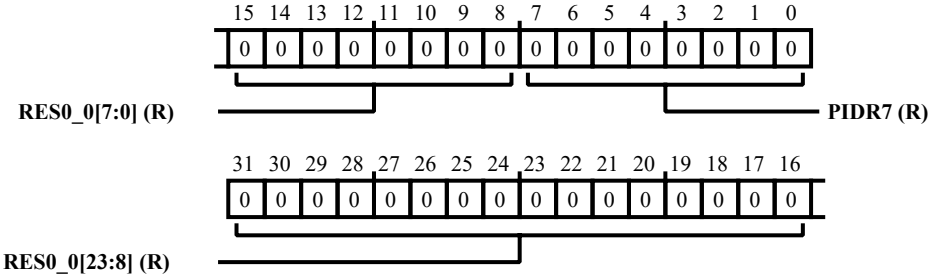


Figure 48-632: CSTF_PIDR7 Register Diagram

Table 48-637: CSTF_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

Priority Control register.

The PRIORITYCONTROL register sets the priority of each port (receiver interface) of the funnel. The programming software requires that the ports are all disabled before the priority control register contents are changed. Changing the port priorities in real time is not supported. If the priority control register is written when one or more of the ports are enabled, then the write is silently rejected and the value in the priority control register remains unchanged. The lower the priority value, the higher is its priority when selecting the next port to be serviced. If two or more ports have the same priority value, then the lowest numbered port is serviced first.

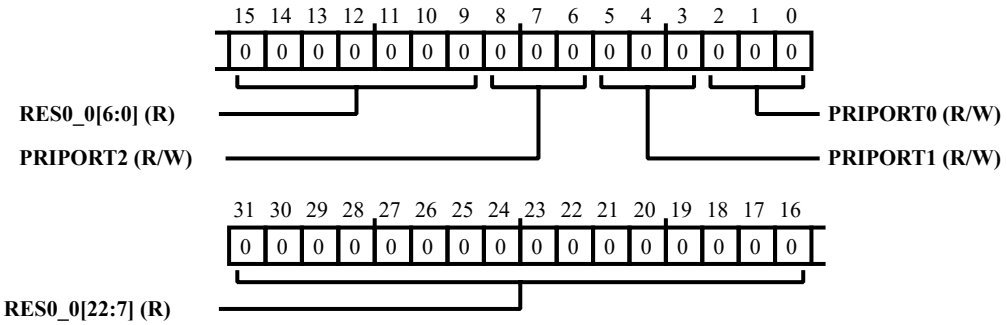


Figure 48-633: CSTF_PRIORITYCONTROL Register Diagram

Table 48-638: CSTF_PRIORITYCONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:9 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
8:6 (R/W)	PRIPORT2	. Priority value for port 2. Field is RES0 if port 2 is not implemented.
5:3 (R/W)	PRIPORT1	. Priority value for port 1.
2:0 (R/W)	PRIPORT0	. Priority value for port 0.

ADSP-2184x A55_SYSTMRRD Register Descriptions

Your module description, here. (A55_SYSTMRRD) contains the following registers.

Table 48-639: ADSP-2184x A55_SYSTMRRD Register List

Name	Description
A55_SYSTMRRD_CIDR0	Component Identification Register 0.
A55_SYSTMRRD_CIDR1	Component Identification Register 1.

Table 48-639: ADSP-2184x A55_SYSTMRRD Register List (Continued)

Name	Description
A55_SYSTMRRD_CIDR2	Component Identification Register 2.
A55_SYSTMRRD_CIDR3	Component Identification Register 3.
A55_SYSTMRRD_CNTCVLREAD	Current value of Counter[31:0].
A55_SYSTMRRD_CNTCVUREAD	Current value of Counter[63:32].
A55_SYSTMRRD_PIDR0	Peripheral Identification Register 0.
A55_SYSTMRRD_PIDR1	Peripheral Identification Register 1.
A55_SYSTMRRD_PIDR2	Peripheral Identification Register 2.
A55_SYSTMRRD_PIDR3	Peripheral Identification Register 3.
A55_SYSTMRRD_PIDR4	Peripheral Identification Register 4.
A55_SYSTMRRD_PIDR5	Peripheral Identification Register 5.
A55_SYSTMRRD_PIDR6	Peripheral Identification Register 6.
A55_SYSTMRRD_PIDR7	Peripheral Identification Register 7.

Component Identification Register 0.

The CIDR0 register is part of the set of component identification registers.

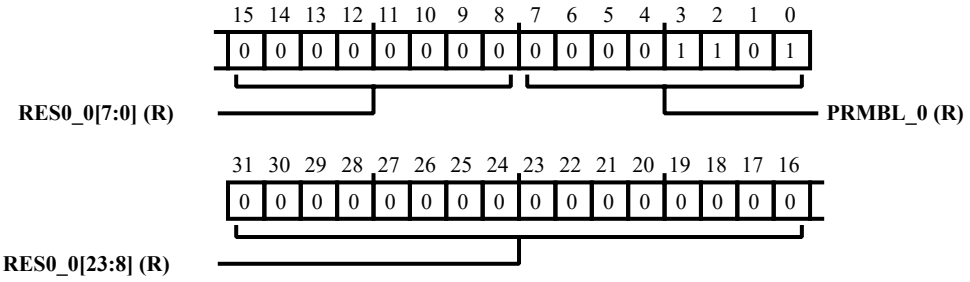


Figure 48-634: A55_SYSTMRRD_CIDR0 Register Diagram

Table 48-640: A55_SYSTMRRD_CIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_0	. Preamble. Returns 0x0D.

Component Identification Register 1.

The CIDR1 register is part of the set of component identification registers.

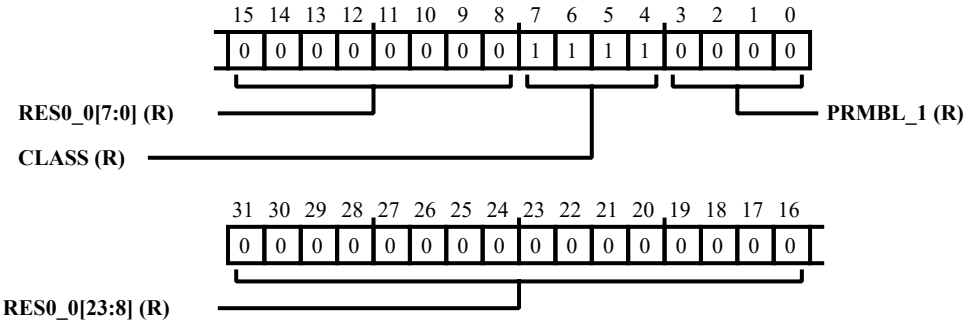


Figure 48-635: A55_SYSTMRRD_CIDR1 Register Diagram

Table 48-641: A55_SYSTMRRD_CIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.	
7:4 (R/NW)	CLASS	. Component class.	
		9	CoreSight component.
		15	CoreLink, PrimeCell, or system component.
3:0 (R/NW)	PRMBL_1	. Preamble. Returns 0x0.	

Component Identification Register 2.

The CIDR2 register is part of the set of component identification registers.

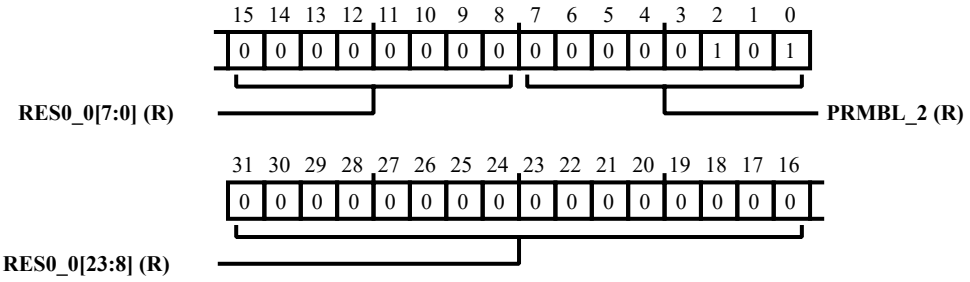


Figure 48-636: A55_SYSTMRRD_CIDR2 Register Diagram

Table 48-642: A55_SYSTMRRD_CIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_2	. Preamble. Returns 0x05.

Component Identification Register 3.

The CIDR3 register is part of the set of component identification registers.

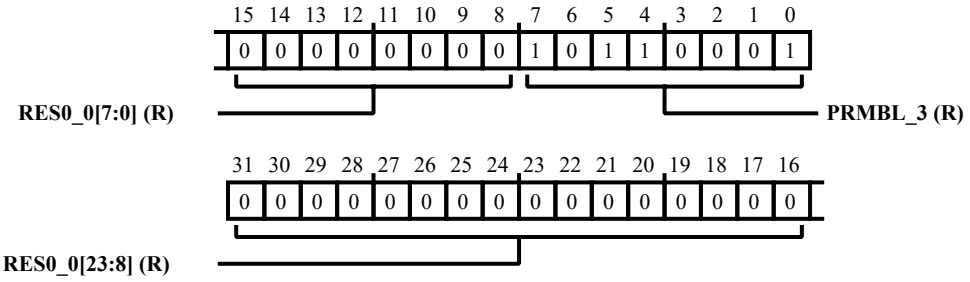


Figure 48-637: A55_SYSTMRRD_CIDR3 Register Diagram

Table 48-643: A55_SYSTMRRD_CIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PRMBL_3	. Preamble. Returns 0xB1.

Current value of Counter[31:0].

The CNTCVLREAD register reads the lower 32 bits of the current counter value.

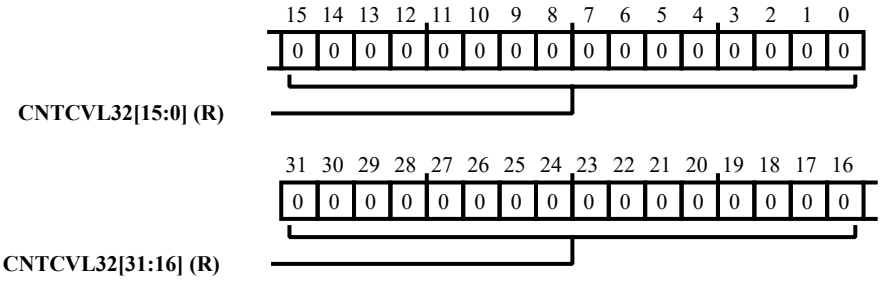


Figure 48-638: A55_SYSTMRRD_CNTCVLREAD Register Diagram

Table 48-644: A55_SYSTMRRD_CNTCVLREAD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CNTCVL32	. The lower 32 bits of the current timestamp counter value.

Current value of Counter[63:32].

The CNTCVUREAD register reads the upper 32 bits of the current counter value.

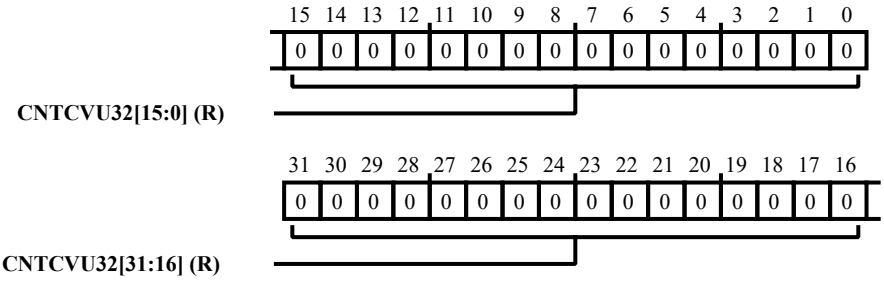


Figure 48-639: A55_SYSTMRRD_CNTCVUREAD Register Diagram

Table 48-645: A55_SYSTMRRD_CNTCVUREAD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CNTCVU32	. The upper 32 bits of the current timestamp counter value.

Peripheral Identification Register 0.

The PIDR0 register is part of the set of peripheral identification registers.

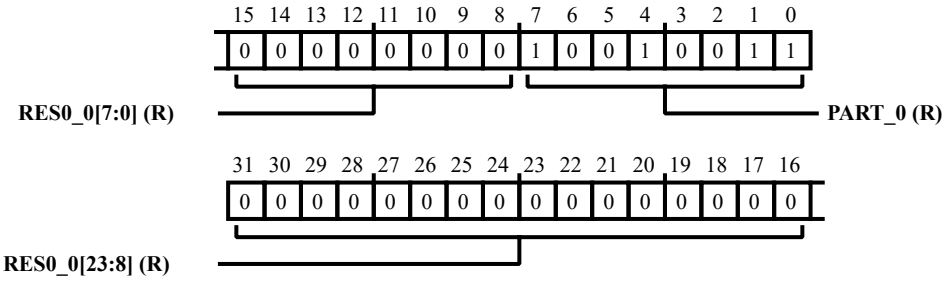


Figure 48-640: A55_SYSTMRRD_PIDR0 Register Diagram

Table 48-646: A55_SYSTMRRD_PIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PART_0	. Part number, bits[7:0]. Taken together with PIDR1.PART_1 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 1.

The PIDR1 register is part of the set of peripheral identification registers.

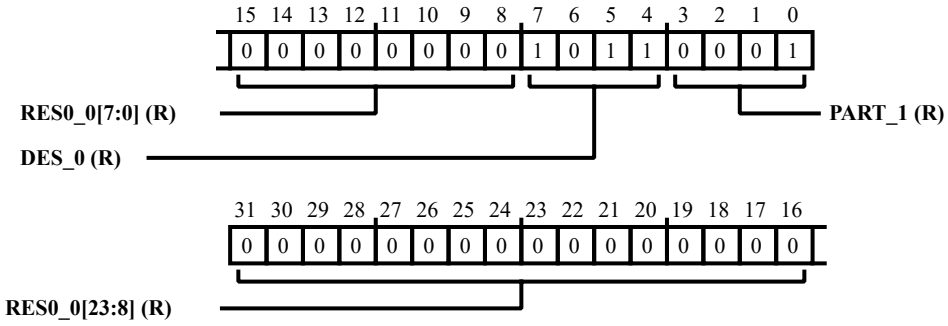


Figure 48-641: A55_SYSTMRRD_PIDR1 Register Diagram

Table 48-647: A55_SYSTMRRD_PIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	DES_0	. JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same.
3:0 (R/NW)	PART_1	. Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral Identification Register 2.

The PIDR2 register is part of the set of peripheral identification registers.

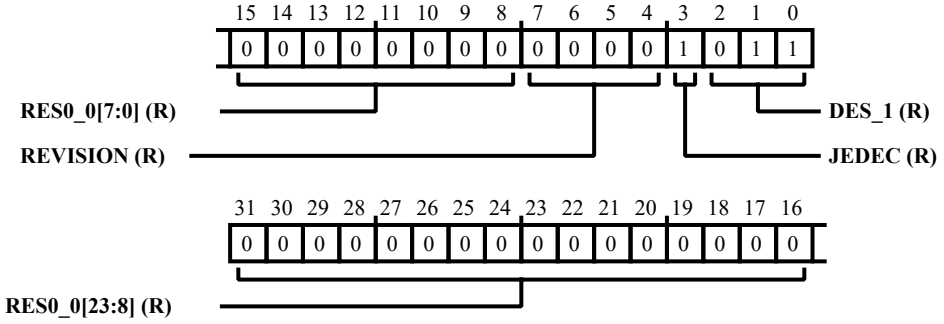


Figure 48-642: A55_SYSTMRRD_PIDR2 Register Diagram

Table 48-648: A55_SYSTMRRD_PIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVISION	. Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component.
3 (R/NW)	JEDEC	. 1 - Always set. Indicates that a JEDEC assigned value is used.
2:0 (R/NW)	DES_1	. JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 3.

The PIDR3 register is part of the set of peripheral identification registers.

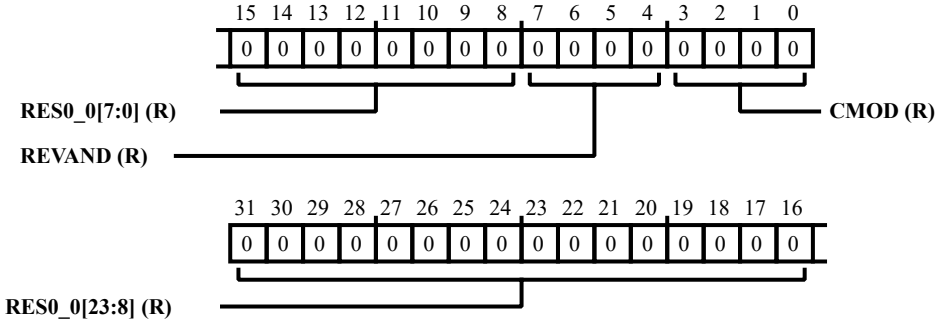


Figure 48-643: A55_SYSTMRRD_PIDR3 Register Diagram

Table 48-649: A55_SYSTMRRD_PIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	REVAND	. This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0.
3:0 (R/NW)	CMOD	. Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral Identification Register 4.

The PIDR4 register is part of the set of peripheral identification registers.

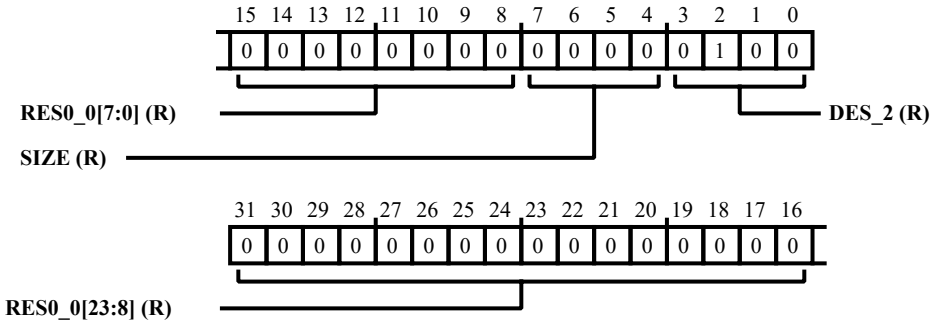


Figure 48-644: A55_SYSTMRRD_PIDR4 Register Diagram

Table 48-650: A55_SYSTMRRD_PIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:4 (R/NW)	SIZE	. Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated.
3:0 (R/NW)	DES_2	. JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral Identification Register 5.

The PIDR5 register is part of the set of peripheral identification registers.

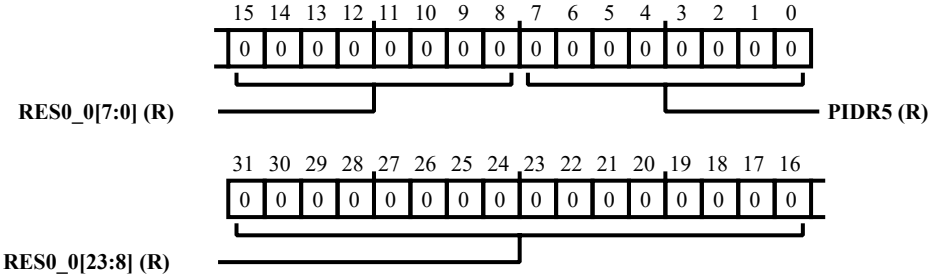


Figure 48-645: A55_SYSTMRRD_PIDR5 Register Diagram

Table 48-651: A55_SYSTMRRD_PIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR5	. Reserved.

Peripheral Identification Register 6.

The PIDR6 register is part of the set of peripheral identification registers.

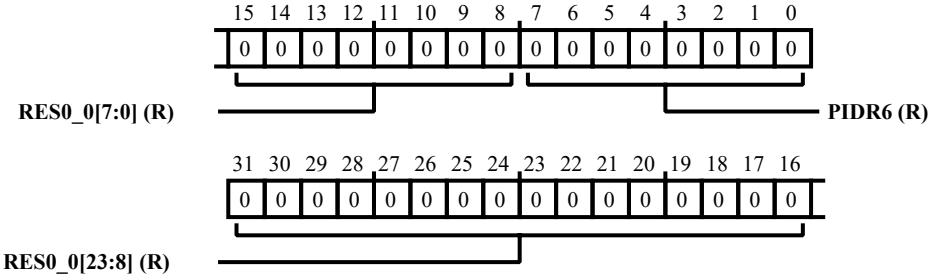


Figure 48-646: A55_SYSTMRRD_PIDR6 Register Diagram

Table 48-652: A55_SYSTMRRD_PIDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR6	. Reserved.

Peripheral Identification Register 7.

The PIDR7 register is part of the set of peripheral identification registers.

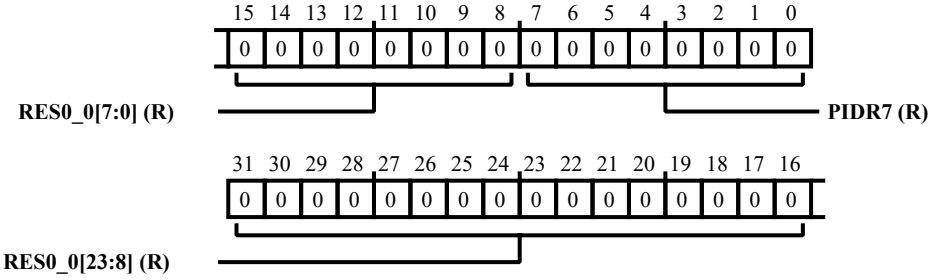


Figure 48-647: A55_SYSTMRRD_PIDR7 Register Diagram

Table 48-653: A55_SYSTMRRD_PIDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/NW)	RES0_0	. Reserved bit or field with SBZP behavior.
7:0 (R/NW)	PIDR7	. Reserved.

ADSP-2184x A55_C0 Register Descriptions

Your module description, here. (A55_C0) contains the following registers.

Table 48-654: ADSP-2184x A55_C0 Register List

Name	Description
A55_C0_DBGBCR0_EL1	Debug Breakpoint Control Registers.
A55_C0_DBGBCR1_EL1	Debug Breakpoint Control Registers.
A55_C0_DBGBCR2_EL1	Debug Breakpoint Control Registers.
A55_C0_DBGBCR3_EL1	Debug Breakpoint Control Registers.
A55_C0_DBGBCR4_EL1	Debug Breakpoint Control Registers.
A55_C0_DBGBCR5_EL1	Debug Breakpoint Control Registers.
A55_C0_DBGCLAIMCLR_EL1	Debug Claim Tag Clear register.
A55_C0_DBGCLAIMSET_EL1	Debug Claim Tag Set register.
A55_C0_DBGDTRRX_EL0	Debug Data Transfer Register, Receive.
A55_C0_DBGDTRTX_EL0	Debug Data Transfer Register, Transmit.

Table 48-654: ADSP-2184x A55_C0 Register List (Continued)

Name	Description
A55_C0_DBGWCR0_EL1	Debug Watchpoint Control Registers.
A55_C0_DBGWCR1_EL1	Debug Watchpoint Control Registers.
A55_C0_DBGWCR2_EL1	Debug Watchpoint Control Registers.
A55_C0_DBGWCR3_EL1	Debug Watchpoint Control Registers.
A55_C0_DBGWVR0_EL1_31_0	Debug Watchpoint Value Registers.
A55_C0_DBGWVR0_EL1_63_32	Debug Watchpoint Value Registers.
A55_C0_DBGWVR1_EL1_31_0	Debug Watchpoint Value Registers.
A55_C0_DBGWVR1_EL1_63_32	Debug Watchpoint Value Registers.
A55_C0_DBGWVR2_EL1_31_0	Debug Watchpoint Value Registers.
A55_C0_DBGWVR2_EL1_63_32	Debug Watchpoint Value Registers.
A55_C0_DBGWVR3_EL1_31_0	Debug Watchpoint Value Registers.
A55_C0_DBGWVR3_EL1_63_32	Debug Watchpoint Value Registers.
A55_C0_EDCIDSR	External Debug Context ID Sample Register.
A55_C0_EDECCR	External Debug Exception Catch Control Register.
A55_C0_EDESR	External Debug Event Status Register.
A55_C0_EDITCTRL	External Debug Integration mode Control register.
A55_C0_EDITR	External Debug Instruction Transfer Register.
A55_C0_EDPRCR	External Debug Power/Reset Control Register.
A55_C0_EDPRSR	External Debug Processor Status Register.
A55_C0_EDRCR	External Debug Reserve Control Register.
A55_C0_EDSCR	External Debug Status and Control Register.
A55_C0_EDWAR_31_0	External Debug Watchpoint Address Register.
A55_C0_EDWAR_63_32	External Debug Watchpoint Address Register.
A55_C0_OSLAR_EL1	OS Lock Access Register.
A55_C0_TRCACATR0_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR0_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR1_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR1_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR2_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR2_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR3_31_0	Trace Address Comparator Access Type Registers.

Table 48-654: ADSP-2184x A55_C0 Register List (Continued)

Name	Description
A55_C0_TRCACATR3_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR4_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR4_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR5_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR5_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR6_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR6_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR7_31_0	Trace Address Comparator Access Type Registers.
A55_C0_TRCACATR7_63_32	Trace Address Comparator Access Type Registers.
A55_C0_TRCACVR0_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR0_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR1_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR1_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR2_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR2_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR3_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR3_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR4_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR4_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR5_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR5_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR6_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR6_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCACVR7_31_0	Trace Address Comparator Value Registers.
A55_C0_TRCACVR7_63_32	Trace Address Comparator Value Registers.
A55_C0_TRCAUXCTLR	Trace Auxiliary Control Register.
A55_C0_TRCBBCTLR	Trace Branch Broadcast Control Register.
A55_C0_TRCCCCTLR	Trace Cycle Count Control Register.
A55_C0_TRCCIDCCTLR0	Trace Context ID Comparator Control Register 0.
A55_C0_TRCCIDCVR0_31_0	Trace Context ID Comparator Value Registers.
A55_C0_TRCCIDCVR0_63_32	Trace Context ID Comparator Value Registers.

Table 48-654: ADSP-2184x A55_C0 Register List (Continued)

Name	Description
A55_C0_TRCCLAIMCLR	Trace Claim Tag Clear Register.
A55_C0_TRCCLAIMSET	Trace Claim Tag Set Register.
A55_C0_TRCCNTCTLR0	Trace Counter Control Registers.
A55_C0_TRCCNTCTLR1	Trace Counter Control Registers.
A55_C0_TRCCNTRLDVR0	Trace Counter Reload Value Registers.
A55_C0_TRCCNTRLDVR1	Trace Counter Reload Value Registers.
A55_C0_TRCCNTVR0	Trace Counter Value Registers.
A55_C0_TRCCNTVR1	Trace Counter Value Registers.
A55_C0_TRCCONFIGR	Trace Configuration Register.
A55_C0_TRCEVENTCTL0R	Trace Event Control 0 Register.
A55_C0_TRCEVENTCTL1R	Trace Event Control 1 Register.
A55_C0_TRCEXTINSELR	Trace External Input Select Register.
A55_C0_TRCIDR0	Trace ID Register 0.
A55_C0_TRCIDR1	Trace ID Register 1.
A55_C0_TRCIDR2	Trace ID Register 2.
A55_C0_TRCIDR3	Trace ID Register 3.
A55_C0_TRCIDR4	Trace ID Register 4.
A55_C0_TRCIDR5	Trace ID Register 5.
A55_C0_TRCIMSPEC0	Trace IMPLEMENTATION DEFINED Register 0.
A55_C0_TRCITATBIDR	Integration ATB Identification Register.
A55_C0_TRCITCTRL	Trace Integration Mode Control Register.
A55_C0_TRCITIATBINR	Integration Instruction ATB In Register.
A55_C0_TRCITIATBOUTR	Integration Instruction ATB Out Register.
A55_C0_TRCITIDATAR	Integration Instruction ATB Data Register.
A55_C0_TRCOSLAR	Trace OS Lock Access Register.
A55_C0_TRCOSLSR	Trace OS Lock Status Register.
A55_C0_TRCPDSR	Trace PowerDown Status Register.
A55_C0_TRCPRGCTLR	Trace Programming Control Register.
A55_C0_TRCRSCTLR10	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR11	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR12	Trace Resource Selection Control Registers.

Table 48-654: ADSP-2184x A55_C0 Register List (Continued)

Name	Description
A55_C0_TRCRSCTLR13	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR14	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR15	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR2	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR3	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR4	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR5	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR6	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR7	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR8	Trace Resource Selection Control Registers.
A55_C0_TRCRSCTLR9	Trace Resource Selection Control Registers.
A55_C0_TRCSEQEVR0	Trace Sequencer State Transition Control Registers.
A55_C0_TRCSEQEVR1	Trace Sequencer State Transition Control Registers.
A55_C0_TRCSEQEVR2	Trace Sequencer State Transition Control Registers.
A55_C0_TRCSEQRSTEVR	Trace Sequencer Reset Control Register.
A55_C0_TRCSEQSTR	Trace Sequencer State Register.
A55_C0_TRCSSCCR0	Trace Single-Shot Comparator Control Registers.
A55_C0_TRCSSCSR0	Trace Single-Shot Comparator Status Registers.
A55_C0_TRCSTALLCTLR	Trace Stall Control Register.
A55_C0_TRCSTATR	Trace Status Register.
A55_C0_TRCSYNCPR	Trace Synchronization Period Register.
A55_C0_TRCTRACEIDR	Trace Trace ID Register.
A55_C0_TRCTSCTLR	Trace Global Timestamp Control Register.
A55_C0_TRCVICTLR	Trace ViewInst Main Control Register.
A55_C0_TRCVIIECTLR	Trace ViewInst Include/Exclude Control Register.
A55_C0_TRCVISSCTLR	Trace ViewInst Start/Stop Control Register.
A55_C0_TRCVMIDCVR0_31_0	Trace VMID Comparator Value Registers.
A55_C0_TRCVMIDCVR0_63_32	Trace VMID Comparator Value Registers.
A55_C0_PMCCFILTR_EL0	Performance Monitors Cycle Counter Filter Register.
A55_C0_PMCCNTR_EL0_31_0	Performance Monitors Cycle Counter.
A55_C0_PMCCNTR_EL0_63_32	Performance Monitors Cycle Counter.

Table 48-654: ADSP-2184x A55_C0 Register List (Continued)

Name	Description
A55_C0_PMCEID0	Performance Monitors Common Event Identification register 0.
A55_C0_PMCEID1	Performance Monitors Common Event Identification register 1.
A55_C0_PMCEID2	Performance Monitors Common Event Identification register 2.
A55_C0_PMCEID3	Performance Monitors Common Event Identification register 3.
A55_C0_PMCFGR	Performance Monitors Configuration Register.
A55_C0_PMCNTENCLR_EL0	Performance Monitors Count Enable Clear register.
A55_C0_PMCNTENSET_EL0	Performance Monitors Count Enable Set register.
A55_C0_PMCR_EL0	Performance Monitors Control Register.
A55_C0_PMEVCNTR0_EL0	Performance Monitors Event Count Registers.
A55_C0_PMEVCNTR1_EL0	Performance Monitors Event Count Registers.
A55_C0_PMEVCNTR2_EL0	Performance Monitors Event Count Registers.
A55_C0_PMEVCNTR3_EL0	Performance Monitors Event Count Registers.
A55_C0_PMEVCNTR4_EL0	Performance Monitors Event Count Registers.
A55_C0_PMEVCNTR5_EL0	Performance Monitors Event Count Registers.
A55_C0_PMEVTYPER0_EL0	Performance Monitors Event Type Registers.
A55_C0_PMEVTYPER1_EL0	Performance Monitors Event Type Registers.
A55_C0_PMEVTYPER2_EL0	Performance Monitors Event Type Registers.
A55_C0_PMEVTYPER3_EL0	Performance Monitors Event Type Registers.
A55_C0_PMEVTYPER4_EL0	Performance Monitors Event Type Registers.
A55_C0_PMEVTYPER5_EL0	Performance Monitors Event Type Registers.
A55_C0_PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear register.
A55_C0_PMINTENSET_EL1	Performance Monitors Interrupt Enable Set register.
A55_C0_PMITCTRL	Performance Monitors Integration mode Control register.
A55_C0_PMOVSCLR_EL0	Performance Monitors Overflow Flag Status Clear register.
A55_C0_PMOVSSET_EL0	Performance Monitors Overflow Flag Status Set register.
A55_C0_PMSWINC_EL0	Performance Monitors Software Increment register.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR0_EL1.

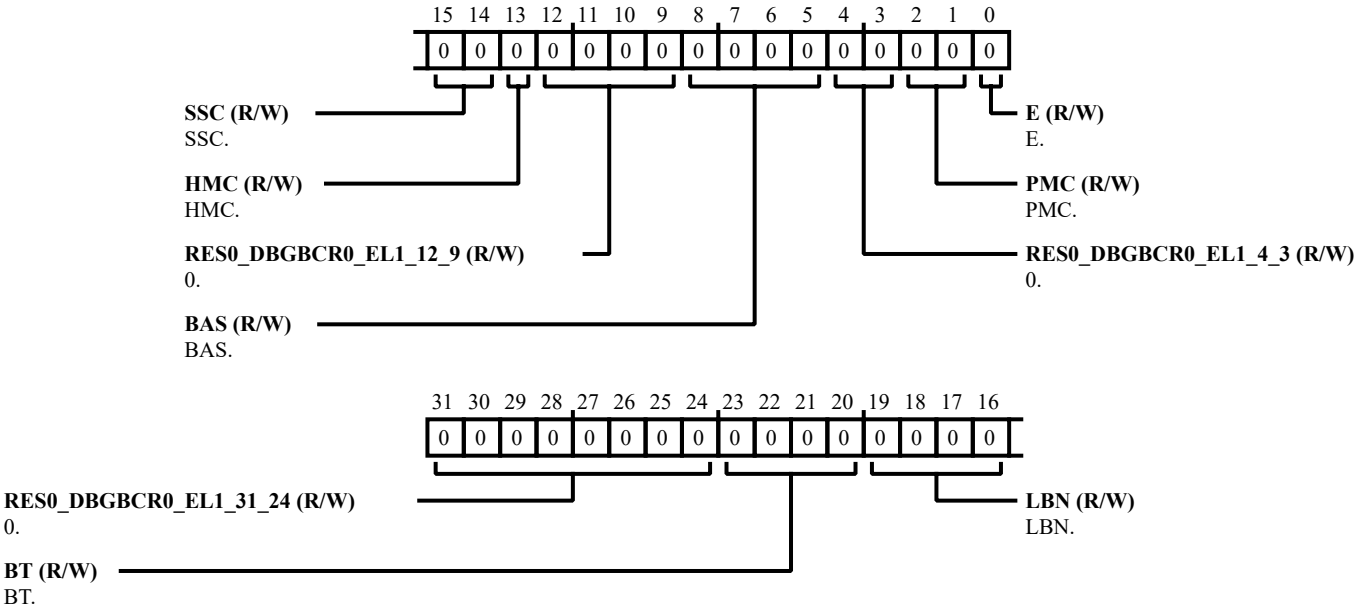


Figure 48-648: A55_C0_DBGBCR0_EL1 Register Diagram

Table 48-655: A55_C0_DBGBCR0_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR0_EL1_31_24	0.. Reserved, RES0.

Table 48-655: A55_C0_DBGBCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-655: A55_C0_DBGBCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR0_EL1_12_9	0.. Reserved, RES0.	

Table 48-655: A55_C0_DBGBCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR0_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR1_EL1.

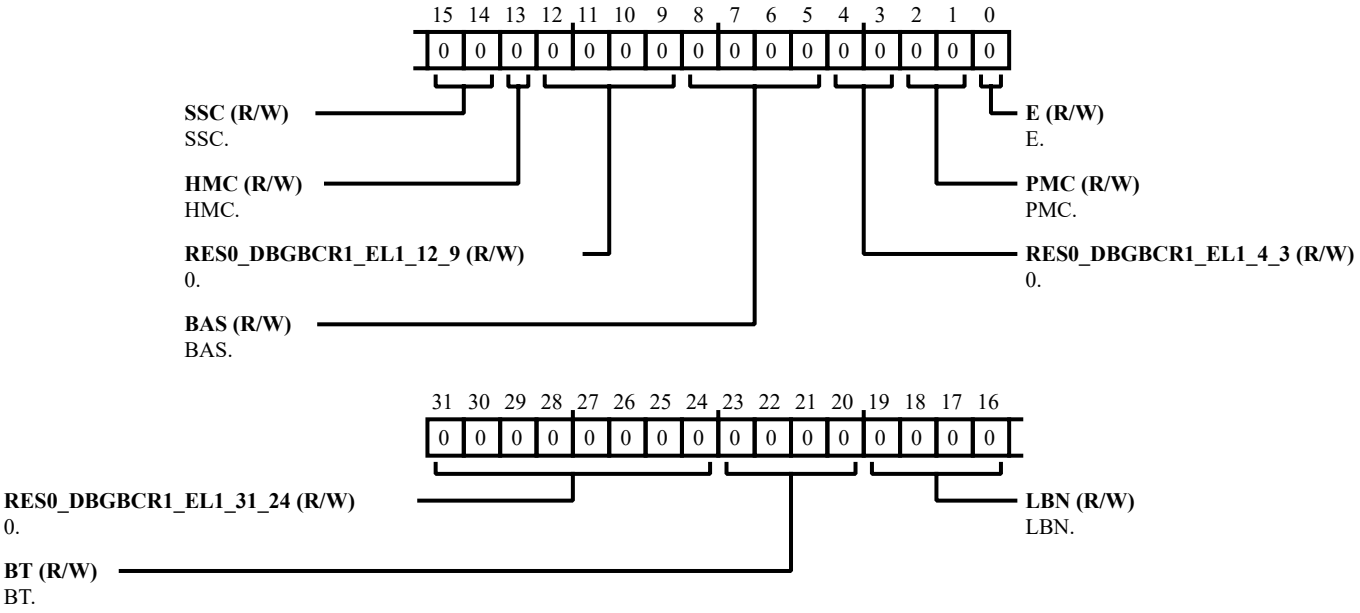


Figure 48-649: A55_C0_DBGBCR1_EL1 Register Diagram

Table 48-656: A55_C0_DBGBCR1_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR1_EL1_31_24	0.. Reserved, RES0.

Table 48-656: A55_C0_DBGBCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-656: A55_C0_DBGBCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR1_EL1_12_9	0.. Reserved, RES0.	

Table 48-656: A55_C0_DBGBCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR1_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR2_EL1.

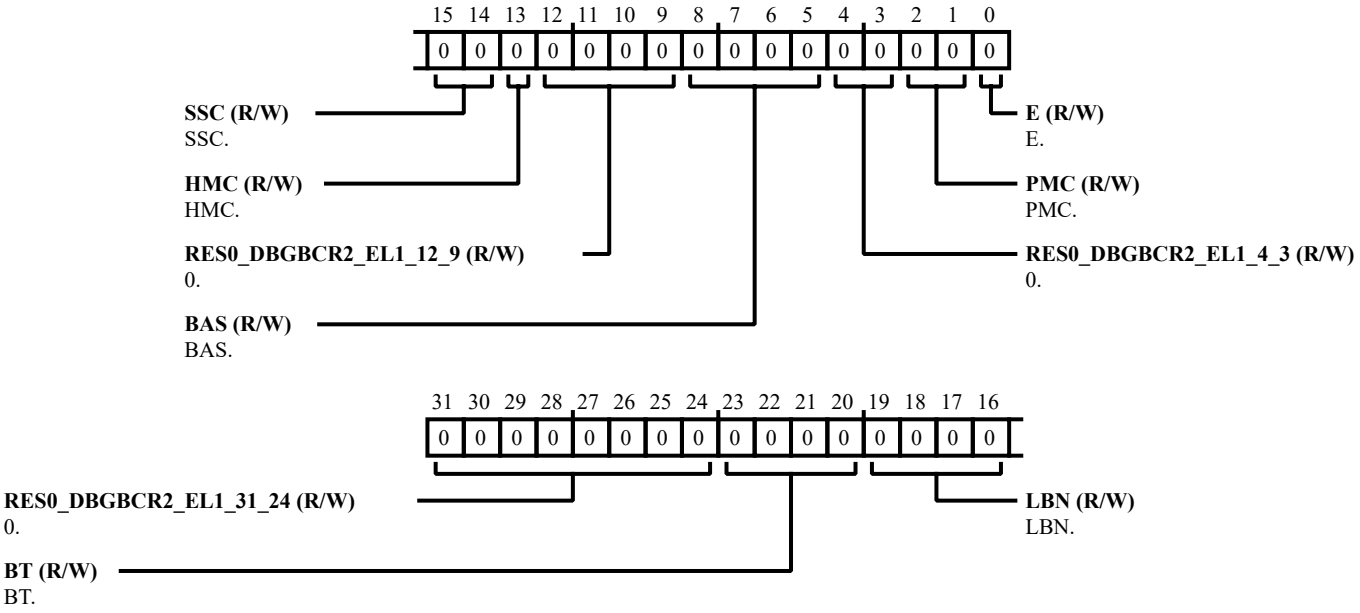


Figure 48-650: A55_C0_DBGBCR2_EL1 Register Diagram

Table 48-657: A55_C0_DBGBCR2_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR2_EL1_31_24	0.. Reserved, RES0.

Table 48-657: A55_C0_DBGBCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-657: A55_C0_DBGBCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR2_EL1_12_9	0.. Reserved, RES0.	

Table 48-657: A55_C0_DBGBCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR2_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR3_EL1.

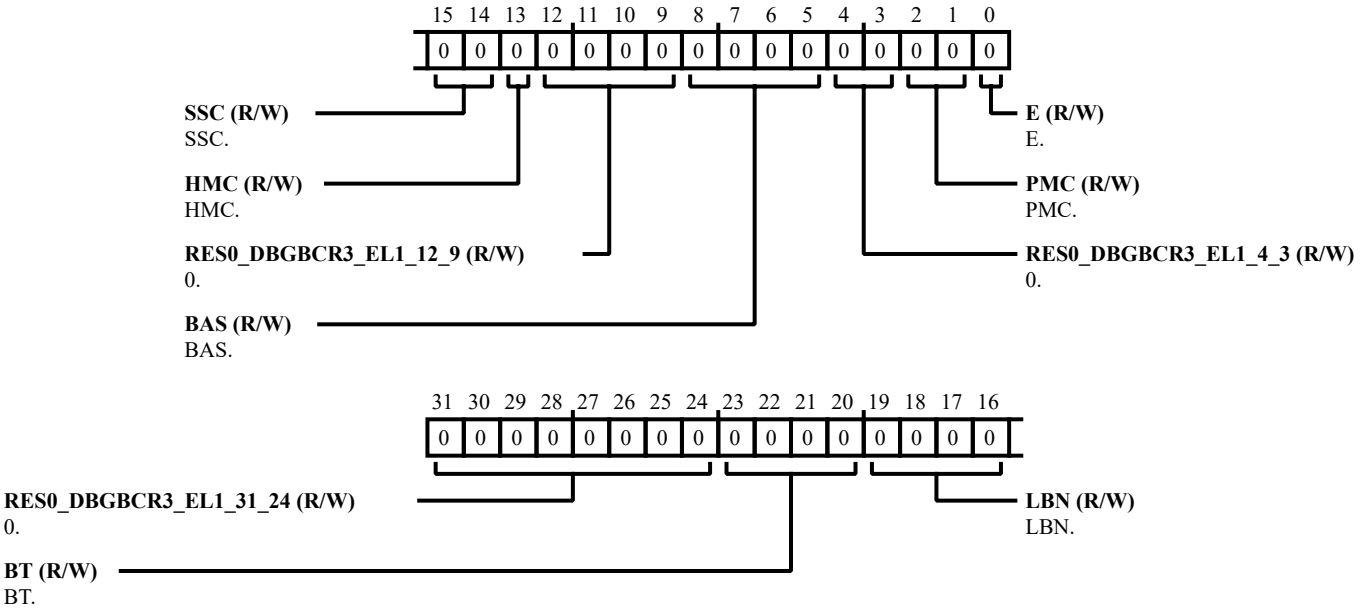


Figure 48-651: A55_C0_DBGBCR3_EL1 Register Diagram

Table 48-658: A55_C0_DBGBCR3_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR3_EL1_31_24	0.. Reserved, RES0.

Table 48-658: A55_C0_DBGBCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-658: A55_C0_DBGBCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR3_EL1_12_9	0.. Reserved, RES0.	

Table 48-658: A55_C0_DBGBCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.
4:3 (R/W)	RES0_DBGBCR3_EL1_4_3	0.. Reserved, RES0.
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled. 0 Breakpoint disabled. 1 Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR4_EL1.

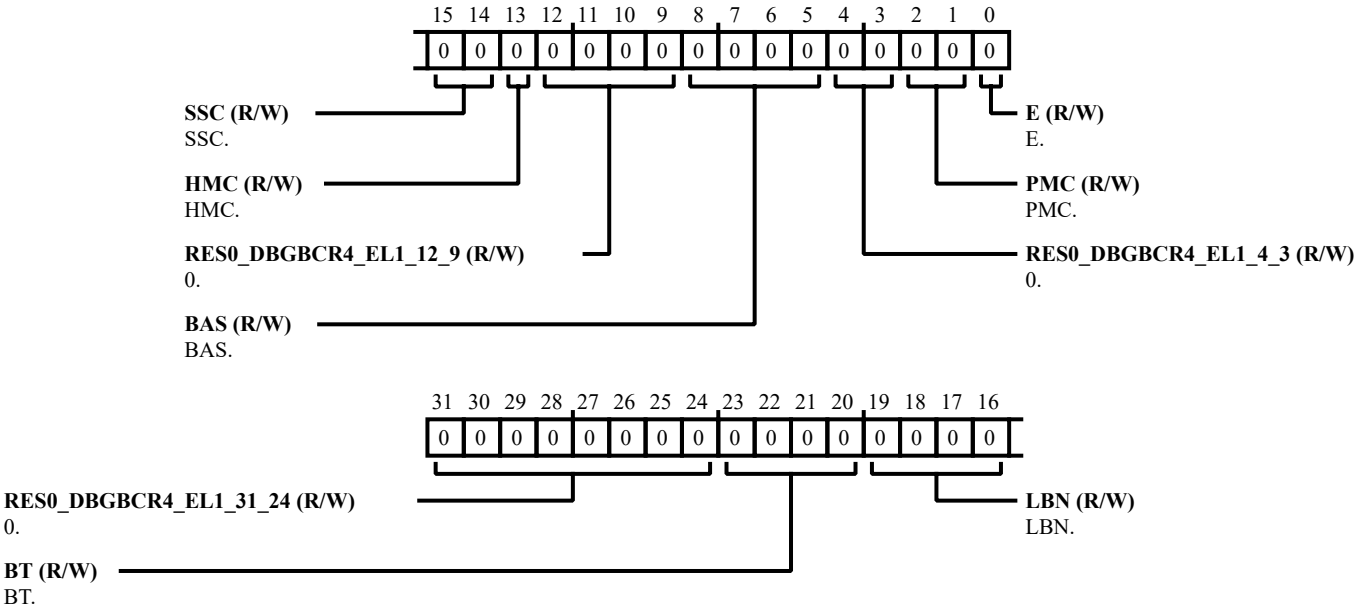


Figure 48-652: A55_C0_DBGBCR4_EL1 Register Diagram

Table 48-659: A55_C0_DBGBCR4_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR4_EL1_31_24	0.. Reserved, RES0.

Table 48-659: A55_C0_DBGBCR4_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-659: A55_C0_DBGBCR4_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR4_EL1_12_9	0.. Reserved, RES0.	

Table 48-659: A55_C0_DBGBCR4_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR4_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR5_EL1.

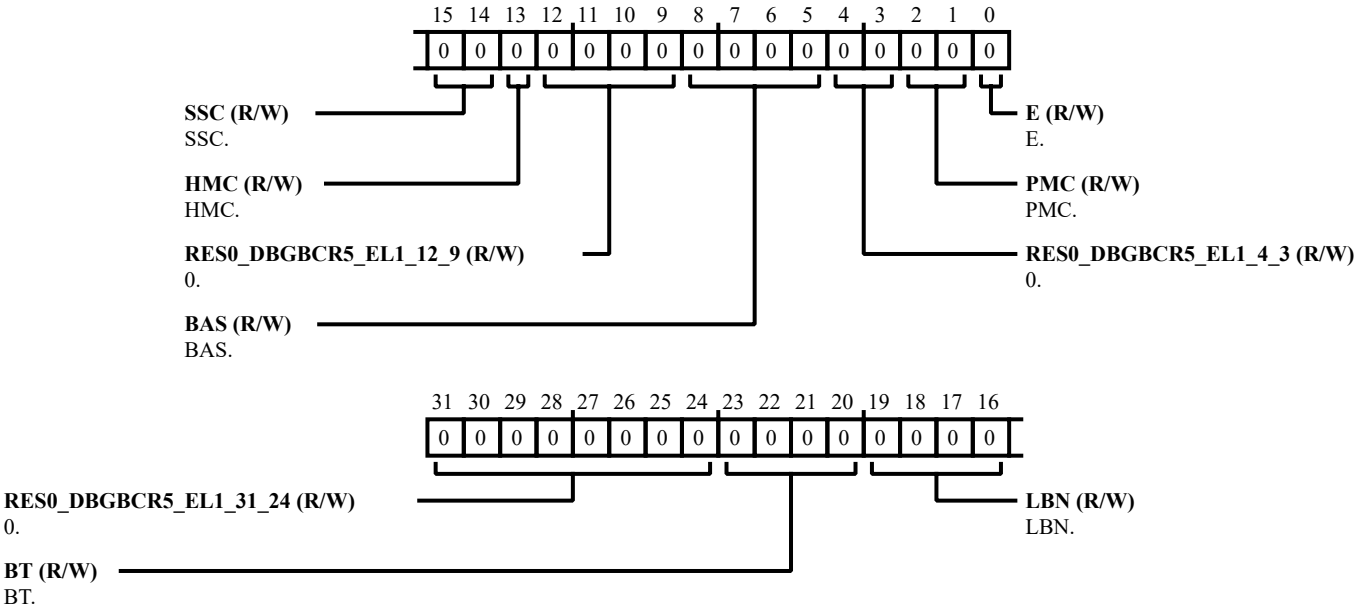


Figure 48-653: A55_C0_DBGBCR5_EL1 Register Diagram

Table 48-660: A55_C0_DBGBCR5_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR5_EL1_31_24	0.. Reserved, RES0.

Table 48-660: A55_C0_DBGBCR5_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-660: A55_C0_DBGBCR5_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR5_EL1_12_9	0.. Reserved, RES0.	

Table 48-660: A55_C0_DBGBCR5_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.
4:3 (R/W)	RES0_DBGBCR5_EL1_4_3	0.. Reserved, RES0.
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled. 0 Breakpoint disabled. 1 Breakpoint enabled.

Debug Claim Tag Clear register.

Used by software to read the values of the CLAIM tag bits, and to clear these bits to 0.

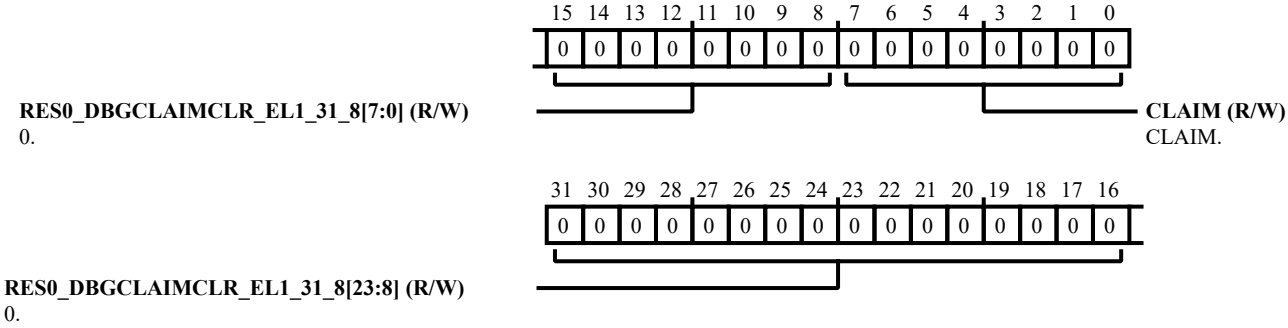


Figure 48-654: A55_C0_DBGCLAIMCLR_EL1 Register Diagram

Table 48-661: A55_C0_DBGCLAIMCLR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_DBGCLAIMCLR_EL1_31_8	0.. Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.
7:0 (R/W)	CLAIM	CLAIM.. Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits. Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0. Writing 0 to one of these bits has no effect. A cold reset clears the CLAIM tag bits to 0.

Debug Claim Tag Set register.

Used by software to set the CLAIM tag bits to 1.

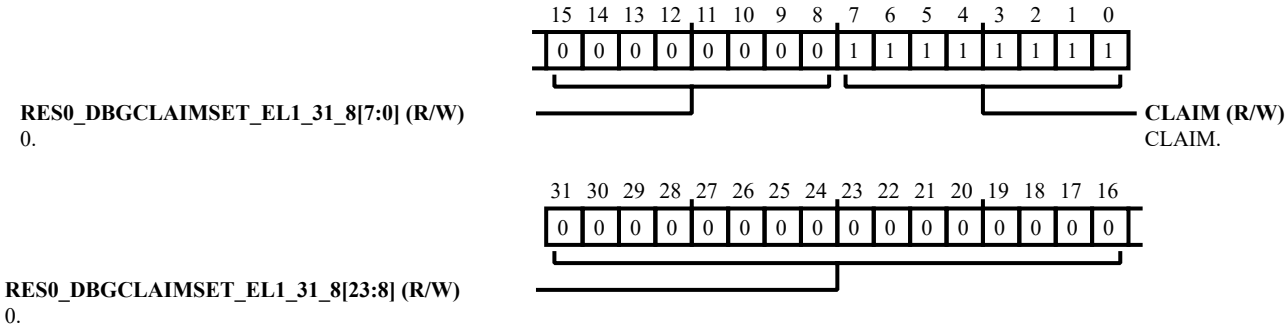


Figure 48-655: A55_C0_DBGCLAIMSET_EL1 Register Diagram

Table 48-662: A55_C0_DBGCLAIMSET_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_DBGCLAIM-SET_EL1_31_8	0.. Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.
7:0 (R/W)	CLAIM	CLAIM.. Set CLAIM tag bits. RAO. Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1. Writing 0 to one of these bits has no effect. A cold reset clears the CLAIM tag bits to 0.

Debug Data Transfer Register, Receive.

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. It is a component of the Debug Communications Channel.

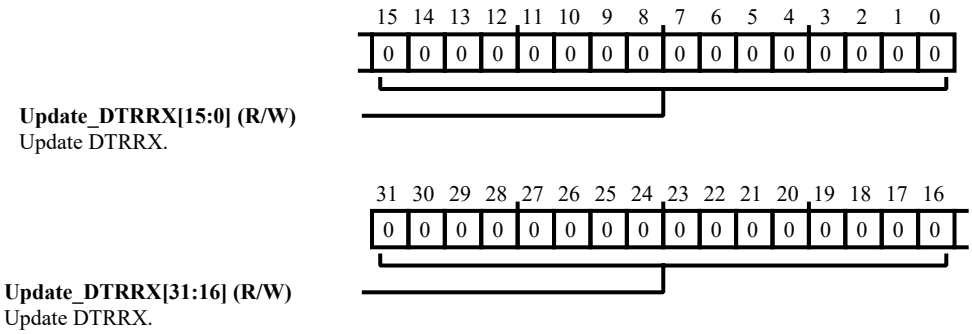


Figure 48-656: A55_C0_DBGDTRRX_EL0 Register Diagram

Table 48-663: A55_C0_DBGDTRRX_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	UPDATE_DTRRX	Update DTRRX.. Update DTRRX. If RXfull is set to 0, then writes to this register update the value in DTRRX and set RXfull to 1. Reads of this register return the last value written to DTRRX and do not change RXfull. For the full behavior of the Debug Communications Channel, see .

Debug Data Transfer Register, Transmit.

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. It is a component of the Debug Communication Channel.

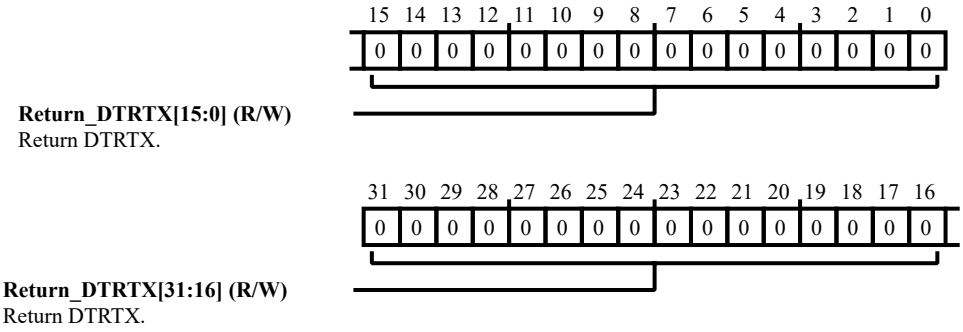


Figure 48-657: A55_C0_DBGDTRTX_EL0 Register Diagram

Table 48-664: A55_C0_DBGDTRTX_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RETURN_DTRTX	Return DTRTX.. Return DTRTX. If TXfull is set to 1, then reads of this register return the value in DTRTX and clear TXfull to 0. Writes of this register update the value in DTRTX and do not change TXfull. For the full behavior of the Debug Communications Channel, see .

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR0_EL1.

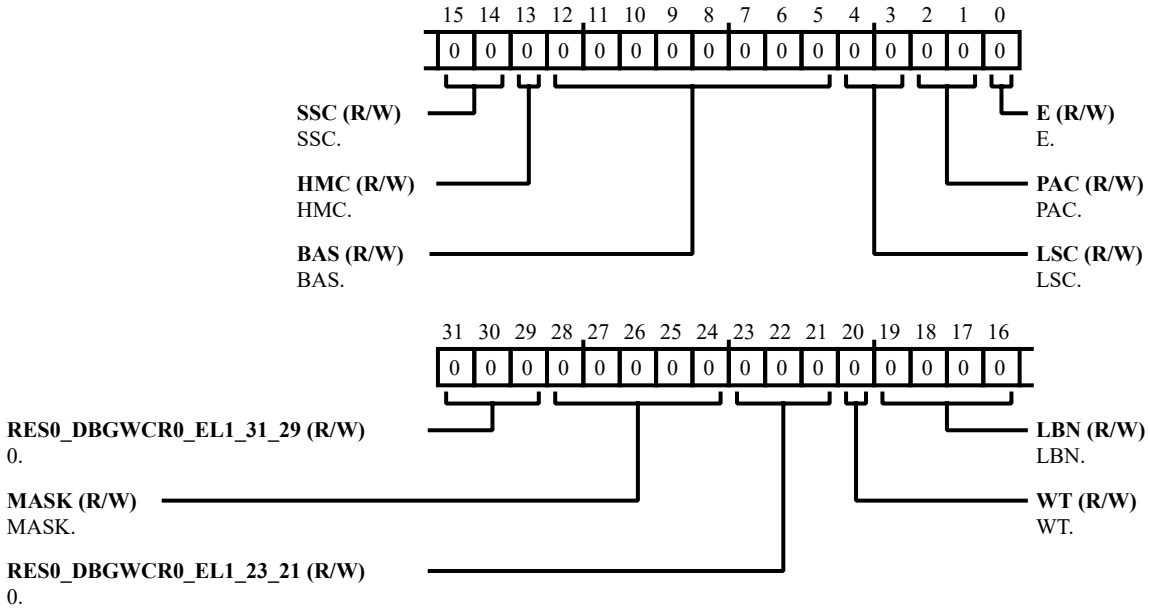


Figure 48-658: A55_C0_DBGWCR0_EL1 Register Diagram

Table 48-665: A55_C0_DBGWCR0_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR0_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-665: A55_C0_DBGWCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR0_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-665: A55_C0_DBGWCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR1_EL1.

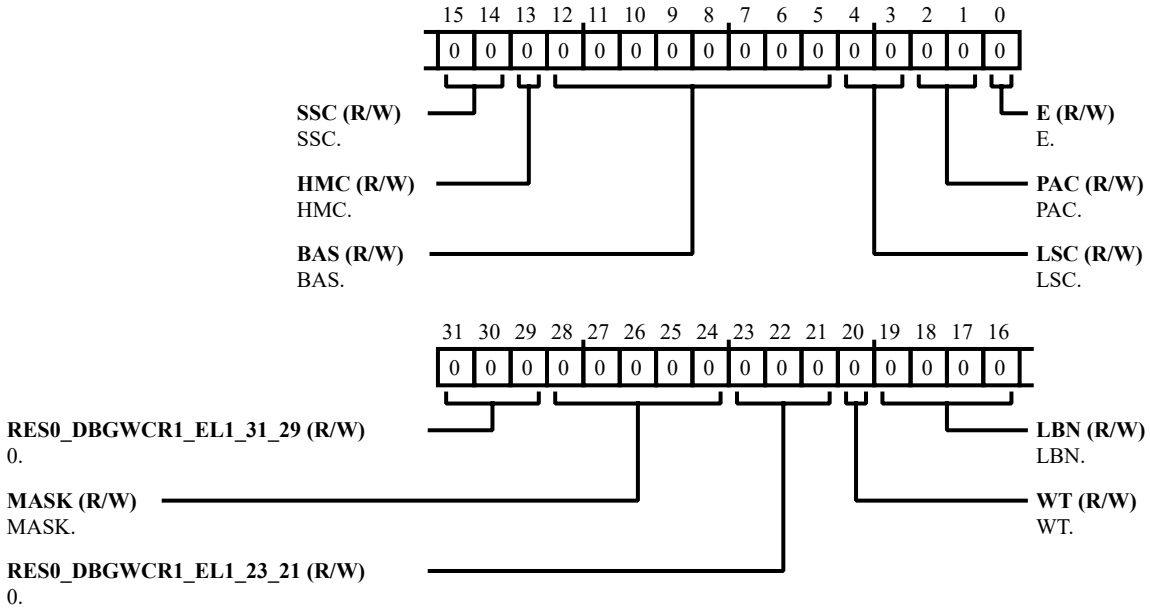


Figure 48-659: A55_C0_DBGWCR1_EL1 Register Diagram

Table 48-666: A55_C0_DBGWCR1_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR1_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-666: A55_C0_DBGWCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR1_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-666: A55_C0_DBGWCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR2_EL1.

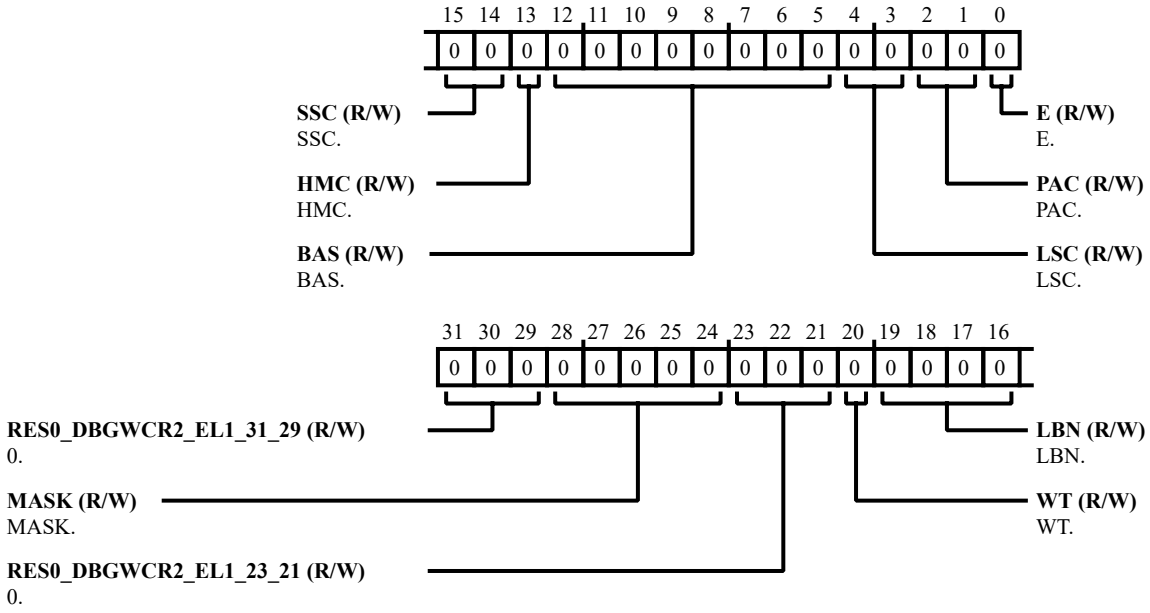


Figure 48-660: A55_C0_DBGWCR2_EL1 Register Diagram

Table 48-667: A55_C0_DBGWCR2_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR2_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-667: A55_C0_DBGWCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR2_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-667: A55_C0_DBGWCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR3_EL1.

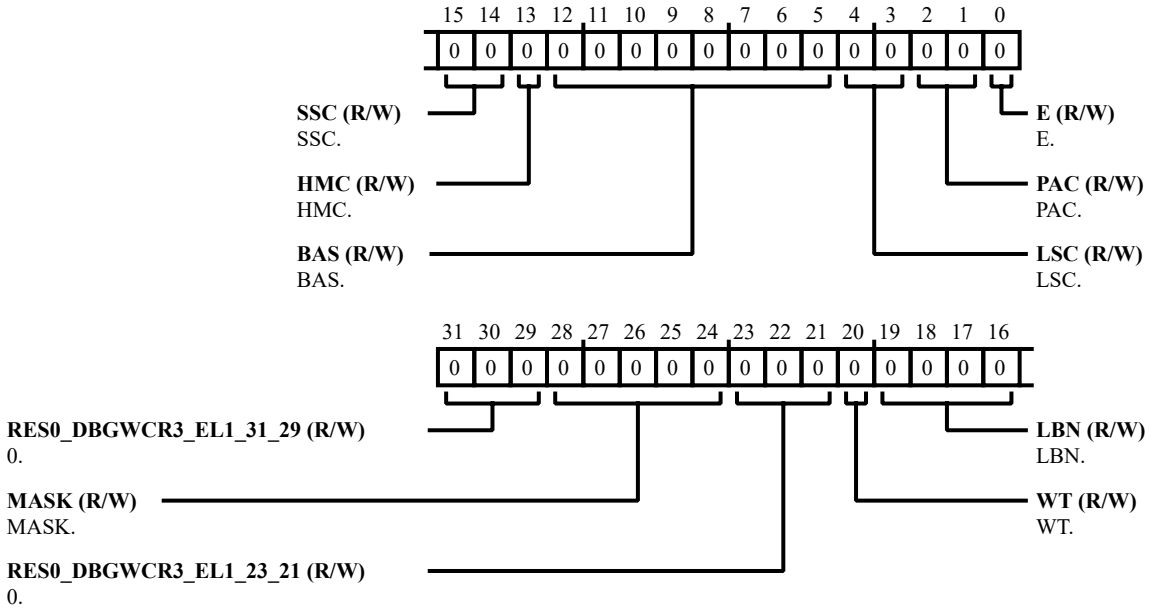


Figure 48-661: A55_C0_DBGWCR3_EL1 Register Diagram

Table 48-668: A55_C0_DBGWCR3_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR3_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-668: A55_C0_DBGWCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR3_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-668: A55_C0_DBGWCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR0_EL1.

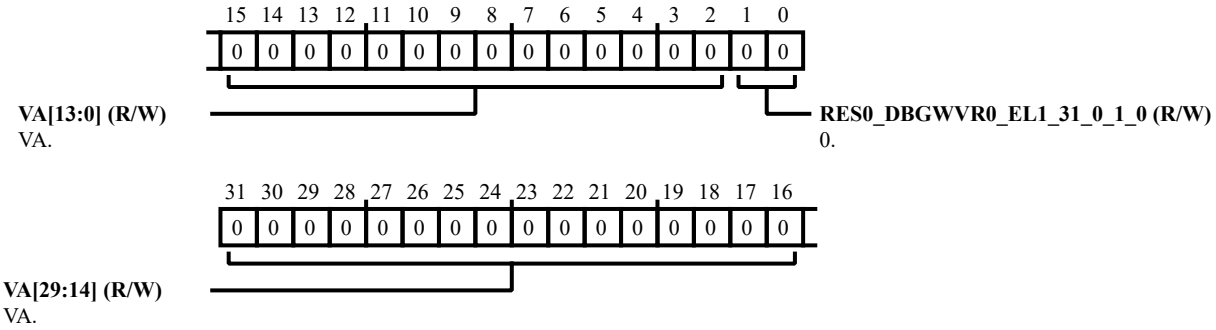


Figure 48-662: A55_C0_DBGWVR0_EL1_31_0 Register Diagram

Table 48-669: A55_C0_DBGWVR0_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR0_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR0_EL1.

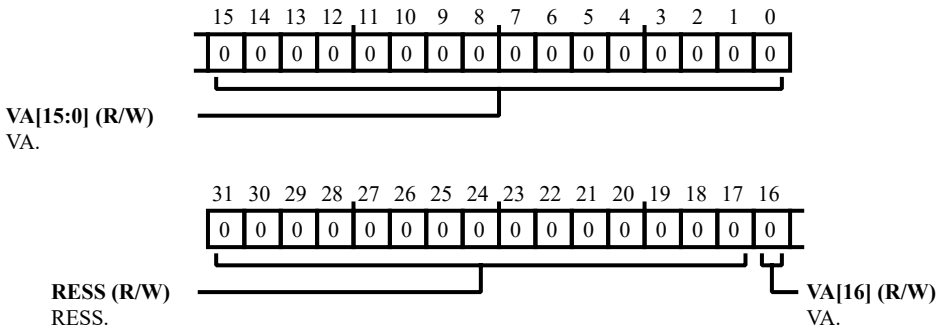


Figure 48-663: A55_C0_DBGWVR0_EL1_63_32 Register Diagram

Table 48-670: A55_C0_DBGWVR0_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR1_EL1.

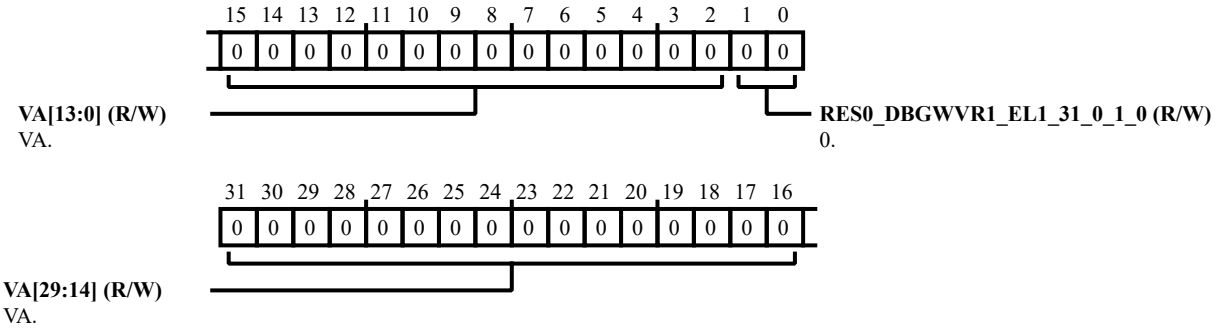


Figure 48-664: A55_C0_DBGWVR1_EL1_31_0 Register Diagram

Table 48-671: A55_C0_DBGWVR1_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR1_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR1_EL1.

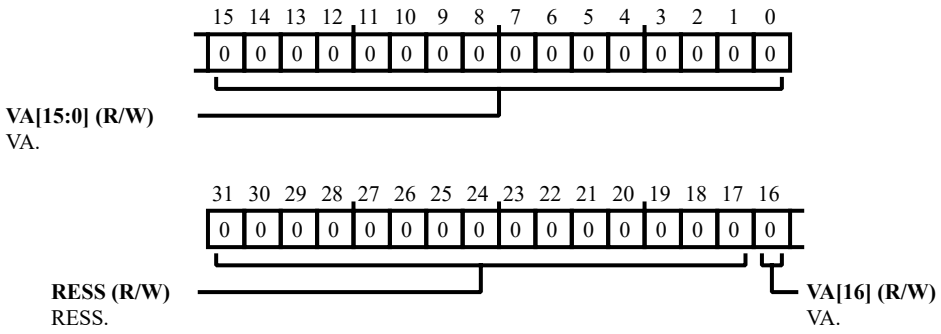


Figure 48-665: A55_C0_DBGWVR1_EL1_63_32 Register Diagram

Table 48-672: A55_C0_DBGWVR1_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR2_EL1.

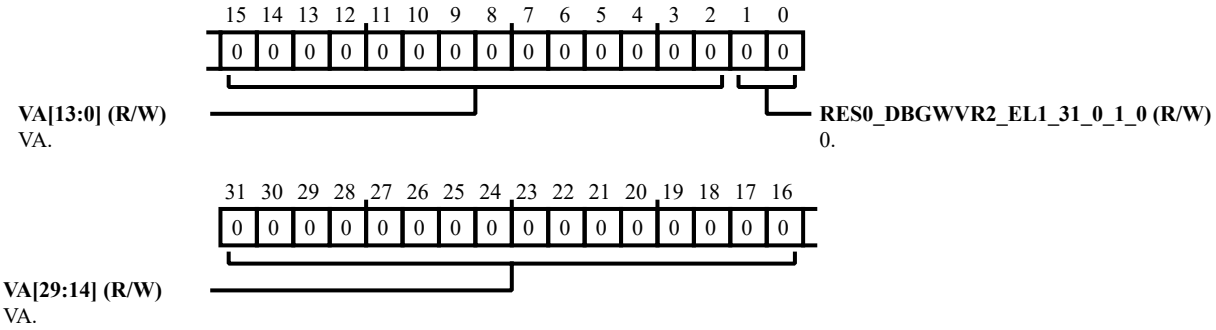


Figure 48-666: A55_C0_DBGWVR2_EL1_31_0 Register Diagram

Table 48-673: A55_C0_DBGWVR2_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR2_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR2_EL1.

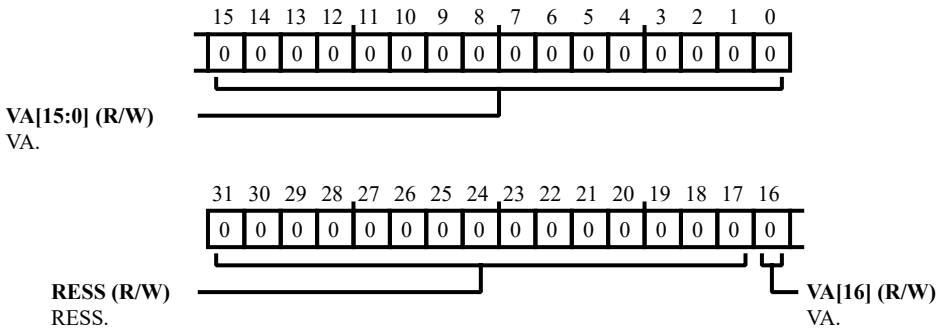


Figure 48-667: A55_C0_DBGWVR2_EL1_63_32 Register Diagram

Table 48-674: A55_C0_DBGWVR2_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR3_EL1.

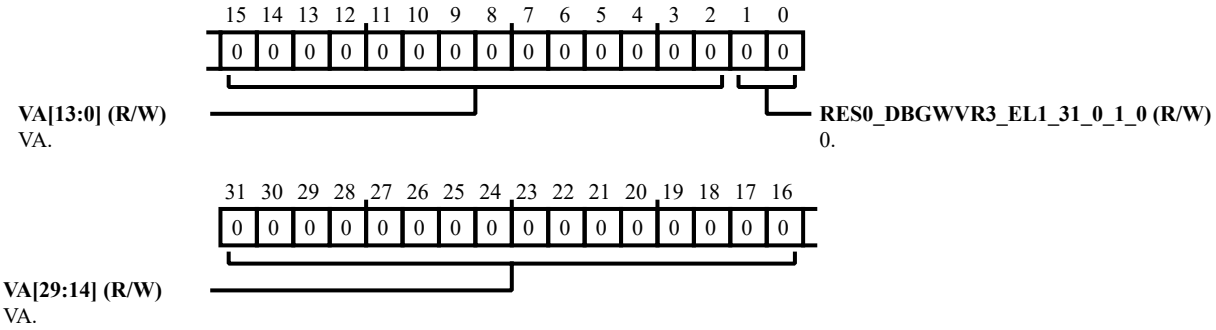


Figure 48-668: A55_C0_DBGWVR3_EL1_31_0 Register Diagram

Table 48-675: A55_C0_DBGWVR3_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR3_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR3_EL1.

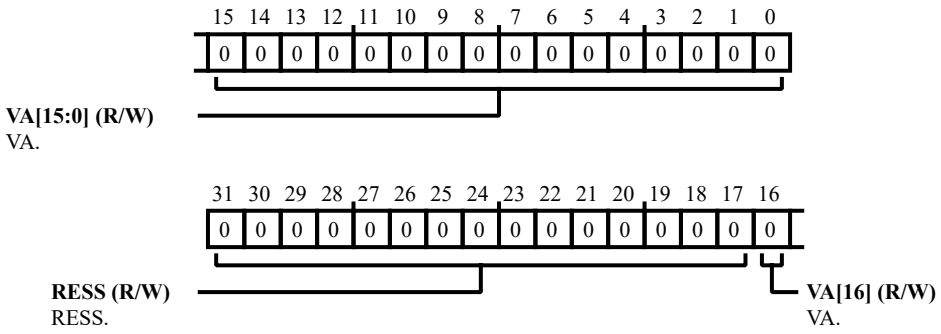


Figure 48-669: A55_C0_DBGWVR3_EL1_63_32 Register Diagram

Table 48-676: A55_C0_DBGWVR3_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

External Debug Context ID Sample Register.

Contains the sampled value of the Context ID, captured on reading the low half of EDPCSR.

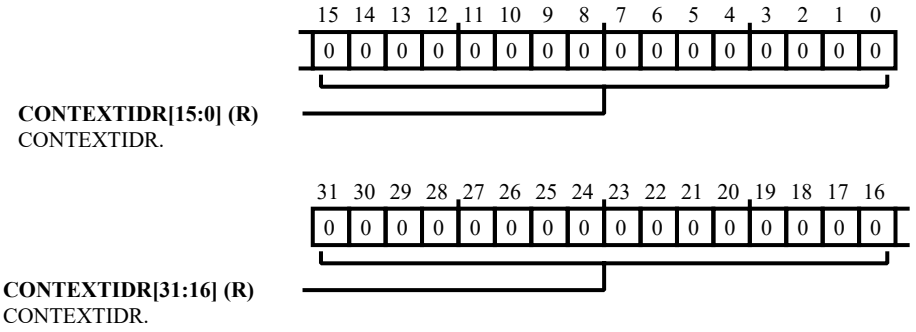


Figure 48-670: A55_C0_EDCIDSR Register Diagram

Table 48-677: A55_C0_EDCIDSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CONTEXTIDR	CONTEXTIDR.. The sampled value of the Context ID, captured on reading the low half of EDPCSR. If EL1 is using AArch64, the Context ID is held in CONTEXTIDR_EL1. If EL1 is using AArch32, the Context ID is held in CONTEXTIDR. If EL3 is implemented and is using AArch32 then CONTEXTIDR is a Banked register, and EDCIDSR samples the current Banked copy of CONTEXTIDR.

External Debug Exception Catch Control Register.

Controls Exception Catch debug events.

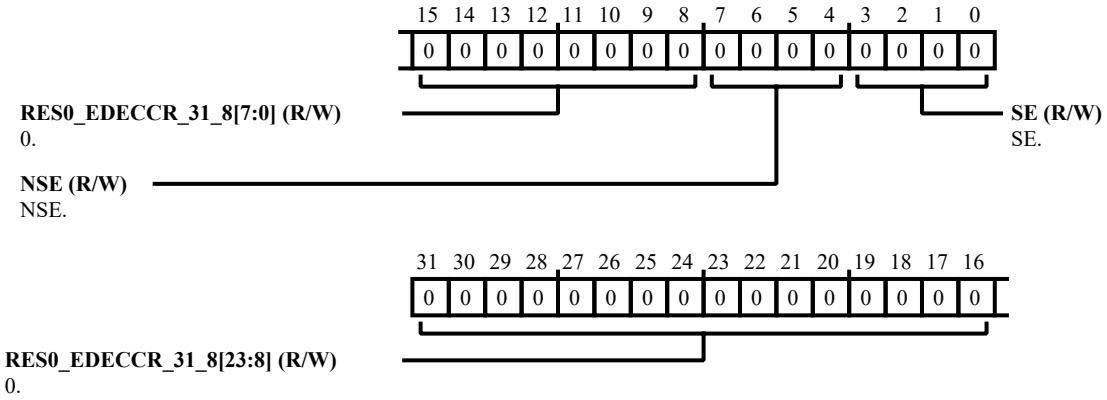


Figure 48-671: A55_C0_EDECCR Register Diagram

Table 48-678: A55_C0_EDECCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	RES0_EDECCR_31_8	0.. Reserved, RES0.	
7:4 (R/W)	NSE	NSE.. Coarse-grained Non-secure exception catch. If EL3 and EL2 are not implemented and the PE behaves as if SCR_EL3.NS is set to 0, this field is reserved, RES0. Otherwise, possible values for this field are: 0000 Exception Catch debug event disabled for Non-secure Exception levels. 0010 Exception Catch debug event enabled for Non-secure EL1. 0100 Exception Catch debug event enabled for Non-secure EL2. 0110 Exception Catch debug event enabled for Non-secure EL1 and EL2. All other values are reserved. A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then: The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR. The value returned for NSE by a read of EDECCR is UNKNOWN.	
		0	Exception Catch debug event disabled for Non-secure Exception levels.
		2	Exception Catch debug event enabled for Non-secure EL1.
		4	Exception Catch debug event enabled for Non-secure EL2.
		6	Exception Catch debug event enabled for Non-secure EL1 and EL2.

Table 48-678: A55_C0_EDECCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	SE	SE.. Coarse-grained Secure exception catch. If EL3 is not implemented and the PE behaves as if SCR_EL3.NS is set to 1, this field is reserved, RES0. Otherwise, possible values for this field are: 0000 Exception Catch debug event disabled for Secure Exception levels. 0010 Exception Catch debug event enabled for Secure EL1. 1000 Exception Catch debug event enabled for Secure EL3. 1010 Exception Catch debug event enabled for Secure EL1 and EL3. All other values are reserved. A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then: The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR. The value returned for SE by a read of EDECCR is UNKNOWN.	
		0	Exception Catch debug event disabled for Secure Exception levels.
		2	Exception Catch debug event enabled for Secure EL1.
		8	Exception Catch debug event enabled for Secure EL3.
		10	Exception Catch debug event enabled for Secure EL1 and EL3.

External Debug Event Status Register.

Indicates the status of internally pending Halting debug events.

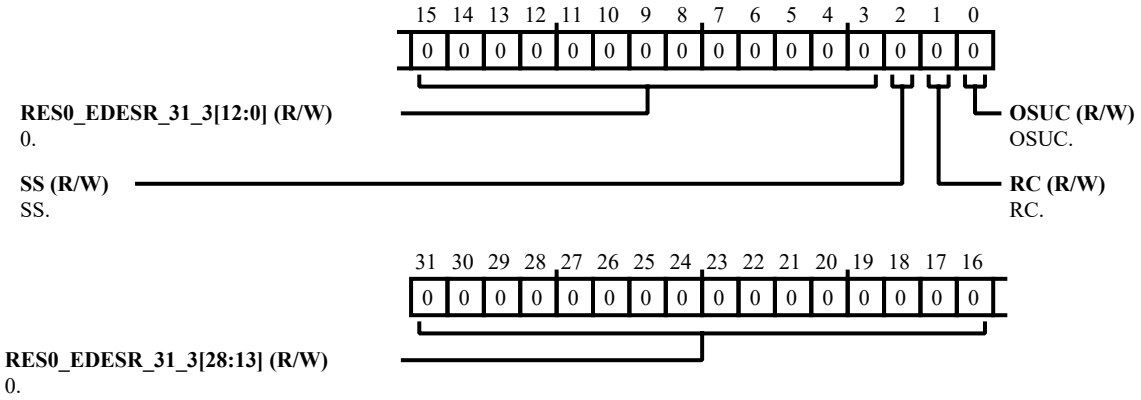


Figure 48-672: A55_C0_EDESR Register Diagram

Table 48-679: A55_C0_EDESR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/W)	RES0_EDESR_31_3	0.. Reserved, RES0.	
2 (R/W)	SS	SS.. Halting step debug event pending. Possible values of this field are: 0 Reading this means that a Halting step debug event is not pending. Writing this means no action. 1 Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.	
		0	Reading this means that a Halting step debug event is not pending. Writing this means no action.
		1	Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.
1 (R/W)	RC	RC.. Reset Catch debug event pending. Possible values of this field are: 0 Reading this means that a Reset Catch debug event is not pending. Writing this means no action. 1 Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.	
		0	Reading this means that a Reset Catch debug event is not pending. Writing this means no action.
		1	Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.

Table 48-679: A55_C0_EDESR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	OSUC	OSUC.. OS Unlock Catch debug event pending. Possible values of this field are: 0 Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action. 1 Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.	
		0	Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action.
		1	Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.

External Debug Integration mode Control register.

Enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

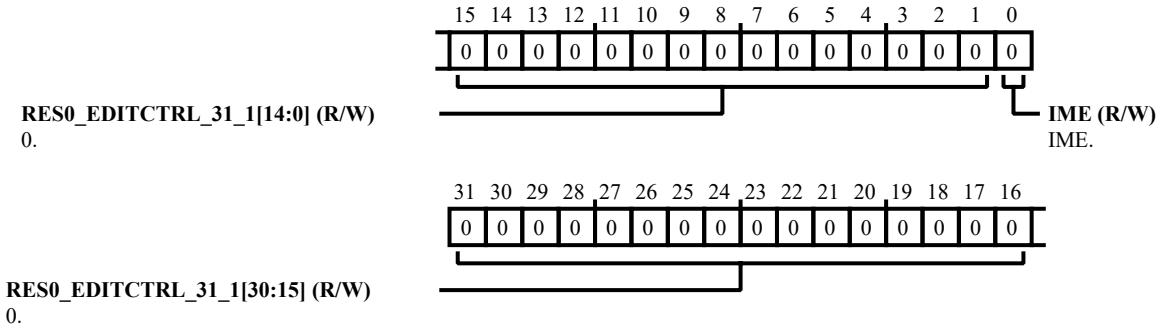


Figure 48-673: A55_C0_EDITCTRL Register Diagram

Table 48-680: A55_C0_EDITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_EDITCTRL_31_1	0.. Reserved, RES0.	
0 (R/W)	IME	IME.. Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED. 0 Normal operation. 1 Integration mode enabled.	
		0	Normal operation.
		1	Integration mode enabled.

External Debug Instruction Transfer Register.

Used in Debug state for passing instructions to the PE for execution.

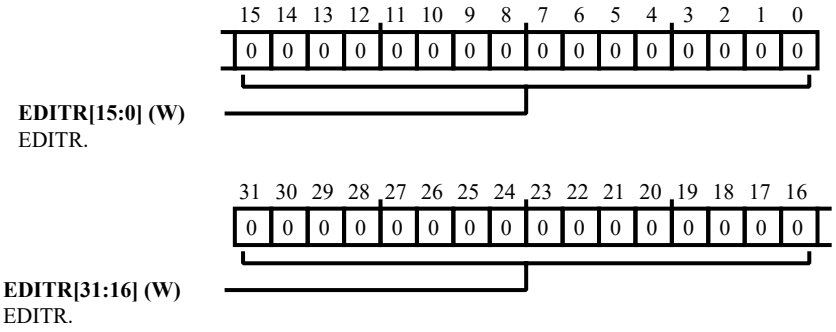


Figure 48-674: A55_C0_EDITR Register Diagram

Table 48-681: A55_C0_EDITR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	EDITR	EDITR.. Used in Debug state for passing instructions to the processor for execution.

External Debug Power/Reset Control Register.

Controls the PE functionality related to powerup, reset, and powerdown.

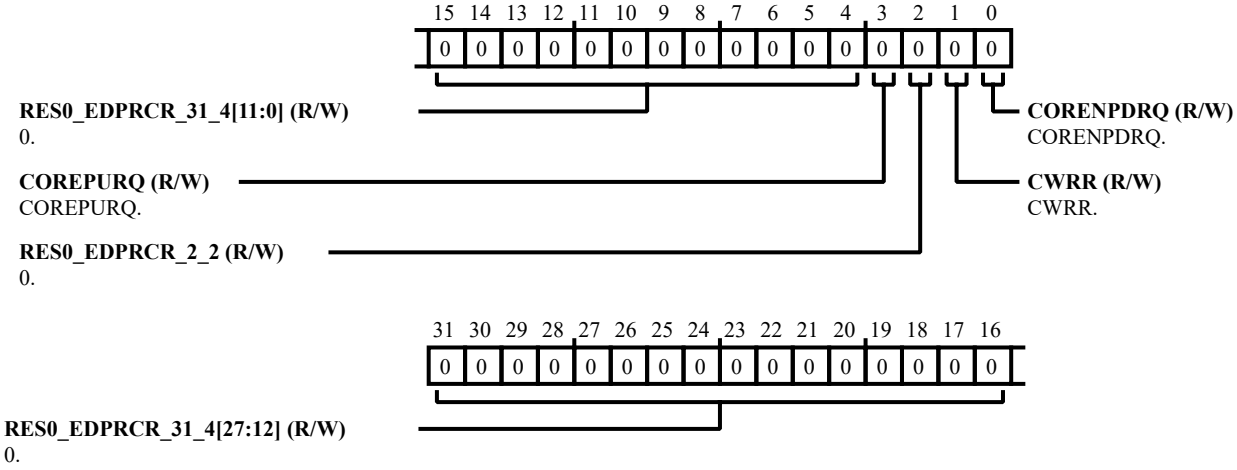


Figure 48-675: A55_C0_EDPRCR Register Diagram

Table 48-682: A55_C0_EDPRCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RES0_EDPRCR_31_4	0.. Reserved, RES0.	
3 (R/W)	COREPURQ	COREPURQ.. Core powerup request. Allows a debugger to request that the power controller power up the core, enabling access to the debug register in the Core power domain. The actions on writing to this bit are: 0 Do not request power up of the Core power domain. 1 Request power up of the Core power domain, and emulation of powerdown. In an implementation that includes the recommended external debug interface, this bit drives the DBGPWRUPREQ signal. Typically, this request is passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. This field is in the Debug power domain and can be read and written when the Core power domain is powered off. On an External debug reset this field resets to 0. The power controller must not allow the Core power domain to switch off while this bit is one.	
		0	Do not request power up of the Core power domain.
		1	Request power up of the Core power domain, and emulation of powerdown.
2 (R/W)	RES0_EDPRCR_2_2	0.. Reserved, RES0.	

Table 48-682: A55_C0_EDPRCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	CWRR	CWRR.. Warm reset request. Write only bit that reads as zero. The actions on writing to this bit are: 0 No action. 1 Request Warm reset. The PE ignores writes to this bit if any of the following are true: ExternalInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Non-Secure state. External-SecureInvasiveDebugEnabled() == FALSE and one of the following is true:EL3 is implemented.The implemented Security state is Secure state. The Core power domain is either off or in a low-power state where the Core power domain registers cannot be accessed. DoubleLockStatus() == TRUE (OS Double Lock is set). OSLSR.OSLK == 1 (OS lock is locked). In an implementation that includes the recommended external debug interface, this bit drives the DBGRSTREQ signal. Whether a write to this bit initiates a Warm Reset, the extent of the reset is IMPLEMENTATION DEFINED, but must be one of: The request is ignored. Only this PE is Warm reset. This PE and other components of the system, possibly including other PEs, are Warm reset. Although the ARM architecture permits the first option from the above list, ARM recommends that either of the other options is implemented.	
		0	No action.
		1	Request Warm reset.
0 (R/W)	CORENPDRQ	CORENPDRQ.. Core no powerdown request. Requests emulation of powerdown. Possible values of this bit are: 0 If the system responds to a powerdown request, it powers down Core power domain. 1 If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. This bit is UNKNOWN, and the PE ignores writes to this bit if any of the following are true: The Core power domain is either off or in a low-power state where the Core power domain registers cannot be accessed. DoubleLockStatus() == TRUE (OS Double Lock is set). OSLSR.OSLK == 1 (OS lock is locked).	
		0	If the system responds to a powerdown request, it powers down Core power domain.
		1	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

External Debug Processor Status Register.

Holds information about the reset and powerdown state of the PE.

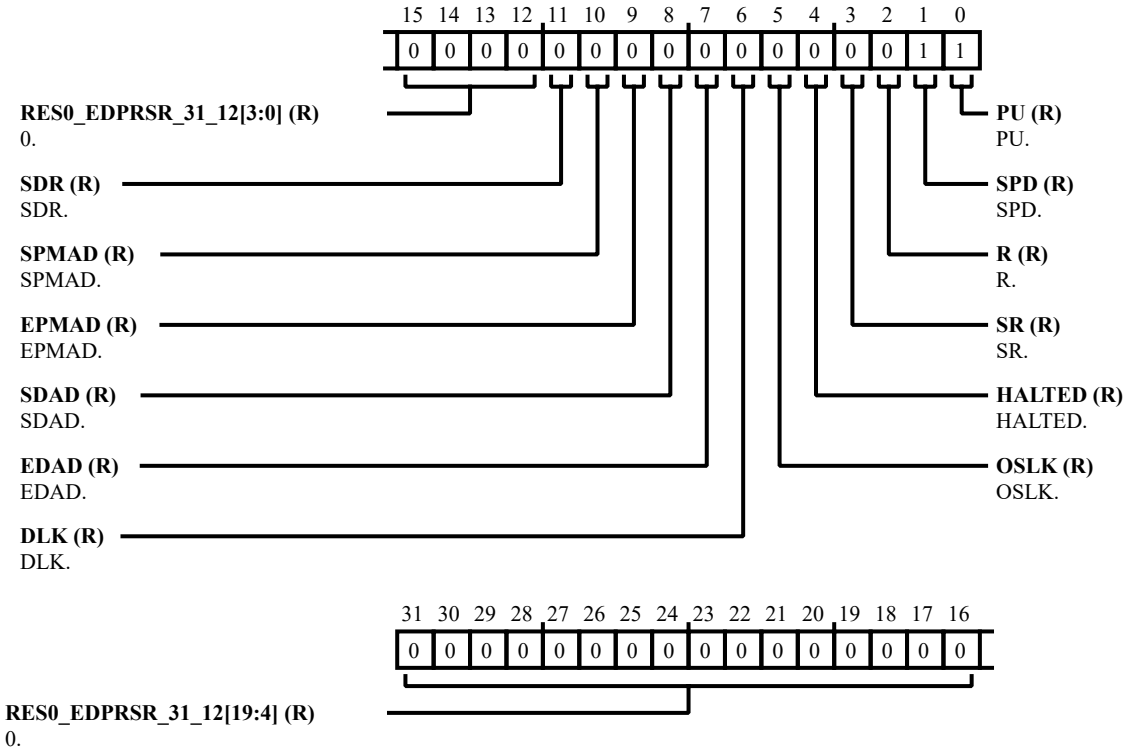


Figure 48-676: A55_C0_EDPRSR Register Diagram

Table 48-683: A55_C0_EDPRSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_EDPRSR_31_12	0.. Reserved, RES0.

Table 48-683: A55_C0_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	SDR	SDR.. Sticky debug restart. Set to 1 when the PE exits Debug state. This bit is UNKNOWN on reads if any of the following are true: DoubleLockStatus() == TRUE (EDPRSR.DLK == 1). The OS double-lock is locked. EDPRSR.R == 1. The PE is in Reset state. EDPRSR.PU == 0. The Core power domain is powered down. Otherwise permitted values are: 0 The PE has not restarted since EDPRSR was last read. 1 The PE has restarted since EDPRSR was last read. If a reset occurs when the PE is in Debug state, the PE exits Debug state. SDR is UNKNOWN on Warm reset, meaning a debugger must also use the SR bit to determine whether the PE has left Debug state. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs.	
		0	The PE has not restarted since EDPRSR was last read.
		1	The PE has restarted since EDPRSR was last read.
10 (R/NW)	SPMAD	SPMAD.. Sticky EPMAD error. Set to 1 if an external debug interface access to a Performance Monitors register returns an error because AllowExternalPMUAccess() == FALSE. This bit is UNKNOWN on reads if any of the following are true: EDPRSR.{DLK, OSLK, R} is 1. EDPRSR.PU is 0. Otherwise permitted values are: 0 No accesses to the external Performance Monitors registers have failed since EDPRSR was last read. 1 At least one access to the external Performance Monitors registers has failed since EDPRSR was last read. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs. The write to SPMAD is an indirect write to EDPRSR that is a side effect of the access. The indirect write might not occur for a memory-mapped access to the external debug interface.	
		0	No accesses to the external Performance Monitors registers have failed since EDPRSR was last read.
		1	At least one access to the external Performance Monitors registers has failed since EDPRSR was last read.

Table 48-683: A55_C0_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	EPMAD	EPMAD.. External Performance Monitors access disable status. This bit is UNKNOWN on reads if any of EDPRSR.{DLK, OSLK, R} is 1, or EDPRSR.PU is 0. Otherwise permitted values are: 0 External Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE. 1 External Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE. If external performance monitors access is not implemented, EPMAD is RAO.	
		0	External Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE.
		1	External Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE.
8 (R/NW)	SDAD	SDAD.. Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because AllowExternalDebugAccess() == FALSE. This bit is UNKNOWN on reads if any of the following are true: Either of EDPRSR.{DLK, R} is 1. EDPRSR.PU is 0. EDPRSR.OSLK is 1 and external debug writes to OSLAR_EL1 do not return an error when AllowExternalDebugAccess() == FALSE. Otherwise permitted values are: 0 No accesses to the external debug registers have failed since EDPRSR was last read. 1 At least one access to the external debug registers has failed since EDPRSR was last read. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs. The write to SDAD is an indirect write to EDPRSR that is a side effect of the access. The indirect write might not occur for a memory-mapped access to the external debug interface.	
		0	No accesses to the external debug registers have failed since EDPRSR was last read.
		1	At least one access to the external debug registers has failed since EDPRSR was last read.
7 (R/NW)	EDAD	EDAD.. External debug access disable status. This bit is UNKNOWN on reads if any of the following are true: Either of EDPRSR.{DLK, R} is 1. EDPRSR.PU is 0. EDPRSR.OSLK is 1 and external debug writes to OSLAR_EL1 do not return an error when AllowExternalDebugAccess() == FALSE. Otherwise permitted values are: 0 External debug access enabled. AllowExternalDebugAccess() == TRUE. 1 External debug access disabled. AllowExternalDebugAccess() == FALSE.	
		0	External debug access enabled. AllowExternalDebugAccess() == TRUE.
		1	External debug access disabled. AllowExternalDebugAccess() == FALSE.

Table 48-683: A55_C0_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	DLK	DLK.. OS Double Lock status bit. Returns the result of the pseudocode function DoubleLockStatus(). This bit is UNKNOWN on reads if EDPRSR.PU is 0. Otherwise reads as zero if any of the following are true, that is when DoubleLockStatus() == FALSE: OSDLR_EL1.DLK == 0. DBGPRCR_EL1.CORENPDRQ == 1. The PE is in Debug state. If the Core power domain is powered up and DoubleLockStatus() == TRUE, it is IMPLEMENTATION DEFINED whether: EDPRSR.PU reads as 1, EDPRSR.DLK reads as 1, and EDPRSR.SPD is UNKNOWN. EDPRSR.PU reads as 0, EDPRSR.DLK is UNKNOWN, and EDPRSR.SPD reads as 0. If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see.	
5 (R/NW)	OSLK	OSLK.. OS lock status bit. This bit is UNKNOWN on reads if either of EDPRSR.{DLK, R} is 1 or EDPRSR.PU is 0. A read of this bit returns the value of OSLSR_EL1.OSLK.	
4 (R/NW)	HALTED	HALTED.. Halted status bit. This bit is UNKNOWN on reads if EDPRSR.PU is 0. Otherwise permitted values are: 0 PE in Non-debug state. 1 PE is in Debug state. Because the OS Double Lock is never set when the PE is in Debug state, this bit is always RAZ when EDPRSR.DLK is set to 1.	
		0	PE in Non-debug state.
		1	PE is in Debug state.
3 (R/NW)	SR	SR.. Sticky core reset status bit. This bit is UNKNOWN on reads if EDPRSR.DLK is 1 or EDPRSR.PU is 0. Otherwise permitted values are: 0 The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read. 1 The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0 if the non-debug logic of the PE is not in reset state. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs.	
		0	The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read.
		1	The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read.

Table 48-683: A55_C0_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	R	R.. PE reset status bit. This bit is UNKNOWN on reads if either EDPRSR.DLK is 1 or EDPRSR.PU is 0. Otherwise permitted values are: 0 The non-debug logic of the PE is not in reset state. 1 The non-debug logic of the PE is in reset state. If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see.	
		0	The non-debug logic of the PE is not in reset state.
		1	The non-debug logic of the PE is in reset state.
1 (R/NW)	SPD	SPD.. Sticky core powerdown status bit. This bit is UNKNOWN on reads if both EDPRSR.DLK and EDPRSR.PU are 1. Otherwise, permitted values are: 0 If EDPRSR.PU is 0, the debugger does not know whether the state of the debug registers in the Core power domain is lost. If the Core power domain is in a powerup state (EDPRSR.PU is 1), the state of the debug registers in the Core power domain has not been lost. 1 The EDPRSR.PU is 1, the state of the debug registers in the Core power domain has been lost. If DoubleLockStatus() == FALSE and the PE is not in the powerdown state, then following a read of EDPRSR, this bit clears to 0. If DoubleLockStatus() == TRUE and the PE is not in the powerdown state, it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs. When the value of EDPRSR.PU is 0 indicating that the Core power domain is in either retention or powerdown state, EDPRSR.SPD is not cleared following a read of EDPRSR, for the IMPLEMENTATION DEFINED behavior see EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see.	
		0	If EDPRSR.PU is 0, the debugger does not know whether the state of the debug registers in the Core power domain is lost. If the Core power domain is in a powerup state (EDPRSR.PU is 1), the state of the debug registers in the Core power domain has not been lost.
		1	The EDPRSR.PU is 1, the state of the debug registers in the Core power domain has been lost.

Table 48-683: A55_C0_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	PU	PU.. Core powerup status bit. Indicates whether the Core power domain debug registers can be accessed. The value of EDPRSR.PU is IMPLEMENTATION DEFINED when the Core power domain is powered-up and OS double-lock is locked. See the description of DLK for more information. Otherwise, permitted values are: 0 Core is in a low-power or powerdown state where the debug registers cannot be accessed. 1 Core is in a powerup state where the debug registers can be accessed. If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see.	
		0	Core is in a low-power or powerdown state where the debug registers cannot be accessed.
		1	Core is in a powerup state where the debug registers can be accessed.

External Debug Reserve Control Register.

This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

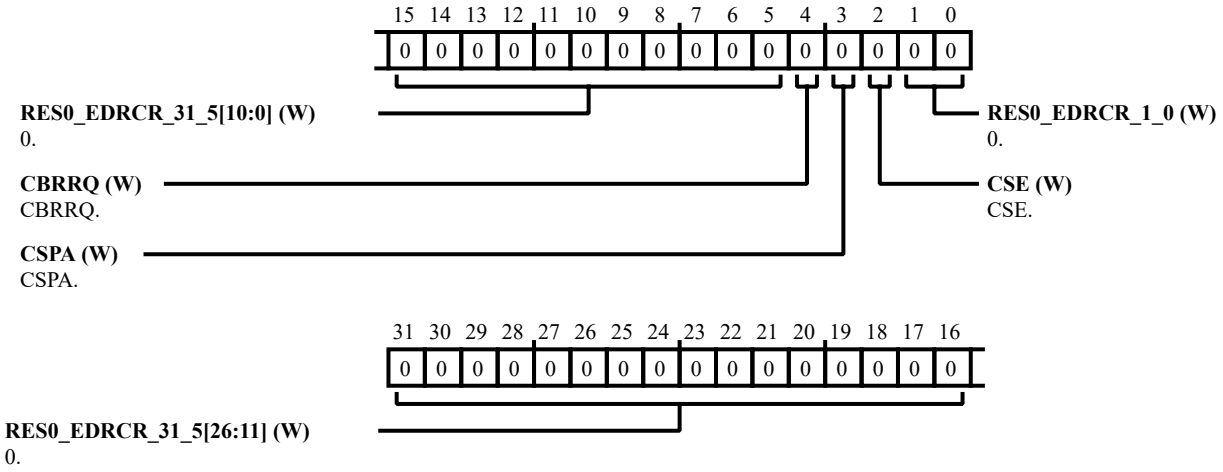


Figure 48-677: A55_C0_EDRCR Register Diagram

Table 48-684: A55_C0_EDRCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (RX/W)	RES0_EDRCR_31_5	0.. Reserved, RES0.	
4 (RX/W)	CBRRQ	CBRRQ.. Allow imprecise entry to Debug state. The actions on writing to this bit are: 0 No action. 1 Allow imprecise entry to Debug state, for example by canceling pending bus accesses. Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1. This feature is optional. If this feature is not implemented, writes to this bit are ignored.	
		0	No action.
		1	Allow imprecise entry to Debug state, for example by canceling pending bus accesses.
3 (RX/W)	CSPA	CSPA.. Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The actions on writing to this bit are: 0 No action. 1 Clear the EDSCR.PipeAdv bit to 0.	
		0	No action.
		1	Clear the EDSCR.PipeAdv bit to 0.

Table 48-684: A55_C0_EDRCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (RX/W)	CSE	CSE.. Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The actions on writing to this bit are: 0 No action. 1 Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the EDSCR.ITO bit, to 0.	
		0	No action.
		1	Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the EDSCR.ITO bit, to 0.
1:0 (RX/W)	RES0_EDRCR_1_0	0.. Reserved, RES0.	

External Debug Status and Control Register.

Main control register for the debug implementation.

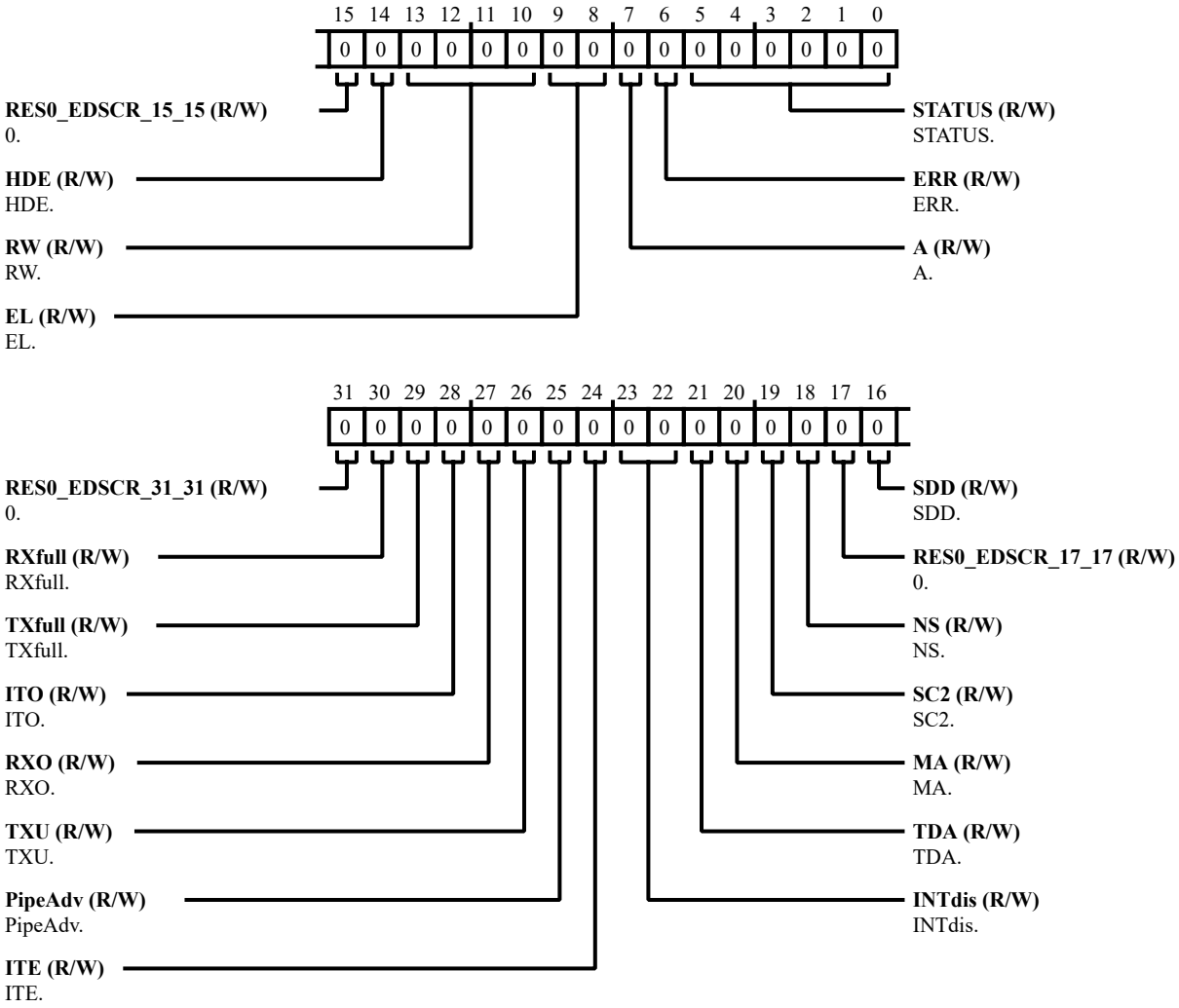


Figure 48-678: A55_C0_EDSCR Register Diagram

Table 48-685: A55_C0_EDSCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	RES0_EDSCR_31_31	0.. Reserved, RES0.
30 (R/W)	RXFULL	RXfull.. DTRRX full. This bit is RO.

Table 48-685: A55_C0_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	TXFULL	TXfull.. DTRTX full. This bit is RO.
28 (R/W)	ITO	ITO.. ITR overrun. This bit is RO. If the PE is in Non-debug state, this bit is UNKNOWN. ITO is set to 0 on entry to Debug state.
27 (R/W)	RXO	RXO.. DTRRX overrun. This bit is RO.
26 (R/W)	TXU	TXU.. DTRTX underrun. This bit is RO.
25 (R/W)	PIPEADV	PipeAdv.. Pipeline advance. This bit is RO. Set to 1 every time the PE pipeline retires one or more instructions. Cleared to 0 by a write to EDRCR.CSPA. The architecture does not define precisely when this bit is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing.
24 (R/W)	ITE	ITE.. ITR empty. This bit is RO. If the PE is in Non-debug state, this bit is UNKNOWN. It is always valid in Debug state.

Table 48-685: A55_C0_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/W)	INTDIS	INTdis.. Interrupt disable. Disables taking interrupts (including virtual interrupts and System Error interrupts) in Non-Debug state. If external invasive debug is disabled, the value of this field is ignored. If external invasive debug is enabled, the possible values of this field are: 00 Do not disable interrupts. 01 Disable interrupts taken to Non-secure EL1. 10 Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable interrupts taken to Secure EL1. 11 Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable all other interrupts. The value of INTdis does not affect whether an interrupt is a WFI wake-up event, but can mask an interrupt as a WFE wake-up event. If EL3 and EL2 are not implemented, the values 0b01 and 0b10 are reserved. If programmed with a reserved value the PE behaves as if INTdis has been programmed with a defined value, other than for a direct read of EDSCR, and the value returned by a read of EDSCR.INTdis is UNKNOWN.	
		0	Do not disable interrupts.
		1	Disable interrupts taken to Non-secure EL1.
		2	Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable interrupts taken to Secure EL1.
		3	Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable all other interrupts.
21 (R/W)	TDA	TDA.. Traps accesses to the following Debug System registers: AArch64: DBGBCR<n>_EL1, DBGBVR<n>_EL1, DBGWCR<n>_EL1, DBGWVR<n>_EL1. AArch32: DBGBCR<n>, DBGBVR<n>, DBGBXVR<n>, DBGWCR<n>, DBGWVR<n>. The possible values of this field are: 0 Accesses to Debug System registers do not generate a Software Access debug event. 1 Accesses to Debug System registers generate a Software Access debug event, if OSLSR.OSLK is 0 and if halting is allowed.	
		0	Accesses to Debug System registers do not generate a Software Access debug event.
		1	Accesses to Debug System registers generate a Software Access debug event, if OSLSR.OSLK is 0 and if halting is allowed.

Table 48-685: A55_C0_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	MA	MA.. Memory access mode. Controls use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state. Possible values of this field are: 0 Normal access mode. 1 Memory access mode.	
		0	Normal access mode.
		1	Memory access mode.
19 (R/W)	SC2	SC2.. Sample CONTEXTIDR_EL2. Controls whether the Sample-based Profiling Extension samples CONTEXTIDR_EL2 or VTTBR_EL2.VMID. 0 Sample VTTBR_EL2.VMID. 1 Sample CONTEXTIDR_EL2.	
		0	Sample VTTBR_EL2.VMID.
		1	Sample CONTEXTIDR_EL2.
18 (R/W)	NS	NS.. Non-secure status. Read-only. When in Debug state, gives the current Security state: 0 Secure state, IsSecure() == TRUE. 1 Non-secure state, IsSecure() == FALSE. In Non-debug state, this bit is UNKNOWN.	
		0	Secure state, IsSecure() == TRUE.
		1	Non-secure state, IsSecure() == FALSE.
17 (R/W)	RES0_EDSCR_17_17	0.. Reserved, RES0.	
16 (R/W)	SDD	SDD.. Secure debug disabled. This bit is RO. On entry to Debug state: If entering in Secure state, SDD is set to 0. If entering in Non-secure state, SDD is set to the inverse of ExternalSecureInvasiveDebugEnabled(). In Debug state, the value of the SDD bit does not change, even if ExternalSecureInvasiveDebugEnabled() changes. In Non-debug state: SDD returns the inverse of ExternalSecureInvasiveDebugEnabled(). If the authentication signals that control ExternalSecureInvasiveDebugEnabled() change, a context synchronization operation is required to guarantee their effect. This bit is unaffected by the Security state of the PE. If EL3 is not implemented and the implementation is Non-secure, this bit is RES1.	
15 (R/W)	RES0_EDSCR_15_15	0.. Reserved, RES0.	

Table 48-685: A55_C0_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	HDE	HDE.. Halting debug enable. The possible values of this field are: 0 Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events. 1 Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.	
		0	Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events.
		1	Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.
13:10 (R/W)	RW	RW.. Exception level Execution state status. Read-only. In Debug state, each bit gives the current Execution state of each EL: RW Meaning 1111 All Exception levels are using AArch64. 1110 EL0 is using AArch32. All other Exception levels are using AArch64. 110x EL0 and EL1 are using AArch32. All other Exception levels are using AArch64. Never seen if EL2 is not implemented in the current Security state. 10xx EL0, EL1, and, if implemented in the current Security state, EL2 are using AArch32. All other Exception levels are using AArch64. 0xxx All Exception levels are using AArch32. However: The value of 1110 is only permitted at EL0. The values 110x are not permitted if either:EL2 is not implemented.EL3 is implemented and SCR_EL3.NS/ SCR.NS == 0. The values 10xx are not permitted if EL3 is not implemented. In Non-debug state, this field is RAO.	
9:8 (R/W)	EL	EL.. Exception level. Read-only. In Debug state, this gives the current EL of the PE. In Non-debug state, this field is RAZ.	
7 (R/W)	A	A.. System Error interrupt pending. Read-only. In Debug state, indicates whether a SError interrupt is pending: If HCR_EL2.{AMO, TGE} = {1, 0} and in Non-secure EL0 or EL1, a virtual SError interrupt. Otherwise, a physical SError interrupt. 0 No SError interrupt pending. 1 SError interrupt pending. A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted. UNKNOWN in Non-debug state.	
		0	No SError interrupt pending.
		1	SError interrupt pending.
6 (R/W)	ERR	ERR.. Cumulative error flag. This field is RO. It is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR.	

Table 48-685: A55_C0_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:0 (R/W)	STATUS	STATUS.. Debug status flags. This field is RO. The possible values of this field are: 000010 PE is in Non-debug state. 000001 PE is restarting, exiting Debug state. 000111 Breakpoint. 010011 External debug request. 011011 Halting step, normal. 011111 Halting step, exclusive. 100011 OS Unlock Catch. 100111 Reset Catch. 101011 Watchpoint. 101111 HLT instruction. 110011 Software access to debug register. 110111 Exception Catch. 111011 Halting step, no syndrome. All other values of STATUS are reserved.	
		1	PE is restarting, exiting Debug state.
		2	PE is in Non-debug state.
		7	Breakpoint.
		19	External debug request.
		27	Halting step, normal.
		31	Halting step, exclusive.
		35	OS Unlock Catch.
		39	Reset Catch.
		43	Watchpoint.
		47	HLT instruction.
		51	Software access to debug register.
		55	Exception Catch.
		59	Halting step, no syndrome.

External Debug Watchpoint Address Register.

Returns the virtual data address being accessed when a watchpoint debug event was triggered.

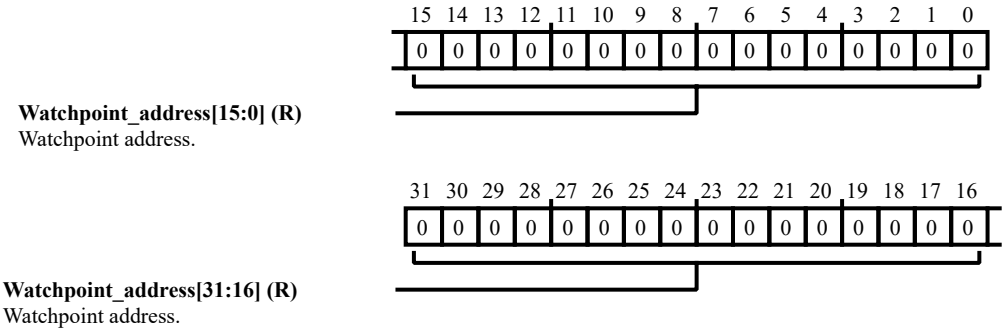


Figure 48-679: A55_C0_EDWAR_31_0 Register Diagram

Table 48-686: A55_C0_EDWAR_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	WATCHPOINT_ADDRESS	Watchpoint address.. Watchpoint address. The data virtual address being accessed when a watchpoint debug event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the DC ZVA block size. The value of this register is UNKNOWN if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event. The value of EDWAR[63:32] is UNKNOWN if Debug state was entered for a Watchpoint debug event taken from AArch32 state. The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See.

External Debug Watchpoint Address Register.

Returns the virtual data address being accessed when a watchpoint debug event was triggered.

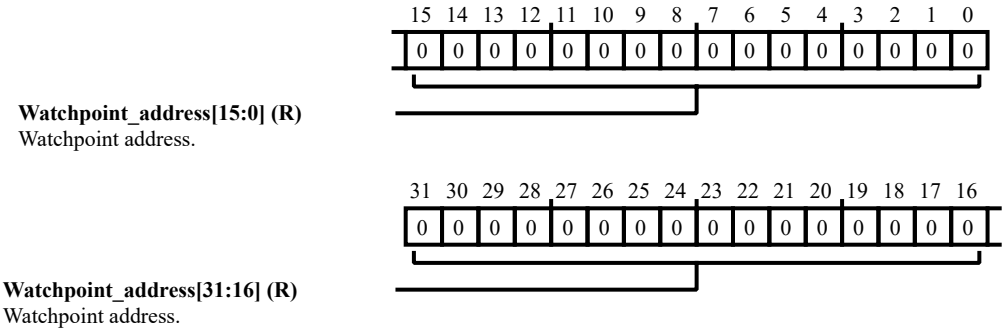


Figure 48-680: A55_C0_EDWAR_63_32 Register Diagram

Table 48-687: A55_C0_EDWAR_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	WATCHPOINT_ADDRESS	Watchpoint address.. Watchpoint address. The data virtual address being accessed when a watchpoint debug event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the DC ZVA block size. The value of this register is UNKNOWN if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event. The value of EDWAR[63:32] is UNKNOWN if Debug state was entered for a Watchpoint debug event taken from AArch32 state. The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See.

OS Lock Access Register.

Used to lock or unlock the OS lock.

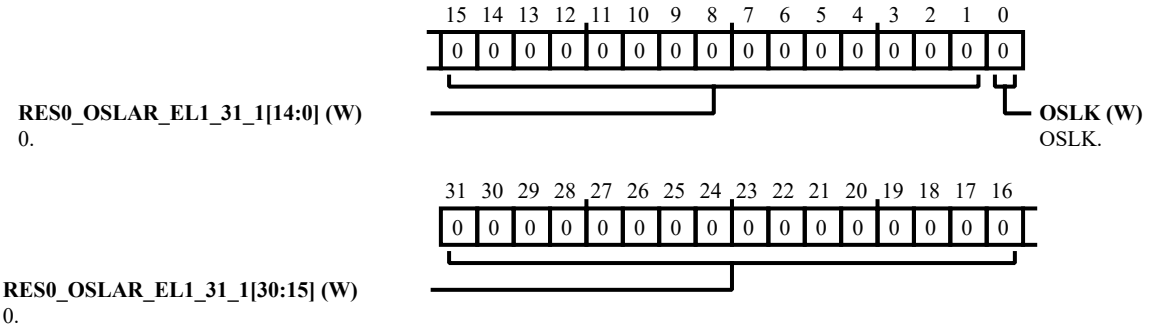


Figure 48-681: A55_C0_OSLAR_EL1 Register Diagram

Table 48-688: A55_C0_OSLAR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:1 (RX/W)	RES0_OSLAR_EL1_31_1	0.. Reserved, RES0.
0 (RX/W)	OSLK	OSLK.. On writes to OSLAR_EL1, bit[0] is copied to the OS lock. Use EDPRSR.OSLK to check the current status of the lock.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

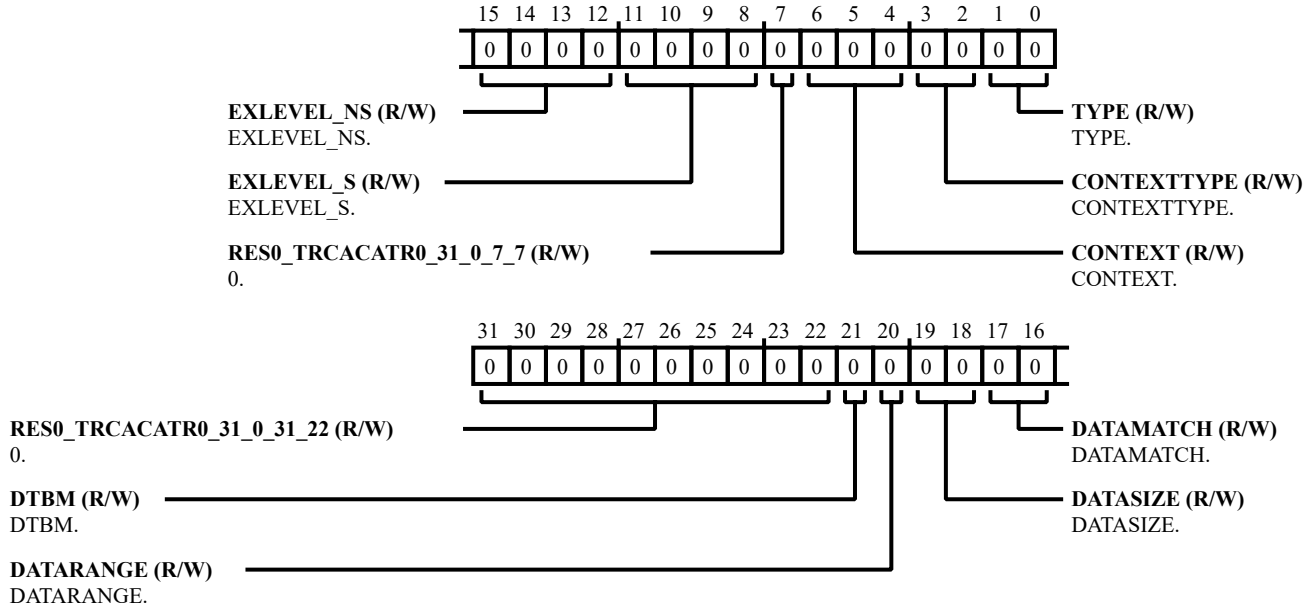


Figure 48-682: A55_C0_TRCACATR0_31_0 Register Diagram

Table 48-689: A55_C0_TRCACATR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR0_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-689: A55_C0_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-689: A55_C0_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR0_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-689: A55_C0_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-689: A55_C0_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

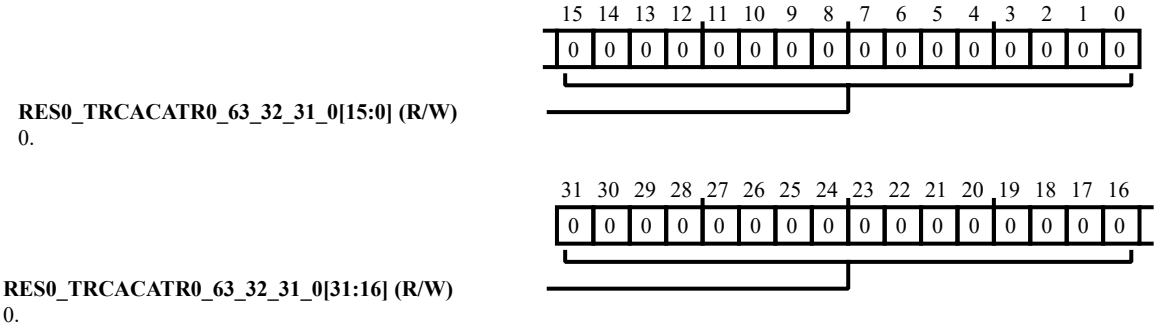


Figure 48-683: A55_C0_TRCACATR0_63_32 Register Diagram

Table 48-690: A55_C0_TRCACATR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR0_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

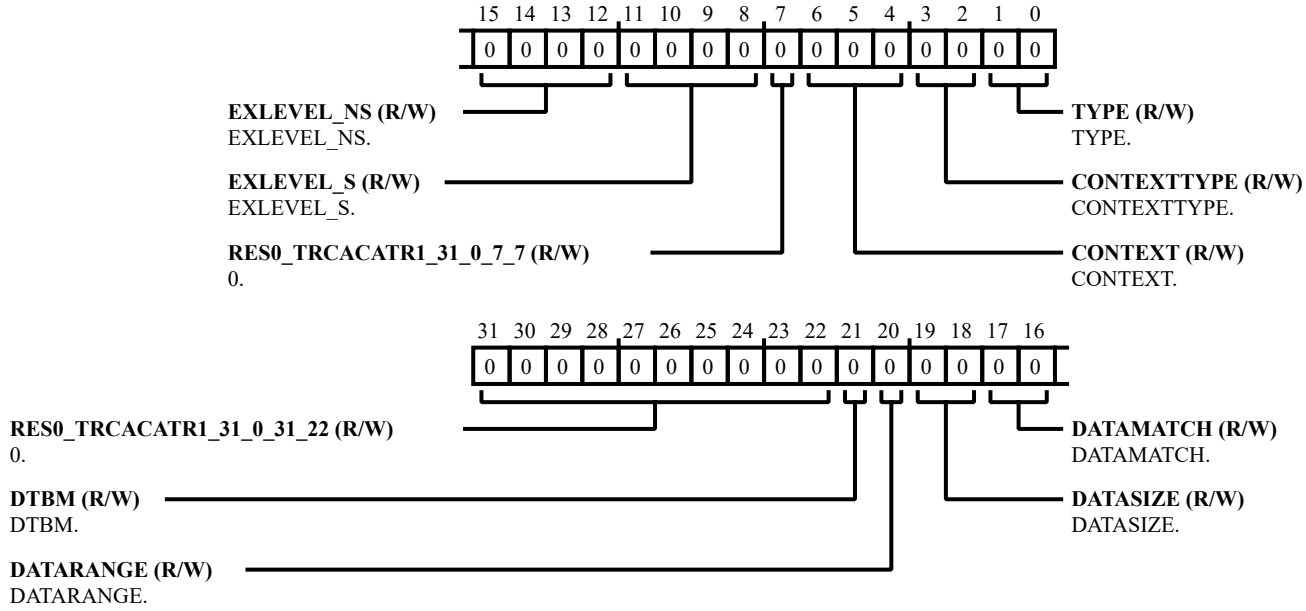


Figure 48-684: A55_C0_TRCACATR1_31_0 Register Diagram

Table 48-691: A55_C0_TRCACATR1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR1_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-691: A55_C0_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-691: A55_C0_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR1_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-691: A55_C0_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-691: A55_C0_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

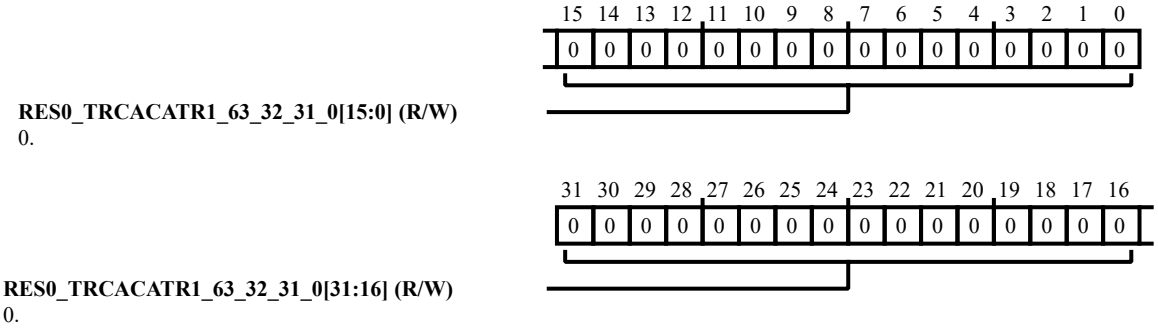


Figure 48-685: A55_C0_TRCACATR1_63_32 Register Diagram

Table 48-692: A55_C0_TRCACATR1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR1_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

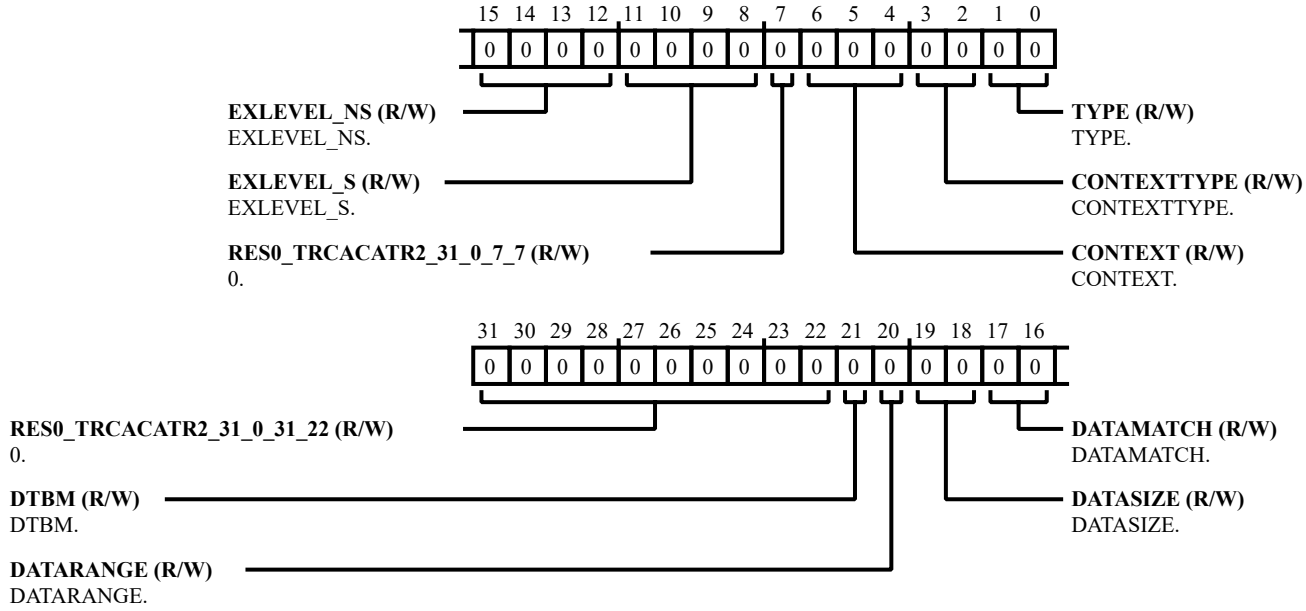


Figure 48-686: A55_C0_TRCACATR2_31_0 Register Diagram

Table 48-693: A55_C0_TRCACATR2_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR2_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-693: A55_C0_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-693: A55_C0_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR2_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-693: A55_C0_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-693: A55_C0_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

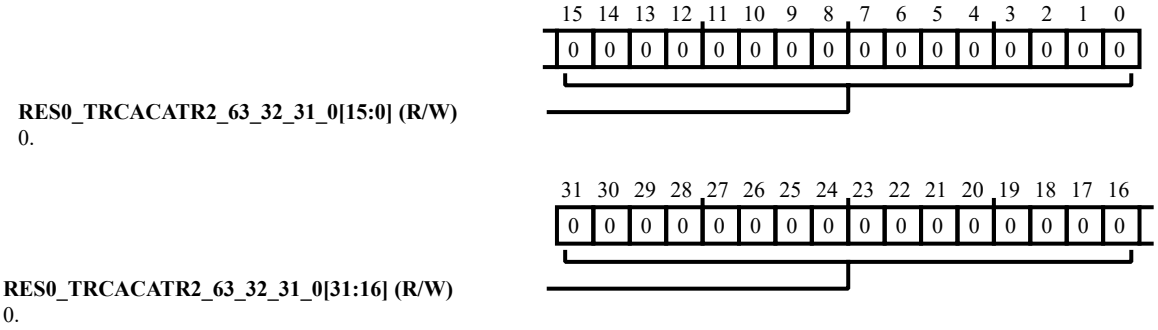


Figure 48-687: A55_C0_TRCACATR2_63_32 Register Diagram

Table 48-694: A55_C0_TRCACATR2_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR2_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

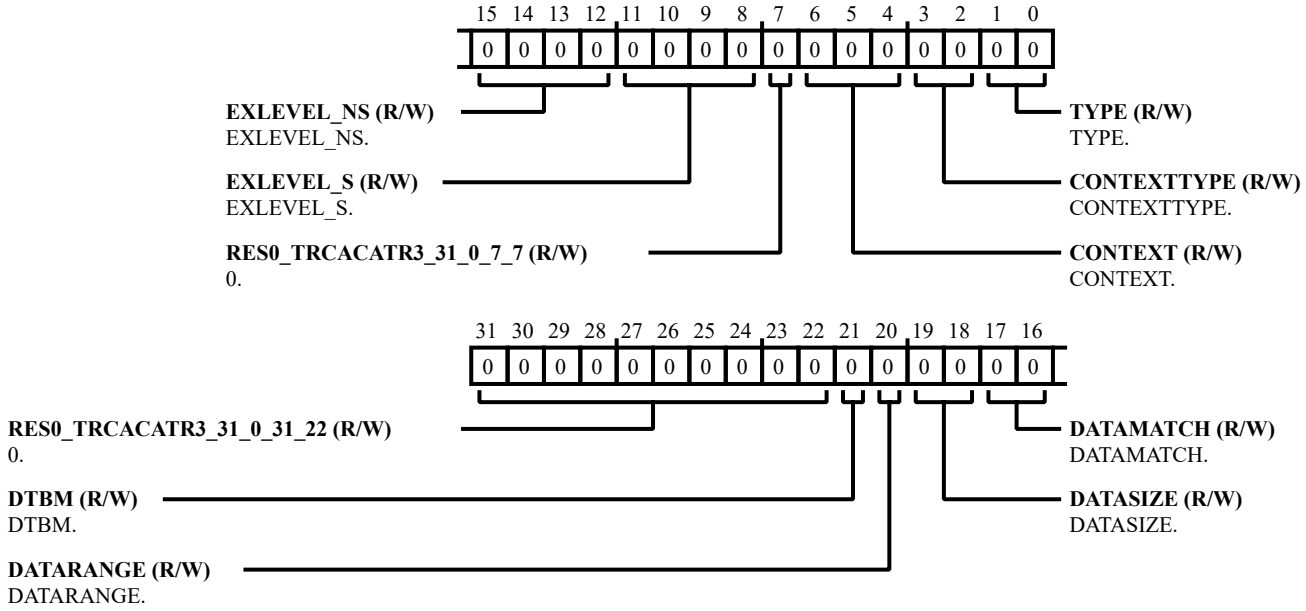


Figure 48-688: A55_C0_TRCACATR3_31_0 Register Diagram

Table 48-695: A55_C0_TRCACATR3_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR3_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-695: A55_C0_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-695: A55_C0_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR3_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-695: A55_C0_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-695: A55_C0_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

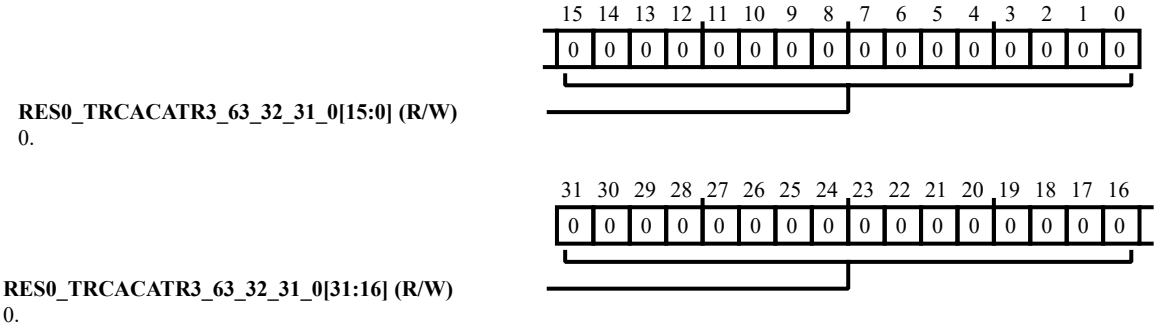


Figure 48-689: A55_C0_TRCACATR3_63_32 Register Diagram

Table 48-696: A55_C0_TRCACATR3_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR3_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

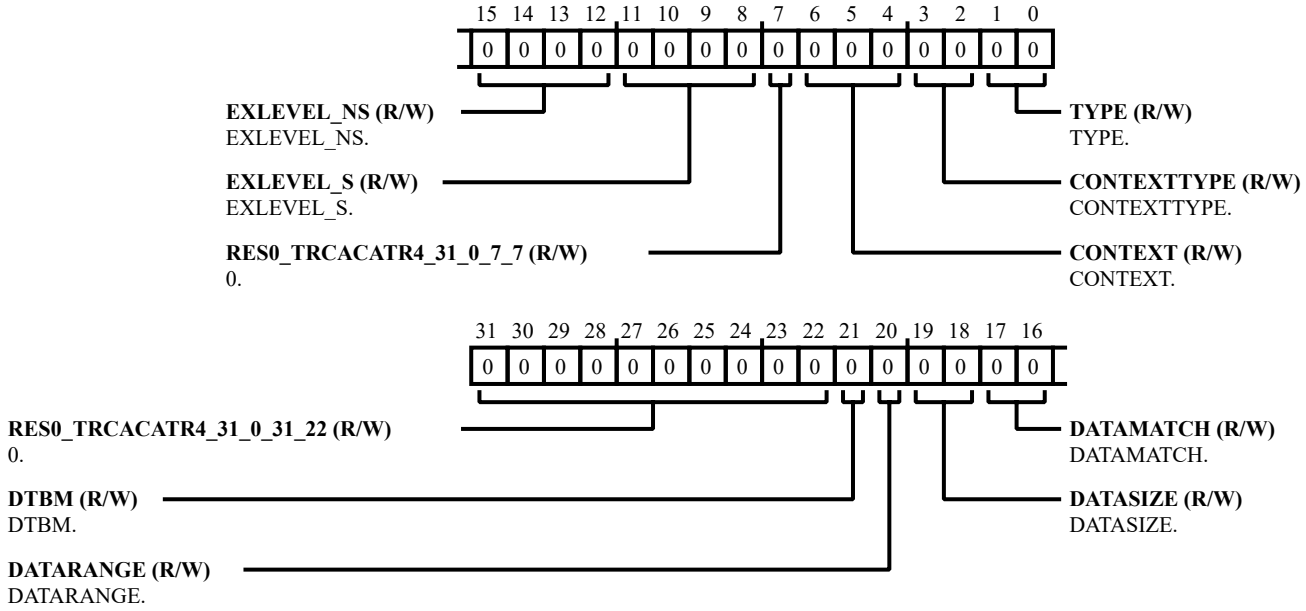


Figure 48-690: A55_C0_TRCACATR4_31_0 Register Diagram

Table 48-697: A55_C0_TRCACATR4_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR4_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-697: A55_C0_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-697: A55_C0_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR4_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-697: A55_C0_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-697: A55_C0_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

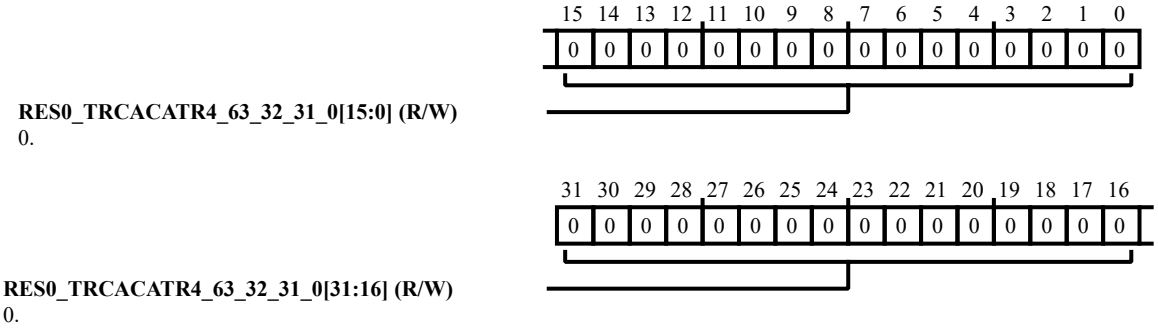


Figure 48-691: A55_C0_TRCACATR4_63_32 Register Diagram

Table 48-698: A55_C0_TRCACATR4_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR4_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

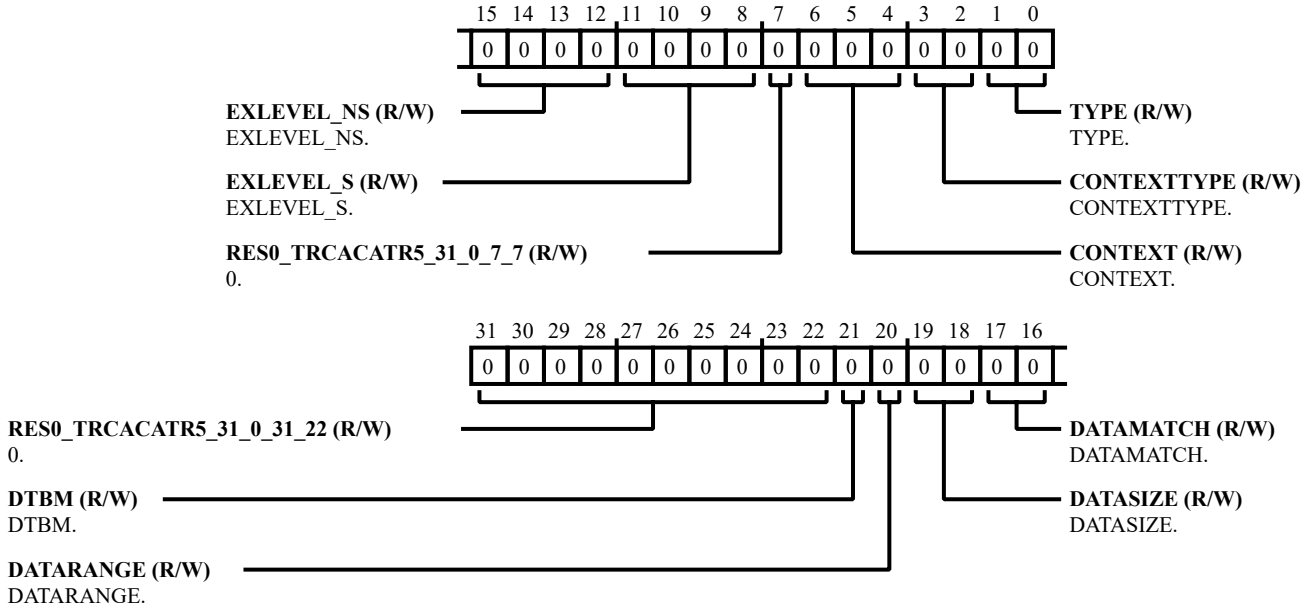


Figure 48-692: A55_C0_TRCACATR5_31_0 Register Diagram

Table 48-699: A55_C0_TRCACATR5_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR5_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-699: A55_C0_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-699: A55_C0_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR5_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-699: A55_C0_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-699: A55_C0_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

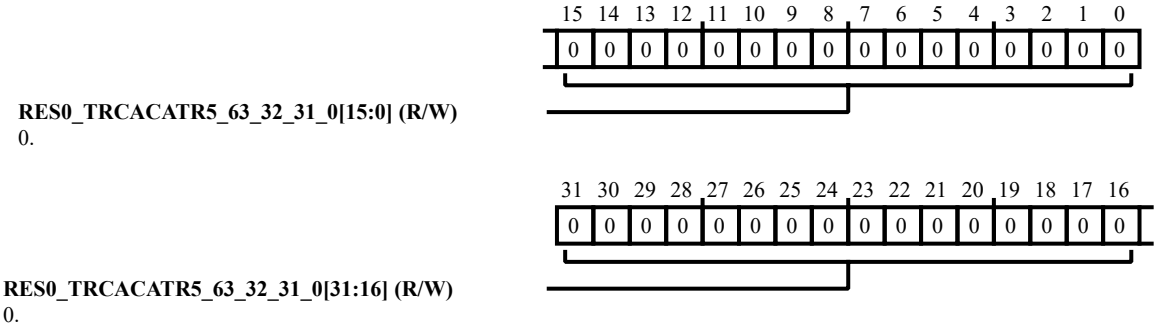


Figure 48-693: A55_C0_TRCACATR5_63_32 Register Diagram

Table 48-700: A55_C0_TRCACATR5_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR5_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

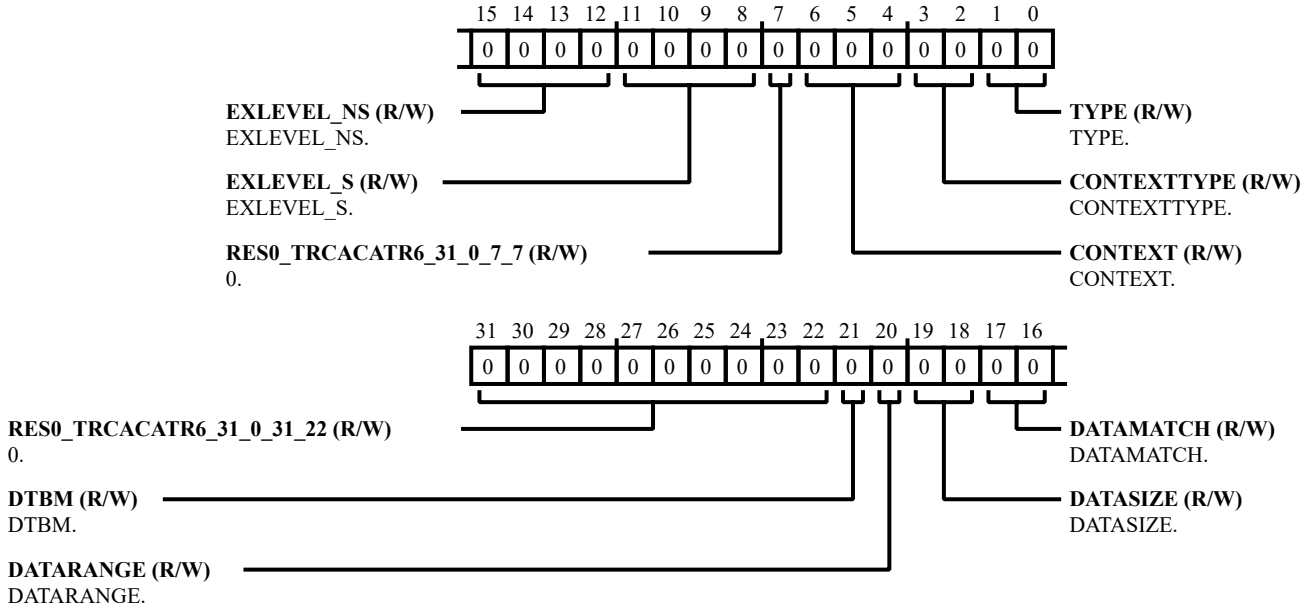


Figure 48-694: A55_C0_TRCACATR6_31_0 Register Diagram

Table 48-701: A55_C0_TRCACATR6_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR6_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-701: A55_C0_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-701: A55_C0_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR6_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-701: A55_C0_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-701: A55_C0_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

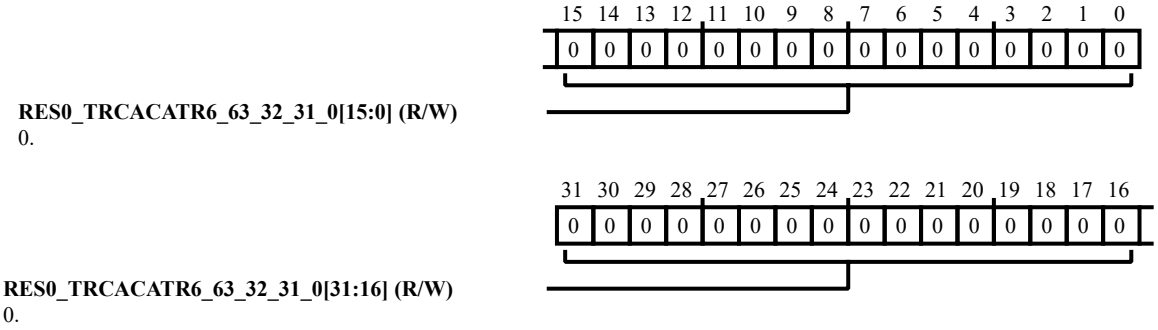


Figure 48-695: A55_C0_TRCACATR6_63_32 Register Diagram

Table 48-702: A55_C0_TRCACATR6_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR6_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

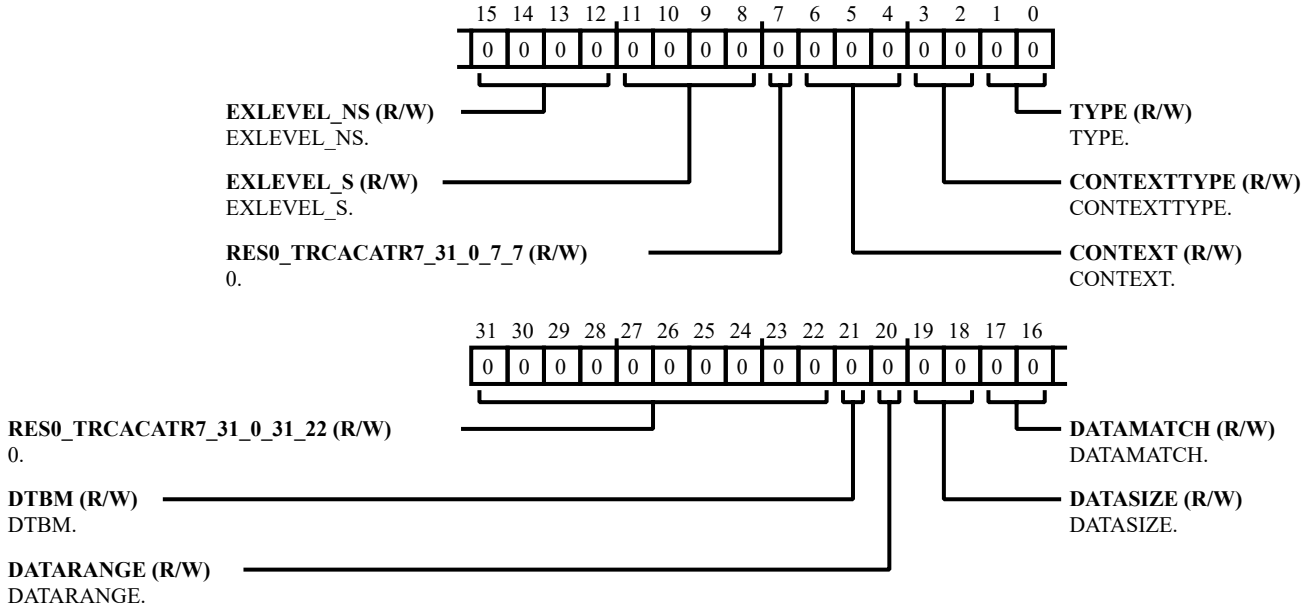


Figure 48-696: A55_C0_TRCACATR7_31_0 Register Diagram

Table 48-703: A55_C0_TRCACATR7_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR7_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-703: A55_C0_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-703: A55_C0_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR7_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-703: A55_C0_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-703: A55_C0_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

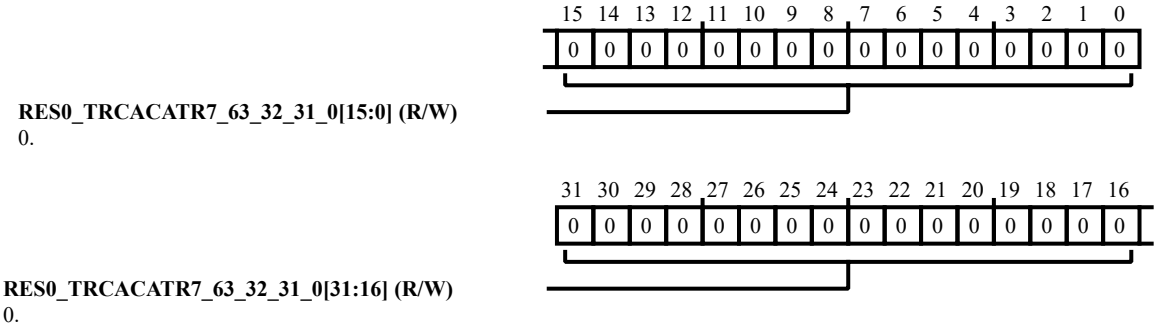


Figure 48-697: A55_C0_TRCACATR7_63_32 Register Diagram

Table 48-704: A55_C0_TRCACATR7_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR7_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Value Registers.

Contains an address value.

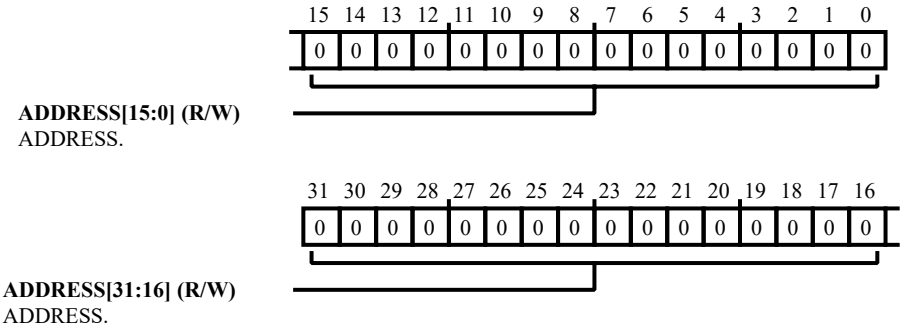


Figure 48-698: A55_C0_TRCACVR0_31_0 Register Diagram

Table 48-705: A55_C0_TRCACVR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

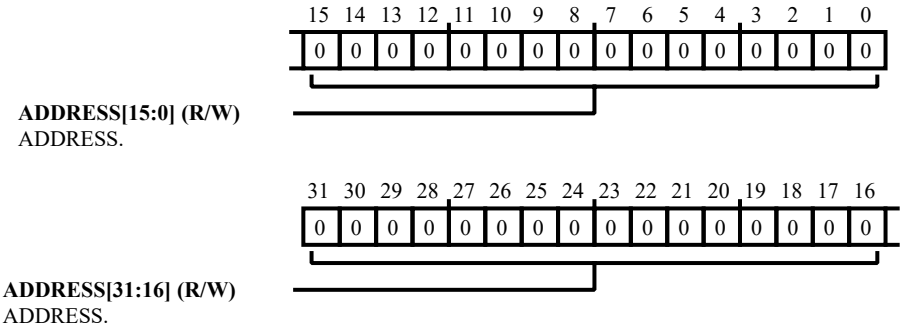


Figure 48-699: A55_C0_TRCACVR0_63_32 Register Diagram

Table 48-706: A55_C0_TRCACVR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

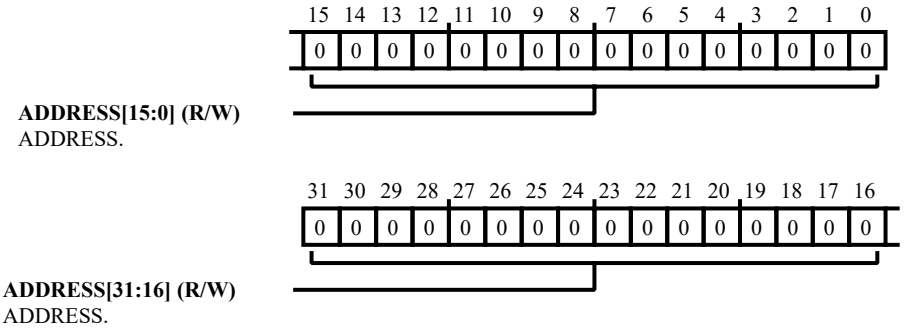


Figure 48-700: A55_C0_TRCACVR1_31_0 Register Diagram

Table 48-707: A55_C0_TRCACVR1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

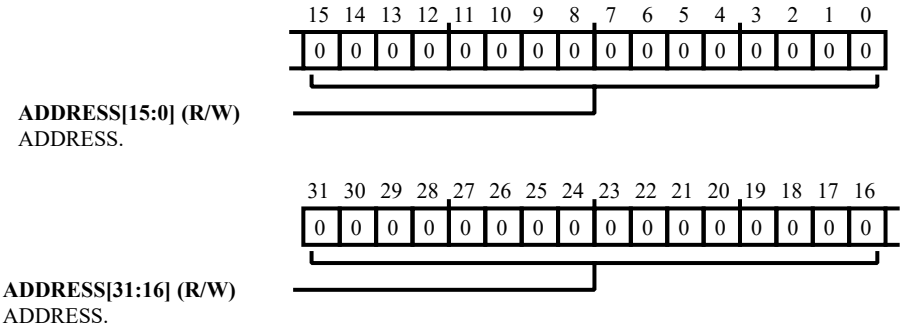


Figure 48-701: A55_C0_TRCACVR1_63_32 Register Diagram

Table 48-708: A55_C0_TRCACVR1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

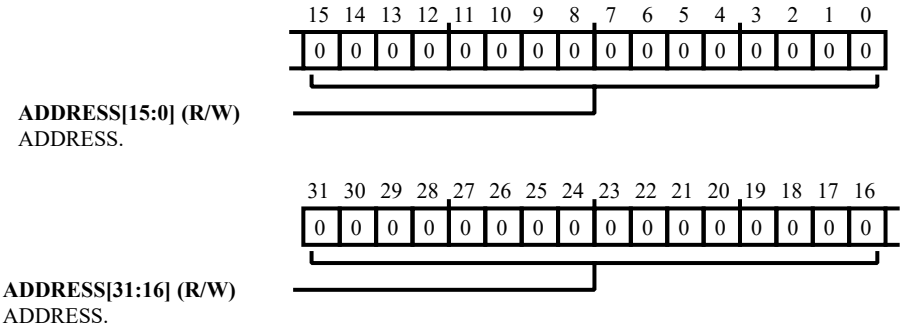


Figure 48-702: A55_C0_TRCACVR2_31_0 Register Diagram

Table 48-709: A55_C0_TRCACVR2_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

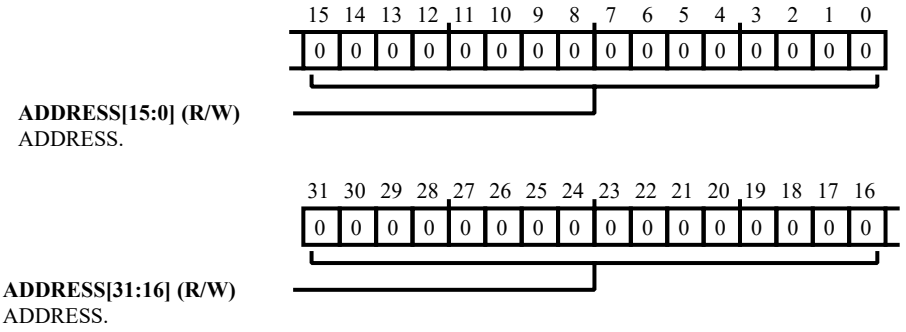


Figure 48-703: A55_C0_TRCACVR2_63_32 Register Diagram

Table 48-710: A55_C0_TRCACVR2_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

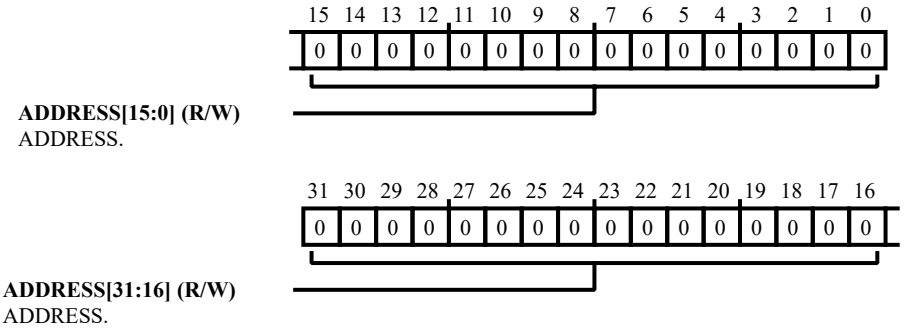


Figure 48-704: A55_C0_TRCACVR3_31_0 Register Diagram

Table 48-711: A55_C0_TRCACVR3_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

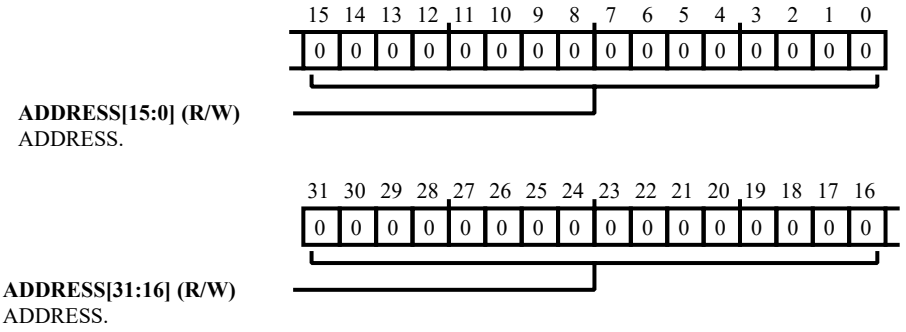


Figure 48-705: A55_C0_TRCACVR3_63_32 Register Diagram

Table 48-712: A55_C0_TRCACVR3_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

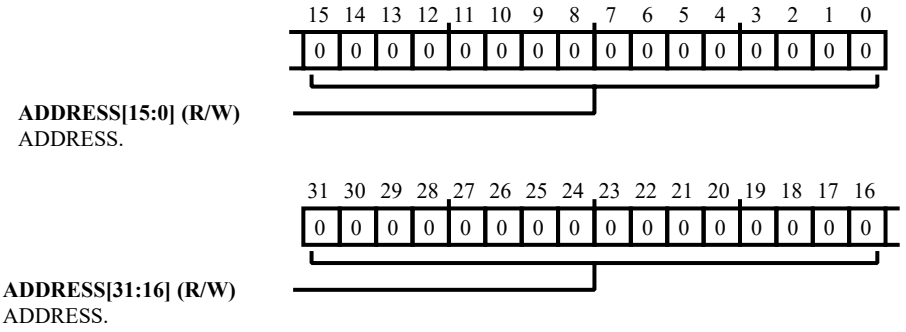


Figure 48-706: A55_C0_TRCACVR4_31_0 Register Diagram

Table 48-713: A55_C0_TRCACVR4_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

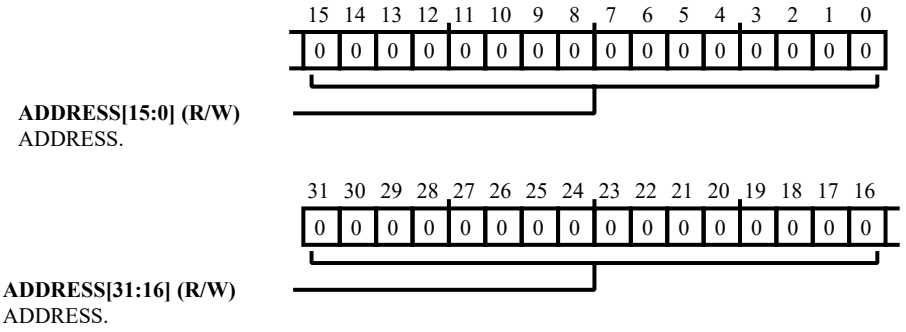


Figure 48-707: A55_C0_TRCACVR4_63_32 Register Diagram

Table 48-714: A55_C0_TRCACVR4_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

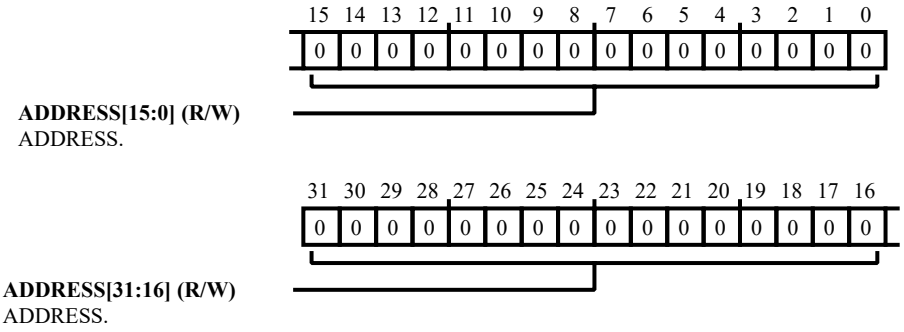


Figure 48-708: A55_C0_TRCACVR5_31_0 Register Diagram

Table 48-715: A55_C0_TRCACVR5_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

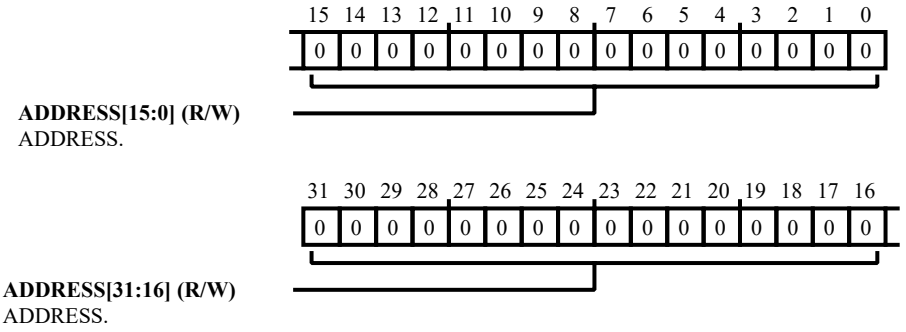


Figure 48-709: A55_C0_TRCACVR5_63_32 Register Diagram

Table 48-716: A55_C0_TRCACVR5_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

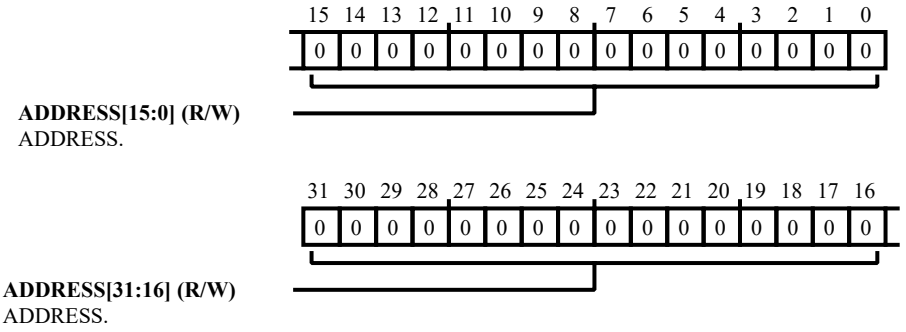


Figure 48-710: A55_C0_TRCACVR6_31_0 Register Diagram

Table 48-717: A55_C0_TRCACVR6_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

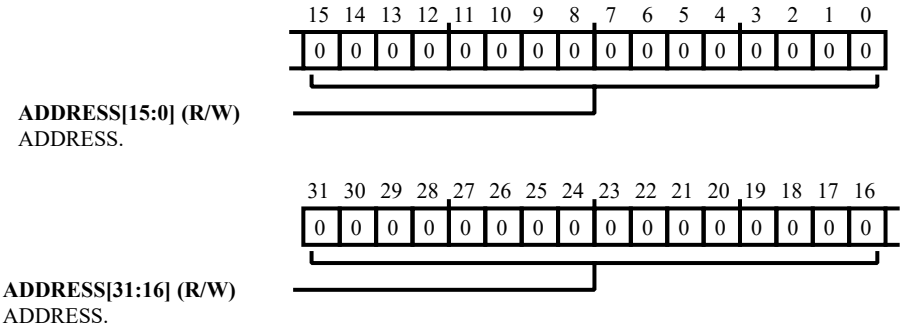


Figure 48-711: A55_C0_TRCACVR6_63_32 Register Diagram

Table 48-718: A55_C0_TRCACVR6_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

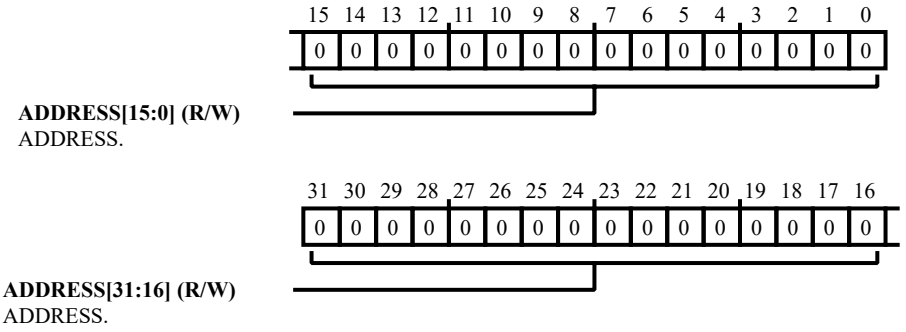


Figure 48-712: A55_C0_TRCACVR7_31_0 Register Diagram

Table 48-719: A55_C0_TRCACVR7_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

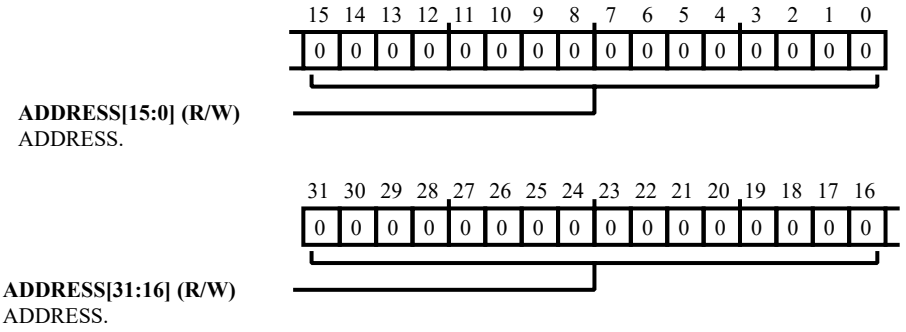


Figure 48-713: A55_C0_TRCACVR7_63_32 Register Diagram

Table 48-720: A55_C0_TRCACVR7_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Auxiliary Control Register.

The function of this register is IMPLEMENTATION DEFINED.

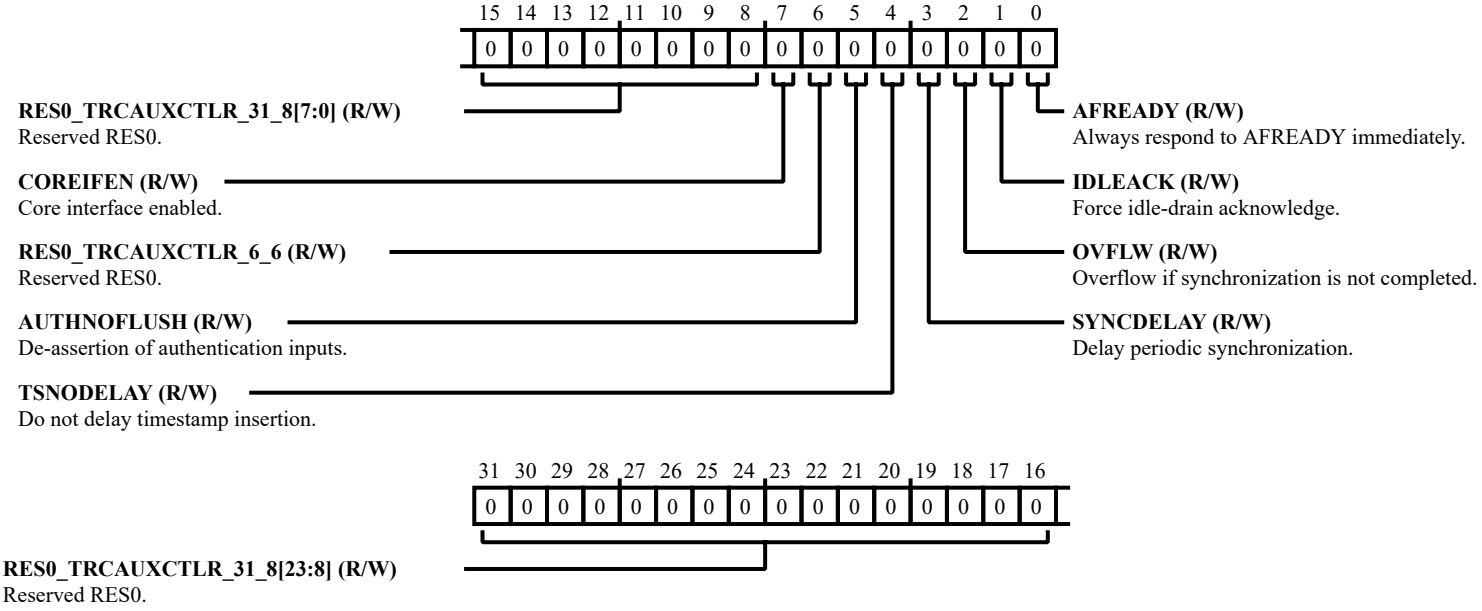


Figure 48-714: A55_C0_TRCAUXCTLR Register Diagram

Table 48-721: A55_C0_TRCAUXCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	RES0_TRCAUXCTLR_31_8	Reserved RES0.. Reserved RES0.	
		0	Reserved.
7 (R/W)	COREIFEN	Core interface enabled.. Keep core interface enabled regardless of trace enable register state.	
		0	Core interface enabled is set by trace enable register state.
		1	Enable core interface regardless of trace enable register state.
6 (R/W)	RES0_TRCAUXCTLR_6_6	Reserved RES0.. Reserved RES0.	
		0	Reserved.

Table 48-721: A55_C0_TRCAUXCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	AUTHNOFLUSH	De-assertion of authentication inputs.. Do not flush trace on de-assertion of authentication inputs. When this bit is set to 1 the trace unit behavior deviates from architecturally-specified behavior.	
		0	ETM trace unit FIFO is flushed and ETM trace unit enters idle state when DBGEN or NIDEN is LOW.
		1	ETM trace unit FIFO is not flushed and ETM trace unit does not enter idle state when DBGEN or NIDEN is LOW.
4 (R/W)	TSNODELAY	Do not delay timestamp insertion.. Do not delay timestamp insertion based on FIFO depth.	
		0	Timestamp packets are inserted into FIFO only when trace activity is LOW.
		1	Timestamp packets are inserted into FIFO irrespective of trace activity.
3 (R/W)	SYNCDELAY	Delay periodic synchronization.. Delay periodic synchronization if FIFO is more than half-full.	
		0	ETM trace unit AFREADYM output is asserted only when the ETM trace unit is in idle state or when all the trace bytes in FIFO before a flush request are output.
		1	ETM trace unit AFREADYM output is always asserted HIGH. When this bit is set to 1 trace unit behavior deviates from architecturally-specified behavior.
2 (R/W)	OVFLW	Overflow if synchronization is not completed.. Force an overflow if synchronization is not completed when second synchronization becomes due. When this bit is set to 1 the trace unit behavior deviates from architecturally-specified behavior.	
		0	ETM trace unit idle acknowledge is asserted only when the ETM trace unit is in idle state.
		1	ETM trace unit idle acknowledge is asserted irrespective of the ETM trace unit idle state.
1 (R/W)	IDLEACK	Force idle-drain acknowledge.. Force idle-drain acknowledge high CPU does not wait for trace to drain before entering WFX state. When this bit is set to 1 trace unit behavior deviates from architecturally-specified behavior.	
		0	No FIFO overflow when SYNC packets are delayed.
		1	Forces FIFO overflow when SYNC packets are delayed.

Table 48-721: A55_C0_TRCAUXCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	AFREADY	Always respond to AFREADY immediately.. Always respond to AFREADY immediately. Does not have any interaction with FIFO draining even in WFI state.	
		0	SYNC packets are inserted into FIFO only when trace activity is low.
		1	SYNC packets are inserted into FIFO irrespective of trace activity.

Trace Branch Broadcast Control Register.

Controls which regions in the memory map are enabled to use branch broadcasting.

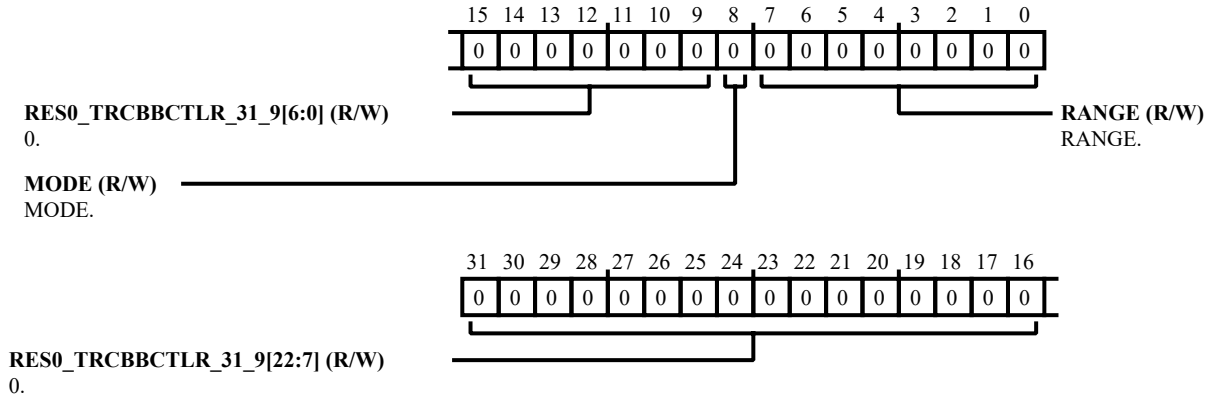


Figure 48-715: A55_C0_TRCBBCTLR Register Diagram

Table 48-722: A55_C0_TRCBBCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:9 (R/W)	RES0_TRCBBCTLR_31_9	0.. Reserved, RES0.	
8 (R/W)	MODE	MODE.. Mode bit: 0 Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines. If RANGE==0 then branch broadcasting is enabled for the entire memory map. 1 Include mode. Branch broadcasting is enabled in the address range that RANGE defines. If RANGE==0 then the branch broadcasting behavior is UNPREDICTABLE.	
		0	Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines. If RANGE==0 then branch broadcasting is enabled for the entire memory map.
		1	Include mode. Branch broadcasting is enabled in the address range that RANGE defines. If RANGE==0 then the branch broadcasting behavior is UNPREDICTABLE.

Table 48-722: A55_C0_TRCBBCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W)	RANGE	RANGE.. Address range field. Selects which address range comparator pairs are in use with branch broadcasting. Each bit represents an address range comparator pair, so bit[n] controls the selection of address range comparator pair n. If bit[n] is: 0 The address range that address range comparator pair n defines is not selected. 1 The address range that address range comparator pair n defines is selected.	
		0	The address range that address range comparator pair n defines is not selected.
		1	The address range that address range comparator pair n defines is selected.

Trace Cycle Count Control Register.

Sets the threshold value for cycle counting.

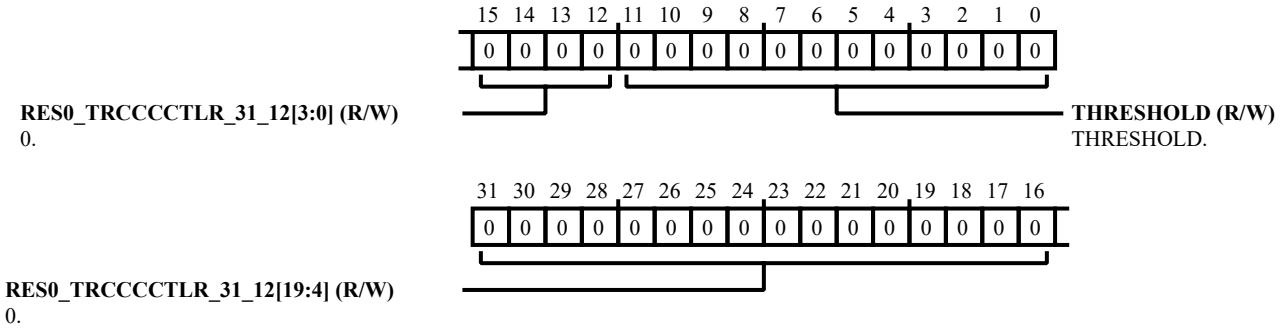


Figure 48-716: A55_C0_TRCCCCTLR Register Diagram

Table 48-723: A55_C0_TRCCCCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/W)	RES0_TRCCCCTLR_31_12	0.. Reserved, RES0.
11:0 (R/W)	THRESHOLD	THRESHOLD.. Sets the threshold value for instruction trace cycle counting.The minimum threshold value that can be programmed into THRESHOLD is given in TRCIDR3.CCITMIN.Writing a value of zero might cause UNPREDICTABLE behaviour.

Trace Context ID Comparator Control Register 0.

Contains Context ID mask values for the TRCCIDCVRn registers, where n=0-3.

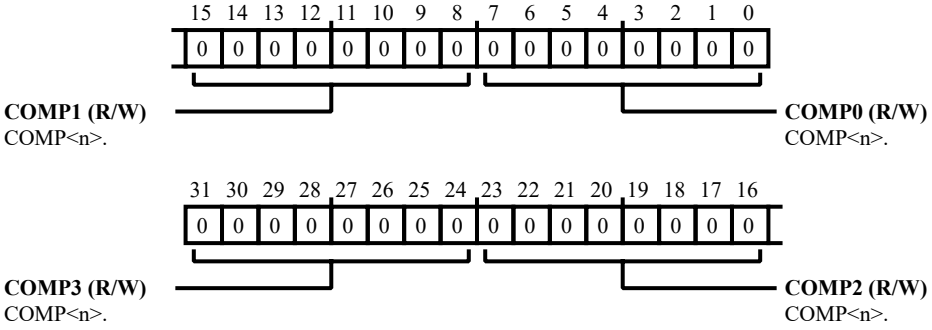


Figure 48-717: A55_C0_TRCCIDCCTLR0 Register Diagram

Table 48-724: A55_C0_TRCCIDCCTLR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	COMP3	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.

Table 48-724: A55_C0_TRCCIDCCTLR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	COMP2	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
15:8 (R/W)	COMP1	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
7:0 (R/W)	COMP0	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.

Trace Context ID Comparator Value Registers.

Contains a Context ID value.

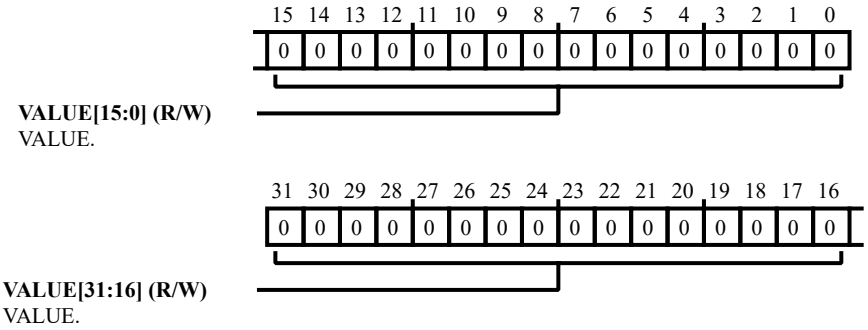


Figure 48-718: A55_C0_TRCCIDCVR0_31_0 Register Diagram

Table 48-725: A55_C0_TRCCIDCVR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. Context ID value. The implemented width of this field is IMPLEMENTATION DEFINED and is set by TRCIDR2.CIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the Context ID is zero until the processor updates the Context ID.

Trace Context ID Comparator Value Registers.

Contains a Context ID value.

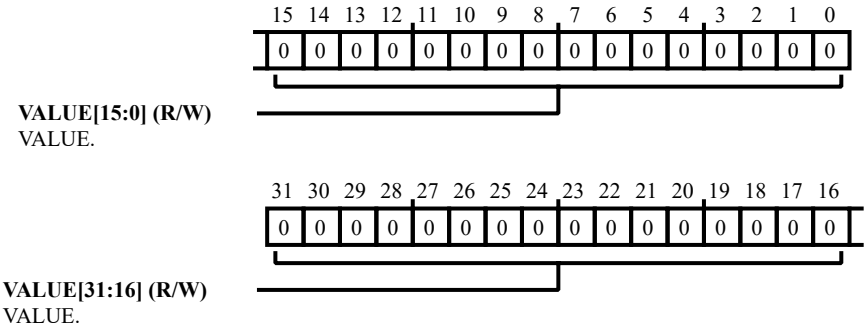


Figure 48-719: A55_C0_TRCCIDCVR0_63_32 Register Diagram

Table 48-726: A55_C0_TRCCIDCVR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. Context ID value. The implemented width of this field is IMPLEMENTATION DEFINED and is set by TRCIDR2.CIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the Context ID is zero until the processor updates the Context ID.

Trace Claim Tag Clear Register.

Software can use this to clear a claim tag bit to 0, and to read the values of the claim tag.

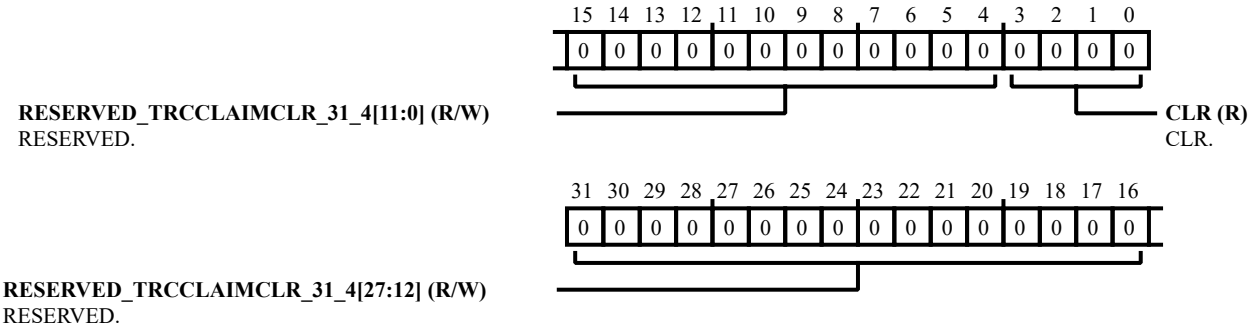


Figure 48-720: A55_C0_TRCCLAIMCLR Register Diagram

Table 48-727: A55_C0_TRCCLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_TRCCLAIMCLR_31_4	RESERVED.. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	CLR	CLR.. Clears bits in the claim tag and determines the current value of the claim tag.	
		0	On reads for each bit Claim tag bit is set.
		1	On reads for each bit Claim tag bit is not set.

Trace Claim Tag Set Register.

Software can use this to set bits in the claim tag to 1, and discover the number of bits that the claim tag supports.

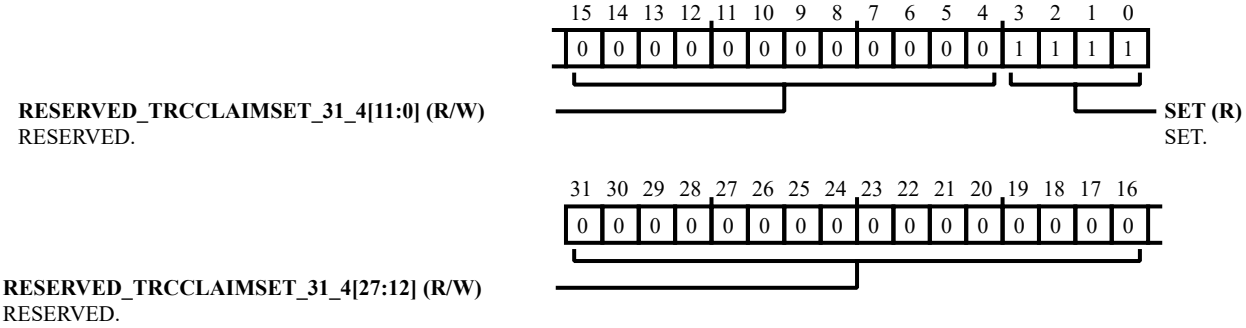


Figure 48-721: A55_C0_TRCCLAIMSET Register Diagram

Table 48-728: A55_C0_TRCCLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_TRCCLAIM-SET_31_4	RESERVED.. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	SET	SET.. Sets bits in the claim tag and determines the number of claim tag bits implemented.	
		0	On reads for each bit Claim tag bit is implemented.
		1	On reads for each bit Claim tag bit is not implemented.

Trace Counter Control Registers.

Trace Counter Control Registers.

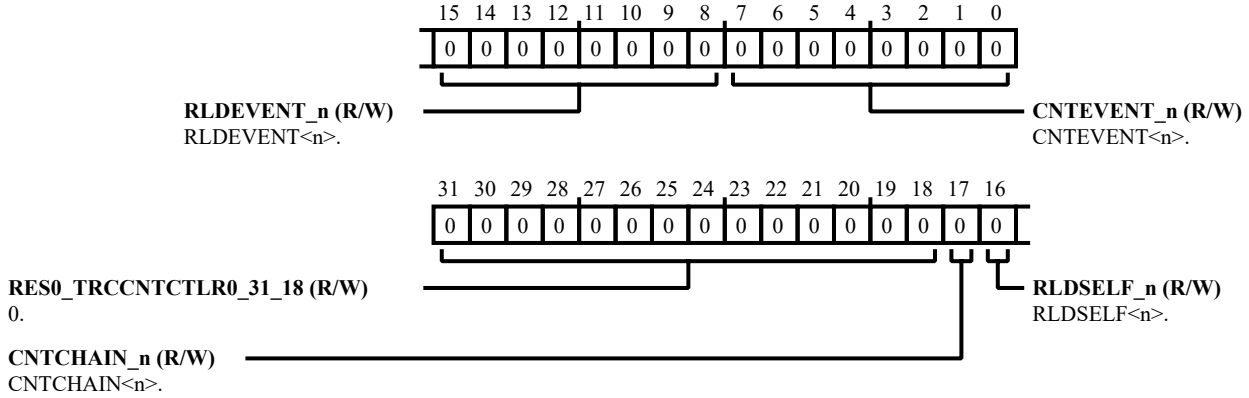


Figure 48-722: A55_C0_TRCCNTCTLR0 Register Diagram

Table 48-729: A55_C0_TRCCNTCTLR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/W)	RES0_TRCCNTCTLR0_31_18	0.. Reserved, RES0.	
17 (R/W)	CNTCHAIN_N	CNTCHAIN<n>.. For TRCCNTCTLR3 and TRCCNTCTLR1, controls whether counter <n> decrements when a reload event occurs for counter <n-1>: 0 1 For TRCCNTCTLR2 and TRCCNTCTLR0, this bit is RES0.	
		0	
		1	
16 (R/W)	RLDSELF_N	RLDSELF<n>.. Controls whether a reload event occurs for counter <n>, when counter <n> reaches zero: 0 The trace unit does not generate a reload event. 1 The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.	
		0	The trace unit does not generate a reload event.
		1	The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.
15:8 (R/W)	RLDEVENT_N	RLDEVENT<n>.. Selects an event, that when it occurs causes a reload event for counter <n>.	

Table 48-729: A55_C0_TRCCNTCTLR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	CNTEVENT_N	CNTEVENT<n>.. Selects an event, that when it occurs causes counter <n> to decrement.

Trace Counter Control Registers.

Trace Counter Control Registers.

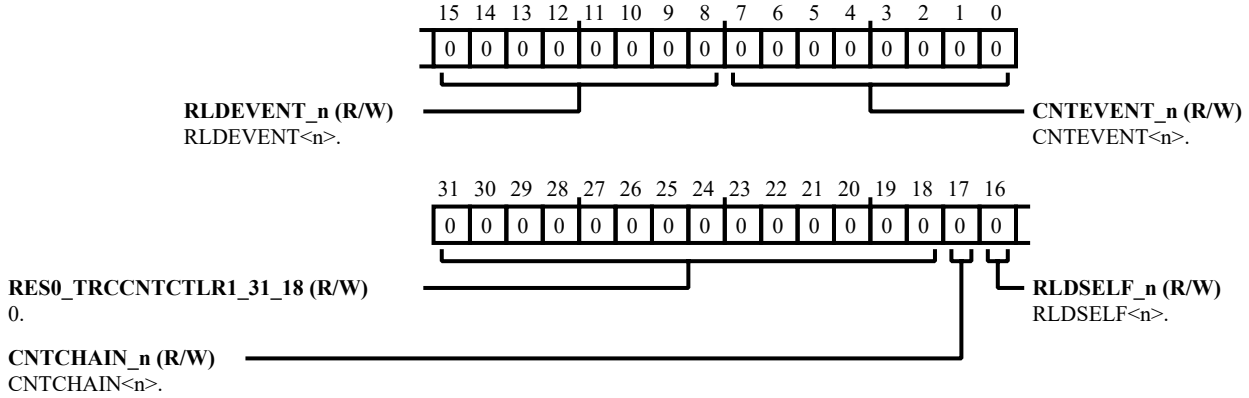


Figure 48-723: A55_C0_TRCCNTCTLR1 Register Diagram

Table 48-730: A55_C0_TRCCNTCTLR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/W)	RES0_TRCCNTCTLR1_31_18	0.. Reserved, RES0.	
17 (R/W)	CNTCHAIN_N	CNTCHAIN<n>.. For TRCCNTCTLR3 and TRCCNTCTLR1, controls whether counter <n> decrements when a reload event occurs for counter <n-1>: 0 1 For TRCCNTCTLR2 and TRCCNTCTLR0, this bit is RES0.	
		0	
		1	
16 (R/W)	RLDSELF_N	RLDSELF<n>.. Controls whether a reload event occurs for counter <n>, when counter <n> reaches zero: 0 The trace unit does not generate a reload event. 1 The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.	
		0	The trace unit does not generate a reload event.
		1	The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.
15:8 (R/W)	RLDEVENT_N	RLDEVENT<n>.. Selects an event, that when it occurs causes a reload event for counter <n>.	

Table 48-730: A55_C0_TRCCNTCTLR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	CNTEVENT_N	CNTEVENT<n>.. Selects an event, that when it occurs causes counter <n> to decrement.

Trace Counter Reload Value Registers.

This sets or returns the reload count value for counter 0.

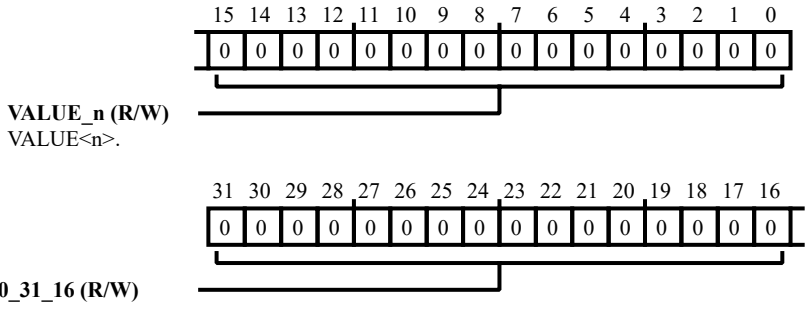


Figure 48-724: A55_C0_TRCCNTRLDVR0 Register Diagram

Table 48-731: A55_C0_TRCCNTRLDVR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTRLDVR0_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the reload value for counter <n>. When a reload event occurs for counter <n> then the trace unit copies the VALUE<n> field into counter <n>.

Trace Counter Reload Value Registers.

This sets or returns the reload count value for counter 1.

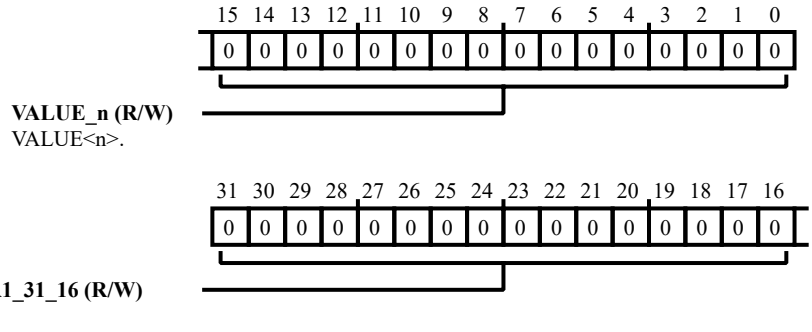


Figure 48-725: A55_C0_TRCCNTRLDVR1 Register Diagram

Table 48-732: A55_C0_TRCCNTRLDVR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTRLDVR1_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the reload value for counter <n>. When a reload event occurs for counter <n> then the trace unit copies the VALUE<n> field into counter <n>.

Trace Counter Value Registers.

This sets or returns the value of counter 0.

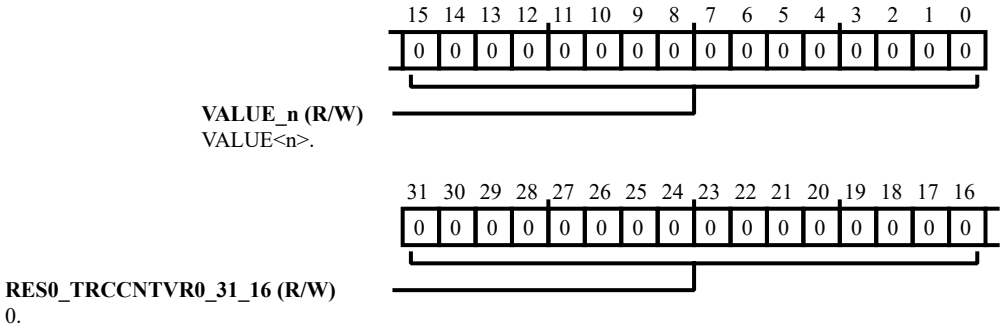


Figure 48-726: A55_C0_TRCCNTVR0 Register Diagram

Table 48-733: A55_C0_TRCCNTVR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTVR0_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the count value of counter <n>.

Trace Counter Value Registers.

This sets or returns the value of counter 1.

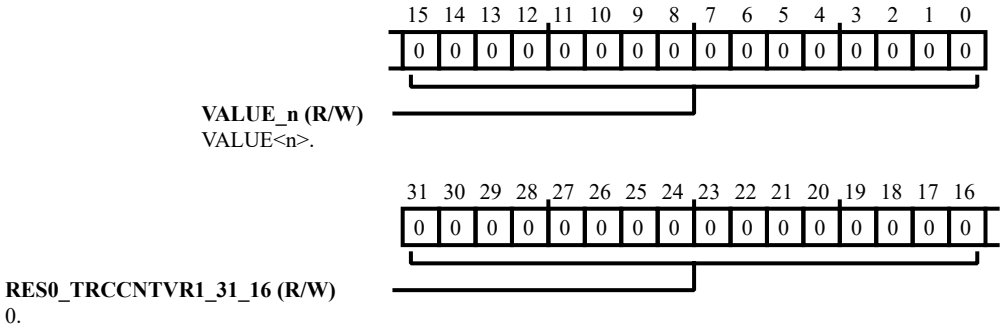


Figure 48-727: A55_C0_TRCCNTVR1 Register Diagram

Table 48-734: A55_C0_TRCCNTVR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTVR1_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the count value of counter <n>.

Trace Configuration Register.

Controls the tracing options.

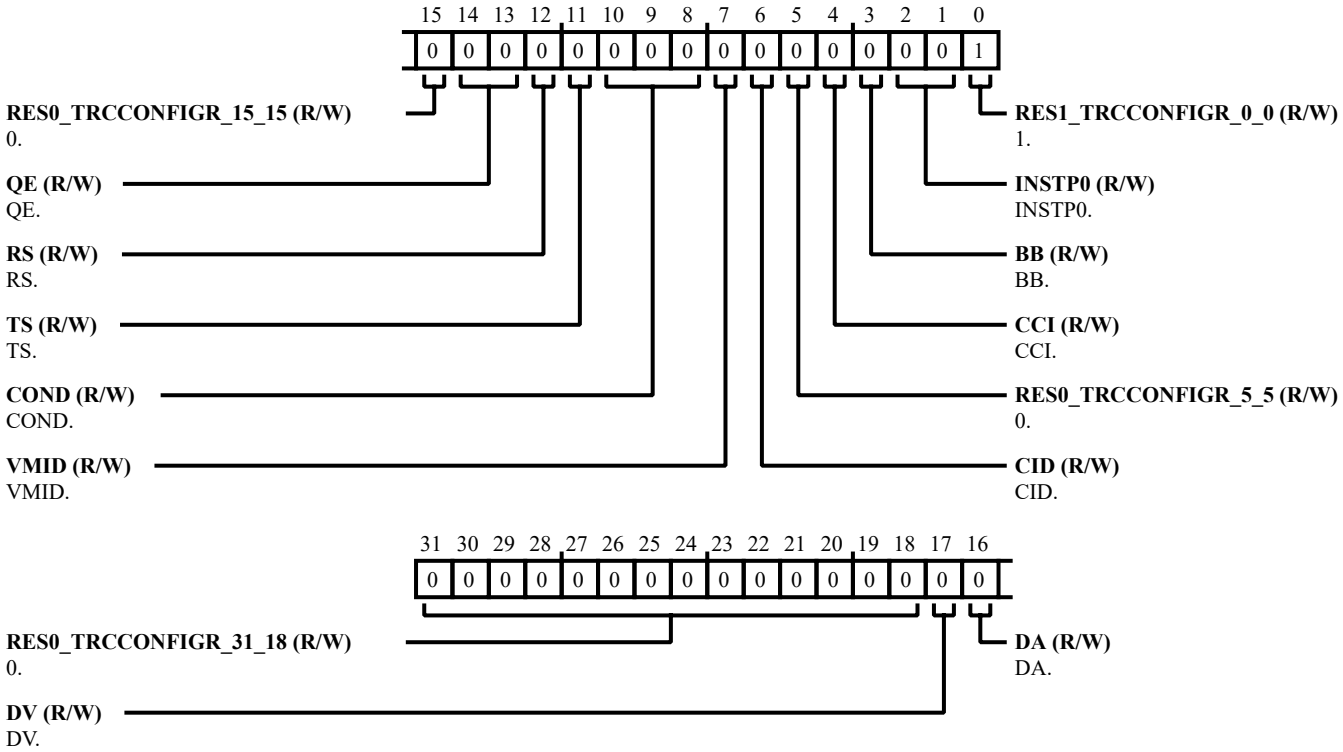


Figure 48-728: A55_C0_TRCCONFIGR Register Diagram

Table 48-735: A55_C0_TRCCONFIGR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/W)	RES0_TRCCON-FIGR_31_18	0.. Reserved, RES0.	
17 (R/W)	DV	DV.. Data value tracing bit: 0 Data value tracing is disabled. 1 Data value tracing is enabled when INSTP0 is not 0b00. TRCIDR0.TRCDATA indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Data value tracing is disabled.
		1	Data value tracing is enabled when INSTP0 is not 0b00.

Table 48-735: A55_C0_TRCCONFIGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	DA	DA.. Data address tracing bit: 0 Data address tracing is disabled. 1 Data address tracing is enabled when INSTP0 is not 0b00. TRCIDR0.TRCDATA indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Data address tracing is disabled.
		1	Data address tracing is enabled when INSTP0 is not 0b00.
15 (R/W)	RES0_TRCCON-FIGR_15_15	0.. Reserved, RES0.	
14:13 (R/W)	QE	QE.. Q element enable field: 00 Q elements are disabled. 01 Q elements with instruction counts are enabled. Q elements without instruction counts are disabled. 11 Q elements with and without instruction counts are enabled. The value 0b10 is reserved.TRCIDR0.QSUPP indicates which values of this field are implemented.TRCCONFIGR.QE must be set to 0b00 if any of the following are true:TRCCONFIGR.INSTP0 is not 0b00.TRCCONFIGR.COND is not 0b000.TRCCONFIGR.BB is not 0.	
		0	Q elements are disabled.
		1	Q elements with instruction counts are enabled. Q elements without instruction counts are disabled.
		3	Q elements with and without instruction counts are enabled.
12 (R/W)	RS	RS.. Return stack enable bit. 0 Return stack is disabled. 1 Return stack is enabled. TRCIDR0.RETSTACK indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Return stack is disabled.
		1	Return stack is enabled.
11 (R/W)	TS	TS.. Global timestamp tracing bit: 0 Global timestamp tracing is disabled. 1 Global timestamp tracing is enabled. TRCTSCTLR controls the insertion of timestamps in the trace. TRCIDR0.TSSIZE indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Global timestamp tracing is disabled.
		1	Global timestamp tracing is enabled. TRCTSCTLR controls the insertion of timestamps in the trace.

Table 48-735: A55_C0_TRCCONFIGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:8 (R/W)	COND	COND.. Conditional instruction tracing bit. The permitted values are: 000 Conditional instruction tracing is disabled. 001 Conditional load instructions are traced. 010 Conditional store instructions are traced. 011 Conditional load and store instructions are traced. 111 All conditional instructions are traced. All other values are reserved.TRCIDR0.TRCCOND indicates whether this field is supported. If it is not supported then this field is RES0.	
		0	Conditional instruction tracing is disabled.
		1	Conditional load instructions are traced.
		2	Conditional store instructions are traced.
		3	Conditional load and store instructions are traced.
		7	All conditional instructions are traced.
7 (R/W)	VMID	VMID.. VMID tracing bit: 0 VMID tracing is disabled. 1 VMID tracing is enabled. TRCIDR2.VMIDSIZE indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	VMID tracing is disabled.
		1	VMID tracing is enabled.
6 (R/W)	CID	CID.. Context ID tracing bit: 0 Context ID tracing is disabled. 1 Context ID tracing is enabled. TRCIDR2.CIDSIZE indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Context ID tracing is disabled.
		1	Context ID tracing is enabled.
5 (R/W)	RES0_TRCCONFIGR_5_5	0.. Reserved, RES0.	
4 (R/W)	CCI	CCI.. Cycle counting instruction trace bit: 0 Cycle counting in the instruction trace is disabled. 1 Cycle counting in the instruction trace is enabled. TRCCCCTLR controls the threshold value for cycle counting. TRCIDR0.TRCCCI indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Cycle counting in the instruction trace is disabled.
		1	Cycle counting in the instruction trace is enabled. TRCCCCTLR controls the threshold value for cycle counting.

Table 48-735: A55_C0_TRCCONFIGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	BB	BB.. Branch broadcast mode bit: 0 Branch broadcast mode is disabled. 1 Branch broadcast mode is enabled. TRCBBCTLR controls which regions of memory are enabled to use branch broadcasting. TRCIDR0.TRCBB indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Branch broadcast mode is disabled.
		1	Branch broadcast mode is enabled. TRCBBCTLR controls which regions of memory are enabled to use branch broadcasting.
2:1 (R/W)	INSTP0	INSTP0.. Instruction P0 bit. This field controls whether load and store instructions are traced as P0 instructions: 00 Do not trace load and store instructions as P0 instructions. 01 Trace load instructions as P0 instructions. 10 Trace store instructions as P0 instructions. 11 Trace load and store instructions as P0 instructions. TRCIDR0.INSTP0 indicates whether this field is supported. If it is not supported then this field is RES0.	
		0	Do not trace load and store instructions as P0 instructions.
		1	Trace load instructions as P0 instructions.
		2	Trace store instructions as P0 instructions.
		3	Trace load and store instructions as P0 instructions.
0 (R/W)	RES1_TRCCONFIGR_0_0	1.. Reserved, RES1.	

Trace Event Control 0 Register.

Trace Event Control 0 Register.

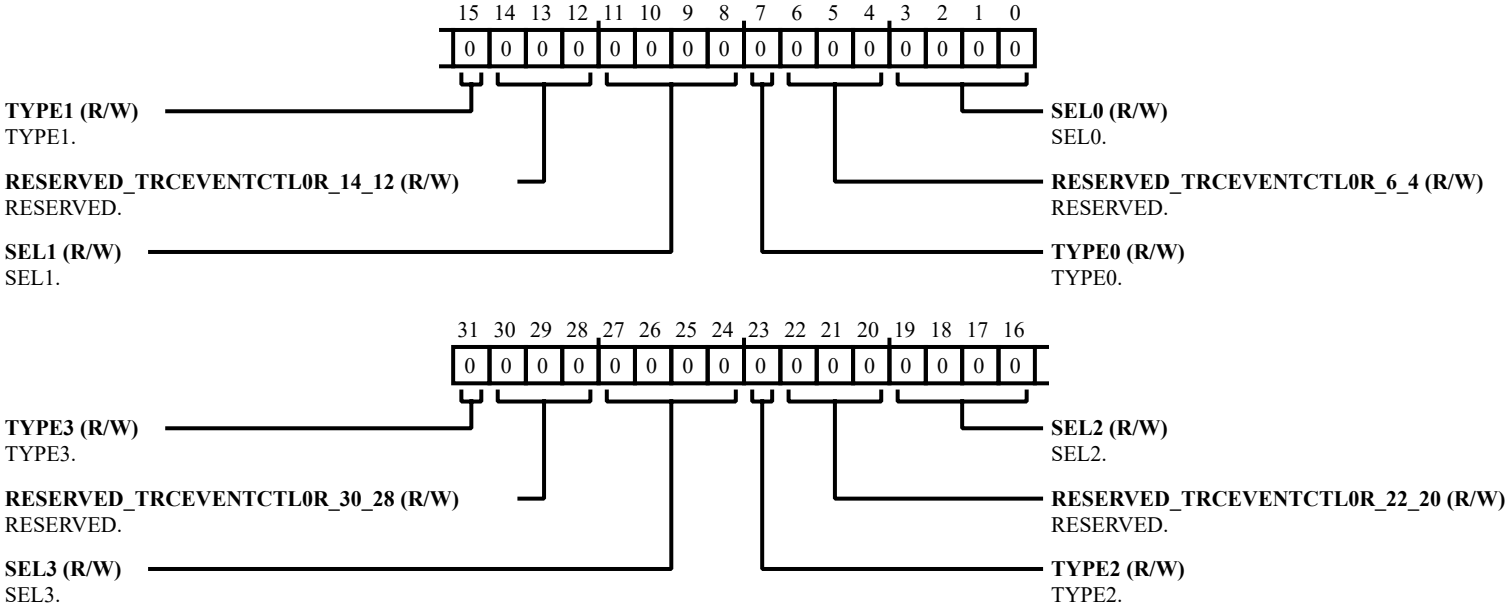


Figure 48-729: A55_C0_TRCEVENTCTL0R Register Diagram

Table 48-736: A55_C0_TRCEVENTCTL0R Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TYPE3	TYPE3.. Selects the resource type for trace event 3.	
		0	Single selected resource.
		1	Boolean combined resource pair.
30:28 (R/W)	RESERVED_TRCE-VENTCTL0R_30_28	RESERVED.. Reserved RES0.	
		0	Reserved.
27:24 (R/W)	SEL3	SEL3.. Selects the resource number based on the value of TYPE3.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Table 48-736: A55_C0_TRCEVENTCTL0R Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	TYPE2	TYPE2.. Selects the resource type for trace event 2.	
		0	Single selected resource.
		1	Boolean combined resource pair.
22:20 (R/W)	RESERVED_TRCE-VENTCTL0R_22_20	RESERVED.. Reserved RES0.	
		0	Reserved.
19:16 (R/W)	SEL2	SEL2.. Selects the resource number based on the value of TYPE2.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
15 (R/W)	TYPE1	TYPE1.. Selects the resource type for trace event 1.	
		0	Single selected resource.
		1	Boolean combined resource pair.
14:12 (R/W)	RESERVED_TRCE-VENTCTL0R_14_12	RESERVED.. Reserved RES0.	
		0	Reserved.
11:8 (R/W)	SEL1	SEL1.. Selects the resource number based on the value of TYPE1.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
7 (R/W)	TYPE0	TYPE0.. Selects the resource type for trace event 0.	
		0	Single selected resource.
		1	Boolean combined resource pair.
6:4 (R/W)	RESERVED_TRCE-VENTCTL0R_6_4	RESERVED.. Reserved RES0.	
		0	Reserved.

Table 48-736: A55_C0_TRCEVENTCTL0R Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	SEL0	SEL0.. Selects the resource number based on the value of TYPE0.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Trace Event Control 1 Register.

Trace Event Control 1 Register.

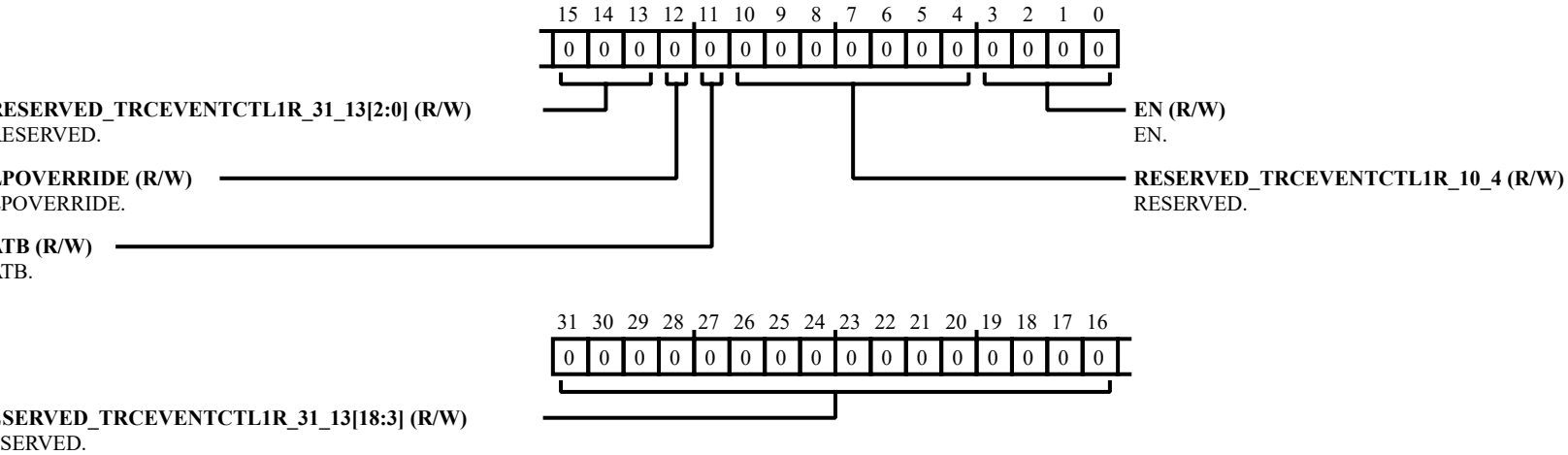


Figure 48-730: A55_C0_TRCEVENTCTL1R Register Diagram

Table 48-737: A55_C0_TRCEVENTCTL1R Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:13 (R/W)	RESERVED_TRCE-VENTCTL1R_31_13	RESERVED.. Reserved RES0.	
		0	Reserved.
12 (R/W)	LPOVERRIDE	LPOVERRIDE.. Low power state behavior override.	
		0	Low power state behavior unaffected.
		1	Low power state behavior overridden. The resources and Event trace generation are unaffected by entry to a low power state.
11 (R/W)	ATB	ATB.. ATB trigger enable.	
		0	ATB trigger disabled.
		1	ATB trigger enabled.
10:4 (R/W)	RESERVED_TRCE-VENTCTL1R_10_4	RESERVED.. Reserved RES0.	
		0	Reserved.

Table 48-737: A55_C0_TRCEVENTCTL1R Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	EN	EN.. One bit per event to enable generation of an event element in the instruction trace stream when the selected event occurs.	
		0	Event does not cause an event element.
		1	Event causes an event element.

Trace External Input Select Register.

Use this to set, or read, which external inputs are resources to the trace unit.

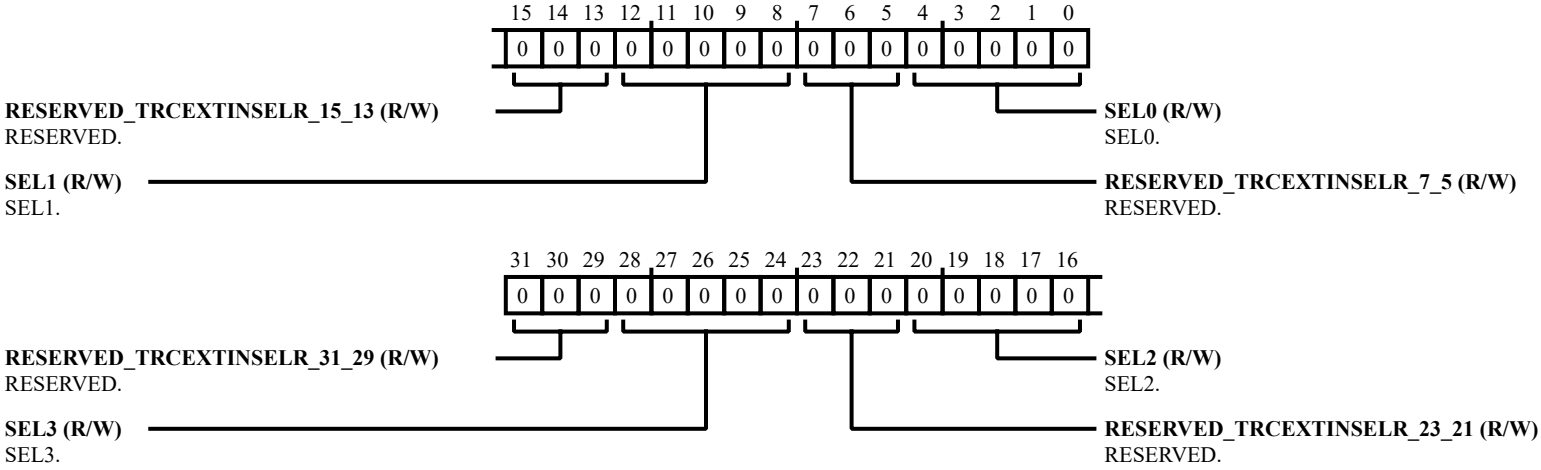


Figure 48-731: A55_C0_TRCEXTINSELR Register Diagram

Table 48-738: A55_C0_TRCEXTINSELR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RESERVED_TRCEXTIN-SELR_31_29	RESERVED.. Reserved RES0.	
		0	Reserved.
28:24 (R/W)	SEL3	SEL3.. Selects an event from the external input bus for External Input Resource 3.	
23:21 (R/W)	RESERVED_TRCEXTIN-SELR_23_21	RESERVED.. Reserved RES0.	
		0	Reserved.
20:16 (R/W)	SEL2	SEL2.. Selects an event from the external input bus for External Input Resource 2.	
15:13 (R/W)	RESERVED_TRCEXTIN-SELR_15_13	RESERVED.. Reserved RES0.	
		0	Reserved.
12:8 (R/W)	SEL1	SEL1.. Selects an event from the external input bus for External Input Resource 1.	

Table 48-738: A55_C0_TRCEXTINSELR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:5 (R/W)	RESERVED_TRCEXTIN-SELR_7_5	RESERVED.. Reserved RES0.	
		0	Reserved.
4:0 (R/W)	SEL0	SEL0.. Selects an event from the external input bus for External Input Resource 0.	

Trace ID Register 0.

Returns the tracing capabilities of the trace unit.

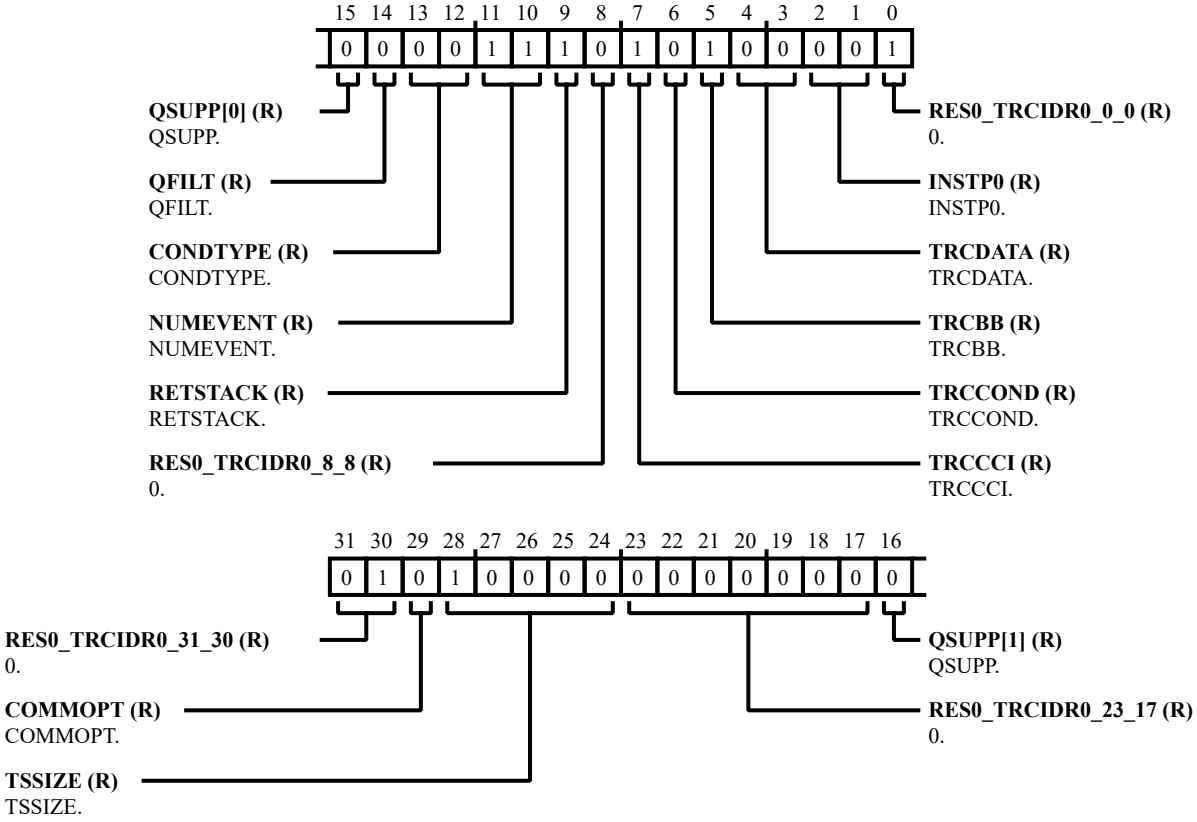


Figure 48-732: A55_C0_TRCIDR0 Register Diagram

Table 48-739: A55_C0_TRCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	RES0_TRCIDR0_31_30	0.. Reserved, RES0.	
29 (R/NW)	COMMOPT	COMMOPT.. Conditional instruction tracing support bit. Indicates if the trace unit supports conditional instruction tracing: 0 Conditional instruction tracing is not supported. 1 Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.	
		0	Conditional instruction tracing is not supported.
		1	Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.

Table 48-739: A55_C0_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28:24 (R/NW)	TSSIZE	TSSIZE.. Global timestamp size field. The permitted values are: 00000 Global timestamping is not implemented. 00110 Implementation supports a maximum global timestamp of 48bits. 01000 Implementation supports a maximum global timestamp of 64bits. All other values are reserved.When global timestamping is implemented then TRCCONFIGR.TS and TRCTSCTLR are supported.	
		0	Global timestamping is not implemented.
		6	Implementation supports a maximum global timestamp of 48bits.
		8	Implementation supports a maximum global timestamp of 64bits.
23:17 (R/NW)	RES0_TRCIDR0_23_17	0.. Reserved, RES0.	
16:15 (R/NW)	QSUPP	QSUPP.. Q element support field. The permitted values are: 00 Q element support is not implemented. TRCCONFIGR.QE is RES0. 01 Q element support is implemented, and only supports Q elements with instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b01. 10 Q element support is implemented, and only supports Q elements without instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b11. 11 Q element support is implemented, and supports both Q elements with instruction counts and Q elements without instruction counts. TRCCONFIGR.QE is fully implemented.	
		0	Q element support is not implemented. TRCCONFIGR.QE is RES0.
		1	Q element support is implemented, and only supports Q elements with instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b01.
		2	Q element support is implemented, and only supports Q elements without instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b11.
		3	Q element support is implemented, and supports both Q elements with instruction counts and Q elements without instruction counts. TRCCONFIGR.QE is fully implemented.

Table 48-739: A55_C0_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/NW)	QFILT	QFILT.. Q element filtering support field. The permitted values are: 0 Q element filtering is not implemented. 1 Q element filtering is implemented. TRCQCTLR is implemented. When QSUPP==0b00, this field is RES0.	
		0	Q element filtering is not implemented.
		1	Q element filtering is implemented. TRCQCTLR is implemented.
13:12 (R/NW)	CONDTYPE	CONDTYPE.. Conditional tracing field. The permitted values are: 00 The trace unit indicates only if a conditional instruction is a pass or fail. 01 The trace unit provides the Current Program Status Register (CPSR) status, for a conditional instruction. All other values are reserved.	
		0	The trace unit indicates only if a conditional instruction is a pass or fail.
		1	The trace unit provides the Current Program Status Register (CPSR) status, for a conditional instruction.
11:10 (R/NW)	NUMEVENT	NUMEVENT.. Number of events field. Indicates how many events the trace unit supports: 00 The trace unit supports 1 event. 01 The trace unit supports 2 events. 10 The trace unit supports 3 events. 11 The trace unit supports 4 events. This field controls how many fields are supported in TRCEVENTCTL0R, and indicates the size of TRCEVENTCTL1R.INSTEN.	
		0	The trace unit supports 1 event.
		1	The trace unit supports 2 events.
		2	The trace unit supports 3 events.
		3	The trace unit supports 4 events.
9 (R/NW)	RETSTACK	RETSTACK.. Return stack bit. Indicates if the implementation supports a return stack: 0 Return stack is not implemented. 1 Return stack is implemented, so TRCCONFIGR.RS is supported.	
		0	Return stack is not implemented.
		1	Return stack is implemented, so TRCCONFIGR.RS is supported.
8 (R/NW)	RES0_TRCIDR0_8_8	0.. Reserved, RES0.	

Table 48-739: A55_C0_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	TRCCCI	TRCCCI.. Cycle counting instruction bit. Indicates if the trace unit supports cycle counting for instructions: 0 Cycle counting in the instruction trace is not implemented. 1 Cycle counting in the instruction trace is implemented, so TRCCONFIGR.CCI and TRCCCCTLR are supported.	
		0	Cycle counting in the instruction trace is not implemented.
		1	Cycle counting in the instruction trace is implemented, so TRCCONFIGR.CCI and TRCCCCTLR are supported.
6 (R/NW)	TRCCOND	TRCCOND.. Conditional instruction tracing support bit. Indicates if the trace unit supports conditional instruction tracing: 0 Conditional instruction tracing is not supported. 1 Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.	
		0	Conditional instruction tracing is not supported.
		1	Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.
5 (R/NW)	TRCBB	TRCBB.. Branch broadcast tracing support bit. Indicates if the trace unit supports branch broadcast tracing: 0 Branch broadcast tracing is not supported. 1 Branch broadcast tracing is supported, so TRCCONFIGR.BB and TRCBBCTLR are supported.	
		0	Branch broadcast tracing is not supported.
		1	Branch broadcast tracing is supported, so TRCCONFIGR.BB and TRCBBCTLR are supported.
4:3 (R/NW)	TRCDATA	TRCDATA.. Conditional tracing field. The permitted values are: 00 Data tracing is not supported. 11 Tracing of data addresses and data values is supported, so TRCCONFIGR.DA, TRCCONFIGR.DV, TRCSTALLCTLR.DATADISCARD, TRCSTALLCTLR.INSTPRIORITY, TRCSTALLCTLR.DSTALL, and TRCEVENTCTL1R.DATAEN are supported. All other values are reserved.	
		0	Data tracing is not supported.
		3	Tracing of data addresses and data values is supported, so TRCCONFIGR.DA, TRCCONFIGR.DV, TRCSTALLCTLR.DATADISCARD, TRCSTALLCTLR.INSTPRIORITY, TRCSTALLCTLR.DSTALL, and TRCEVENTCTL1R.DATAEN are supported.

Table 48-739: A55_C0_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:1 (R/NW)	INSTP0	INSTP0.. P0 tracing support field. The permitted values are: 00 Tracing of load and store instructions as P0 elements is not supported. 11 Tracing of load and store instructions as P0 elements is supported, so TRCCONFIGR.INSTP0 is supported. All other values are reserved.	
		0	Tracing of load and store instructions as P0 elements is not supported.
		3	Tracing of load and store instructions as P0 elements is supported, so TRCCONFIGR.INSTP0 is supported.
0 (R/NW)	RES0_TRCIDR0_0_0	0.. Reserved, RES0.	

Trace ID Register 1.

Returns the base architecture of the trace unit.

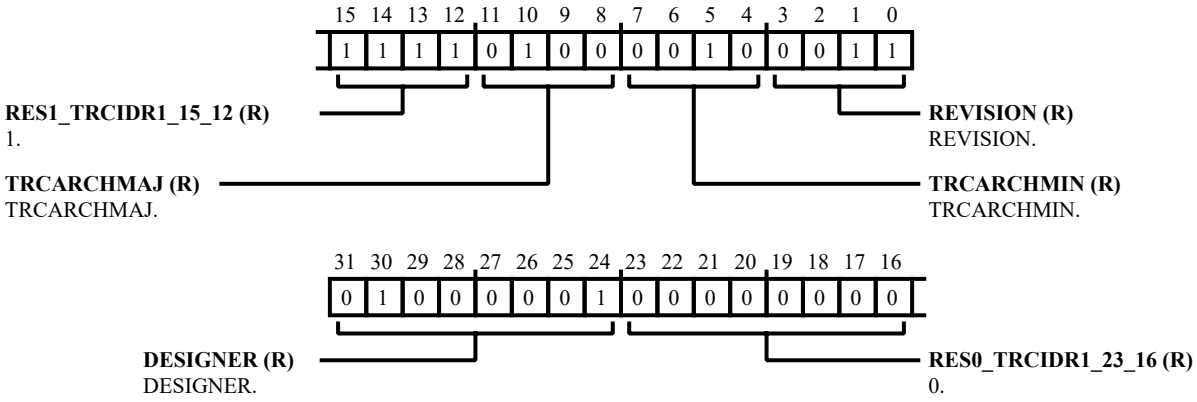


Figure 48-733: A55_C0_TRCIDR1 Register Diagram

Table 48-740: A55_C0_TRCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/NW)	DESIGNER	DESIGNER.. Indicates which company designed the trace unit. The permitted values are: 01000001 ARM Limited. 01000100 Digital Equipment Corporation. 01001101 Motorola, Freescale Semiconductor Inc. 01010001 QUALCOMM Inc. 01010110 Marvell Semiconductor Inc. 01101001 Intel Corporation. All other values are reserved.	
		65	ARM Limited.
		68	Digital Equipment Corporation.
		77	Motorola, Freescale Semiconductor Inc.
		81	QUALCOMM Inc.
		86	Marvell Semiconductor Inc.
		105	Intel Corporation.
23:16 (R/NW)	RES0_TRCIDR1_23_16	0.. Reserved, RES0.	
15:12 (R/NW)	RES1_TRCIDR1_15_12	1.. Reserved, RES1.	
11:8 (R/NW)	TRCARCHMAJ	TRCARCHMAJ.. Indicates the major version of the ETM architecture. The permitted value is: 100 ETMv4. All other values are reserved.	
		4	ETMv4.

Table 48-740: A55_C0_TRCIDR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	TRCARCHMIN	TRCARCHMIN.. Indicates the minor version of the ETM architecture. The permitted value is: 0 ETMv4 minor version 0. All other values are reserved.	
		0	ETMv4 minor version 0.
3:0 (R/NW)	REVISION	REVISION.. Returns an IMPLEMENTATION DEFINED value that identifies the revision of the trace registers and the OS Save and Restore registers.ARM recommends:That the initial implementation sets REVISION==0x0 and the field then increments for any subsequent implementations. However, it is acceptable to omit some values or use another scheme to identify the revision number.That TRCPIDR2.REVISION==TRCIDR1.REVISION. However, in situations where it is difficult to align these fields, such as with a metal layer fix, then it is acceptable to change the REVISION fields independently.	

Trace ID Register 2.

Returns the maximum size of the data value, data address, VMID, Context ID and instruction address in the trace unit.

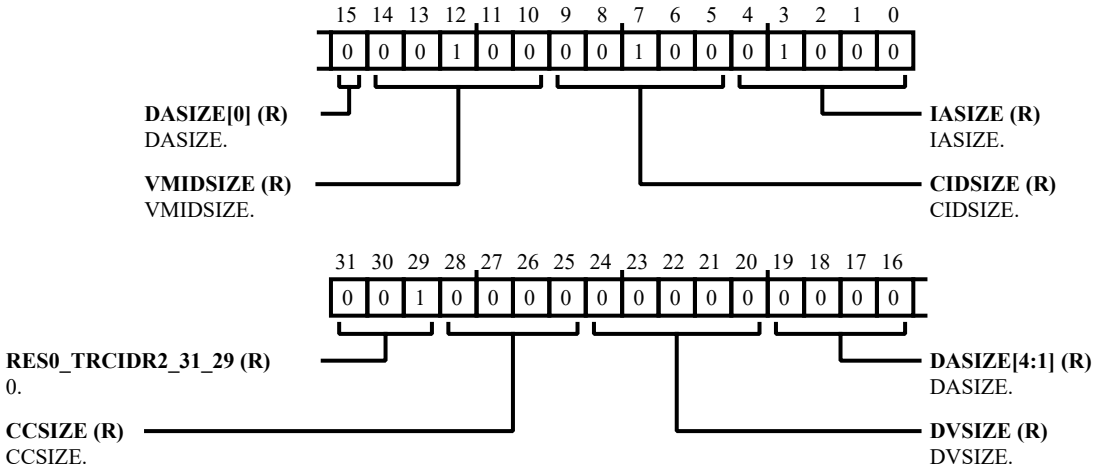


Figure 48-734: A55_C0_TRCIDR2 Register Diagram

Table 48-741: A55_C0_TRCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/NW)	RES0_TRCIDR2_31_29	0.. Reserved, RES0.	
28:25 (R/NW)	CCSIZE	CCSIZE.. Indicates the size of the cycle counter in bits minus 12. 0000 The cycle counter is 12 bits in length. 0001 The cycle counter is 13 bits in length. and so on up to 0b1000, indicating the cycle counter is 20 bits in length.All other values are reserved.If cycle counting is not implemented, as indicated by TRCIDR0.TRCCCI, this field is 0b0000.	
		0	The cycle counter is 12 bits in length.
		1	The cycle counter is 13 bits in length.

Table 48-741: A55_C0_TRCIDR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24:20 (R/NW)	DVSIZE	DVSIZE.. Indicates the data value size in bytes. The permitted values are: 00000 Data value tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00. 00100 Maximum of 32-bit data value size. 01000 Maximum of 64-bit data value size. All other values are reserved.	
		0	Data value tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00.
		4	Maximum of 32-bit data value size.
		8	Maximum of 64-bit data value size.
19:15 (R/NW)	DASIZE	DASIZE.. Indicates the data address size in bytes. The permitted values are: 00000 Data address tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00. 00100 Maximum of 32-bit data address size. 01000 Maximum of 64-bit data address size. All other values are reserved.	
		0	Data address tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00.
		4	Maximum of 32-bit data address size.
		8	Maximum of 64-bit data address size.
14:10 (R/NW)	VMIDSIZE	VMIDSIZE.. Indicates the VMID size. The permitted values are: 00000 VMID tracing is not supported. 00001 Maximum of 8-bit VMID size, so TRCCONFIGR.VMID is supported. All other values are reserved.	
		0	VMID tracing is not supported.
		1	Maximum of 8-bit VMID size, so TRCCONFIGR.VMID is supported.
9:5 (R/NW)	CIDSIZE	CIDSIZE.. Indicates the Context ID size. The permitted values are: 00000 Context ID tracing is not supported. 00100 Maximum of 32-bit Context ID size, so TRCCONFIGR.CID is supported. All other values are reserved.	
		0	Context ID tracing is not supported.
		4	Maximum of 32-bit Context ID size, so TRCCONFIGR.CID is supported.

Table 48-741: A55_C0_TRCIDR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:0 (R/NW)	IASIZE	IASIZE.. Indicates the instruction address size. The permitted values are: 00100 Maximum of 32-bit address size. 01000 Maximum of 64-bit address size. All other values are reserved.	
		4	Maximum of 32-bit address size.
		8	Maximum of 64-bit address size.

Trace ID Register 3.

Indicates the features that the trace unit supports.

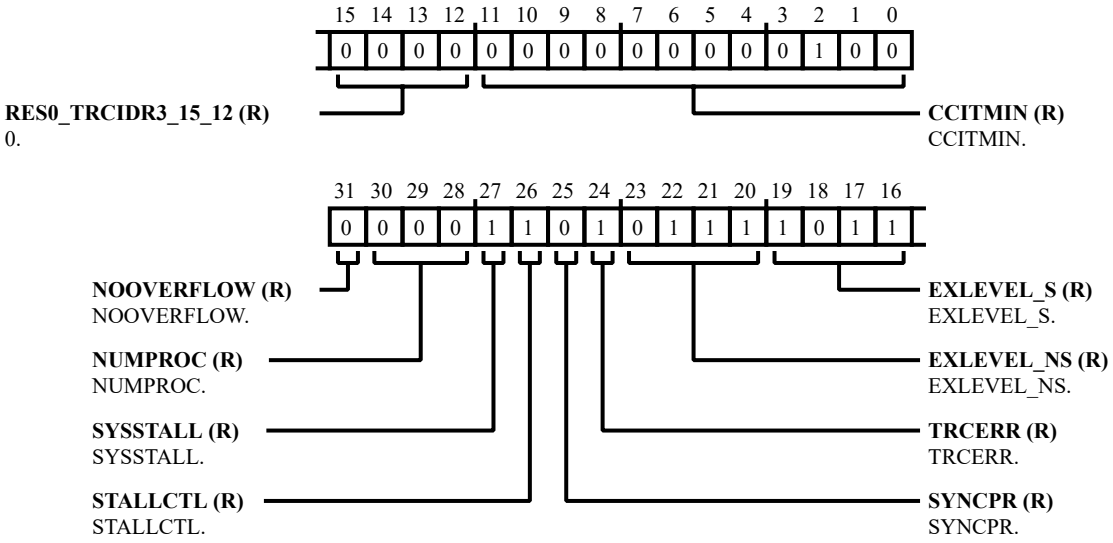


Figure 48-735: A55_C0_TRCIDR3 Register Diagram

Table 48-742: A55_C0_TRCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	NOOVERFLOW	NOOVERFLOW.. Indicates if TRCSTALLCTLR.NOOVERFLOW is supported: 0 TRCSTALLCTLR.NOOVERFLOW is not supported, or STALLCTL==0. 1 TRCSTALLCTLR.NOOVERFLOW is supported.	
		0	TRCSTALLCTLR.NOOVERFLOW is not supported, or STALLCTL==0.
		1	TRCSTALLCTLR.NOOVERFLOW is supported.
30:28 (R/NW)	NUMPROC	NUMPROC.. Indicates the number of processors available for tracing. The possible values are: 000 The trace unit can trace one processor. 001 The trace unit can trace two processors. 010 The trace unit can trace three processors. and so on up to 0b111, which indicates the trace unit can trace eight processors.This field sets the maximum value of TRCPROCSELR.PROCSEL.	
		0	The trace unit can trace one processor.
		1	The trace unit can trace two processors.
		2	The trace unit can trace three processors.

Table 48-742: A55_C0_TRCIDR3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	SYSSTALL	SYSSTALL.. Indicates if the implementation can support stall control: 0 The system does not support stall control of the processor. 1 The system can support stall control of the processor. The system supports stalling of the processor only when SYSSTALL==1 and STALLCTL==1.	
		0	The system does not support stall control of the processor.
		1	The system can support stall control of the processor.
26 (R/NW)	STALLCTL	STALLCTL.. Indicates if TRCSTALLCTLR is supported: 0 TRCSTALLCTLR is not supported. 1 TRCSTALLCTLR is supported.	
		0	TRCSTALLCTLR is not supported.
		1	TRCSTALLCTLR is supported.
25 (R/NW)	SYNCPR	SYNCPR.. Indicates if an implementation has a fixed synchronization period: 0 TRCSYNCPR is read-write so software can change the synchronization period. 1 TRCSYNCPR is read-only so the synchronization period is fixed.	
		0	TRCSYNCPR is read-write so software can change the synchronization period.
		1	TRCSYNCPR is read-only so the synchronization period is fixed.
24 (R/NW)	TRCERR	TRCERR.. Indicates if TRCVICTLR.TRCERR is supported: 0 TRCVICTLR.TRCERR is not supported 1 TRCVICTLR.TRCERR is supported.	
		0	TRCVICTLR.TRCERR is not supported.
		1	TRCVICTLR.TRCERR is supported.

Table 48-742: A55_C0_TRCIDR3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/NW)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit indicates whether instruction tracing is supported for the corresponding exception level: 0 In Non-secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are not supported. 1 In Non-secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are supported. The exception levels are:Bit[20]Exception level 0.Bit[21]Exception level 1.Bit[22]Exception level 2.Bit[23]SBZ. EXLEVEL_NS[3] is never implemented.	
		0	In Non-secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are not supported.
		1	In Non-secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are supported.
19:16 (R/NW)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit indicates whether instruction tracing is supported for the corresponding exception level: 0 In Secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are not supported. 1 In Secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are supported. The exception levels are:Bit[16]Exception level 0.Bit[17]Exception level 1.Bit[18]SBZ. EXLEVEL_S[2] is never implemented.Bit[19]Exception level 3.	
		0	In Secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are not supported.
		1	In Secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are supported.
15:12 (R/NW)	RES0_TRCIDR3_15_12	0.. Reserved, RES0.	
11:0 (R/NW)	CCITMIN	CCITMIN.. Indicates the minimum value that can be programmed in TRCCCCTLR.THRESHOLD.When cycle counting in the instruction trace is supported, that is TRCIDR0.TRCCCI==1, then the minimum value of this field is 0x001, otherwise it is 0x000.	

Trace ID Register 4.

Returns how many resources the trace unit supports.

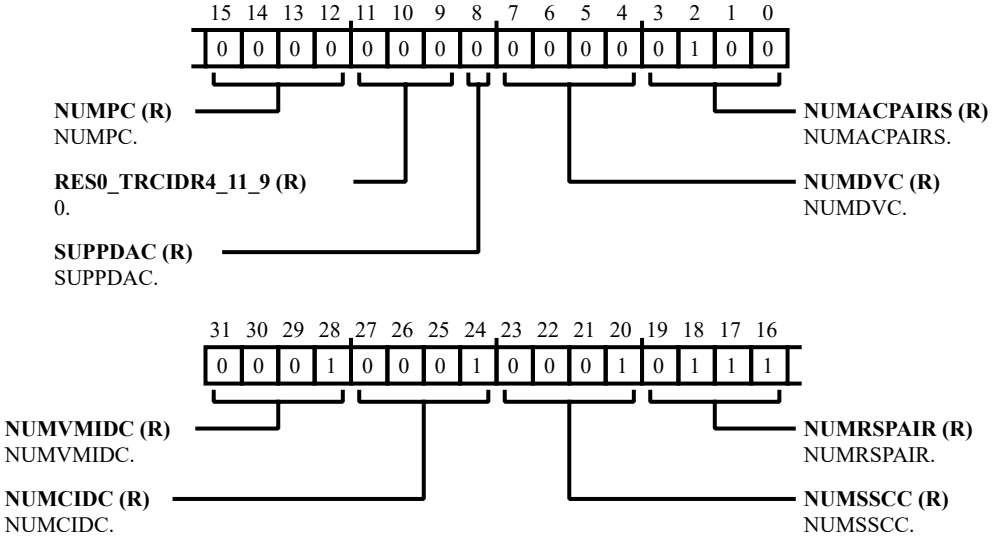


Figure 48-736: A55_C0_TRCIDR4 Register Diagram

Table 48-743: A55_C0_TRCIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	NUMVMIDC	NUMVMIDC.. Indicates the number of VMID comparators that are available for tracing. The permitted values are: 0000 No VMID comparators are available. 0001 The implementation has one VMID comparator. 0010 The implementation has two VMID comparators. and so on up to 0b1000, which indicates that the implementation has eight VMID comparators.All other values are reserved.	
		0	No VMID comparators are available.
		1	The implementation has one VMID comparator.
		2	The implementation has two VMID comparators.
27:24 (R/NW)	NUMCIDC	NUMCIDC.. Indicates the number of Context ID comparators that are available for tracing. The permitted values are: 0000 No Context ID comparators are available. 0001 The implementation has one Context ID comparator. 0010 The implementation has two Context ID comparators. and so on up to 0b1000, which indicates that the implementation has eight Context ID comparators.All other values are reserved.	
		0	No Context ID comparators are available.
		1	The implementation has one Context ID comparator.
		2	The implementation has two Context ID comparators.

Table 48-743: A55_C0_TRCIDR4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/NW)	NUMSSCC	NUMSSCC.. Indicates the number of single-shot comparator controls that are available for tracing. The permitted values are: 0000 No single-shot comparator controls are available. 0001 The implementation has one single-shot comparator control. 0010 The implementation has two single-shot comparator controls. and so on up to 0b1000, which indicates that the implementation has eight single-shot comparator controls.All other values are reserved.	
		0	No single-shot comparator controls are available.
		1	The implementation has one single-shot comparator control.
		2	The implementation has two single-shot comparator controls.
19:16 (R/NW)	NUMRSPAIR	NUMRSPAIR.. Indicates the number of resource selection pairs that are available for tracing. The permitted values are: 0000 The implementation has one resource selection pair. 0001 The implementation has two resource selection pairs. 0010 The implementation has three resource selection pairs. and so on up to 0b1111, which indicates that the implementation has 16 resource selection pairs.Implementations always have at least one resource selection pair so that they can support the FALSE and TRUE resource selectors, 0 and 1.	
		0	The implementation has one resource selection pair.
		1	The implementation has two resource selection pairs.
		2	The implementation has three resource selection pairs.
15:12 (R/NW)	NUMPC	NUMPC.. Indicates the number of processor comparator inputs that are available for tracing. The permitted values are: 0000 No processor comparator inputs are available. 0001 The implementation has one processor comparator input. 0010 The implementation has two processor comparator inputs. and so on up to 0b1000, which indicates that the implementation has eight processor comparator inputs.All other values are reserved.	
		0	No processor comparator inputs are available.
		1	The implementation has one processor comparator input.
		2	The implementation has two processor comparator inputs.
11:9 (R/NW)	RES0_TRCIDR4_11_9	0.. Reserved, RES0.	

Table 48-743: A55_C0_TRCIDR4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	SUPPDAC	SUPPDAC.. Indicates if the implementation can support data address comparisons: 0 The implementation does not support data address comparisons. 1 The implementation can support data address comparisons.	
		0	The implementation does not support data address comparisons.
		1	The implementation can support data address comparisons.
7:4 (R/NW)	NUMDVC	NUMDVC.. Indicates the number of data value comparators that are available for tracing. The permitted values are: 0000 No data value comparators are available. 0001 The implementation has one data value comparator. 0010 The implementation has two data value comparators. and so on up to 0b1000, which indicates that the implementation has eight data value comparators.All other values are reserved.	
		0	No data value comparators are available.
		1	The implementation has one data value comparator.
		2	The implementation has two data value comparators.
3:0 (R/NW)	NUMACPAIRS	NUMACPAIRS.. Indicates the number of address comparator pairs that are available for tracing. The permitted values are: 0000 No address comparator pairs are available. 0001 The implementation has one address comparator pair. 0010 The implementation has two address comparator pairs. and so on up to 0b1000, which indicates that the implementation has eight address comparator pairs.All other values are reserved.	
		0	No address comparator pairs are available.
		1	The implementation has one address comparator pair.
		2	The implementation has two address comparator pairs.

Trace ID Register 5.

Indicates the features that the trace unit supports.

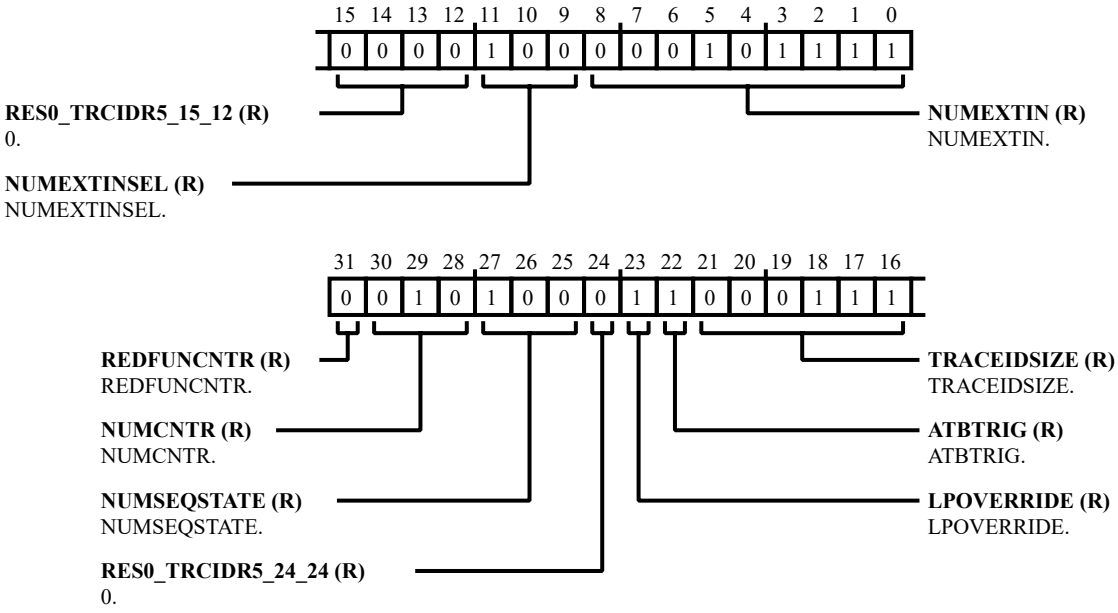


Figure 48-737: A55_C0_TRCIDR5 Register Diagram

Table 48-744: A55_C0_TRCIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	REDFUNCNTR	REDFUNCNTR.. Indicates if the reduced function counter is implemented: 0 The reduced function counter is not supported. 1 Counter 0 is implemented as a reduced function counter.	
		0	The reduced function counter is not supported.
		1	Counter 0 is implemented as a reduced function counter.
30:28 (R/NW)	NUMCNTR	NUMCNTR.. Indicates the number of counters that are available for tracing. The permitted values are: 000 No counters are available. 001 The implementation has one counter. 010 The implementation has two counters. 011 The implementation has three counters. 100 The implementation has four counters. All other values are reserved.	
		0	No counters are available.
		1	The implementation has one counter.
		2	The implementation has two counters.
		3	The implementation has three counters.

Table 48-744: A55_C0_TRCIDR5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	The implementation has four counters.
27:25 (R/NW)	NUMSEQSTATE	NUMSEQSTATE.. Indicates the number of sequencer states that are implemented. The permitted values are: 000 No sequencer states are implemented. 100 The implementation has four sequencer states. All other values are reserved.	
		0	No sequencer states are implemented.
		4	The implementation has four sequencer states.
24 (R/NW)	RES0_TRCIDR5_24_24	0.. Reserved, RES0.	
23 (R/NW)	LPOVERRIDE	LPOVERRIDE.. Indicates if the implementation can support low-power state override: 0 The implementation does not support low-power state override. 1 The implementation supports low-power state override, and the TRCEVENTCTL1R.LPOVERRIDE field is implemented. The trace unit must support low-power state override if it can enter a low-power mode where the resources and event trace generation are disabled.	
		0	The implementation does not support low-power state override.
		1	The implementation supports low-power state override, and the TRCEVENTCTL1R.LPOVERRIDE field is implemented.
22 (R/NW)	ATBTRIG	ATBTRIG.. Indicates if the implementation can support ATB triggers: 0 The implementation does not support ATB triggers. 1 The implementation supports ATB triggers, and the TRCEVENTCTL1R.ATBTRIG field is implemented.	
		0	The implementation does not support ATB triggers.
		1	The implementation supports ATB triggers, and the TRCEVENTCTL1R.ATBTRIG field is implemented.
21:16 (R/NW)	TRACEIDSIZE	TRACEIDSIZE.. Indicates the trace ID width. The permitted value is: 111 The implementation supports a 7-bit trace ID. This sets the width of the TRCTRACEIDR.TRACEID field. All other values are reserved.The CoreSight ATB requires a 7-bit trace ID width.	
		7	The implementation supports a 7-bit trace ID. This sets the width of the TRCTRACEIDR.TRACEID field.
15:12 (R/NW)	RES0_TRCIDR5_15_12	0.. Reserved, RES0.	

Table 48-744: A55_C0_TRCIDR5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:9 (R/NW)	NUMEXTINSEL	NUMEXTINSEL.. Indicates how many external input select resources are implemented. The permitted values are: 000 No external input select resources are available. If NUMEXTINSEL is zero, NUMEXTIN must also be zero. 001 The implementation has one external input select resource. 010 The implementation has two external input select resources. 011 The implementation has three external input select resources. 100 The implementation has four external input select resources. All other values are reserved.See TRCEXTINSELR for how to select an input select resource.	
		0	No external input select resources are available. If NUMEXTINSEL is zero, NUMEXTIN must also be zero.
		1	The implementation has one external input select resource.
		2	The implementation has two external input select resources.
		3	The implementation has three external input select resources.
		4	The implementation has four external input select resources.
8:0 (R/NW)	NUMEXTIN	NUMEXTIN.. Indicates how many external inputs are implemented. The permitted values are: 000000000 No external inputs are available. If NUMEXTIN is zero, NUMEXTINSEL must also be zero. 000000001 The implementation has one external input. 000000010 The implementation has two external inputs. and so on up to 0b100000000, which indicates that the implementation has 256 external inputs.All other values >0b100000000 are reserved.See TRCEXTINSELR for how to select an external input.	
		0	No external inputs are available. If NUMEXTIN is zero, NUMEXTINSEL must also be zero.
		1	The implementation has one external input.
		2	The implementation has two external inputs.

Trace IMPLEMENTATION DEFINED Register 0.

TRCIMSPEC0 shows the presence of any IMPLEMENTATION DEFINED features, and provides an interface to enable the features that are provided. The TRCIMSPECn registers are reserved for the future implementation of up to eight IMPLEMENTATION DEFINED registers. When a trace unit does not implement these registers, TRCIMSPEC0 must still be partially implemented so that a debugger can implement a general mechanism for detecting the IMPLEMENTATION DEFINED registers.

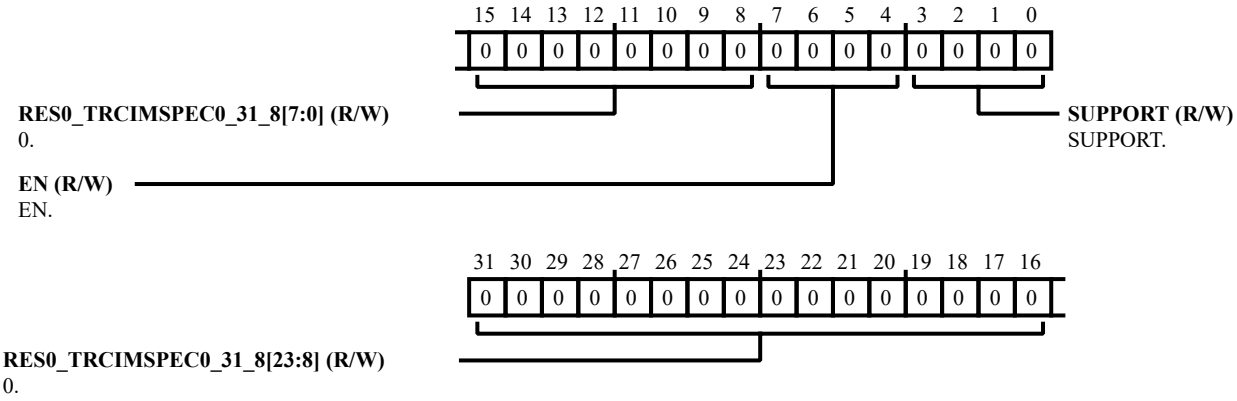


Figure 48-738: A55_C0_TRCIMSPEC0 Register Diagram

Table 48-745: A55_C0_TRCIMSPEC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	RES0_TRCIM-SPEC0_31_8	0.. Reserved, RES0.	
7:4 (R/W)	EN	EN.. If SUPPORT is not 0b0000, controls whether the IMPLEMENTATION DEFINED features are enabled. The permitted values are: 0000 The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported. and any other value, which indicates that the trace unit behavior is IMPLEMENTATION DEFINED.If SUPPORT is 0b0000, this field is RES0.	
		0	The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported.

Table 48-745: A55_C0_TRCIMSPEC0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	SUPPORT	SUPPORT.. Indicates whether the implementation supports IMPLEMENTATION DEFINED features. The permitted values are: 0000 No IMPLEMENTATION DEFINED features are supported. The EN field is RES0. and any other value, which indicates that IMPLEMENTATION DEFINED features are supported. Use of these values requires written permission from ARM.	
		0	No IMPLEMENTATION DEFINED features are supported. The EN field is RES0.

Integration ATB Identification Register.

Integration ATB Identification Register.

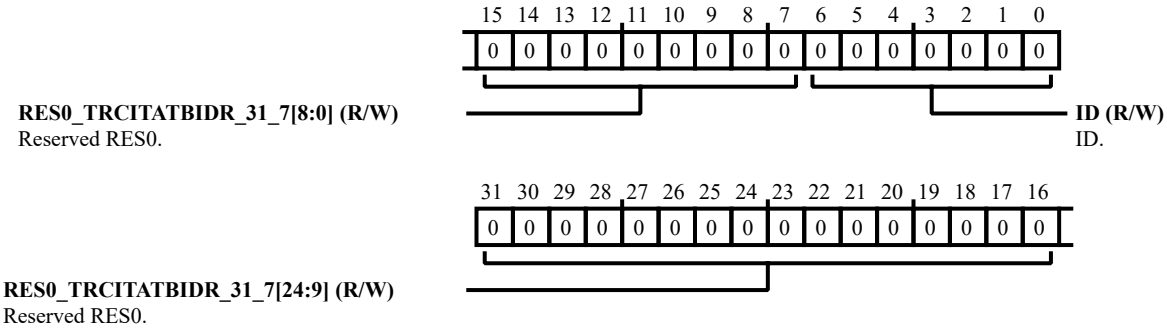


Figure 48-739: A55_C0_TRCITATBIDR Register Diagram

Table 48-746: A55_C0_TRCITATBIDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/W)	RES0_TRCITAT-BIDR_31_7	Reserved RES0.. Reserved RES0.
		0
6:0 (R/W)	ID	ID.. Drives the ATIDMn[6:0] output pins.

Trace Integration Mode Control Register.

Controls whether the trace unit is in integration mode.

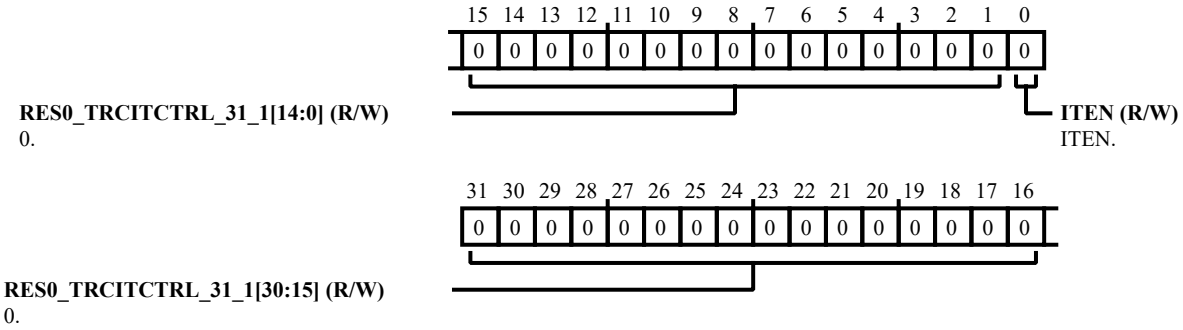


Figure 48-740: A55_C0_TRCITCTRL Register Diagram

Table 48-747: A55_C0_TRCITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_TRCITCTRL_31_1	0.. Reserved, RES0.	
0 (R/W)	ITEN	ITEN.. Integration mode enable bit: 0 The trace unit is not in integration mode. 1 The trace unit is in integration mode. This mode enables a debug agent to perform topology detection, and System-on-Chip (SoC) test software to perform integration testing.	
		0	The trace unit is not in integration mode.
		1	The trace unit is in integration mode. This mode enables a debug agent to perform topology detection, and System-on-Chip (SoC) test software to perform integration testing.

Integration Instruction ATB In Register.

Integration Instruction ATB In Register.

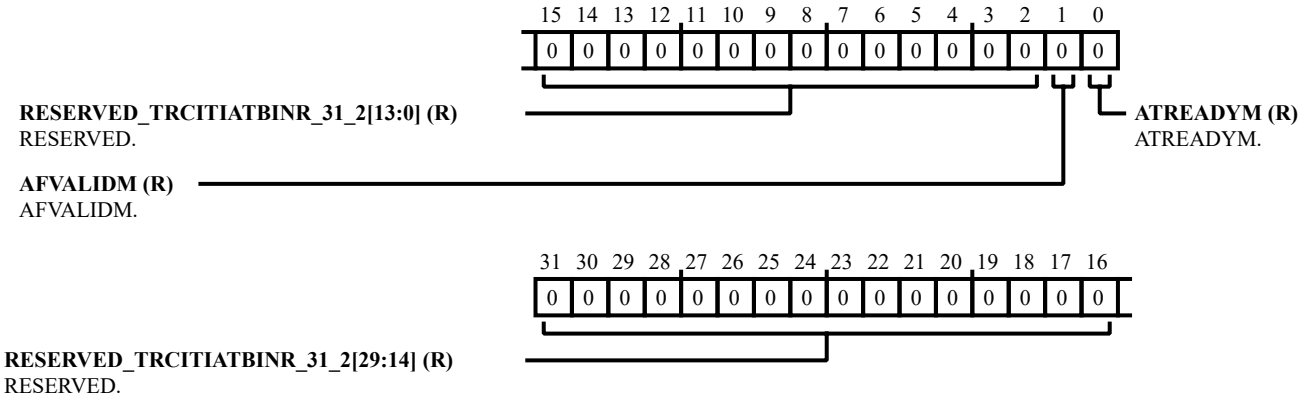


Figure 48-741: A55_C0_TRCITIATBINR Register Diagram

Table 48-748: A55_C0_TRCITIATBINR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/NW)	RESERVED_TRCITIAT-BINR_31_2	RESERVED.. Reserved. Read undefined.
1 (R/NW)	AFVALIDM	AFVALIDM.. Returns the value of the AFVALIDMn input pin.
0 (R/NW)	ATREADYM	ATREADYM.. Returns the value of the ATREADYMn input pin.

Integration Instruction ATB Out Register.

Integration Instruction ATB Out Register.

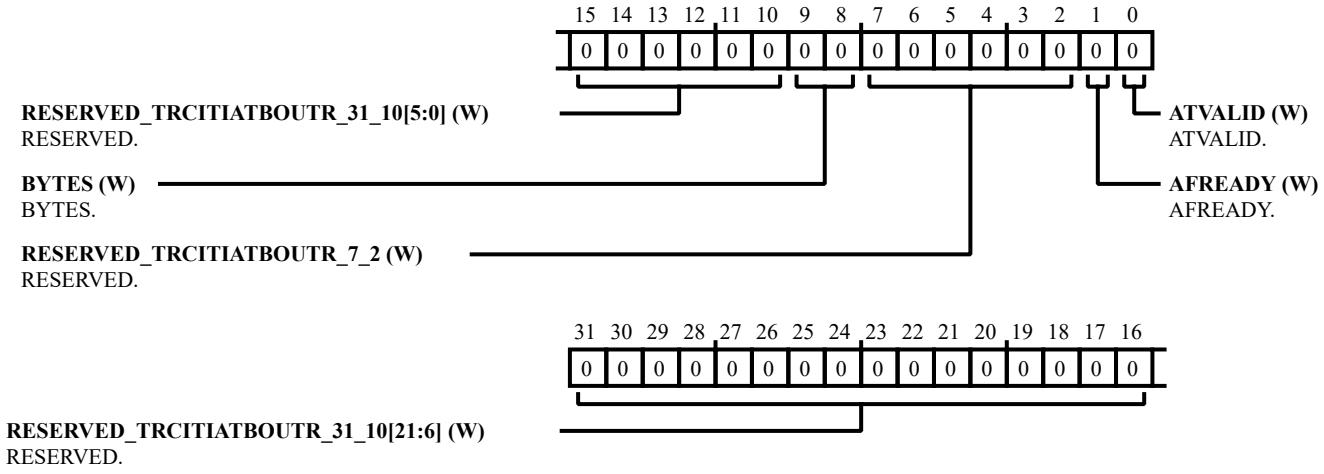


Figure 48-742: A55_C0_TRCITIATBOUTR Register Diagram

Table 48-749: A55_C0_TRCITIATBOUTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:10 (RX/W)	RESERVED_TRCITIAT-BOUTR_31_10	RESERVED.. Reserved. Read undefined.
9:8 (RX/W)	BYTES	BYTES.. Drives the ATBYTESMn[1:0] output pins.
7:2 (RX/W)	RESERVED_TRCITIAT-BOUTR_7_2	RESERVED.. Reserved. Read undefined.
1 (RX/W)	AFREADY	AFREADY.. Drives the AFREADYMn output pin.
0 (RX/W)	ATVALID	ATVALID.. Drives the ATVALIDMn output pin.

Integration Instruction ATB Data Register.

Integration Instruction ATB Data Register.

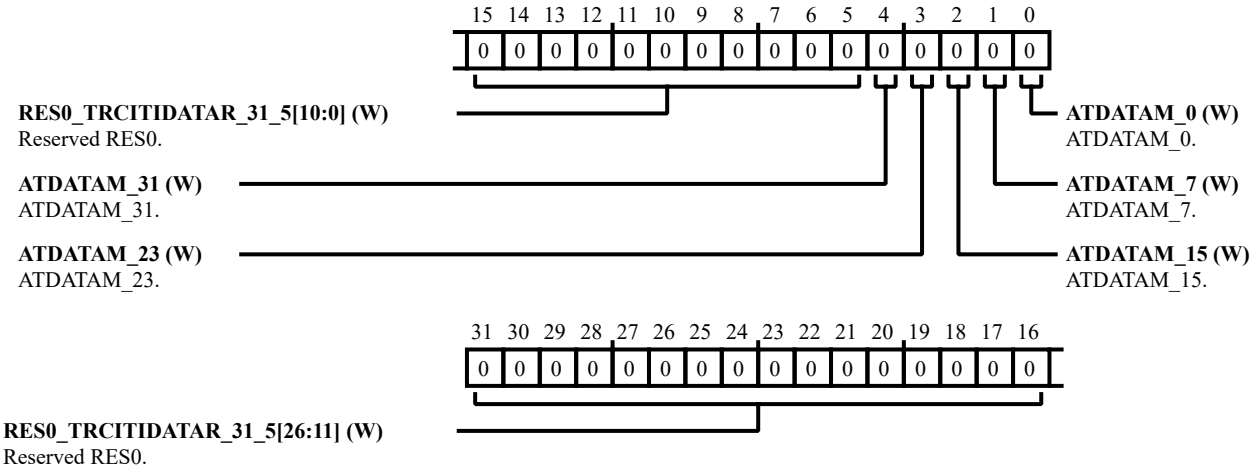


Figure 48-743: A55_C0_TRCITIDATAR Register Diagram

Table 48-750: A55_C0_TRCITIDATAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (RX/W)	RES0_TRCITIDA-TAR_31_5	Reserved RES0.. Reserved RES0.	
		0	Reserved.
4 (RX/W)	ATDATAM_31	ATDATAM_31.. Drives the ATDATAM[31] output.	
3 (RX/W)	ATDATAM_23	ATDATAM_23.. Drives the ATDATAM[23] output.	
2 (RX/W)	ATDATAM_15	ATDATAM_15.. Drives the ATDATAM[15] output.	
1 (RX/W)	ATDATAM_7	ATDATAM_7.. Drives the ATDATAM[7] output.	
0 (RX/W)	ATDATAM_0	ATDATAM_0.. Drives the ATDATAM[0] output.	

Trace OS Lock Access Register.

Controls whether the OS Lock is locked.

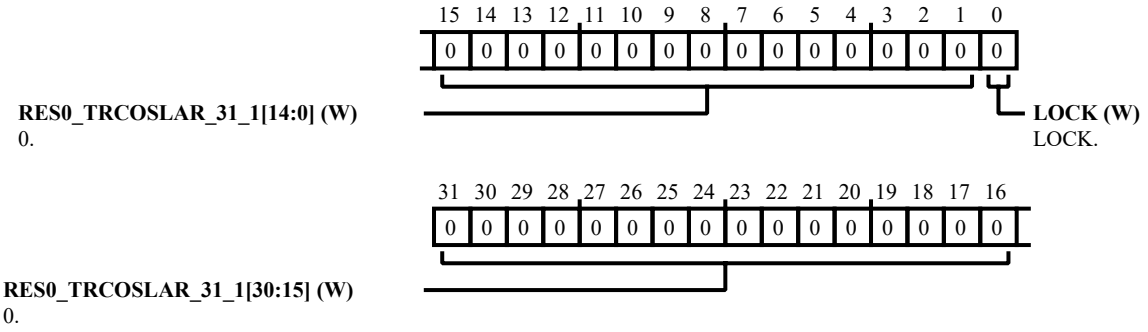


Figure 48-744: A55_C0_TRCOSLAR Register Diagram

Table 48-751: A55_C0_TRCOSLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (RX/W)	RES0_TRCOSLAR_31_1	0.. Reserved, RES0.	
0 (RX/W)	LOCK	LOCK.. OS Lock control bit: 0 Unlocks the OS Lock. 1 Locks the OS Lock. This setting disables the trace unit.	
		0	Unlocks the OS Lock.
		1	Locks the OS Lock. This setting disables the trace unit.

Trace OS Lock Status Register.

Returns the status of the OS Lock.

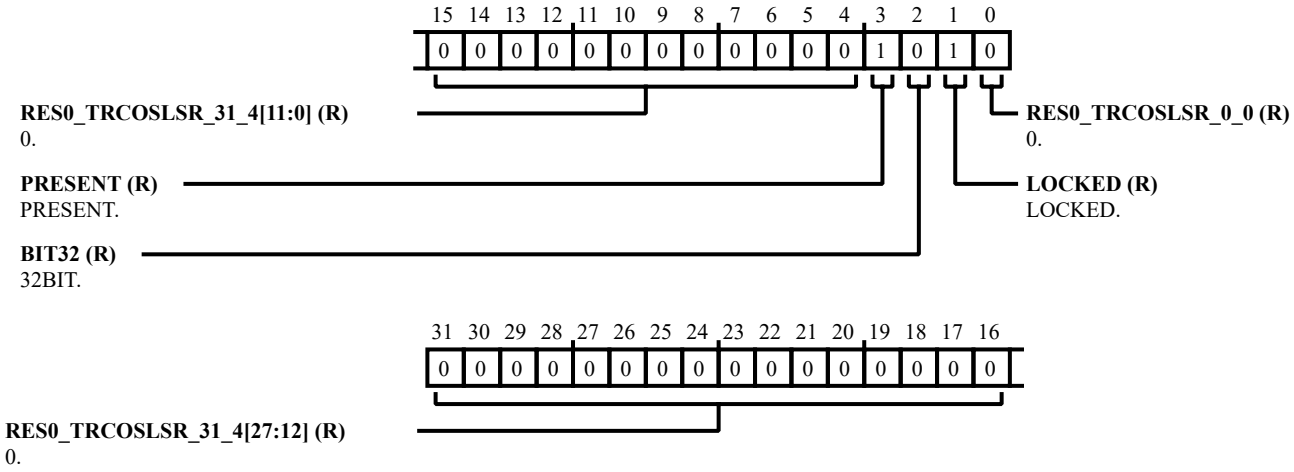


Figure 48-745: A55_C0_TRCOSLSR Register Diagram

Table 48-752: A55_C0_TRCOSLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_TRCOSLSR_31_4	0.. Reserved, RES0.	
3 (R/NW)	PRESENT	PRESENT.. Indicates whether the OS Lock is implemented.This bit is RES1, which indicates that the OS Lock is always implemented.	
2 (R/NW)	BIT32	32BIT.. This bit is RES0, which indicates that software must perform a 32-bit write to update the TRCOSLAR.	
1 (R/NW)	LOCKED	LOCKED.. OS Lock status bit: 0 The OS Lock is unlocked. 1 The OS Lock is locked. When the trace unit core power domain is powered down the value is UNKNOWN. The TRCPDSR indicates if the trace unit core power domain is powered down.	
		0	The OS Lock is unlocked.
		1	The OS Lock is locked.
0 (R/NW)	RES0_TRCOSLSR_0_0	0.. Reserved, RES0.	

Trace PowerDown Status Register.

Returns information about the OS Lock status, core power domain status and power interruption status for the trace unit.

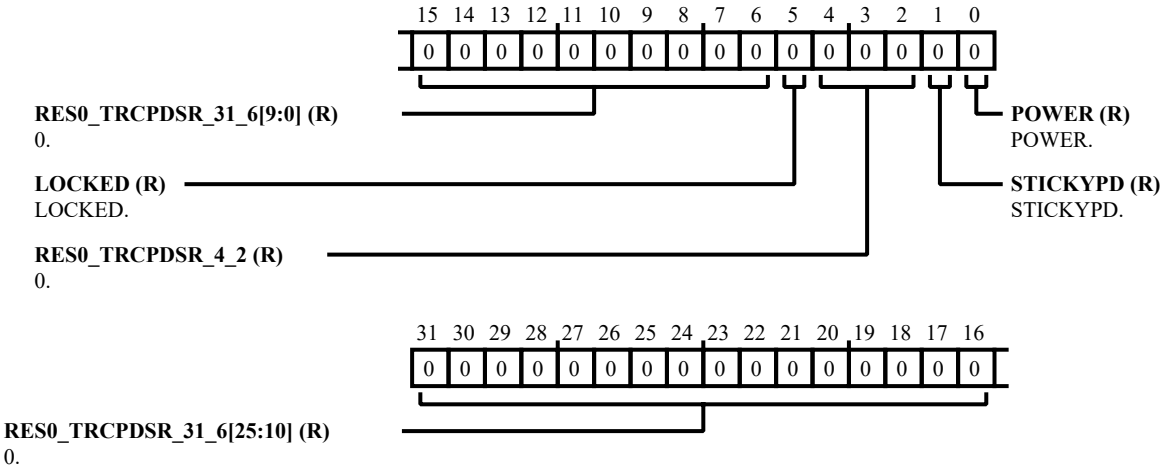


Figure 48-746: A55_C0_TRCPDSR Register Diagram

Table 48-753: A55_C0_TRCPDSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:6 (R/NW)	RES0_TRCPDSR_31_6	0.. Reserved, RES0.	
5 (R/NW)	LOCKED	LOCKED.. OS Lock status bit: 0 The OS Lock is unlocked. 1 The OS Lock is locked. The value is UNKNOWN when the trace unit core power domain is powered down, that is, when POWER==0.	
		0	The OS Lock is unlocked.
		1	The OS Lock is locked.
4:2 (R/NW)	RES0_TRCPDSR_4_2	0.. Reserved, RES0.	

Table 48-753: A55_C0_TRCPDSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	STICKYPD	STICKYPD.. Sticky powerdown status bit. Indicates whether the trace register state is valid: 0 If POWER==1 then the state of TRCOSLSR and the trace registers are valid. If POWER==0 then it is UNKNOWN whether the state of TRCOSLSR and the trace registers are valid. 1 The state of TRCOSLSR and the trace registers might not be valid. The trace unit sets this bit to 1 if either the trace unit is reset, or the power to the trace unit core power domain is removed and the trace register state is not valid. After this register is read, if the Software Lock is unlocked and the trace unit core power domain is powered up, then the trace unit sets this bit to 0. The TRCLAR controls whether the Software Lock is locked.	
		0	If POWER==1 then the state of TRCOSLSR and the trace registers are valid. If POWER==0 then it is UNKNOWN whether the state of TRCOSLSR and the trace registers are valid.
		1	The state of TRCOSLSR and the trace registers might not be valid. The trace unit sets this bit to 1 if either the trace unit is reset, or the power to the trace unit core power domain is removed and the trace register state is not valid.
0 (R/NW)	POWER	POWER.. Power status bit: 0 The trace unit core power domain is not powered. The trace registers are not accessible and they all return an error response. 1 The trace unit core power domain is powered. The trace registers are accessible.	
		0	The trace unit core power domain is not powered. The trace registers are not accessible and they all return an error response.
		1	The trace unit core power domain is powered. The trace registers are accessible.

Trace Programming Control Register.

Enables the trace unit.

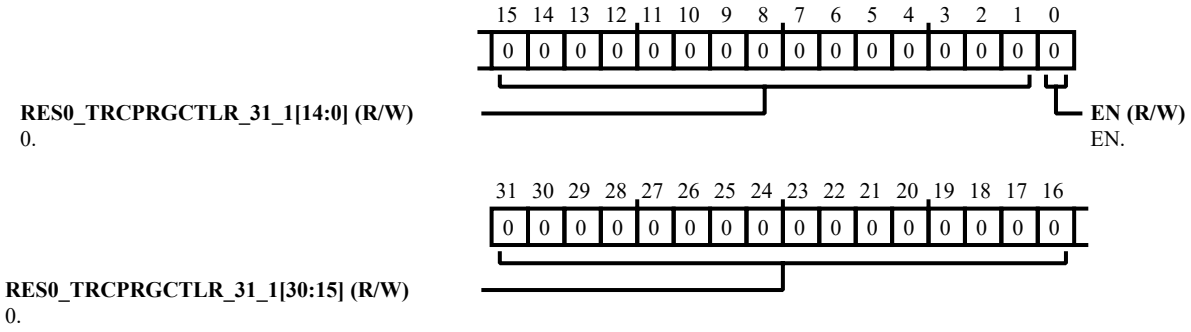


Figure 48-747: A55_C0_TRCPRGCTLR Register Diagram

Table 48-754: A55_C0_TRCPRGCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_TRCPRGCTLR_31_1	0.. Reserved, RES0.	
0 (R/W)	EN	EN.. Trace unit enable bit. Possible values are: 0 The trace unit is disabled. All trace resources are inactive and no trace is generated. 1 The trace unit is enabled.	
		0	The trace unit is disabled. All trace resources are inactive and no trace is generated.
		1	The trace unit is enabled.

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

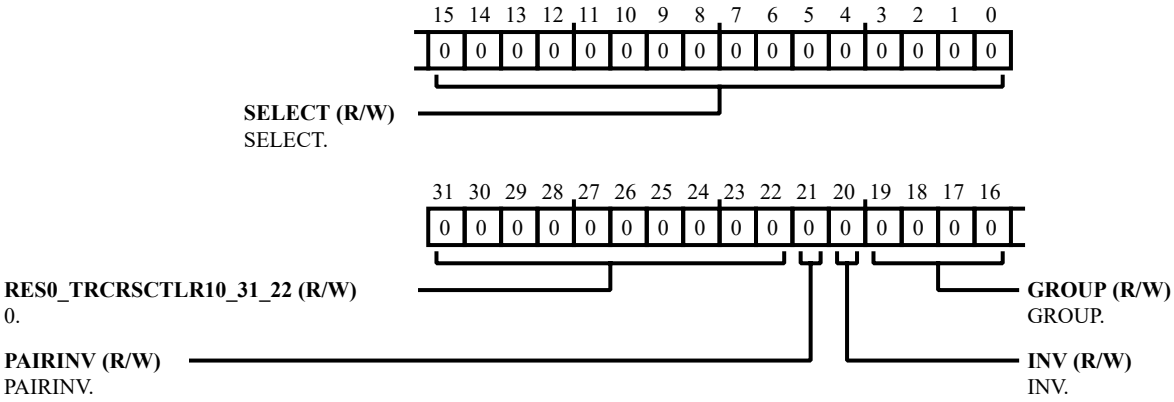


Figure 48-748: A55_C0_TRCRSCTLR10 Register Diagram

Table 48-755: A55_C0_TRCRSCTLR10 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR10_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-755: A55_C0_TRCRSCTLR10 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

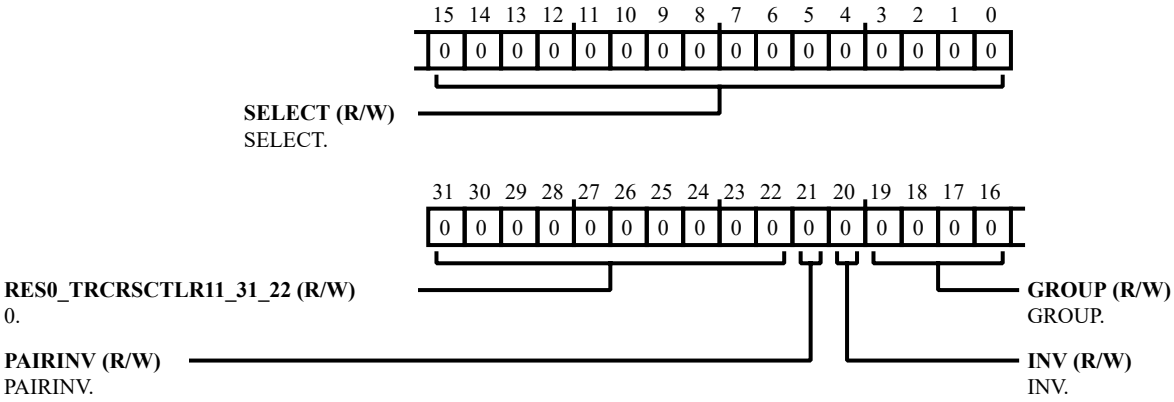


Figure 48-749: A55_C0_TRCRSCTLR11 Register Diagram

Table 48-756: A55_C0_TRCRSCTLR11 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR11_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-756: A55_C0_TRCRSCTLR11 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

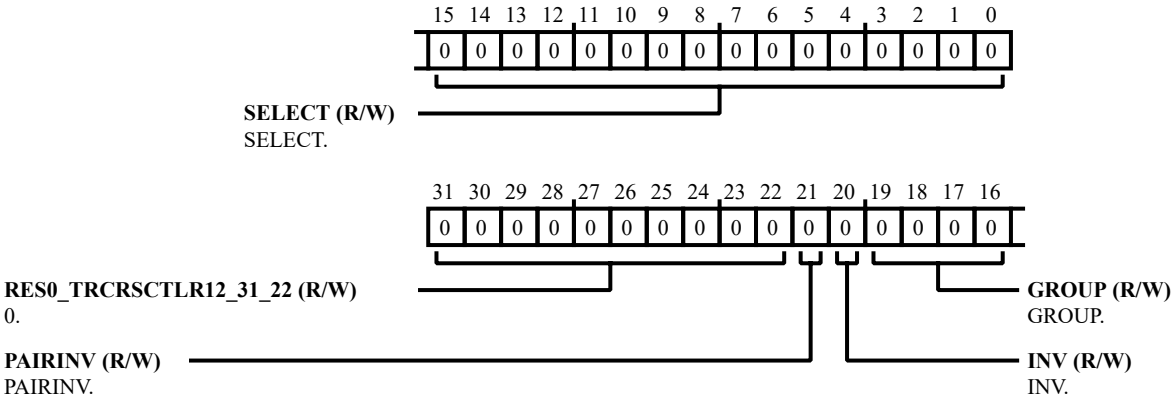


Figure 48-750: A55_C0_TRCRSCTLR12 Register Diagram

Table 48-757: A55_C0_TRCRSCTLR12 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR12_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-757: A55_C0_TRCRSCTLR12 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

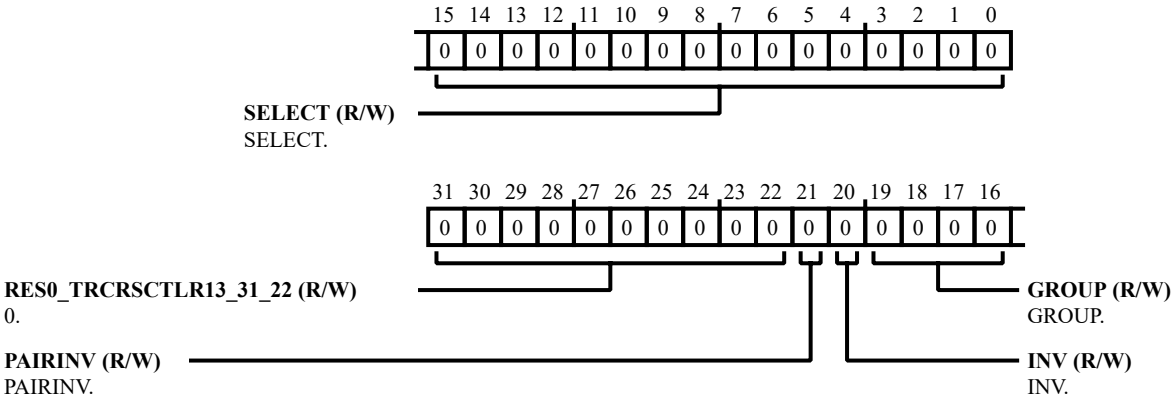


Figure 48-751: A55_C0_TRCRSCTLR13 Register Diagram

Table 48-758: A55_C0_TRCRSCTLR13 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR13_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-758: A55_C0_TRCRSCTLR13 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

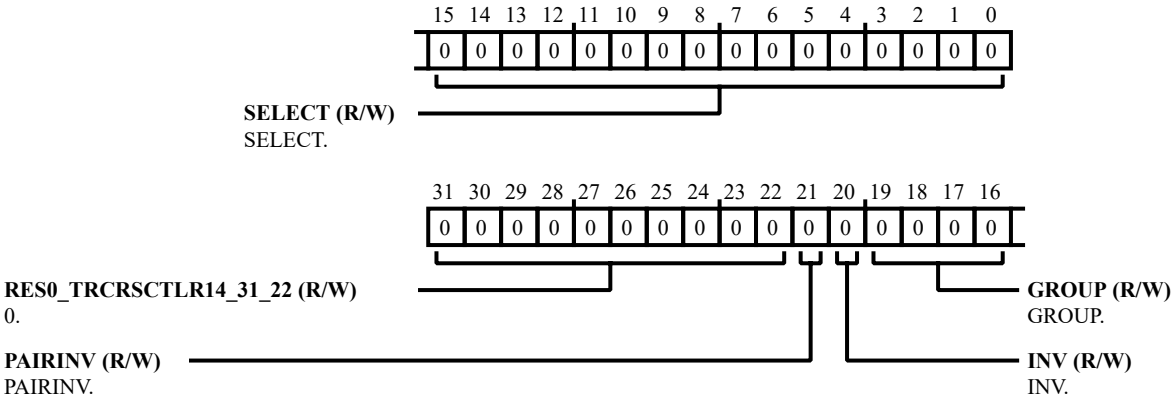


Figure 48-752: A55_C0_TRCRSCTLR14 Register Diagram

Table 48-759: A55_C0_TRCRSCTLR14 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR14_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-759: A55_C0_TRCRSCTLR14 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

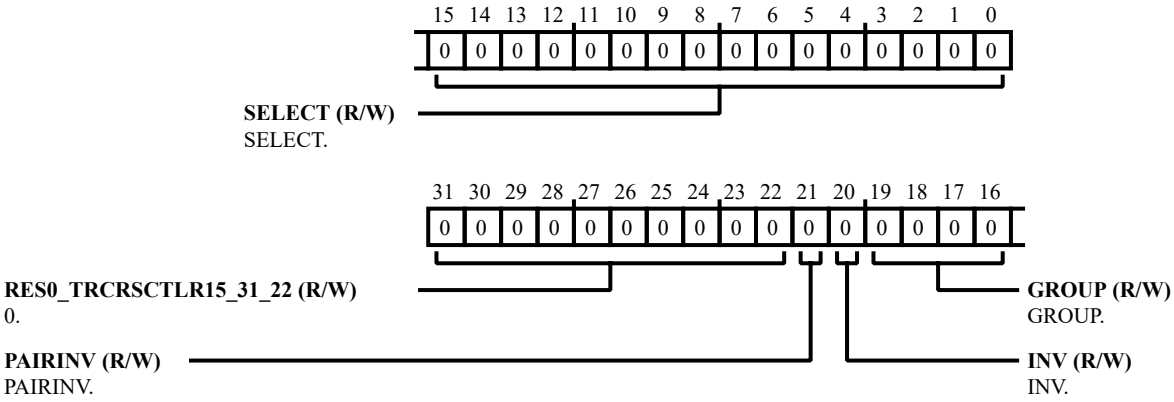


Figure 48-753: A55_C0_TRCRSCTLR15 Register Diagram

Table 48-760: A55_C0_TRCRSCTLR15 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR15_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-760: A55_C0_TRCRSCTLR15 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

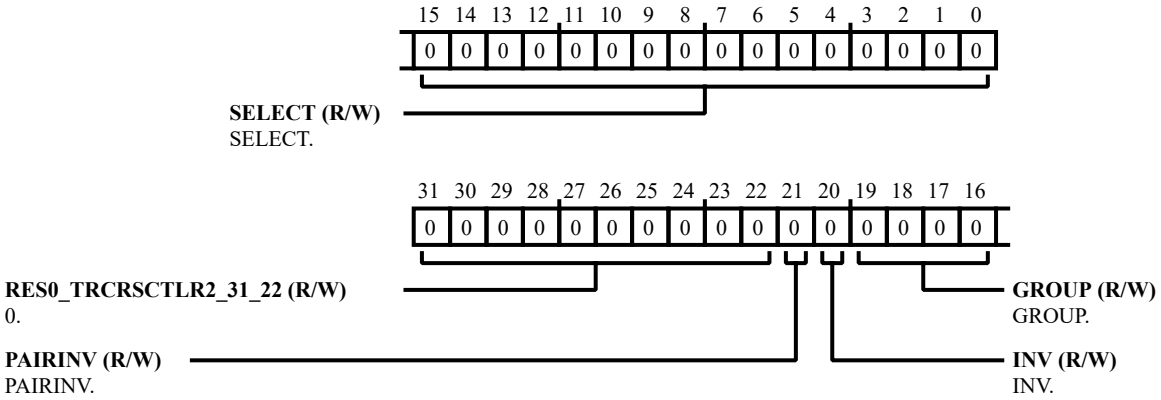


Figure 48-754: A55_C0_TRCRSCTLR2 Register Diagram

Table 48-761: A55_C0_TRCRSCTLR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR2_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-761: A55_C0_TRCRSCTLR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

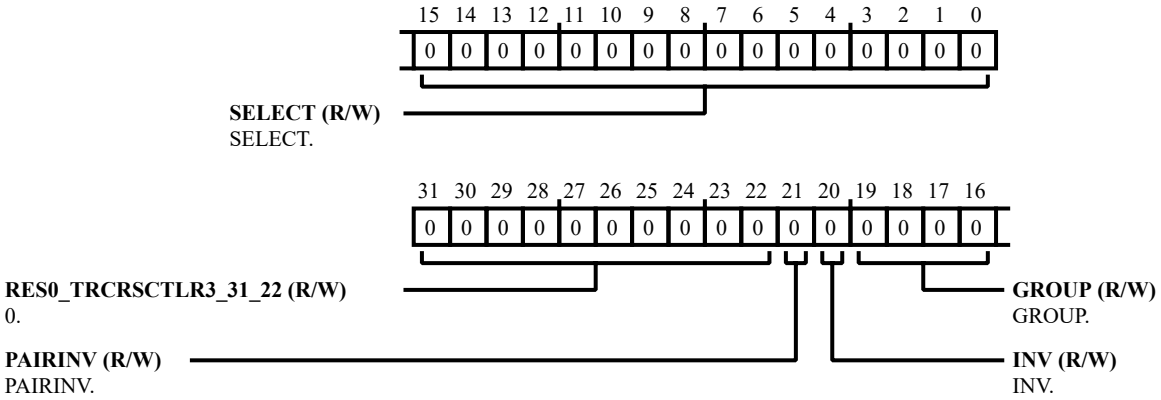


Figure 48-755: A55_C0_TRCRSCTLR3 Register Diagram

Table 48-762: A55_C0_TRCRSCTLR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR3_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-762: A55_C0_TRCRSCTLR3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

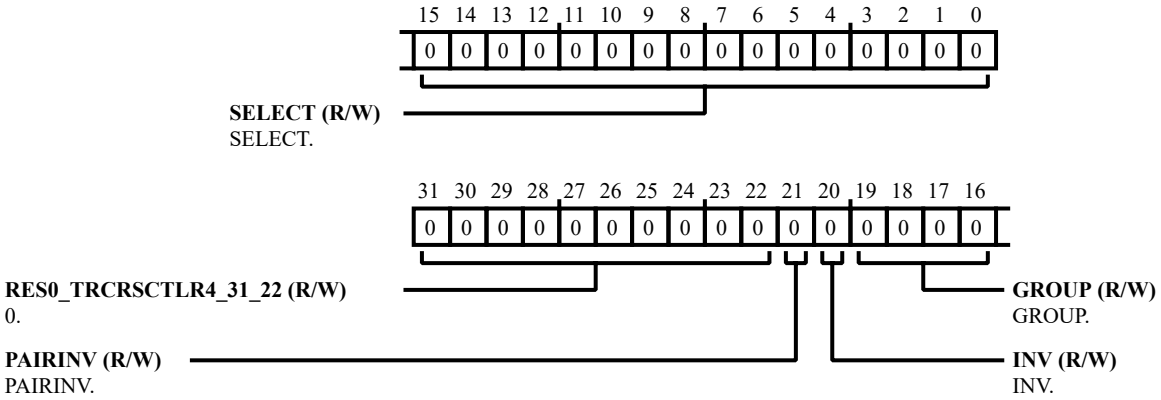


Figure 48-756: A55_C0_TRCRSCTLR4 Register Diagram

Table 48-763: A55_C0_TRCRSCTLR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR4_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-763: A55_C0_TRCRSCTLR4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

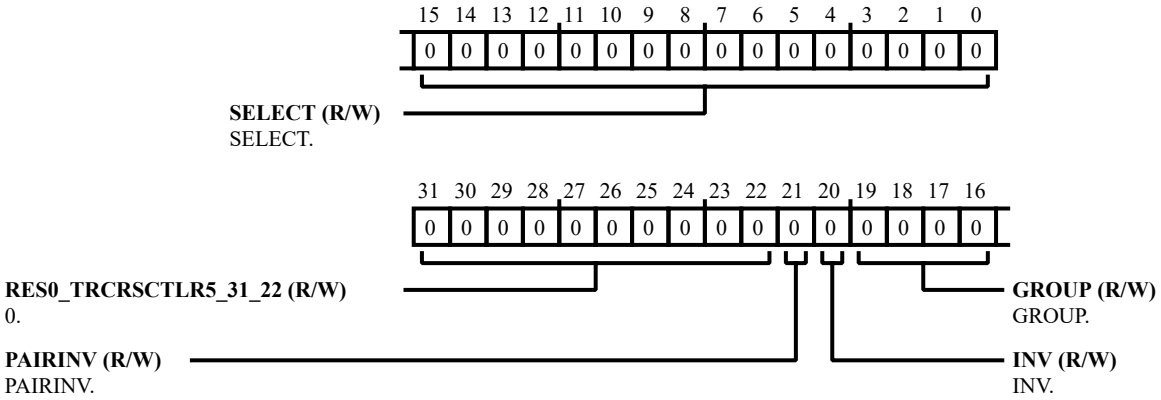


Figure 48-757: A55_C0_TRCRSCTLR5 Register Diagram

Table 48-764: A55_C0_TRCRSCTLR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR5_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-764: A55_C0_TRCRSCTLR5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

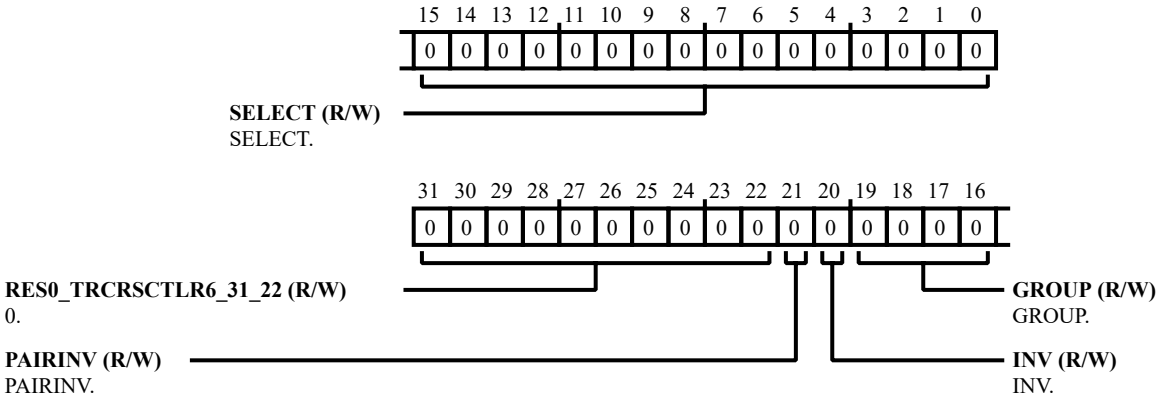


Figure 48-758: A55_C0_TRCRSCTLR6 Register Diagram

Table 48-765: A55_C0_TRCRSCTLR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR6_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-765: A55_C0_TRCRSCTLR6 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

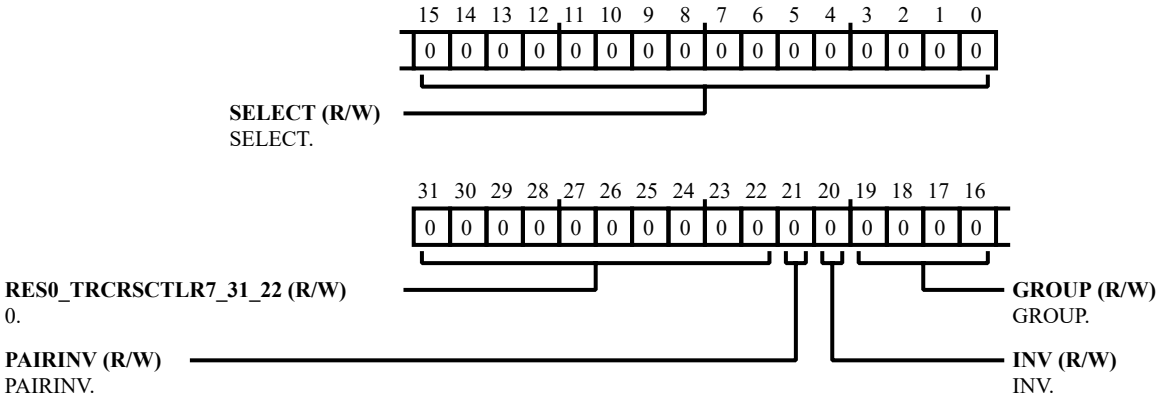


Figure 48-759: A55_C0_TRCRSCTLR7 Register Diagram

Table 48-766: A55_C0_TRCRSCTLR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR7_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-766: A55_C0_TRCRSCTLR7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

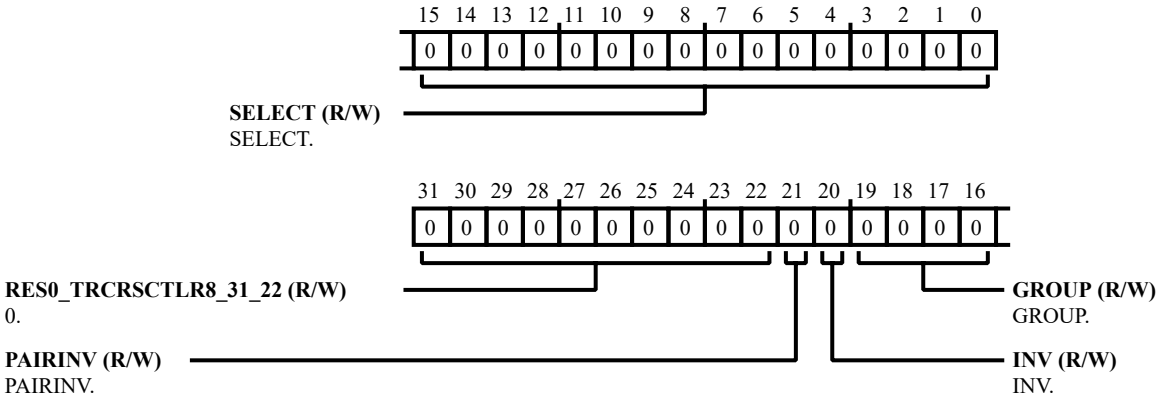


Figure 48-760: A55_C0_TRCRSCTLR8 Register Diagram

Table 48-767: A55_C0_TRCRSCTLR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR8_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-767: A55_C0_TRCRSCTLR8 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

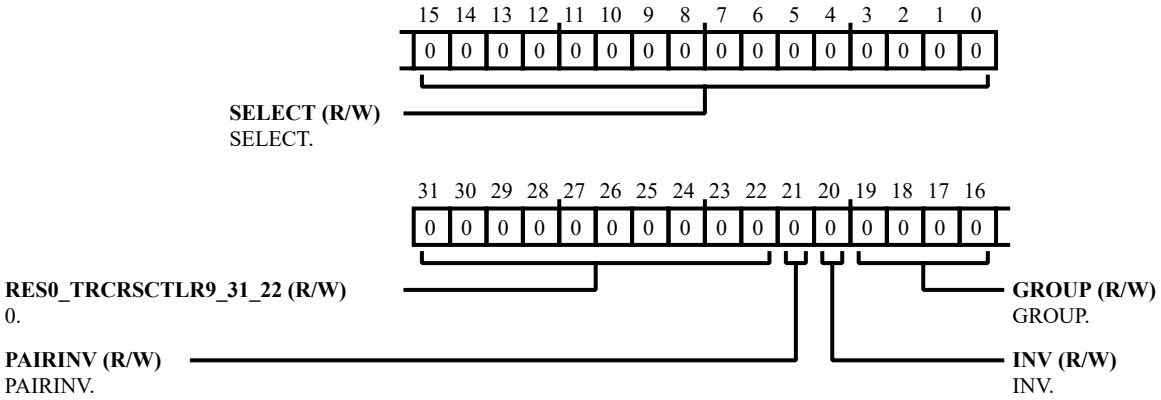


Figure 48-761: A55_C0_TRCRSCTLR9 Register Diagram

Table 48-768: A55_C0_TRCRSCTLR9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR9_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-768: A55_C0_TRCRSCTLR9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Sequencer State Transition Control Registers.

Moves the sequencer state backwards (from state n+1 to state n) when a programmed event occurs, and forwards (from state n to state n+1) when a programmed event occurs.

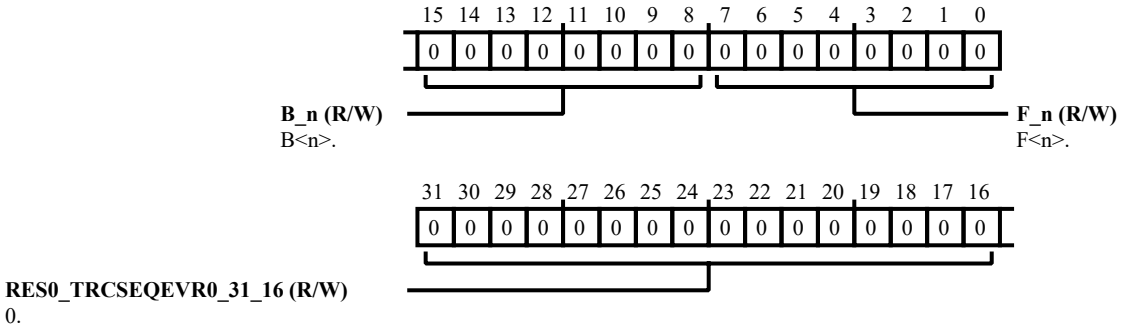


Figure 48-762: A55_C0_TRCSEQEVR0 Register Diagram

Table 48-769: A55_C0_TRCSEQEVR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCSE-QEVR0_31_16	0.. Reserved, RES0.
15:8 (R/W)	B_N	B<n>.. Backward field. Contains an event number. When the event occurs then the sequencer state moves from state n+1 to state n.For example, for TRCSEQEVR2, if B2==0x14 then when event 0x14 occurs, the sequencer moves from state 3 to state 2.
7:0 (R/W)	F_N	F<n>.. Forward field. Contains an event number. When the event occurs then the sequencer state moves from state n to state n+1.For example, for TRCSEQEVR1, if F1==0x12 then when event 0x12 occurs, the sequencer moves from state 1 to state 2.

Trace Sequencer State Transition Control Registers.

Moves the sequencer state backwards (from state n+1 to state n) when a programmed event occurs, and forwards (from state n to state n+1) when a programmed event occurs.

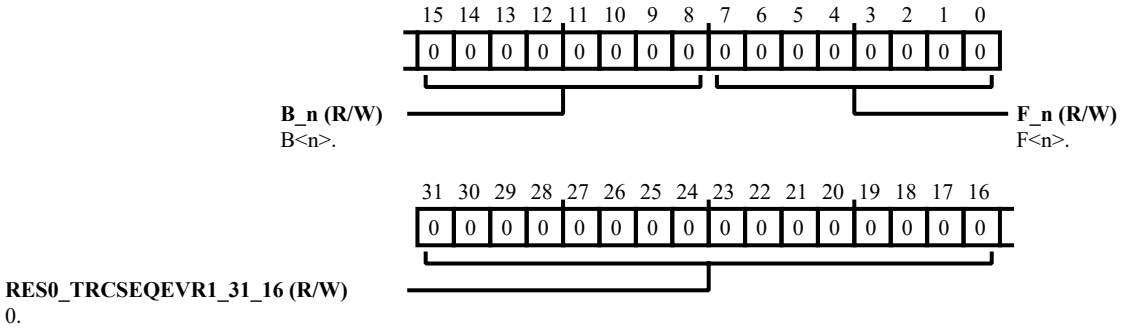


Figure 48-763: A55_C0_TRCSEQEVR1 Register Diagram

Table 48-770: A55_C0_TRCSEQEVR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCSE-QEVR1_31_16	0.. Reserved, RES0.
15:8 (R/W)	B_N	B<n>.. Backward field. Contains an event number. When the event occurs then the sequencer state moves from state n+1 to state n.For example, for TRCSEQEVR2, if B2==0x14 then when event 0x14 occurs, the sequencer moves from state 3 to state 2.
7:0 (R/W)	F_N	F<n>.. Forward field. Contains an event number. When the event occurs then the sequencer state moves from state n to state n+1.For example, for TRCSEQEVR1, if F1==0x12 then when event 0x12 occurs, the sequencer moves from state 1 to state 2.

Trace Sequencer State Transition Control Registers.

Moves the sequencer state backwards (from state n+1 to state n) when a programmed event occurs, and forwards (from state n to state n+1) when a programmed event occurs.

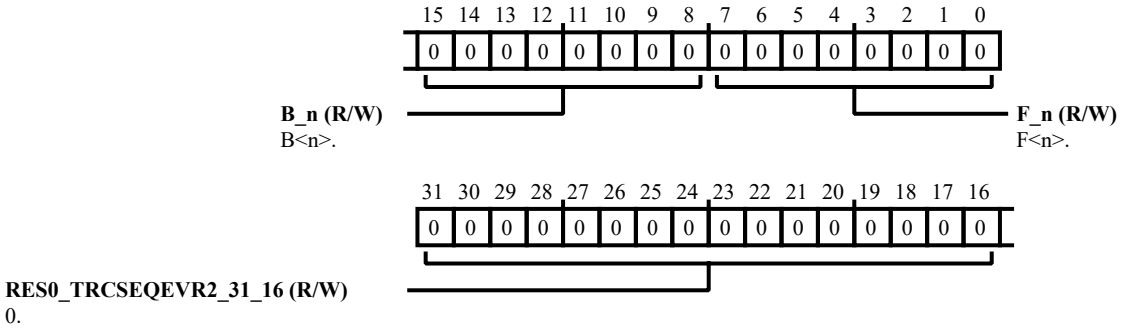


Figure 48-764: A55_C0_TRCSEQEVR2 Register Diagram

Table 48-771: A55_C0_TRCSEQEVR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCSE-QEVR2_31_16	0.. Reserved, RES0.
15:8 (R/W)	B_N	B<n>.. Backward field. Contains an event number. When the event occurs then the sequencer state moves from state n+1 to state n.For example, for TRCSEQEVR2, if B2==0x14 then when event 0x14 occurs, the sequencer moves from state 3 to state 2.
7:0 (R/W)	F_N	F<n>.. Forward field. Contains an event number. When the event occurs then the sequencer state moves from state n to state n+1.For example, for TRCSEQEVR1, if F1==0x12 then when event 0x12 occurs, the sequencer moves from state 1 to state 2.

Trace Sequencer Reset Control Register.

Moves the sequencer to state 0 when a programmed event occurs.

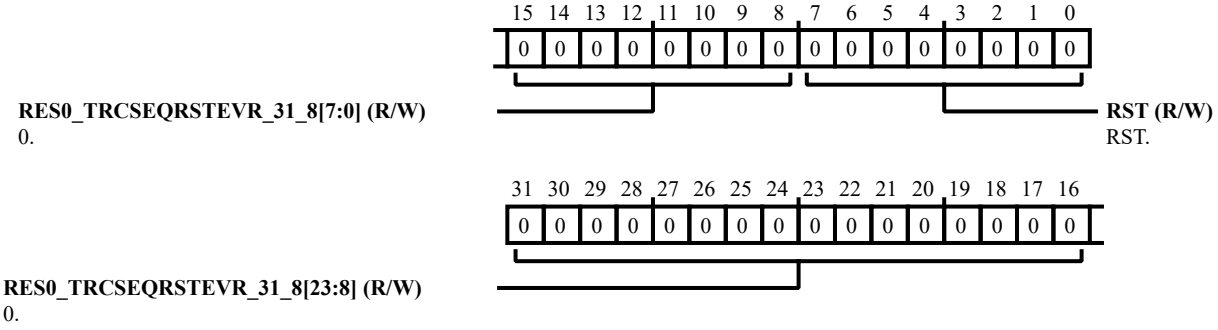


Figure 48-765: A55_C0_TRCSEQRSTEVR Register Diagram

Table 48-772: A55_C0_TRCSEQRSTEVR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_TRCSEQR-STEVR_31_8	0.. Reserved, RES0.
7:0 (R/W)	RST	RST.. Contains an event number. When the event occurs then the sequencer state moves to state 0.

Trace Sequencer State Register.

Use this to set, or read, the sequencer state.

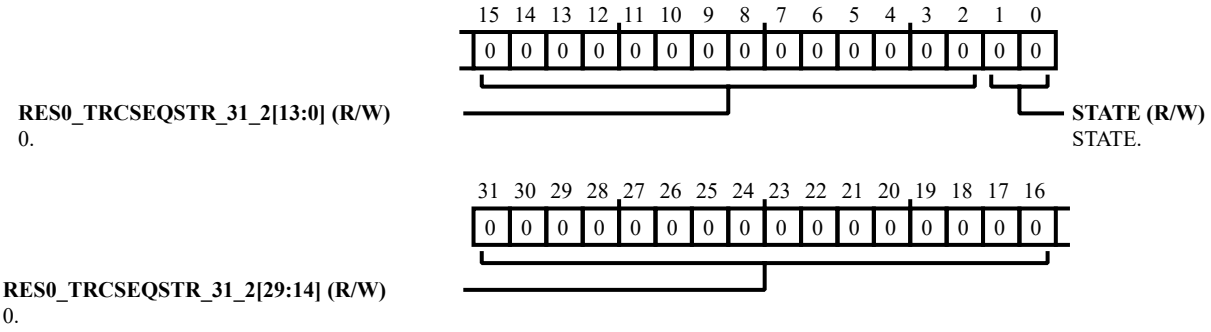


Figure 48-766: A55_C0_TRCSEQSTR Register Diagram

Table 48-773: A55_C0_TRCSEQSTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RES0_TRCSEQSTR_31_2	0.. Reserved, RES0.	
1:0 (R/W)	STATE	STATE.. Sets or returns the state of the sequencer: 00 State 0. 01 State 1. 10 State 2. 11 State 3.	
		0	State 0.
		1	State 1.
		2	State 2.
		3	State 3.

Trace Single-Shot Comparator Control Registers.

Controls the corresponding single-shot comparator resource.

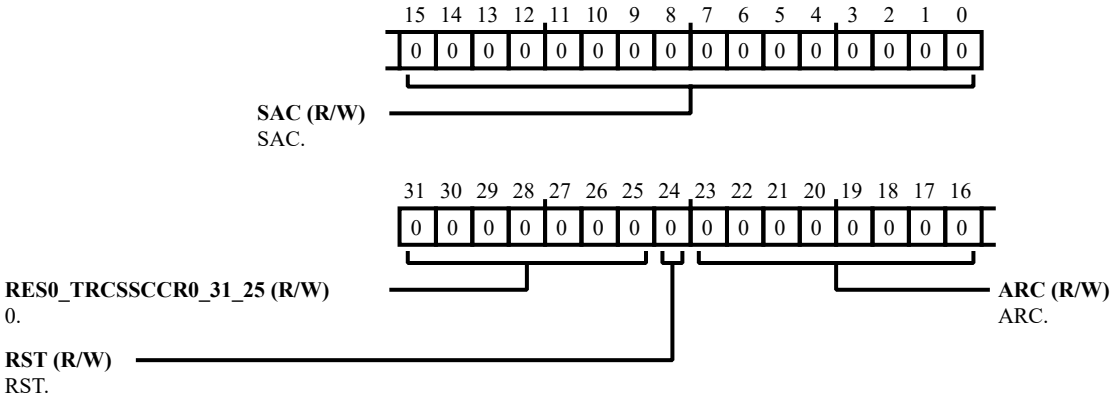


Figure 48-767: A55_C0_TRCSSCCR0 Register Diagram

Table 48-774: A55_C0_TRCSSCCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:25 (R/W)	RES0_TRCSSCCR0_31_25	0.. Reserved, RES0.	
24 (R/W)	RST	RST.. Controls whether the single-shot comparator resource is reset when it fires. 0 When the single-shot comparator resource fires, it is not reset. 1 When the single-shot comparator resource fires, it is reset. This enables the single-shot comparator resource to fire multiple times.	
		0	When the single-shot comparator resource fires, it is not reset.
		1	When the single-shot comparator resource fires, it is reset. This enables the single-shot comparator resource to fire multiple times.

Table 48-774: A55_C0_TRCSSCCR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	ARC	ARC.. Selects one or more address range comparators for single-shot control.Each bit represents an address range comparator pair, so bit[n-16] controls the selection of address range comparator pair n-16. If bit[n-16] is: 0 The address range comparator pair n-16 is not selected for single-shot control. 1 The address range comparator pair n-16 is selected for single-shot control. The width of this field is IMPLEMENTATION DEFINED. The field contains a number of implemented bits equal to TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The address range comparator pair n-16 is not selected for single-shot control.
		1	The address range comparator pair n-16 is selected for single-shot control.
15:0 (R/W)	SAC	SAC.. Selects one or more single address comparators for single-shot control.Each bit represents a single address comparator, so bit[n] controls the selection of single address comparator n. If bit[n] is: 0 The single address comparator n, is not selected for single-shot control. 1 The single address comparator n, is selected for single-shot control. The width of this field is IMPLEMENTATION DEFINED. The field contains a number of implemented bits equal to 2 x TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The single address comparator n, is not selected for single-shot control.
		1	The single address comparator n, is selected for single-shot control.

Trace Single-Shot Comparator Status Registers.

Returns the status of the corresponding single-shot comparator.

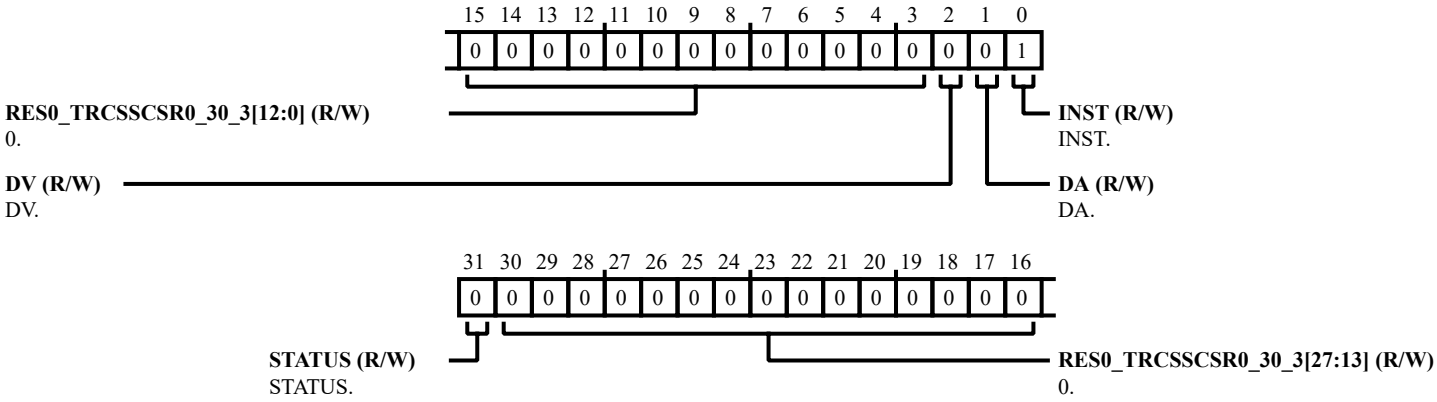


Figure 48-768: A55_C0_TRCSSCSR0 Register Diagram

Table 48-775: A55_C0_TRCSSCSR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	STATUS	STATUS.. Single-shot status bit. Indicates if any of the comparators that TRCSSCCRn.SAC or TRCSSCCRn.ARC selects have matched: 0 No match has occurred. 1 One or more matches has occurred. If TRCSSCCRn.RST==0 then there is only one match and no more matches are possible, and software must reset this bit to 0 to re-enable the single-shot control. STATUS must be written to set an initial state when configuring the trace unit, if the single-shot comparator is to be used.	
		0	No match has occurred.
		1	One or more matches has occurred. If TRCSSCCRn.RST==0 then there is only one match and no more matches are possible, and software must reset this bit to 0 to re-enable the single-shot control.
30:3 (R/W)	RES0_TRCSSCSR0_30_3	0.. Reserved, RES0.	

Table 48-775: A55_C0_TRCSSCSR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	DV	DV.. Data value comparator support bit. Indicates if the trace unit supports data address with data value comparisons: 0 Single-shot data address with data value comparisons are not supported. 1 Single-shot data address with data value comparisons are supported.	
		0	Single-shot data address with data value comparisons are not supported.
		1	Single-shot data address with data value comparisons are supported.
1 (R/W)	DA	DA.. Data address comparator support bit. Indicates if the trace unit supports data address comparisons: 0 Single-shot data address comparisons are not supported. 1 Single-shot data address comparisons are supported.	
		0	Single-shot data address comparisons are not supported.
		1	Single-shot data address comparisons are supported.
0 (R/W)	INST	INST.. Instruction address comparator support bit. Indicates if the trace unit supports instruction address comparisons: 0 Single-shot instruction address comparisons are not supported. 1 Single-shot instruction address comparisons are supported.	
		0	Single-shot instruction address comparisons are not supported.
		1	Single-shot instruction address comparisons are supported.

Trace Stall Control Register.

Trace Stall Control Register.

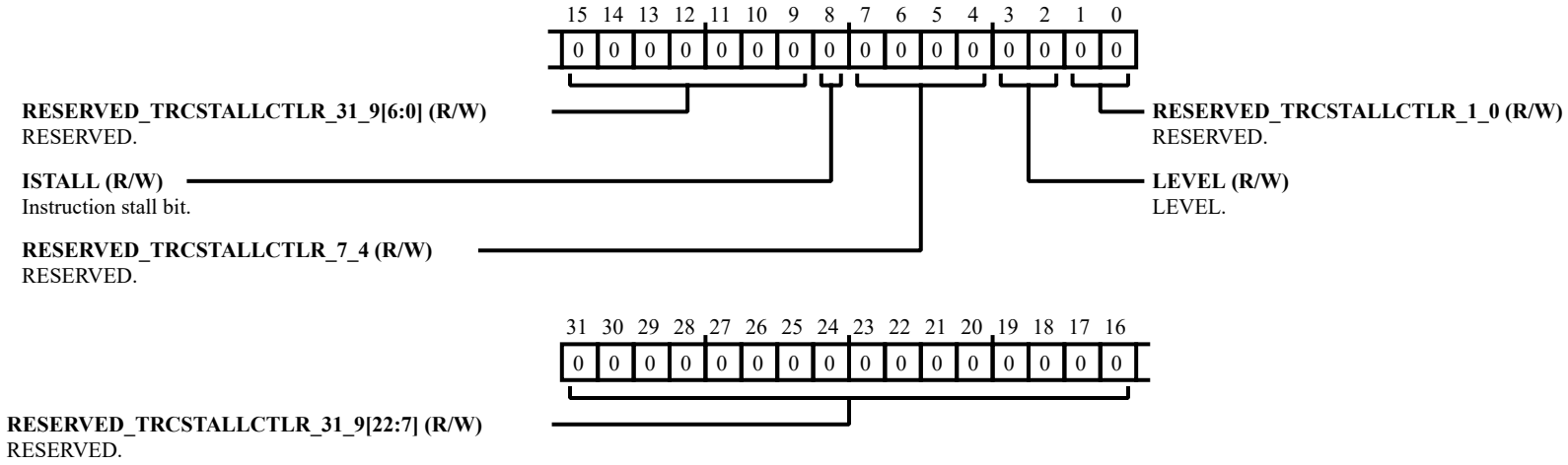


Figure 48-769: A55_C0_TRCSTALLCTLR Register Diagram

Table 48-776: A55_C0_TRCSTALLCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:9 (R/W)	RESERVED_TRCSTALLCTLR_31_9	RESERVED.. Reserved RES0.	
		0	Reserved.
8 (R/W)	ISTALL	Instruction stall bit.. Controls if the trace unit can stall the processor when the instruction trace buffer space is less than LEVEL.	
		0	The trace unit does not stall the processor.
		1	The trace unit can stall the processor.
7:4 (R/W)	RESERVED_TRCSTALLCTLR_7_4	RESERVED.. Reserved RES0.	
		0	Reserved.
3:2 (R/W)	LEVEL	LEVEL.. The field can support 4 monotonic levels from 0b00 to 0b11.	
		0	Minimum invasion. This setting has a greater risk of an ETM trace unit FIFO overflow.
		3	Maximum invasion occurs but there is less risk of a FIFO overflow.

Table 48-776: A55_C0_TRCSTALLCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	RESERVED_TRCSTALLCTLR_1_0	RESERVED.. Reserved RES0.	
		0	Reserved.

Trace Status Register.

Returns the trace unit status.

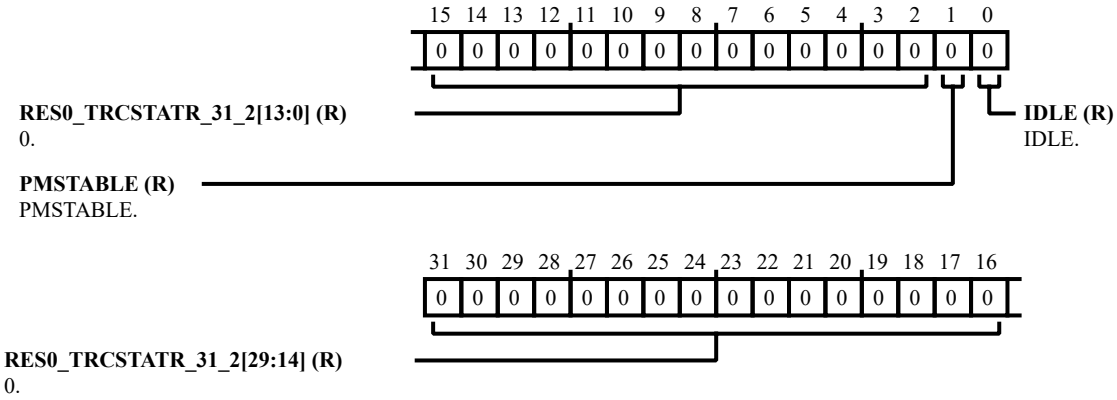


Figure 48-770: A55_C0_TRCSTATR Register Diagram

Table 48-777: A55_C0_TRCSTATR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RES0_TRCSTATR_31_2	0.. Reserved, RES0.	
1 (R/NW)	PMSTABLE	PMSTABLE.. Programmer's model stable bit: 0 The programmer's model is not stable. 1 The programmer's model is stable. When polled, the trace unit trace registers return stable data. The programmer's model is stable when all of the following are true:TRCPRGCTLR.EN==0 or the OS Lock is locked.Reads from trace unit registers return stable data.	
		0	The programmer's model is not stable.
		1	The programmer's model is stable. When polled, the trace unit trace registers return stable data.
0 (R/NW)	IDLE	IDLE.. Idle status bit: 0 The trace unit is not idle. 1 The trace unit is idle. The trace unit is idle when all of the following are true:TRCPRGCTLR.EN==0 or the OS Lock is locked.The trace unit is drained of any trace.With the exception of the programming interfaces, all external interfaces on the trace unit are quiescent.	
		0	The trace unit is not idle.
		1	The trace unit is idle.

Trace Synchronization Period Register.

Controls how often periodic trace synchronization requests occur.

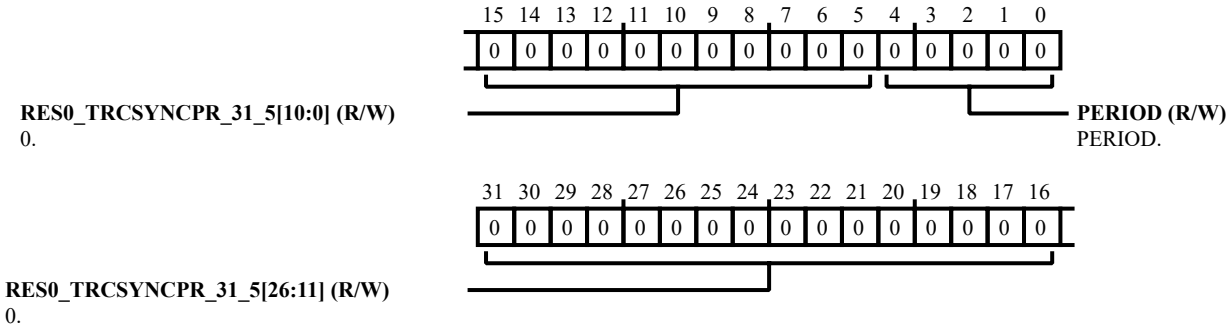


Figure 48-771: A55_C0_TRCSYNCPR Register Diagram

Table 48-778: A55_C0_TRCSYNCPR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (R/W)	RES0_TRCSYNCPR_31_5	0.. Reserved, RES0.	
4:0 (R/W)	PERIOD	PERIOD.. Controls how many bytes of trace, the sum of instruction and data, that a trace unit can generate before a periodic trace synchronization request occurs. The number of bytes is always a power of two and the permitted values are: 00000 Periodic trace synchronization requests are disabled. This setting does not disable other types of trace synchronization request. 01000 Periodic trace synchronization request occurs after 2^8, or 256, bytes of trace. 01001 Periodic trace synchronization request occurs after 2^9, or 512, bytes of trace. 01010 Periodic trace synchronization request occurs after 2^10, or 1024, bytes of trace. and so on up to 0b10100, for which the request occurs after 2^20, or 1048576, bytes of trace.Values between 0b00001 and 0b001111 are reserved, as are values from 0b10101 onwards.	
		0	Periodic trace synchronization requests are disabled. This setting does not disable other types of trace synchronization request.
		8	Periodic trace synchronization request occurs after 2^8, or 256, bytes of trace.
		9	Periodic trace synchronization request occurs after 2^9, or 512, bytes of trace.
		10	Periodic trace synchronization request occurs after 2^10, or 1024, bytes of trace.

Trace Trace ID Register.

Sets the trace ID for instruction trace. If data trace is enabled then it also sets the trace ID for data trace, to (trace ID for instruction trace) + 1.

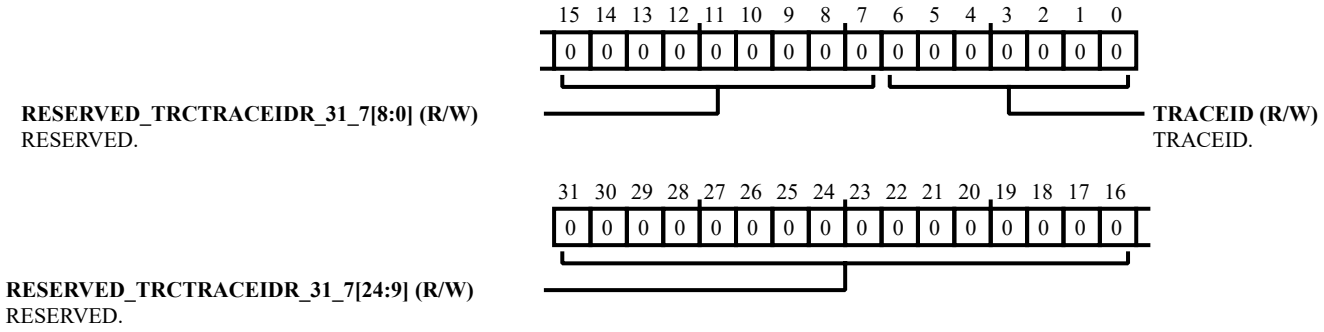


Figure 48-772: A55_C0_TRCTRACEIDR Register Diagram

Table 48-779: A55_C0_TRCTRACEIDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:7 (R/W)	RESERVED_TRCTRA-CEIDR_31_7	RESERVED.. Reserved RES0.	
		0	Reserved.
6:0 (R/W)	TRACEID	TRACEID.. Trace ID value. When only instruction tracing is enabled this provides the trace ID.	

Trace Global Timestamp Control Register.

Controls the insertion of global timestamps in the trace streams.

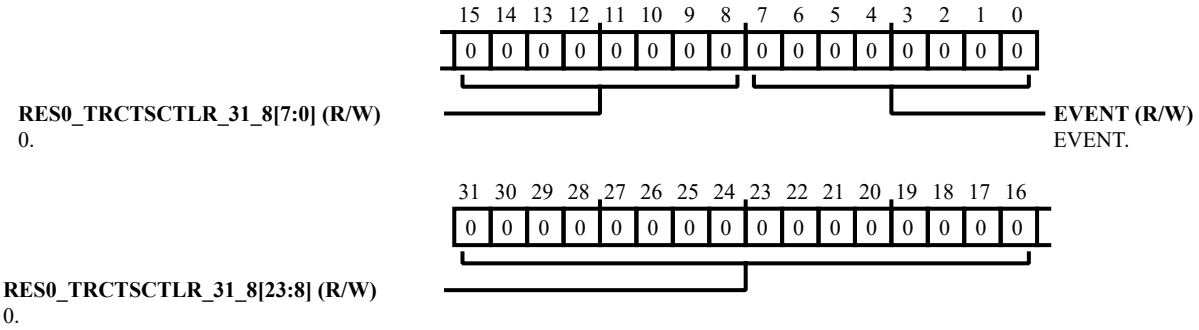


Figure 48-773: A55_C0_TRCTSCTLR Register Diagram

Table 48-780: A55_C0_TRCTSCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_TRCTSCTLR_31_8	0.. Reserved, RES0.
7:0 (R/W)	EVENT	EVENT.. An event selector. When the selected event is triggered, the trace unit inserts a global timestamp into the trace streams.

Trace ViewInst Main Control Register.

Controls instruction trace filtering.

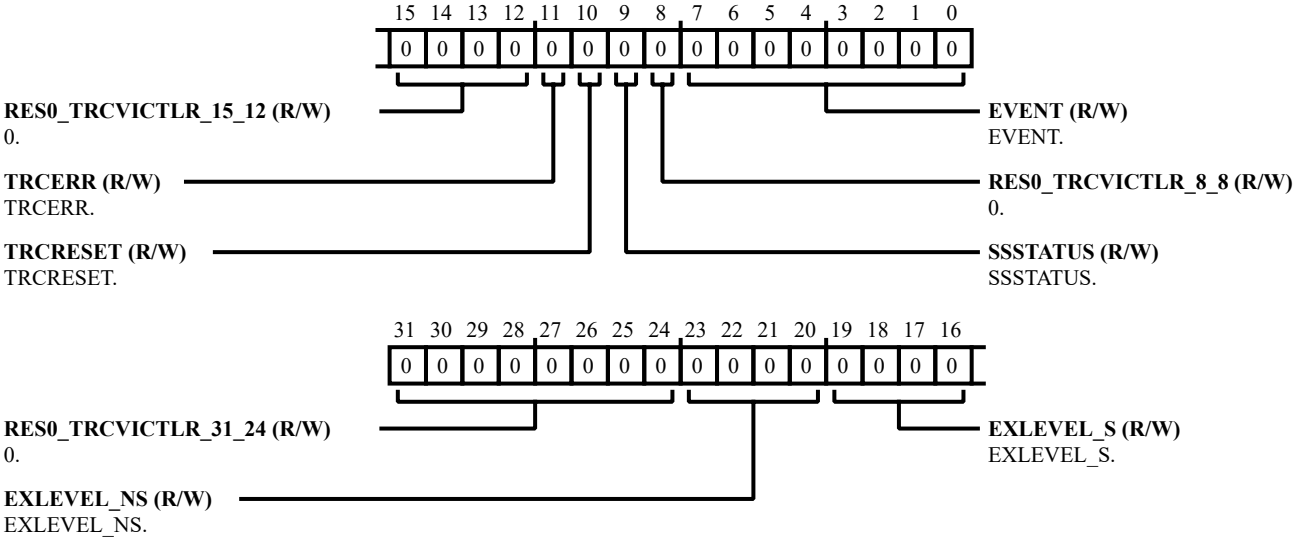


Figure 48-774: A55_C0_TRCVICTLR Register Diagram

Table 48-781: A55_C0_TRCVICTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	RES0_TRCVICTLR_31_24	0.. Reserved, RES0.	
23:20 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding exception level: 0 The trace unit generates instruction trace, in Non-secure state, for exception level n. 1 The trace unit does not generate instruction trace, in Non-secure state, for exception level n. The exception levels are:Bit[20]Exception level 0.Bit[21]Exception level 1.Bit[22]Exception level 2.Bit[23]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. If instruction tracing is not implemented for a given exception level, the corresponding bit in this field is not implemented. Unimplemented bits are RAZ/WI.	
		0	The trace unit generates instruction trace, in Non-secure state, for exception level n.
		1	The trace unit does not generate instruction trace, in Non-secure state, for exception level n.

Table 48-781: A55_C0_TRCVICTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether instruction tracing is enabled for the corresponding exception level: 0 The trace unit generates instruction trace, in Secure state, for exception level n. 1 The trace unit does not generate instruction trace, in Secure state, for exception level n. The exception levels are:Bit[16]Exception level 0.Bit[17]Exception level 1.Bit[18]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[19]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. If instruction tracing is not implemented for a given exception level, the corresponding bit in this field is not implemented. Unimplemented bits are RAZ/WI.	
		0	The trace unit generates instruction trace, in Secure state, for exception level n.
		1	The trace unit does not generate instruction trace, in Secure state, for exception level n.
15:12 (R/W)	RES0_TRCVICTLR_15_12	0.. Reserved, RES0.	
11 (R/W)	TRCERR	TRCERR.. If TRCIDR3.TRCERR==1, this bit controls whether a trace unit must trace a system error exception: 0 The trace unit does not trace a system error exception unless it traces the exception or instruction immediately prior to the system error exception. 1 The trace unit always traces a system error exception. If TRCIDR3.TRCERR==0, this bit is RES0.	
		0	The trace unit does not trace a system error exception unless it traces the exception or instruction immediately prior to the system error exception.
		1	The trace unit always traces a system error exception.
10 (R/W)	TRCRESET	TRCRESET.. Controls whether a trace unit must trace a Reset exception: 0 The trace unit does not trace a Reset exception unless it traces the exception or instruction immediately prior to the Reset exception. 1 The trace unit always traces a Reset exception.	
		0	The trace unit does not trace a Reset exception unless it traces the exception or instruction immediately prior to the Reset exception.
		1	The trace unit always traces a Reset exception.

Table 48-781: A55_C0_TRCVICTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SSSTATUS	SSSTATUS.. IF TRCIDR4.NUMACPAIRS>0 or TRCIDR.NUMPC>0, this bit returns the status of the start-stop logic: 0 The start-stop logic is in the stopped state. 1 The start-stop logic is in the started state. The bit only returns stable data when TRCSTATR.PMSTABLE==1.Before software enables the trace unit, TRCPRGCTLR.EN==1, it must write to this bit to set the initial state of the start-stop logic. If the start-stop logic is not used then set this bit to 1. ARM recommends that the value of this bit is set before each trace run begins.If TRCIDR4.NUMACPAIRS==0 and TRCIDR4.NUMPC==0, this bit is RES0, indicating that the start-stop logic is not implemented.	
		0	The start-stop logic is in the stopped state.
		1	The start-stop logic is in the started state.
8 (R/W)	RES0_TRCVICTLR_8_8	0.. Reserved, RES0.	
7:0 (R/W)	EVENT	EVENT.. An event selector. [TODO: Add the bit assignments for EVENT fields into the descriptions directly?].	

Trace ViewInst Include/Exclude Control Register.

Use this to set or read the address range comparators for the ViewInst include and exclude controls.

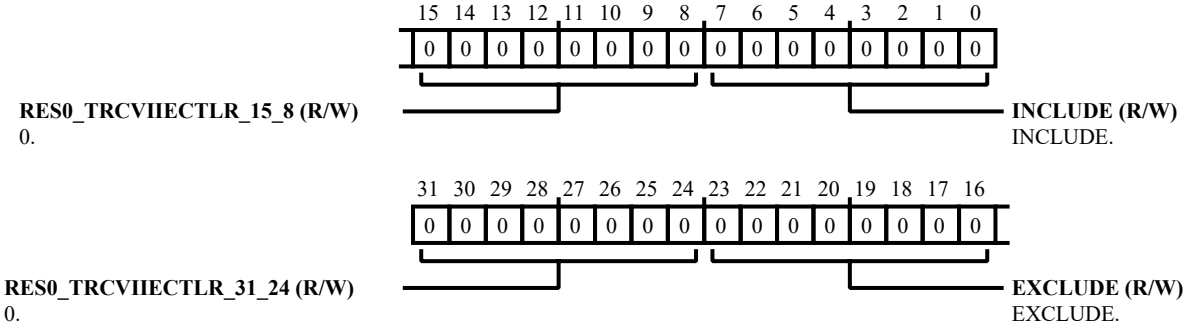


Figure 48-775: A55_C0_TRCVIIECTLR Register Diagram

Table 48-782: A55_C0_TRCVIIECTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	RES0_TRCVIIECTLR_31_24	0.. Reserved, RES0.	
23:16 (R/W)	EXCLUDE	EXCLUDE.. 0 1 The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	
		1	
15:8 (R/W)	RES0_TRCVIIECTLR_15_8	0.. Reserved, RES0.	

Table 48-782: A55_C0_TRCVIIECTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W)	INCLUDE	INCLUDE.. Include range field. Selects which address range comparator pairs are in use with ViewInst include control. Each bit represents an address range comparator pair, so bit[m] controls the selection of address range comparator pair m. If bit[m] is: 0 The address range that address range comparator pair m defines is not selected for ViewInst include control. 1 The address range that address range comparator pair m defines is selected for ViewInst include control. The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.Selecting no include comparators indicates that all instructions are included by default. The exclude control then indicates which ranges are excluded.	
		0	The address range that address range comparator pair m defines is not selected for ViewInst include control.
		1	The address range that address range comparator pair m defines is selected for ViewInst include control.

Trace ViewInst Start/Stop Control Register.

Use this to set, or read, the single address comparators that control the ViewInst start-stop logic. The start-stop logic is active for an instruction which causes a start and remains active up to and including an instruction which causes a stop, and then the start-stop logic becomes inactive.

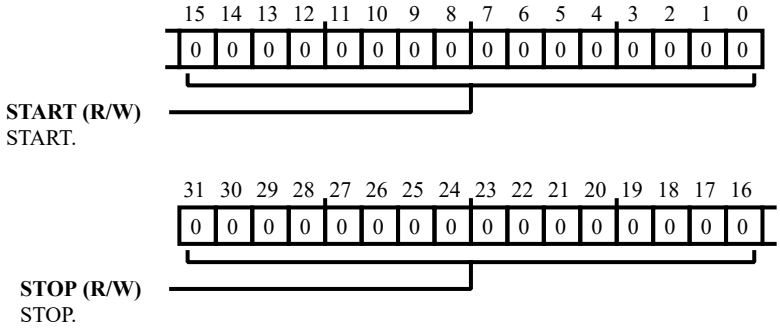


Figure 48-776: A55_C0_TRCVISSCTLR Register Diagram

Table 48-783: A55_C0_TRCVISSCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/W)	STOP	STOP.. Selects which single address comparators are in use with ViewInst start-stop control, for the purpose of stopping trace. Each bit represents a single address comparator, so bit[m] controls the selection of single address comparator m-16. If bit[m] is: 0 The single address comparator m-16 is not selected as a stop resource. 1 The single address comparator m-16 is selected as a stop resource. The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of 2 x TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The single address comparator m-16 is not selected as a stop resource.
		1	The single address comparator m-16 is selected as a stop resource.

Table 48-783: A55_C0_TRCVISSCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:0 (R/W)	START	START.. Selects which single address comparators are in use with ViewInst start-stop control, for the purpose of starting trace. Each bit represents a single address comparator, so bit[n] controls the selection of single address comparator n. If bit[n] is: 0 The single address comparator n is not selected as a start resource. 1 The single address comparator n is selected as a start resource. The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of 2 x TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The single address comparator n is not selected as a start resource.
		1	The single address comparator n is selected as a start resource.

Trace VMID Comparator Value Registers.

Contains a VMID value.

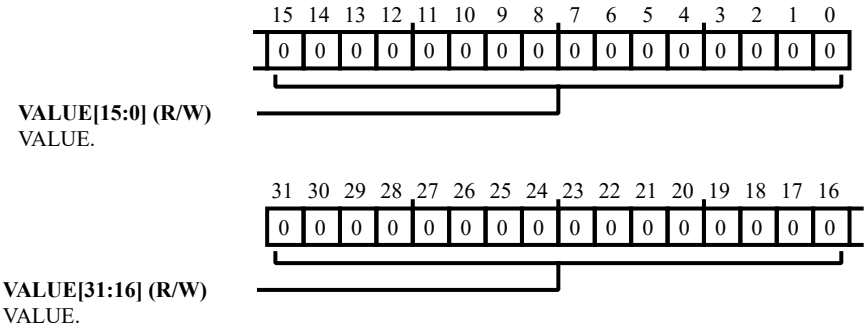


Figure 48-777: A55_C0_TRCVMIDCVR0_31_0 Register Diagram

Table 48-784: A55_C0_TRCVMIDCVR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. VMID value. The implemented width of this field is IMPLEMENTATION DEFINED, and is set by TRCIDR2.VMIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the VMID is zero until the processor updates the VMID.

Trace VMID Comparator Value Registers.

Contains a VMID value.

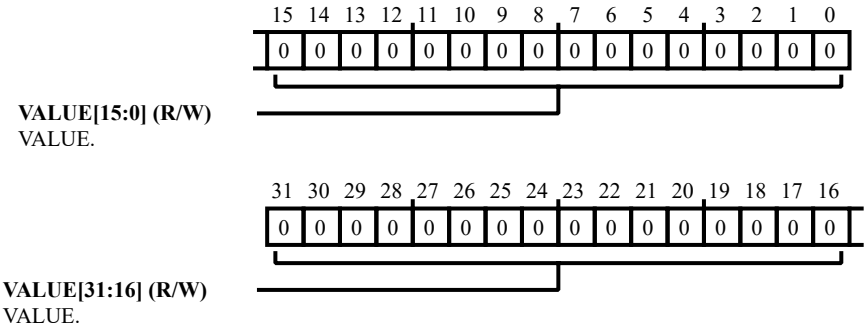


Figure 48-778: A55_C0_TRCVMIDCVR0_63_32 Register Diagram

Table 48-785: A55_C0_TRCVMIDCVR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. VMID value. The implemented width of this field is IMPLEMENTATION DEFINED, and is set by TRCIDR2.VMIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the VMID is zero until the processor updates the VMID.

Performance Monitors Cycle Counter Filter Register.

Determines the modes in which the Cycle Counter, PMCCNTR_EL0, increments.

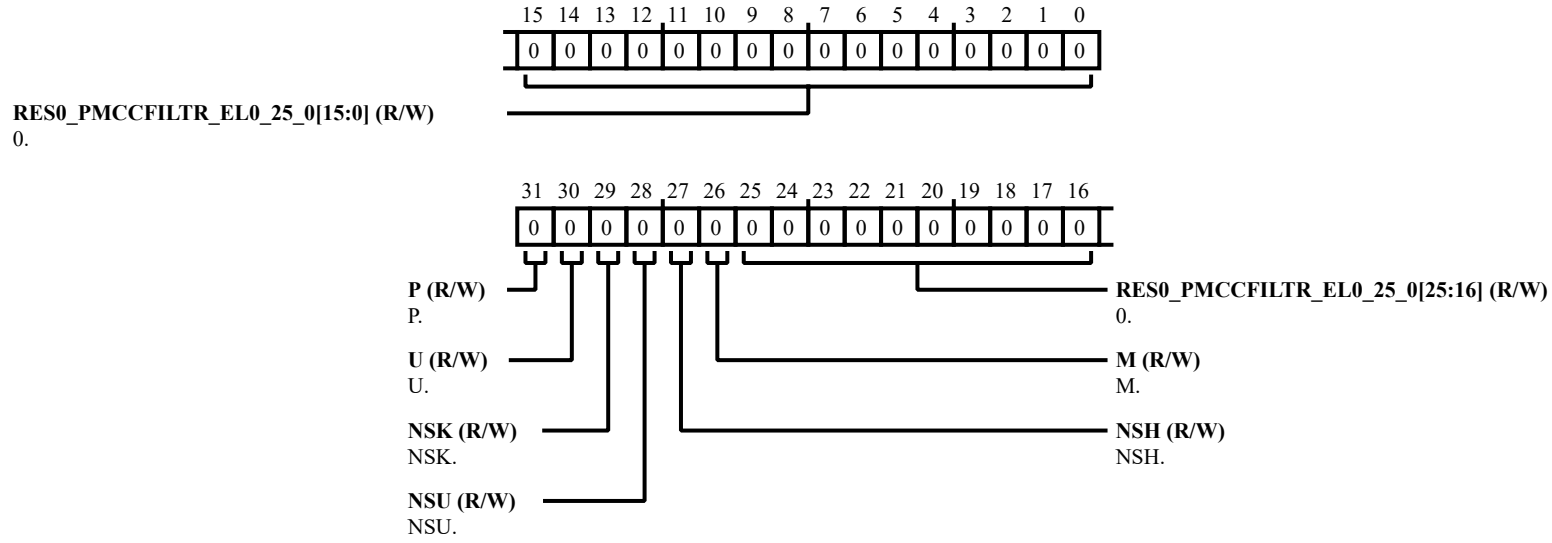


Figure 48-779: A55_C0_PMCCFILTR_EL0 Register Diagram

Table 48-786: A55_C0_PMCCFILTR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged modes filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count cycles in EL1. 1 Do not count cycles in EL1.	
		0	Count cycles in EL1.
		1	Do not count cycles in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count cycles in EL0. 1 Do not count cycles in EL0.	
		0	Count cycles in EL0.
		1	Do not count cycles in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Non-secure EL1 are counted. Otherwise, cycles in Non-secure EL1 are not counted.	

Table 48-786: A55_C0_PMCCFILTR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, cycles in Non-secure EL0 are counted. Otherwise, cycles in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering bit. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count cycles in EL2. 1 Count cycles in EL2.	
		0	Do not count cycles in EL2.
		1	Count cycles in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMCCFILTR System register.	
25:0 (R/W)	RES0_PMCCFILTR_EL0_25_0	0.. Reserved, RES0.	

Performance Monitors Cycle Counter.

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See for more information.

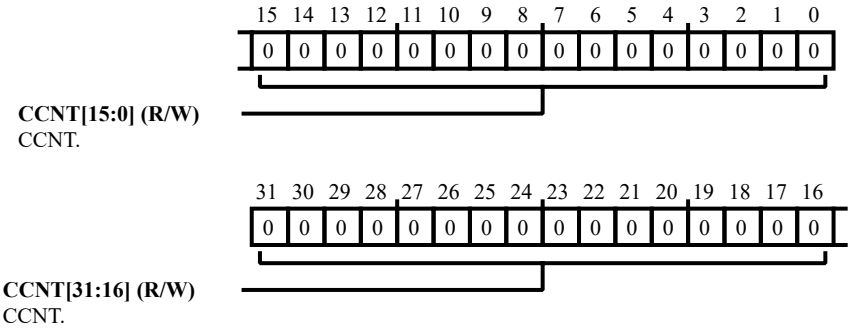


Figure 48-780: A55_C0_PMCCNTR_EL0_31_0 Register Diagram

Table 48-787: A55_C0_PMCCNTR_EL0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CCNT	CCNT.. Cycle count. Depending on the values of PMCR_EL0.{LC,D}, the cycle count increments in one of the following ways: Every processor clock cycle. Every 64th processor clock cycle. Writing 1 to PMCR_EL0.C sets this field to 0.

Performance Monitors Cycle Counter.

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See for more information.

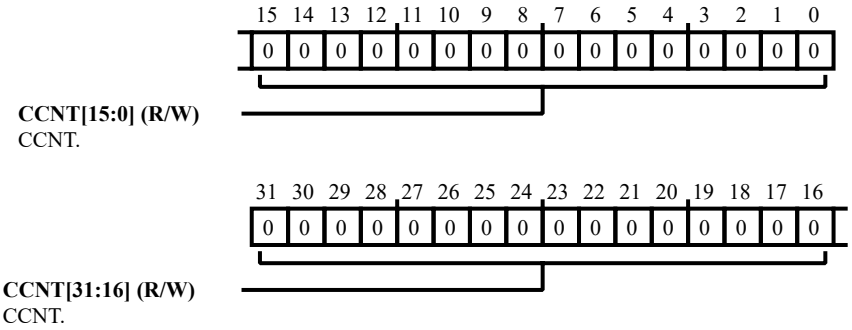


Figure 48-781: A55_C0_PMCCNTR_EL0_63_32 Register Diagram

Table 48-788: A55_C0_PMCCNTR_EL0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CCNT	CCNT.. Cycle count. Depending on the values of PMCR_EL0.{LC,D}, the cycle count increments in one of the following ways: Every processor clock cycle. Every 64th processor clock cycle. Writing 1 to PMCR_EL0.C sets this field to 0.

Performance Monitors Common Event Identification register 0.

Defines which common architectural and common microarchitectural feature events in the range 0x000 to 0x01F are implemented. If a particular bit is set to 1, then the event for that bit is implemented. This view of the register has previously been called PMCEID0_EL0.

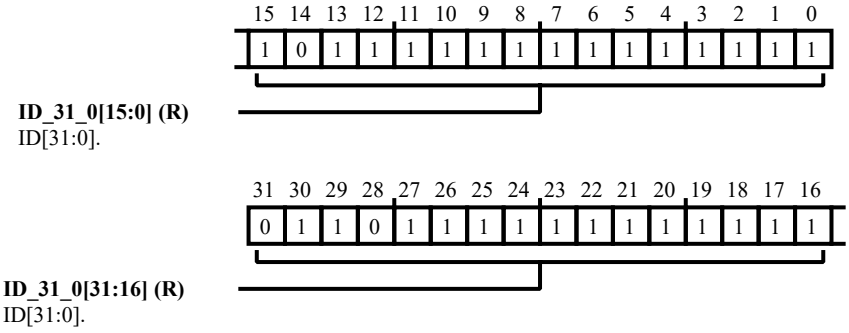


Figure 48-782: A55_C0_PMCEID0 Register Diagram

Table 48-789: A55_C0_PMCEID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_31_0	ID[31:0].. PMCEID0[n] maps to event n. For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Common Event Identification register 1.

Defines which common architectural and common microarchitectural feature events in the range 0x020 to 0x03F are implemented. If a particular bit is set to 1, then the event for that bit is implemented. This view of the register has previously been called PMCEID1_EL0.

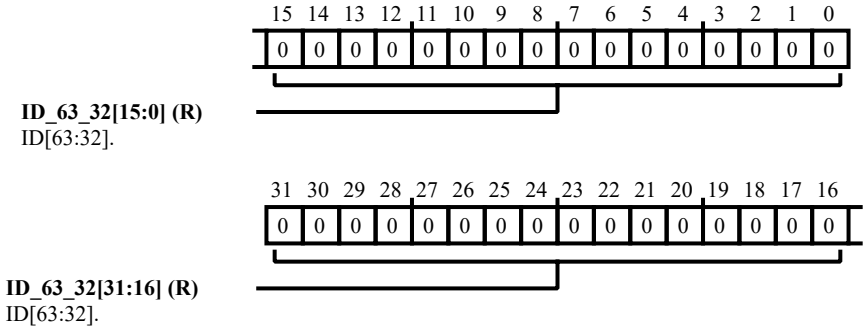


Figure 48-783: A55_C0_PMCEID1 Register Diagram

Table 48-790: A55_C0_PMCEID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_63_32	ID[63:32].. PMCEID1[n] maps to event (n + 32). For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Common Event Identification register 2.

Defines which common architectural and common microarchitectural feature events in the range 0x4000 to 0x401F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

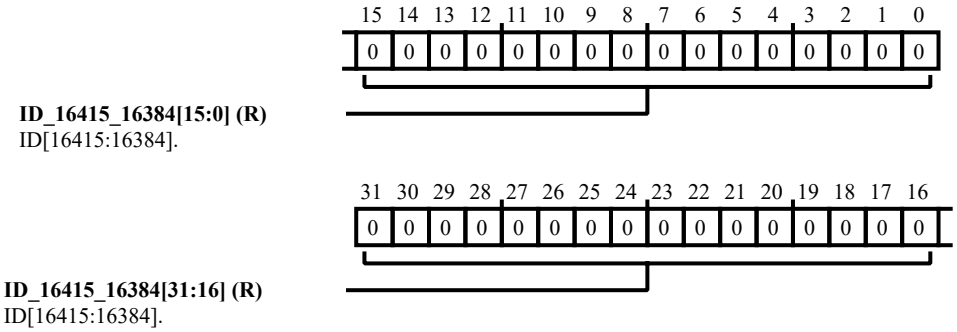


Figure 48-784: A55_C0_PMCEID2 Register Diagram

Table 48-791: A55_C0_PMCEID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_16415_16384	ID[16415:16384].. PMCEID2[31:0] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Common Event Identification register 3.

Defines which common architectural and common microarchitectural feature events in the range 0x4020 to 0x403F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

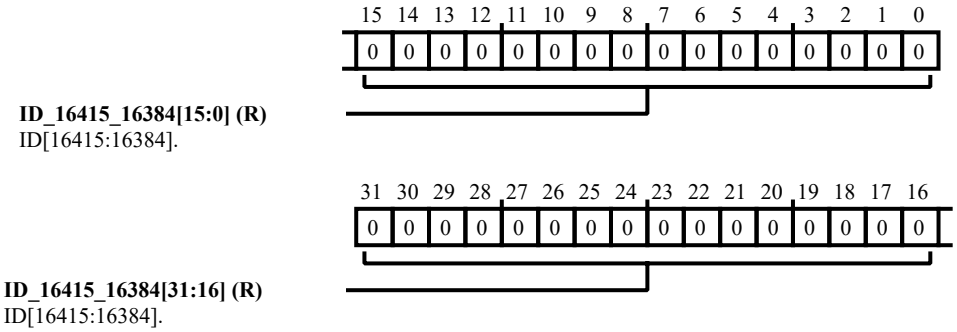


Figure 48-785: A55_C0_PMCEID3 Register Diagram

Table 48-792: A55_C0_PMCEID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_16415_16384	ID[16415:16384].. PMCEID2[31:0] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Configuration Register.

Contains PMU-specific configuration data.

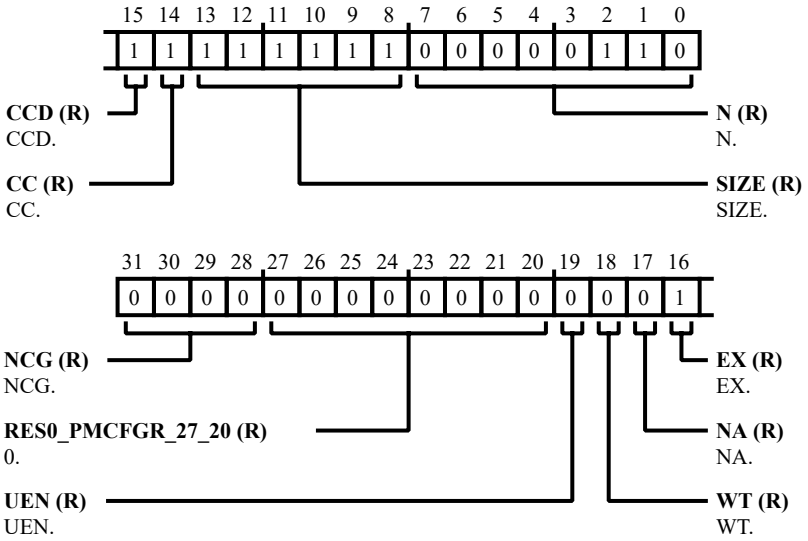


Figure 48-786: A55_C0_PMCFGR Register Diagram

Table 48-793: A55_C0_PMCFGR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	NCG	NCG.. This feature is not supported, so this field is RAZ.	
27:20 (R/NW)	RES0_PMCFGR_27_20	0.. Reserved, RES0.	
19 (R/NW)	UEN	UEN.. User-mode Enable Register supported. PMUSERENR_EL0 is not visible in the external debug interface, so this bit is RAZ.	
18 (R/NW)	WT	WT.. This feature is not supported, so this bit is RAZ.	
17 (R/NW)	NA	NA.. This feature is not supported, so this bit is RAZ.	
16 (R/NW)	EX	EX.. Export supported. Value is IMPLEMENTATION DEFINED. 0 PMCR_EL0.X is RES0. 1 PMCR_EL0.X is read/write.	
		0	PMCR_EL0.X is RES0.
		1	PMCR_EL0.X is read/write.

Table 48-793: A55_C0_PMCFGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	CCD	CCD.. Cycle counter has prescale. This is RES1 if AArch32 is supported at any EL, and RAZ otherwise. 0 PMCR_EL0.D is RES0. 1 PMCR_EL0.D is read/write.	
		0	PMCR_EL0.D is RES0.
		1	PMCR_EL0.D is read/write.
14 (R/NW)	CC	CC.. Dedicated cycle counter (counter 31) supported. This bit is RAO.	
13:8 (R/NW)	SIZE	SIZE.. Size of counters. This field determines the spacing of counters in the memory-map. In ARMv8 the counters are at doubleword-aligned addresses, and the largest counter is 64-bits, so this field is 0b111111.	
7:0 (R/NW)	N	N.. Number of counters implemented in addition to the cycle counter, PMCCNTR_EL0. The maximum number of event counters is 31. 00000000 Only PMCCNTR_EL0 implemented. 00000001 PMCCNTR_EL0 plus one event counter implemented. and so on up to 0b00011111, which indicates PMCCNTR_EL0 and 31 event counters implemented.	
		0	Only PMCCNTR_EL0 implemented.
		1	PMCCNTR_EL0 plus one event counter implemented.

Performance Monitors Count Enable Clear register.

Disables the Cycle Count Register, PMCCNTR_EL0, and any implemented event counters PMEVCNTR0. Reading this register shows which counters are enabled.

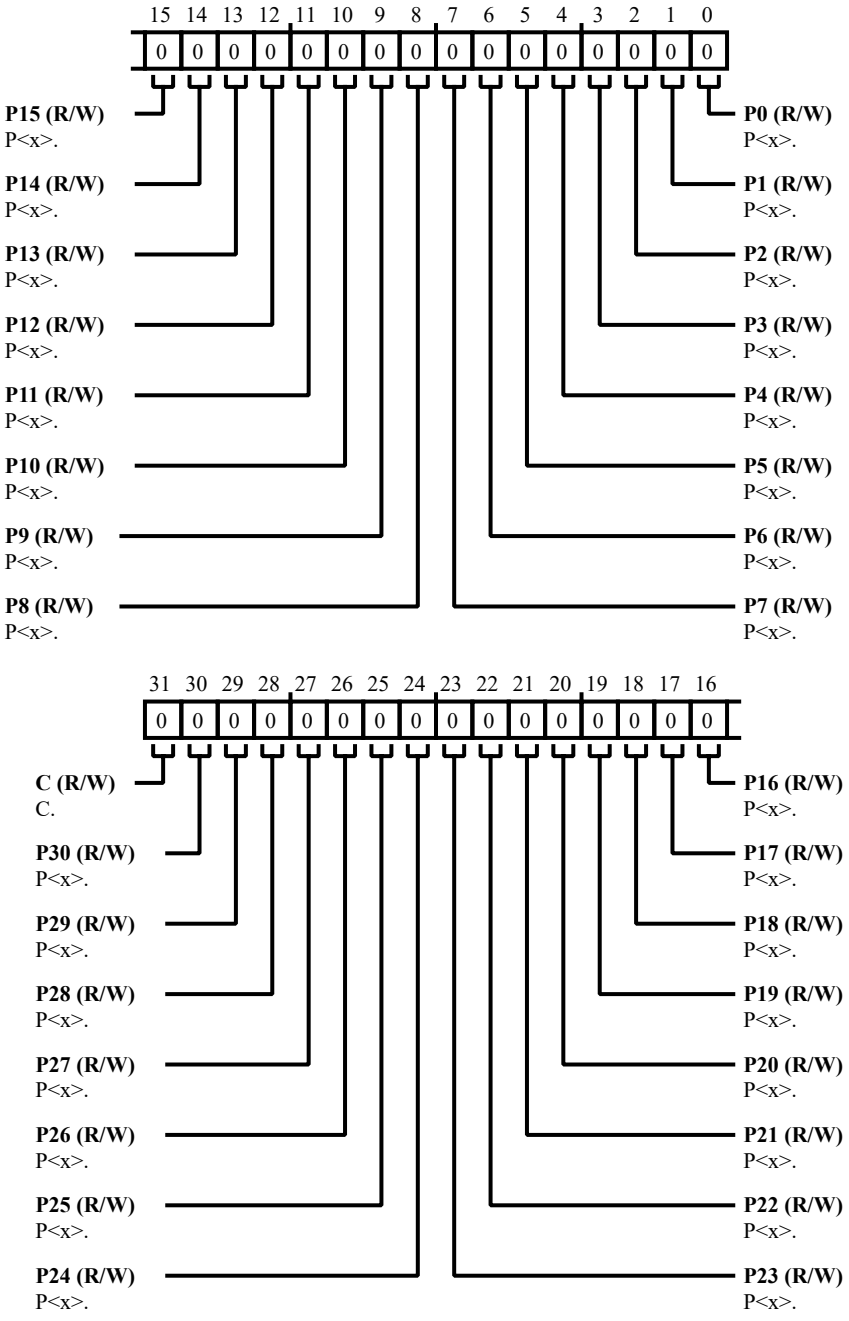


Figure 48-787: A55_C0_PMCNTENCLR_EL0 Register Diagram

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 disable bit. Disables the cycle counter register. Possible values are: 0 When read, means the cycle counter is disabled. When written, has no effect. 1 When read, means the cycle counter is enabled. When written, disables the cycle counter.	
		0	When read, means the cycle counter is disabled. When written, has no effect.
		1	When read, means the cycle counter is enabled. When written, disables the cycle counter.
30 (R/W)	P30	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
29 (R/W)	P29	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
28 (R/W)	P28	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	P27	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
26 (R/W)	P26	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
25 (R/W)	P25	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
24 (R/W)	P24	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	P23	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
22 (R/W)	P22	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
21 (R/W)	P21	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
20 (R/W)	P20	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
18 (R/W)	P18	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
17 (R/W)	P17	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
16 (R/W)	P16	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	P15	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
14 (R/W)	P14	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
13 (R/W)	P13	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
12 (R/W)	P12	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	P11	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
10 (R/W)	P10	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
9 (R/W)	P9	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
8 (R/W)	P8	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
6 (R/W)	P6	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
5 (R/W)	P5	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
4 (R/W)	P4	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-794: A55_C0_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	P3	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
2 (R/W)	P2	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
1 (R/W)	P1	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
0 (R/W)	P0	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Performance Monitors Count Enable Set register.

Enables the Cycle Count Register, PMCCNTR_EL0, and any implemented event counters PMEVCNTR0. Reading this register shows which counters are enabled.

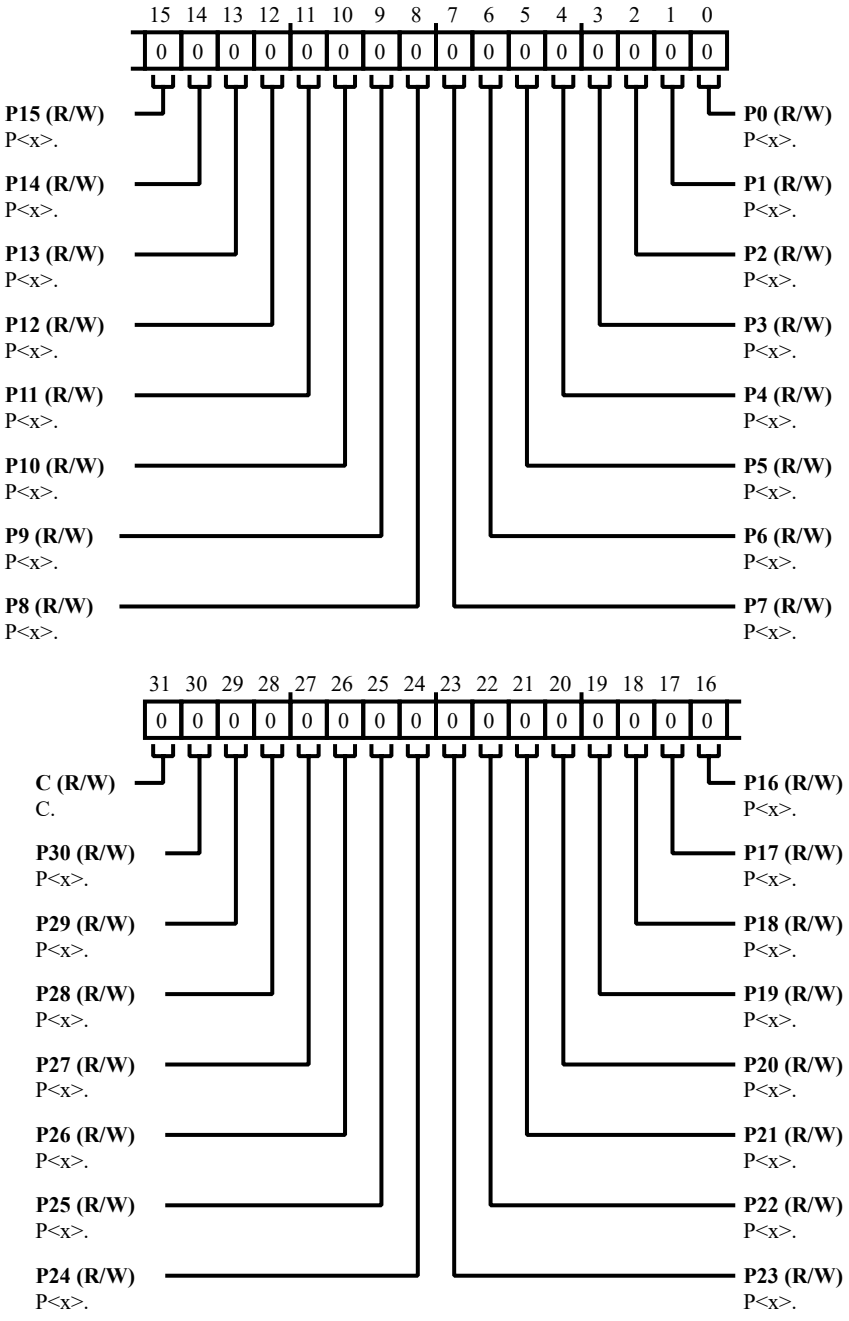


Figure 48-788: A55_C0_PMCNTENSET_EL0 Register Diagram

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 enable bit. Enables the cycle counter register. Possible values are: 0 When read, means the cycle counter is disabled. When written, has no effect. 1 When read, means the cycle counter is enabled. When written, enables the cycle counter.	
		0	When read, means the cycle counter is disabled. When written, has no effect.
		1	When read, means the cycle counter is enabled. When written, enables the cycle counter.
30 (R/W)	P30	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
29 (R/W)	P29	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
28 (R/W)	P28	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	P27	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
26 (R/W)	P26	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
25 (R/W)	P25	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
24 (R/W)	P24	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	P23	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
22 (R/W)	P22	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
21 (R/W)	P21	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
20 (R/W)	P20	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
18 (R/W)	P18	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
17 (R/W)	P17	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
16 (R/W)	P16	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	P15	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
14 (R/W)	P14	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
13 (R/W)	P13	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
12 (R/W)	P12	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	P11	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
10 (R/W)	P10	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
9 (R/W)	P9	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
8 (R/W)	P8	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
6 (R/W)	P6	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
5 (R/W)	P5	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
4 (R/W)	P4	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-795: A55_C0_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	P3	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
2 (R/W)	P2	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
1 (R/W)	P1	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
0 (R/W)	P0	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Performance Monitors Control Register.

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

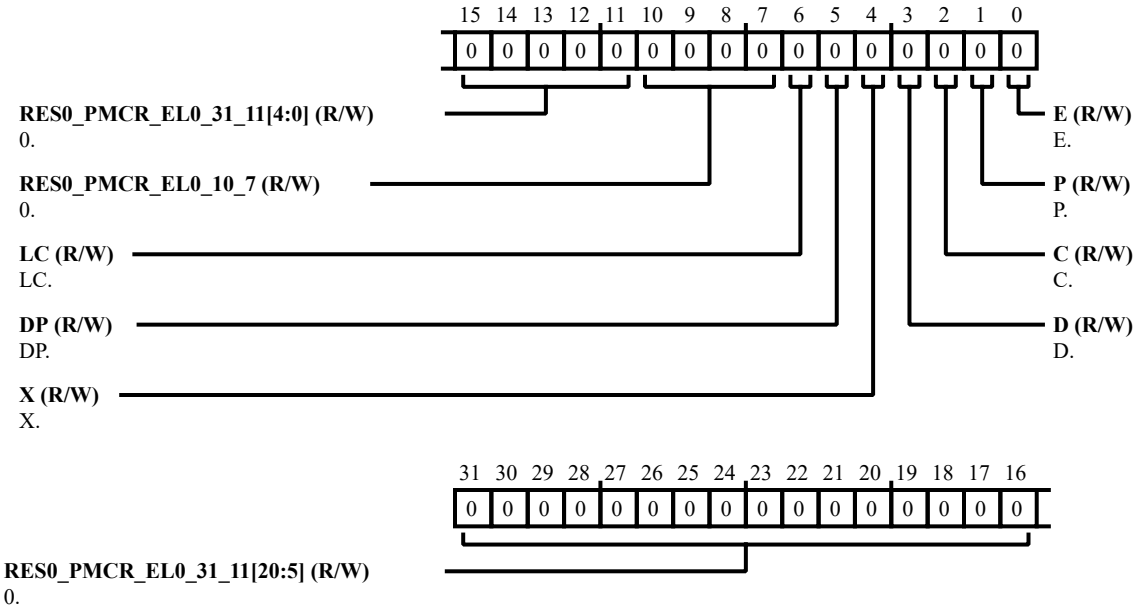


Figure 48-789: A55_C0_PMCR_EL0 Register Diagram

Table 48-796: A55_C0_PMCR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:11 (R/W)	RES0_PMCR_EL0_31_11	0.. Reserved, RAZ/WI.	
10:7 (R/W)	RES0_PMCR_EL0_10_7	0.. Reserved, RES0.	
6 (R/W)	LC	LC.. Long cycle counter enable. Determines which PMCCNTR_EL0 bit generates an overflow recorded by PMOVSR[31]. 0 Cycle counter overflow on increment that changes PMCCNTR_EL0[31] from 1 to 0. 1 Cycle counter overflow on increment that changes PMCCNTR_EL0[63] from 1 to 0. ARM deprecates use of PMCR_EL0.LC = 0. In an AArch64-only implementation, this field is RES1.	
		0	Cycle counter overflow on increment that changes PMCCNTR_EL0[31] from 1 to 0.
		1	Cycle counter overflow on increment that changes PMCCNTR_EL0[63] from 1 to 0.

Table 48-796: A55_C0_PMCR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DP	DP.. Disable cycle counter when event counting is prohibited. The possible values of this bit are: 0 PMCCNTR_EL0, if enabled, counts when event counting is prohibited. 1 PMCCNTR_EL0 does not count when event counting is prohibited. Event counting is prohibited when ProfilingProhibited(IsSecure(),PSTATE.EL) == TRUE. When EL3 is not implemented, this field is RES0: In ARMv8.0. In ARMv8.1, only if EL2 is not implemented. Otherwise this field is RW.	
		0	PMCCNTR_EL0, if enabled, counts when event counting is prohibited.
		1	PMCCNTR_EL0 does not count when event counting is prohibited.
4 (R/W)	X	X.. Enable export of events in an IMPLEMENTATION DEFINED event stream. The possible values of this bit are: 0 Do not export events. 1 Export events where not prohibited. This field enables the exporting of events over an event bus to another device, for example to an OPTIONAL trace macrocell. If the implementation does not include such an event bus then this field is RAZ/WI, otherwise it is an RW field. In an implementation that includes an event bus, no events are exported when counting is prohibited. This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.	
		0	Do not export events.
		1	Export events where not prohibited.
3 (R/W)	D	D.. Clock divider. The possible values of this bit are: 0 When enabled, PMCCNTR_EL0 counts every clock cycle. 1 When enabled, PMCCNTR_EL0 counts once every 64 clock cycles. In an AArch64-only implementation this field is RES0, otherwise it is an RW field.If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.	
		0	When enabled, PMCCNTR_EL0 counts every clock cycle.
		1	When enabled, PMCCNTR_EL0 counts once every 64 clock cycles.

Table 48-796: A55_C0_PMCR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	C	C.. Cycle counter reset. This bit is WO. The effects of writing to this bit are: 0 No action. 1 Reset PMCCNTR_EL0 to zero. This bit is always RAZ. Resetting PMCCNTR_EL0 does not clear the PMCCNTR_EL0 overflow bit to 0.	
		0	No action.
		1	Reset PMCCNTR_EL0 to zero.
1 (R/W)	P	P.. Event counter reset. This bit is WO. The effects of writing to this bit are: 0 No action. 1 Reset all event counters, not including PMCCNTR_EL0, to zero. This bit is always RAZ. Resetting the event counters does not clear any overflow bits to 0.	
		0	No action.
		1	Reset all event counters, not including PMCCNTR_EL0, to zero.
0 (R/W)	E	E.. Enable. The possible values of this bit are: 0 All counters, including PMCCNTR_EL0, are disabled. 1 All counters are enabled by PMCNTENSET_EL0. This bit is RW.	
		0	All counters, including PMCCNTR_EL0, are disabled.
		1	All counters are enabled by PMCNTENSET_EL0.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

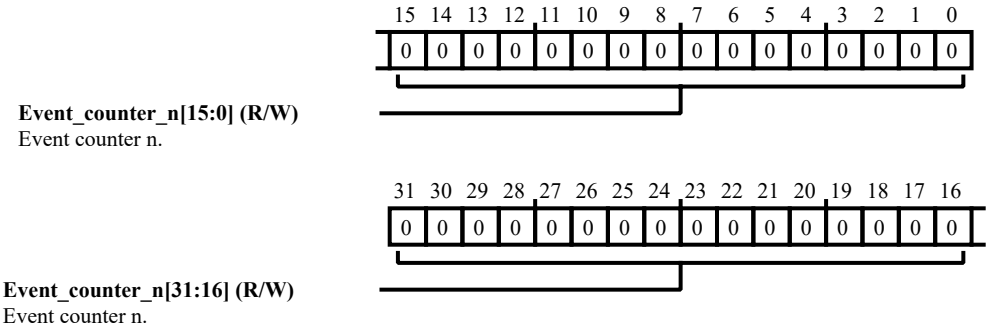


Figure 48-790: A55_C0_PMEVCNTR0_EL0 Register Diagram

Table 48-797: A55_C0_PMEVCNTR0_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

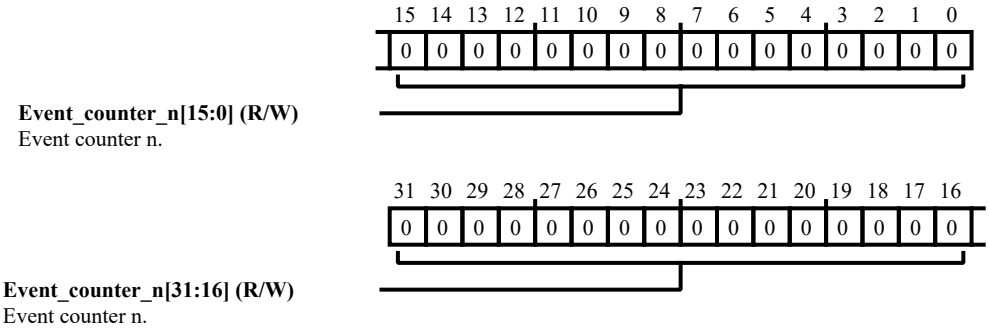


Figure 48-791: A55_C0_PMEVCNTR1_EL0 Register Diagram

Table 48-798: A55_C0_PMEVCNTR1_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

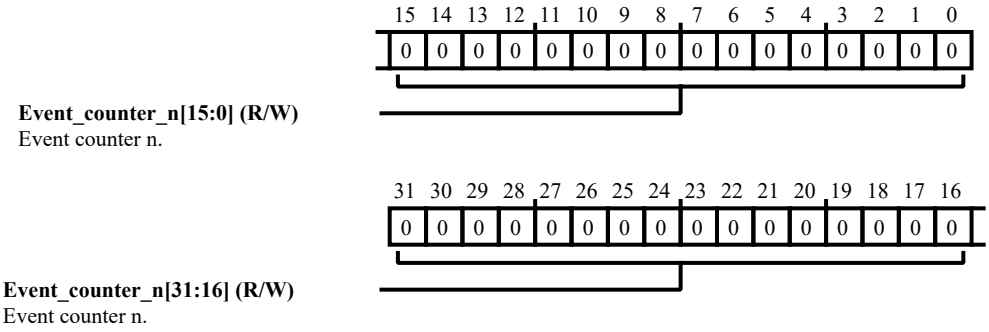


Figure 48-792: A55_C0_PMEVCNTR2_EL0 Register Diagram

Table 48-799: A55_C0_PMEVCNTR2_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

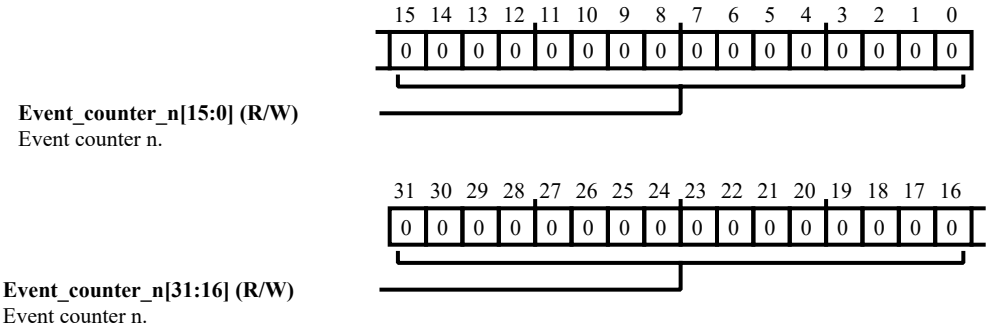


Figure 48-793: A55_C0_PMEVCNTR3_EL0 Register Diagram

Table 48-800: A55_C0_PMEVCNTR3_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

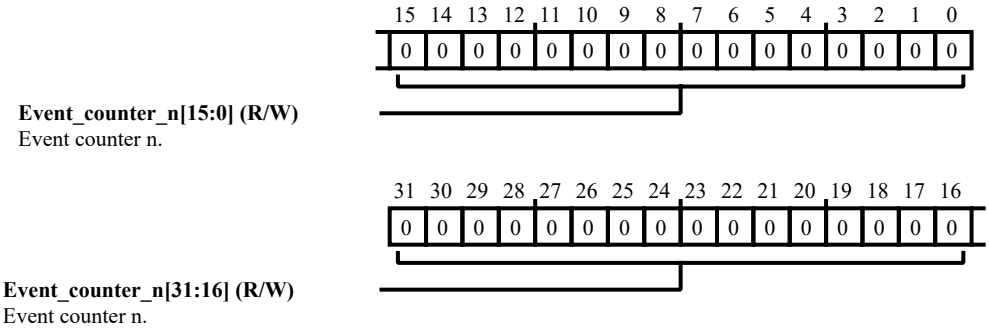


Figure 48-794: A55_C0_PMEVCNTR4_EL0 Register Diagram

Table 48-801: A55_C0_PMEVCNTR4_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

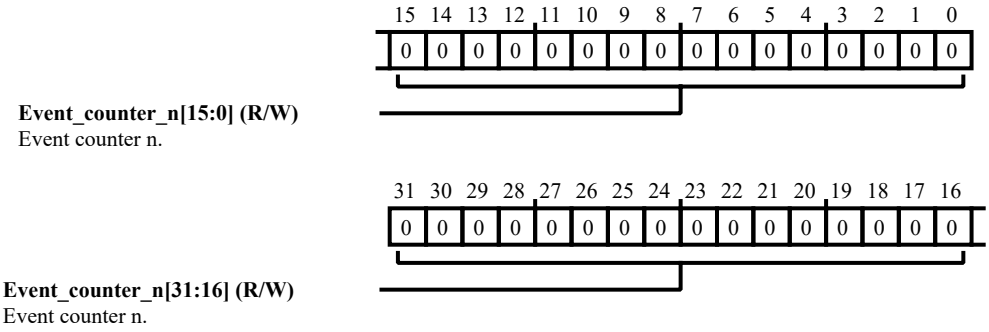


Figure 48-795: A55_C0_PMEVCNTR5_EL0 Register Diagram

Table 48-802: A55_C0_PMEVCNTR5_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

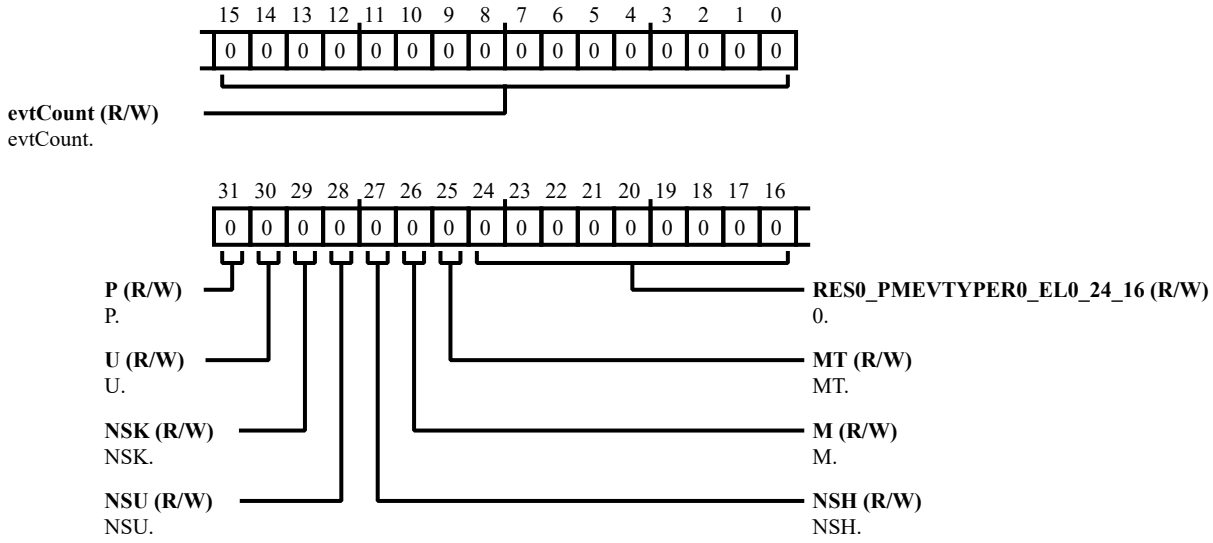


Figure 48-796: A55_C0_PMEVTYPER0_EL0 Register Diagram

Table 48-803: A55_C0_PMEVTYPER0_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-803: A55_C0_PMEVTYPER0_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER0_EL0_24_16	0.. Reserved, RES0.	

Table 48-803: A55_C0_PMEVTYPER0_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

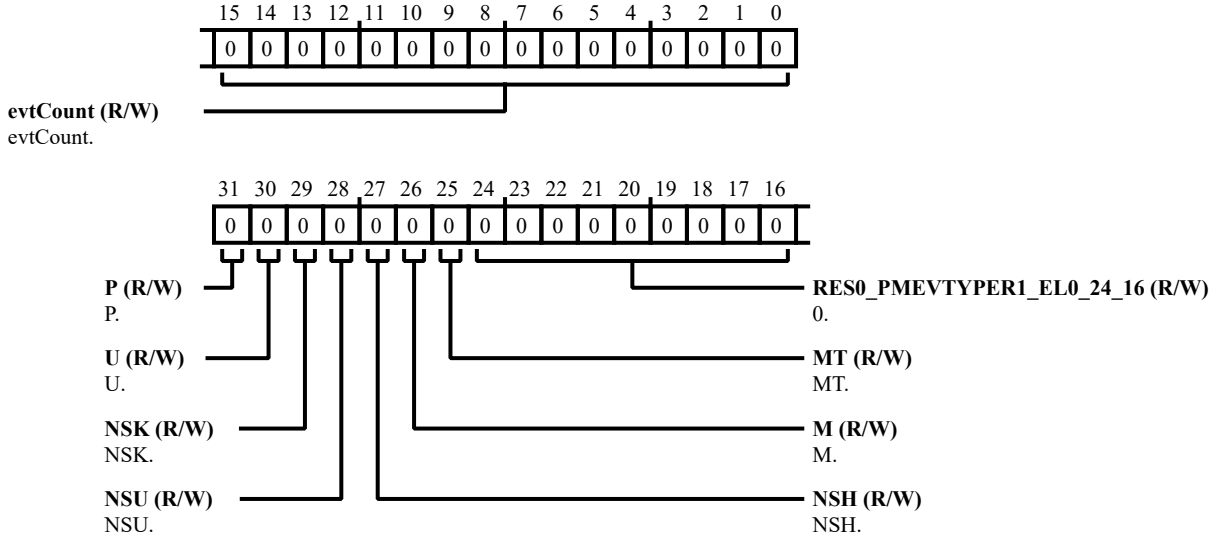


Figure 48-797: A55_C0_PMEVTYPER1_EL0 Register Diagram

Table 48-804: A55_C0_PMEVTYPER1_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-804: A55_C0_PMEVTYPER1_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER1_EL0_24_16	0.. Reserved, RES0.	

Table 48-804: A55_C0_PMEVTYPER1_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

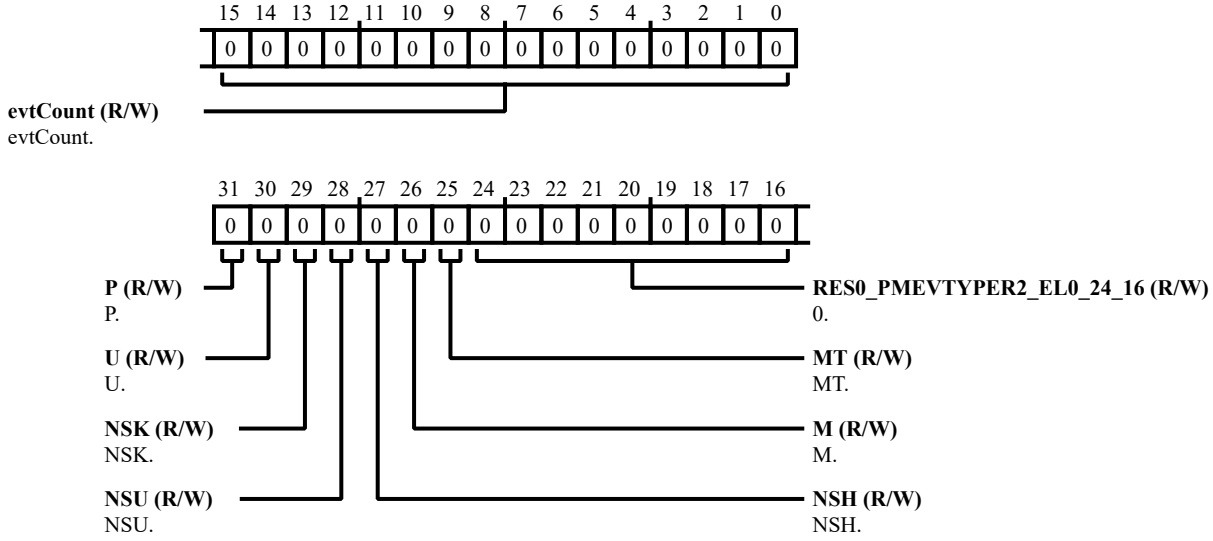


Figure 48-798: A55_C0_PMEVTYPER2_EL0 Register Diagram

Table 48-805: A55_C0_PMEVTYPER2_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-805: A55_C0_PMEVTYPER2_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER2_EL0_24_16	0.. Reserved, RES0.	

Table 48-805: A55_C0_PMEVTYPER2_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

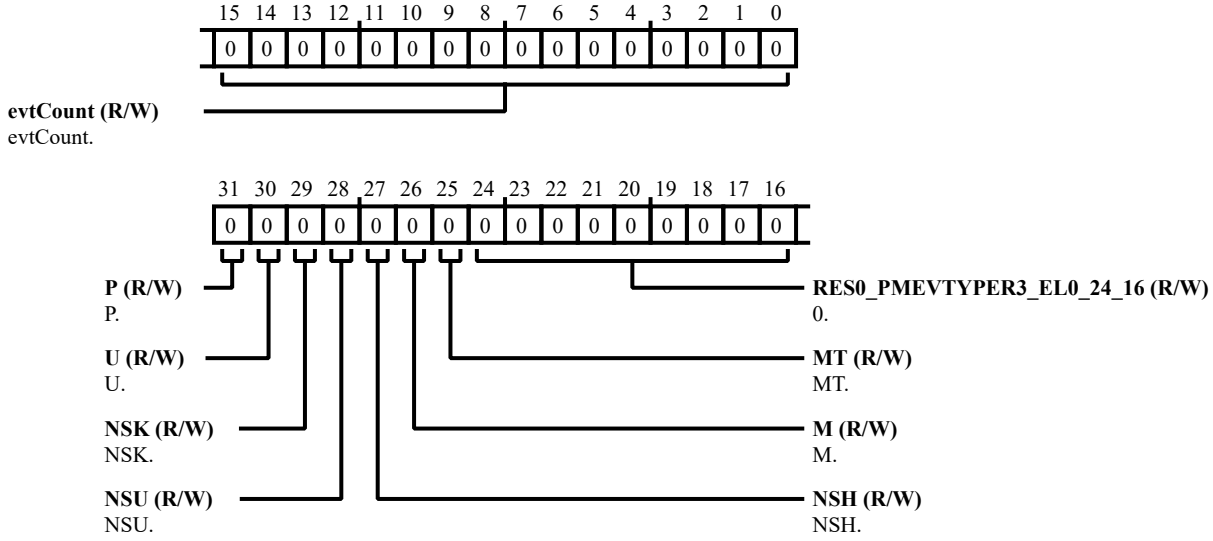


Figure 48-799: A55_C0_PMEVTYPER3_EL0 Register Diagram

Table 48-806: A55_C0_PMEVTYPER3_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-806: A55_C0_PMEVTYPER3_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER3_EL0_24_16	0.. Reserved, RES0.	

Table 48-806: A55_C0_PMEVTYPER3_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

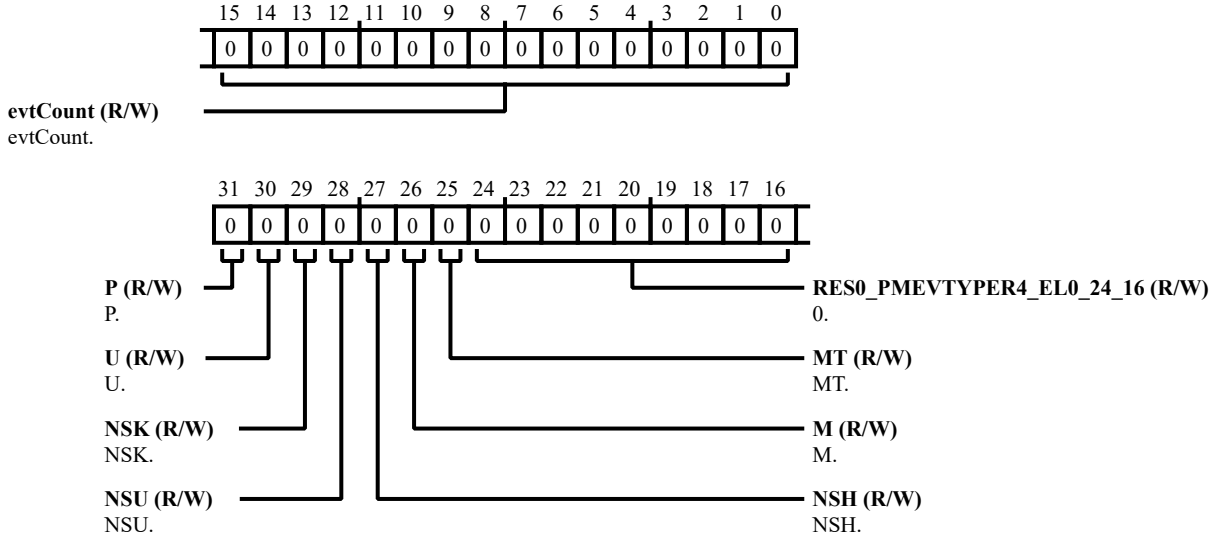


Figure 48-800: A55_C0_PMEVTYPER4_EL0 Register Diagram

Table 48-807: A55_C0_PMEVTYPER4_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-807: A55_C0_PMEVTYPER4_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER4_EL0_24_16	0.. Reserved, RES0.	

Table 48-807: A55_C0_PMEVTYPER4_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

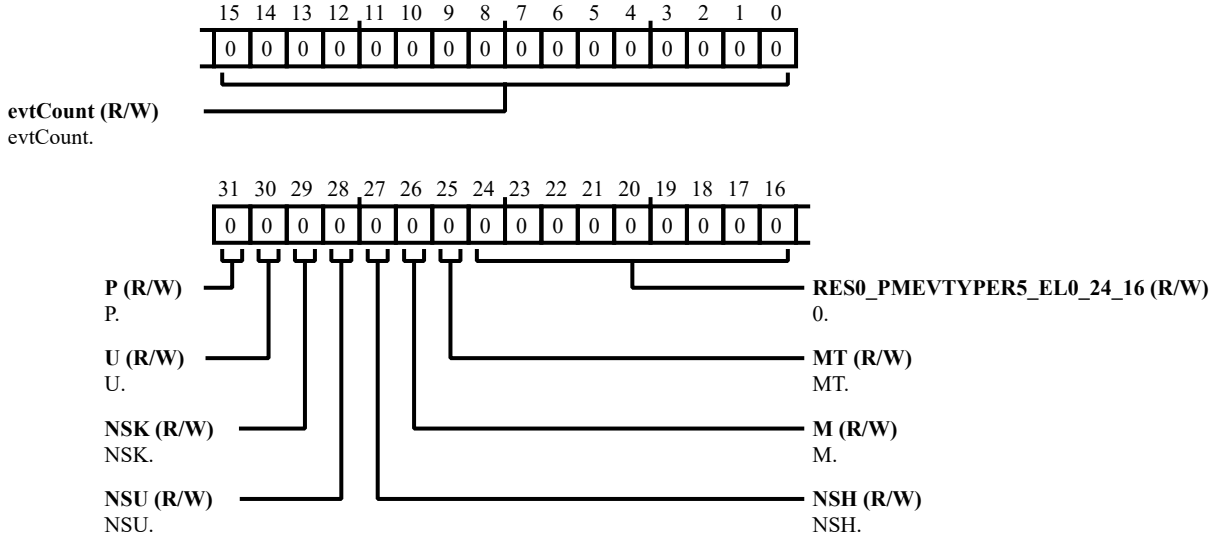


Figure 48-801: A55_C0_PMEVTYPER5_EL0 Register Diagram

Table 48-808: A55_C0_PMEVTYPER5_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-808: A55_C0_PMEVTYPER5_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER5_EL0_24_16	0.. Reserved, RES0.	

Table 48-808: A55_C0_PMEVTYPER5_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Interrupt Enable Clear register.

Disables the generation of interrupt requests on overflows from the Cycle Count Register, PMCCNTR_EL0, and the event counters PMEVCNTR0_EL0. Reading the register shows which overflow interrupt requests are enabled.

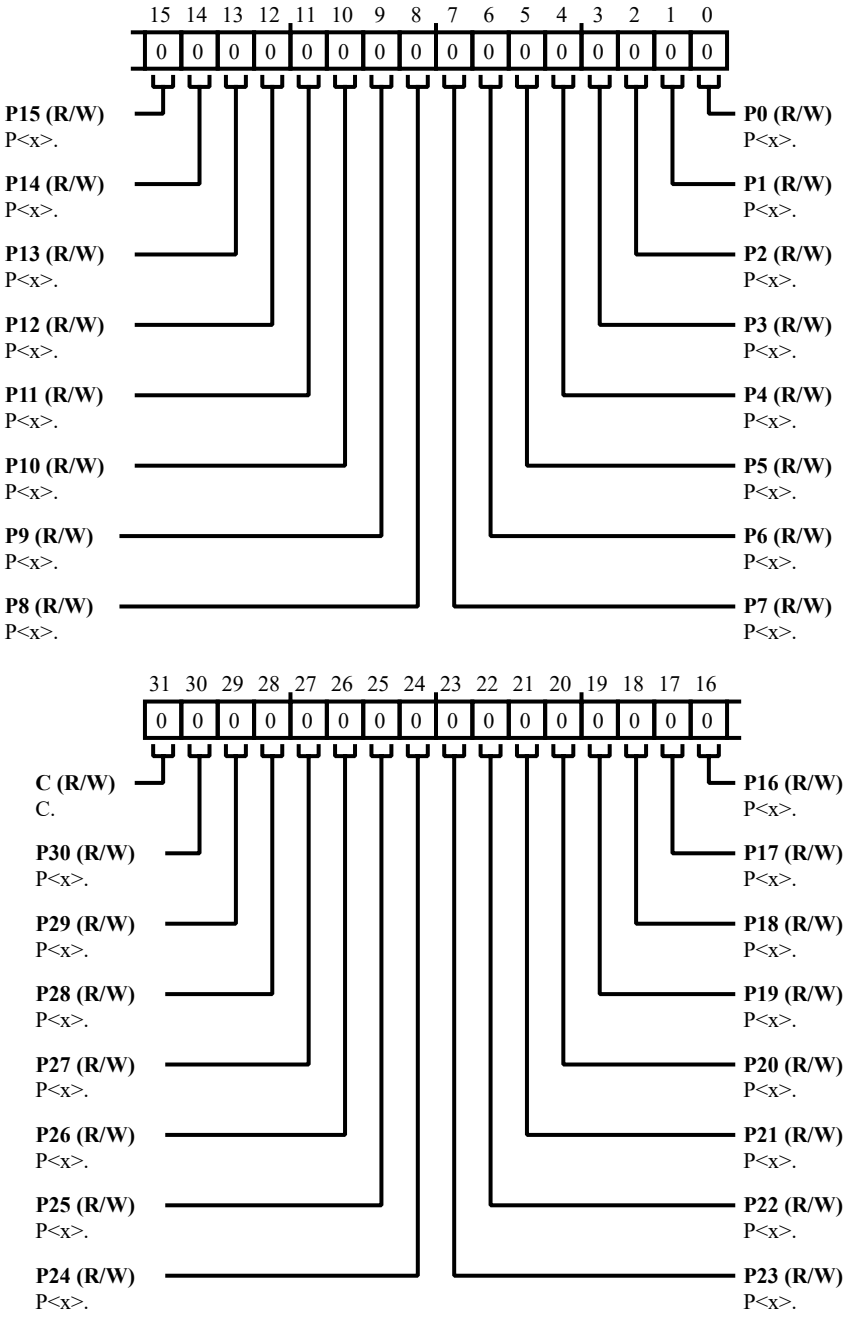


Figure 48-802: A55_C0_PMINTENCLR_EL1 Register Diagram

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow interrupt request disable bit. Possible values are: 0 When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. 1 When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.	
		0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
		1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.
30 (R/W)	P30	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
29 (R/W)	P29	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
27 (R/W)	P27	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	P26	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
25 (R/W)	P25	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	P24	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
23 (R/W)	P23	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
21 (R/W)	P21	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	P20	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
19 (R/W)	P19	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	P18	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
17 (R/W)	P17	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
15 (R/W)	P15	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	P14	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
13 (R/W)	P13	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	P12	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
11 (R/W)	P11	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
9 (R/W)	P9	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	P8	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
7 (R/W)	P7	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	P6	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
5 (R/W)	P5	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
3 (R/W)	P3	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	P2	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
1 (R/W)	P1	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-809: A55_C0_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	P0	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Performance Monitors Interrupt Enable Set register.

Enables the generation of interrupt requests on overflows from the Cycle Count Register, PMCCNTR_EL0, and the event counters PMEVCNTR0_EL0. Reading the register shows which overflow interrupt requests are enabled.

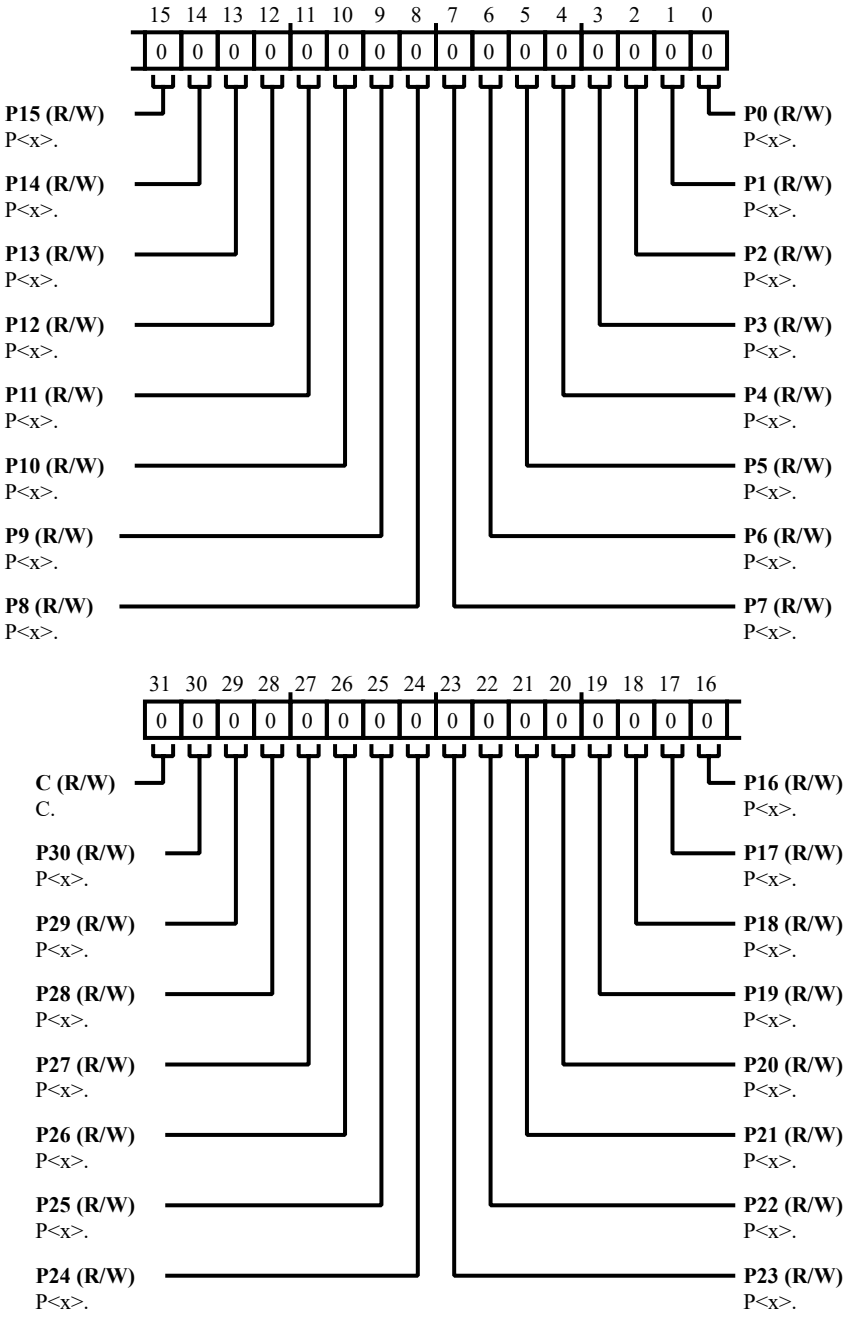


Figure 48-803: A55_C0_PMINTENSET_EL1 Register Diagram

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow interrupt request enable bit. Possible values are: 0 When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. 1 When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.	
		0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
		1	When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.
30 (R/W)	P30	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
29 (R/W)	P29	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
27 (R/W)	P27	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	P26	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
25 (R/W)	P25	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	P24	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
23 (R/W)	P23	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
21 (R/W)	P21	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	P20	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
19 (R/W)	P19	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	P18	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
17 (R/W)	P17	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
15 (R/W)	P15	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	P14	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
13 (R/W)	P13	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	P12	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
11 (R/W)	P11	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
9 (R/W)	P9	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	P8	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
7 (R/W)	P7	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	P6	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
5 (R/W)	P5	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
3 (R/W)	P3	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	P2	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
1 (R/W)	P1	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-810: A55_C0_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	P0	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Performance Monitors Integration mode Control register.

Enables the Performance Monitors to switch from default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

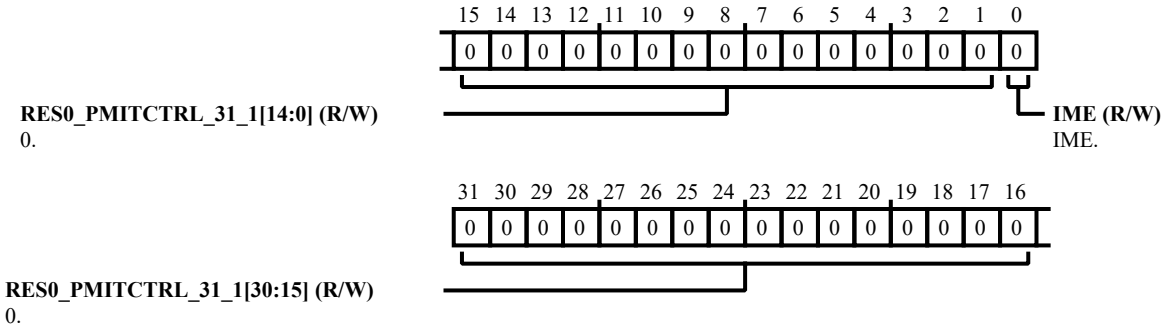


Figure 48-804: A55_C0_PMITCTRL Register Diagram

Table 48-811: A55_C0_PMITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_PMITCTRL_31_1	0.. Reserved, RES0.	
0 (R/W)	IME	IME.. Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED. 0 Normal operation. 1 Integration mode enabled.	
		0	Normal operation.
		1	Integration mode enabled.

Performance Monitors Overflow Flag Status Clear register.

Contains the state of the overflow bit for the Cycle Count Register, PMCCNTR_EL0, and each of the implemented event counters PMEVCNTR0. Writing to this register clears these bits.

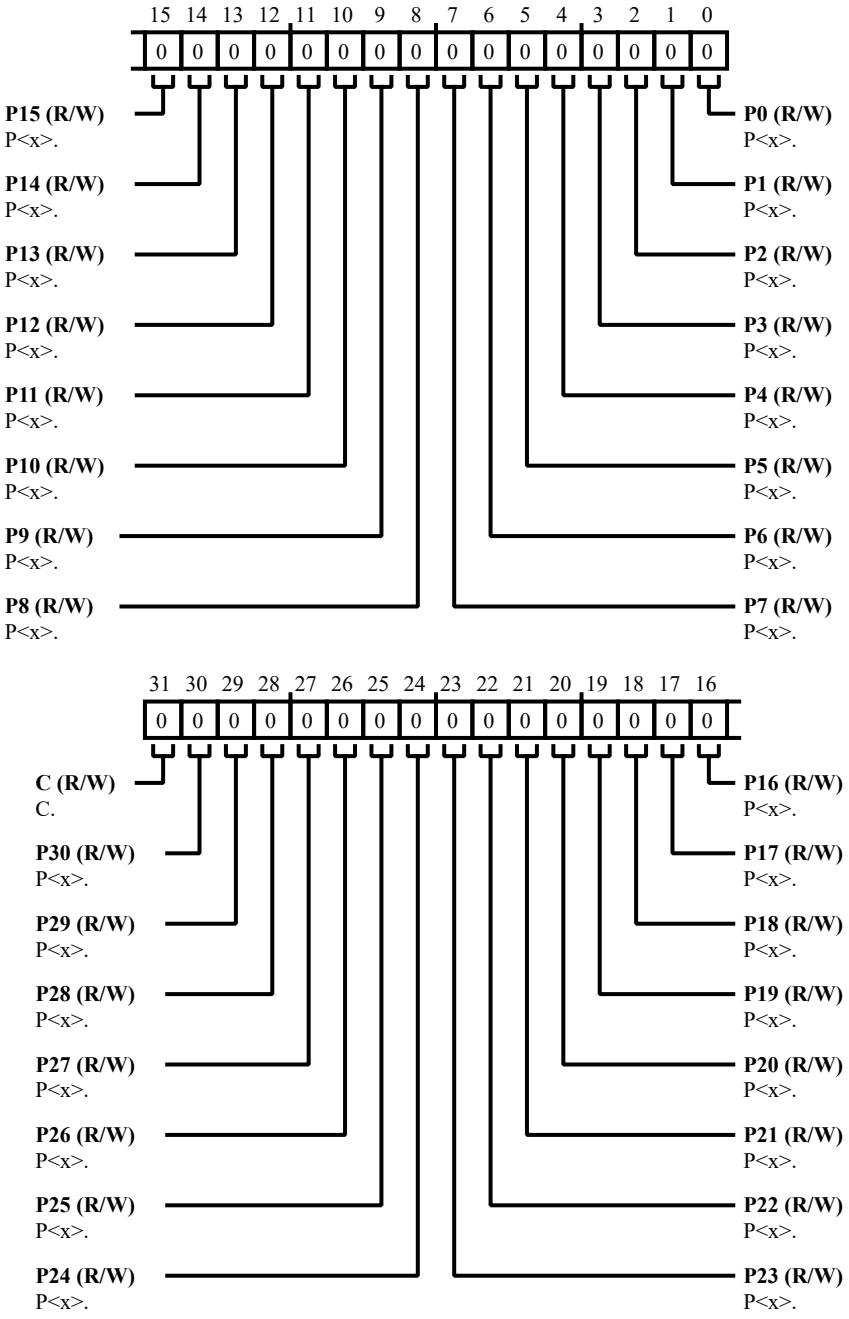


Figure 48-805: A55_C0_PMOVSCLR_EL0 Register Diagram

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow bit. Possible values are: 0 When read, means the cycle counter has not overflowed. When written, has no effect. 1 When read, means the cycle counter has overflowed. When written, clears the overflow bit to 0. PMCR_EL0.LC controls whether an overflow is detected from PMCCNTR_EL0[31] or from PMCCNTR_EL0[63].	
		0	When read, means the cycle counter has not overflowed. When written, has no effect.
		1	When read, means the cycle counter has overflowed. When written, clears the overflow bit to 0.
30 (R/W)	P30	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
29 (R/W)	P29	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
27 (R/W)	P27	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
26 (R/W)	P26	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	P25	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
24 (R/W)	P24	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
23 (R/W)	P23	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
21 (R/W)	P21	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
20 (R/W)	P20	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
18 (R/W)	P18	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
17 (R/W)	P17	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
15 (R/W)	P15	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
14 (R/W)	P14	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	P13	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
12 (R/W)	P12	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
11 (R/W)	P11	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
9 (R/W)	P9	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
8 (R/W)	P8	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
6 (R/W)	P6	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
5 (R/W)	P5	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
3 (R/W)	P3	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
2 (R/W)	P2	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-812: A55_C0_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	P1	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
0 (R/W)	P0	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Performance Monitors Overflow Flag Status Set register.

Sets the state of the overflow bit for the Cycle Count Register, PMCCNTR_EL0, and each of the implemented event counters PMEVCNTR0.

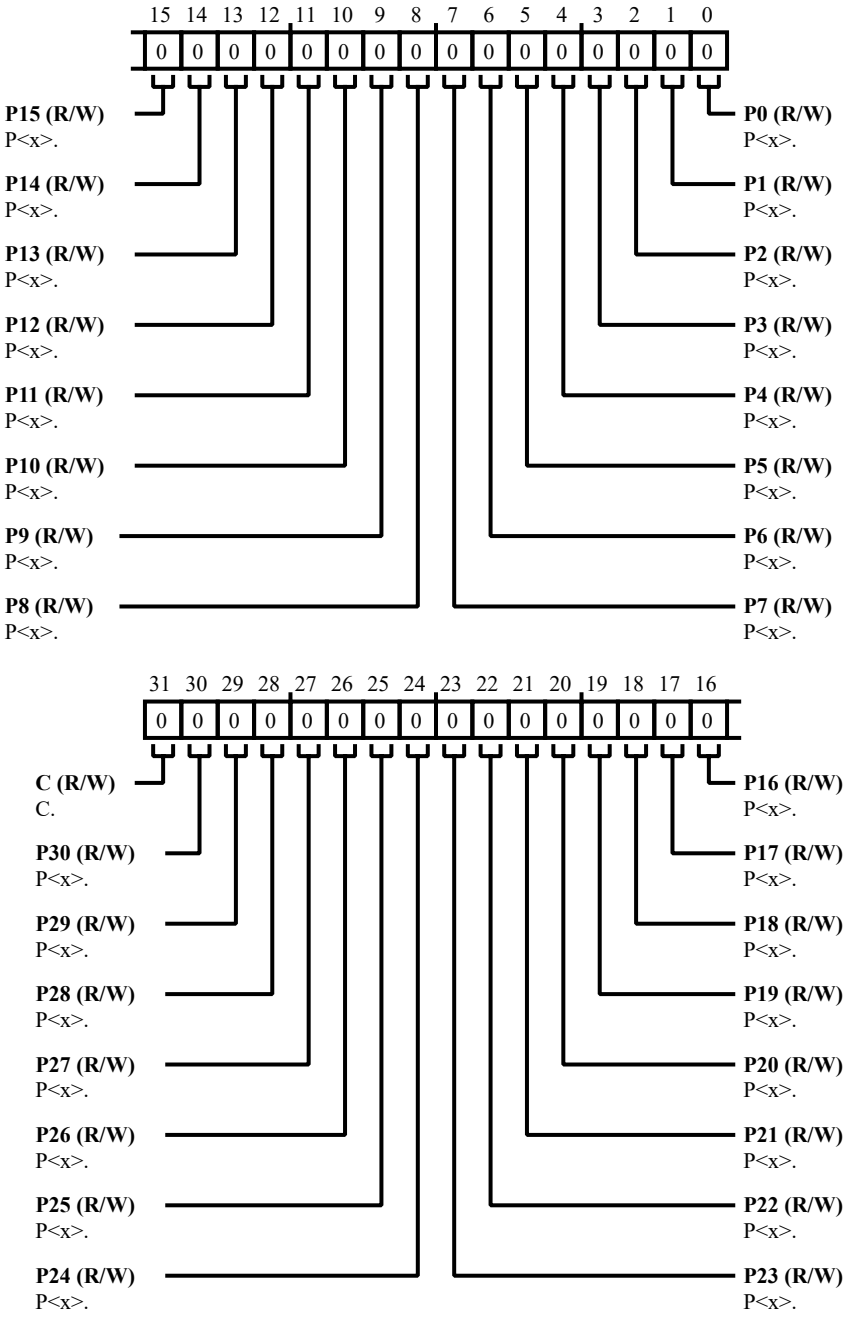


Figure 48-806: A55_C0_PMOVSSET_EL0 Register Diagram

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow bit. Possible values are: 0 When read, means the cycle counter has not overflowed. When written, has no effect. 1 When read, means the cycle counter has overflowed. When written, sets the overflow bit to 1.	
		0	When read, means the cycle counter has not overflowed. When written, has no effect.
		1	When read, means the cycle counter has overflowed. When written, sets the overflow bit to 1.
30 (R/W)	P30	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
29 (R/W)	P29	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
27 (R/W)	P27	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
26 (R/W)	P26	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	P25	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
24 (R/W)	P24	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
23 (R/W)	P23	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
21 (R/W)	P21	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
20 (R/W)	P20	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
18 (R/W)	P18	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
17 (R/W)	P17	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
15 (R/W)	P15	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
14 (R/W)	P14	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	P13	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
12 (R/W)	P12	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
11 (R/W)	P11	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
9 (R/W)	P9	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
8 (R/W)	P8	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
6 (R/W)	P6	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
5 (R/W)	P5	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
3 (R/W)	P3	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
2 (R/W)	P2	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-813: A55_C0_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	P1	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
0 (R/W)	P0	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Performance Monitors Software Increment register.

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see .

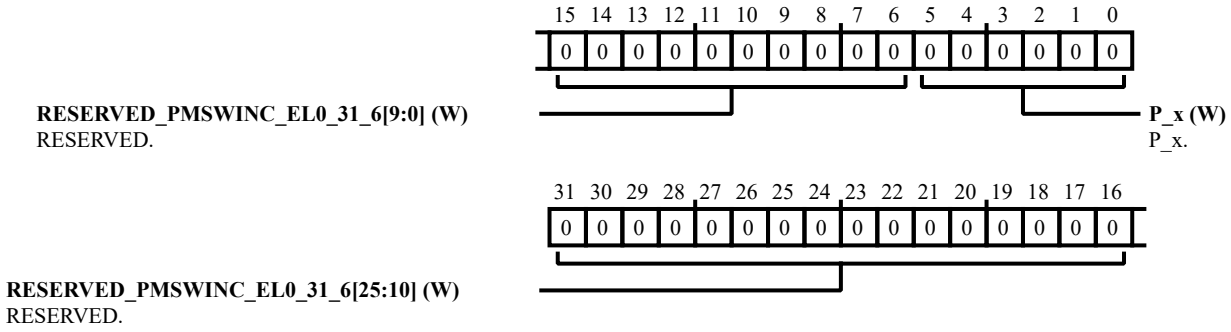


Figure 48-807: A55_C0_PMSWINC_EL0 Register Diagram

Table 48-814: A55_C0_PMSWINC_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:6 (RX/W)	RESERVED_PMSWINC_EL0_31_6	RESERVED.. Reserved RES0.	
		0	Reserved.
5:0 (RX/W)	P_X	P_x.. Event counter software increment bit for PMEVCNTR<x>.	
		0	No action. The write to this bit is ignored.
		1	It is CONSTRAINED UNPREDICTABLE whether a SW_INCR event is generated for event counter x.

ADSP-2184x A55_C2 Register Descriptions

Your module description, here. (A55_C2) contains the following registers.

Table 48-815: ADSP-2184x A55_C2 Register List

Name	Description
A55_C2_DBGBCR0_EL1	Debug Breakpoint Control Registers.
A55_C2_DBGBCR1_EL1	Debug Breakpoint Control Registers.
A55_C2_DBGBCR2_EL1	Debug Breakpoint Control Registers.
A55_C2_DBGBCR3_EL1	Debug Breakpoint Control Registers.
A55_C2_DBGBCR4_EL1	Debug Breakpoint Control Registers.

Table 48-815: ADSP-2184x A55_C2 Register List (Continued)

Name	Description
A55_C2_DBGBCR5_EL1	Debug Breakpoint Control Registers.
A55_C2_DBGCLAIMCLR_EL1	Debug Claim Tag Clear register.
A55_C2_DBGCLAIMSET_EL1	Debug Claim Tag Set register.
A55_C2_DBGDTRRX_EL0	Debug Data Transfer Register, Receive.
A55_C2_DBGDTRTX_EL0	Debug Data Transfer Register, Transmit.
A55_C2_DBGWCR0_EL1	Debug Watchpoint Control Registers.
A55_C2_DBGWCR1_EL1	Debug Watchpoint Control Registers.
A55_C2_DBGWCR2_EL1	Debug Watchpoint Control Registers.
A55_C2_DBGWCR3_EL1	Debug Watchpoint Control Registers.
A55_C2_DBGWVR0_EL1_31_0	Debug Watchpoint Value Registers.
A55_C2_DBGWVR0_EL1_63_32	Debug Watchpoint Value Registers.
A55_C2_DBGWVR1_EL1_31_0	Debug Watchpoint Value Registers.
A55_C2_DBGWVR1_EL1_63_32	Debug Watchpoint Value Registers.
A55_C2_DBGWVR2_EL1_31_0	Debug Watchpoint Value Registers.
A55_C2_DBGWVR2_EL1_63_32	Debug Watchpoint Value Registers.
A55_C2_DBGWVR3_EL1_31_0	Debug Watchpoint Value Registers.
A55_C2_DBGWVR3_EL1_63_32	Debug Watchpoint Value Registers.
A55_C2_EDCIDSR	External Debug Context ID Sample Register.
A55_C2_EDECCR	External Debug Exception Catch Control Register.
A55_C2_EDESR	External Debug Event Status Register.
A55_C2_EDITCTRL	External Debug Integration mode Control register.
A55_C2_EDITR	External Debug Instruction Transfer Register.
A55_C2_EDPRCR	External Debug Power/Reset Control Register.
A55_C2_EDPRSR	External Debug Processor Status Register.
A55_C2_EDRCR	External Debug Reserve Control Register.
A55_C2_EDSCR	External Debug Status and Control Register.
A55_C2_EDWAR_31_0	External Debug Watchpoint Address Register.
A55_C2_EDWAR_63_32	External Debug Watchpoint Address Register.
A55_C2_OSLAR_EL1	OS Lock Access Register.
A55_C2_TRCACATR0_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR0_63_32	Trace Address Comparator Access Type Registers.

Table 48-815: ADSP-2184x A55_C2 Register List (Continued)

Name	Description
A55_C2_TRCACATR1_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR1_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR2_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR2_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR3_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR3_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR4_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR4_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR5_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR5_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR6_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR6_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR7_31_0	Trace Address Comparator Access Type Registers.
A55_C2_TRCACATR7_63_32	Trace Address Comparator Access Type Registers.
A55_C2_TRCACVR0_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR0_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR1_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR1_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR2_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR2_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR3_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR3_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR4_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR4_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR5_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR5_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR6_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR6_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCACVR7_31_0	Trace Address Comparator Value Registers.
A55_C2_TRCACVR7_63_32	Trace Address Comparator Value Registers.
A55_C2_TRCAUXCTLR	Trace Auxiliary Control Register.

Table 48-815: ADSP-2184x A55_C2 Register List (Continued)

Name	Description
A55_C2_TRCBBCTLR	Trace Branch Broadcast Control Register.
A55_C2_TRCCCCTLR	Trace Cycle Count Control Register.
A55_C2_TRCCIDCCTLR0	Trace Context ID Comparator Control Register 0.
A55_C2_TRCCIDCVR0_31_0	Trace Context ID Comparator Value Registers.
A55_C2_TRCCIDCVR0_63_32	Trace Context ID Comparator Value Registers.
A55_C2_TRCCLAIMCLR	Trace Claim Tag Clear Register.
A55_C2_TRCCLAIMSET	Trace Claim Tag Set Register.
A55_C2_TRCCNTCTLR0	Trace Counter Control Registers.
A55_C2_TRCCNTCTLR1	Trace Counter Control Registers.
A55_C2_TRCCNTRLDVR0	Trace Counter Reload Value Registers.
A55_C2_TRCCNTRLDVR1	Trace Counter Reload Value Registers.
A55_C2_TRCCNTVR0	Trace Counter Value Registers.
A55_C2_TRCCNTVR1	Trace Counter Value Registers.
A55_C2_TRCCONFIGR	Trace Configuration Register.
A55_C2_TRCEVENTCTL0R	Trace Event Control 0 Register.
A55_C2_TRCEVENTCTL1R	Trace Event Control 1 Register.
A55_C2_TRCEXTINSELR	Trace External Input Select Register.
A55_C2_TRCIDR0	Trace ID Register 0.
A55_C2_TRCIDR1	Trace ID Register 1.
A55_C2_TRCIDR2	Trace ID Register 2.
A55_C2_TRCIDR3	Trace ID Register 3.
A55_C2_TRCIDR4	Trace ID Register 4.
A55_C2_TRCIDR5	Trace ID Register 5.
A55_C2_TRCIMSPEC0	Trace IMPLEMENTATION DEFINED Register 0.
A55_C2_TRCITATBIDR	Integration ATB Identification Register.
A55_C2_TRCITCTRL	Trace Integration Mode Control Register.
A55_C2_TRCITIATBINR	Integration Instruction ATB In Register.
A55_C2_TRCITIATBOUTR	Integration Instruction ATB Out Register.
A55_C2_TRCITIDATAR	Integration Instruction ATB Data Register.
A55_C2_TRCOSLAR	Trace OS Lock Access Register.
A55_C2_TRCOSLSR	Trace OS Lock Status Register.

Table 48-815: ADSP-2184x A55_C2 Register List (Continued)

Name	Description
A55_C2_TRCPDSR	Trace PowerDown Status Register.
A55_C2_TRCPRGCTLR	Trace Programming Control Register.
A55_C2_TRCRSCTLR10	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR11	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR12	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR13	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR14	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR15	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR2	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR3	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR4	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR5	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR6	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR7	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR8	Trace Resource Selection Control Registers.
A55_C2_TRCRSCTLR9	Trace Resource Selection Control Registers.
A55_C2_TRCSEQEVR0	Trace Sequencer State Transition Control Registers.
A55_C2_TRCSEQEVR1	Trace Sequencer State Transition Control Registers.
A55_C2_TRCSEQEVR2	Trace Sequencer State Transition Control Registers.
A55_C2_TRCSEQRSTEVR	Trace Sequencer Reset Control Register.
A55_C2_TRCSEQSTR	Trace Sequencer State Register.
A55_C2_TRCSSCCR0	Trace Single-Shot Comparator Control Registers.
A55_C2_TRCSSCSR0	Trace Single-Shot Comparator Status Registers.
A55_C2_TRCSTALLCTLR	Trace Stall Control Register.
A55_C2_TRCSTATR	Trace Status Register.
A55_C2_TRCSYNCPR	Trace Synchronization Period Register.
A55_C2_TRCTRACEIDR	Trace Trace ID Register.
A55_C2_TRCTSCTLR	Trace Global Timestamp Control Register.
A55_C2_TRCVICTLR	Trace ViewInst Main Control Register.
A55_C2_TRCVIIECTLR	Trace ViewInst Include/Exclude Control Register.
A55_C2_TRCVISSCTLR	Trace ViewInst Start/Stop Control Register.

Table 48-815: ADSP-2184x A55_C2 Register List (Continued)

Name	Description
A55_C2_TRCVMIDCVR0_31_0	Trace VMID Comparator Value Registers.
A55_C2_TRCVMIDCVR0_63_32	Trace VMID Comparator Value Registers.
A55_C2_PMCCFILTR_EL0	Performance Monitors Cycle Counter Filter Register.
A55_C2_PMCCNTR_EL0_31_0	Performance Monitors Cycle Counter.
A55_C2_PMCCNTR_EL0_63_32	Performance Monitors Cycle Counter.
A55_C2_PMCEID0	Performance Monitors Common Event Identification register 0.
A55_C2_PMCEID1	Performance Monitors Common Event Identification register 1.
A55_C2_PMCEID2	Performance Monitors Common Event Identification register 2.
A55_C2_PMCEID3	Performance Monitors Common Event Identification register 3.
A55_C2_PMCFGR	Performance Monitors Configuration Register.
A55_C2_PMCNTENCLR_EL0	Performance Monitors Count Enable Clear register.
A55_C2_PMCNTENSET_EL0	Performance Monitors Count Enable Set register.
A55_C2_PMCR_EL0	Performance Monitors Control Register.
A55_C2_PMEVCNTR0_EL0	Performance Monitors Event Count Registers.
A55_C2_PMEVCNTR1_EL0	Performance Monitors Event Count Registers.
A55_C2_PMEVCNTR2_EL0	Performance Monitors Event Count Registers.
A55_C2_PMEVCNTR3_EL0	Performance Monitors Event Count Registers.
A55_C2_PMEVCNTR4_EL0	Performance Monitors Event Count Registers.
A55_C2_PMEVCNTR5_EL0	Performance Monitors Event Count Registers.
A55_C2_PMEVTYPER0_EL0	Performance Monitors Event Type Registers.
A55_C2_PMEVTYPER1_EL0	Performance Monitors Event Type Registers.
A55_C2_PMEVTYPER2_EL0	Performance Monitors Event Type Registers.
A55_C2_PMEVTYPER3_EL0	Performance Monitors Event Type Registers.
A55_C2_PMEVTYPER4_EL0	Performance Monitors Event Type Registers.
A55_C2_PMEVTYPER5_EL0	Performance Monitors Event Type Registers.
A55_C2_PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear register.
A55_C2_PMINTENSET_EL1	Performance Monitors Interrupt Enable Set register.
A55_C2_PMITCTRL	Performance Monitors Integration mode Control register.
A55_C2_PMOVSCLR_EL0	Performance Monitors Overflow Flag Status Clear register.
A55_C2_PMOVSSET_EL0	Performance Monitors Overflow Flag Status Set register.
A55_C2_PMSWINC_EL0	Performance Monitors Software Increment register.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR0_EL1.

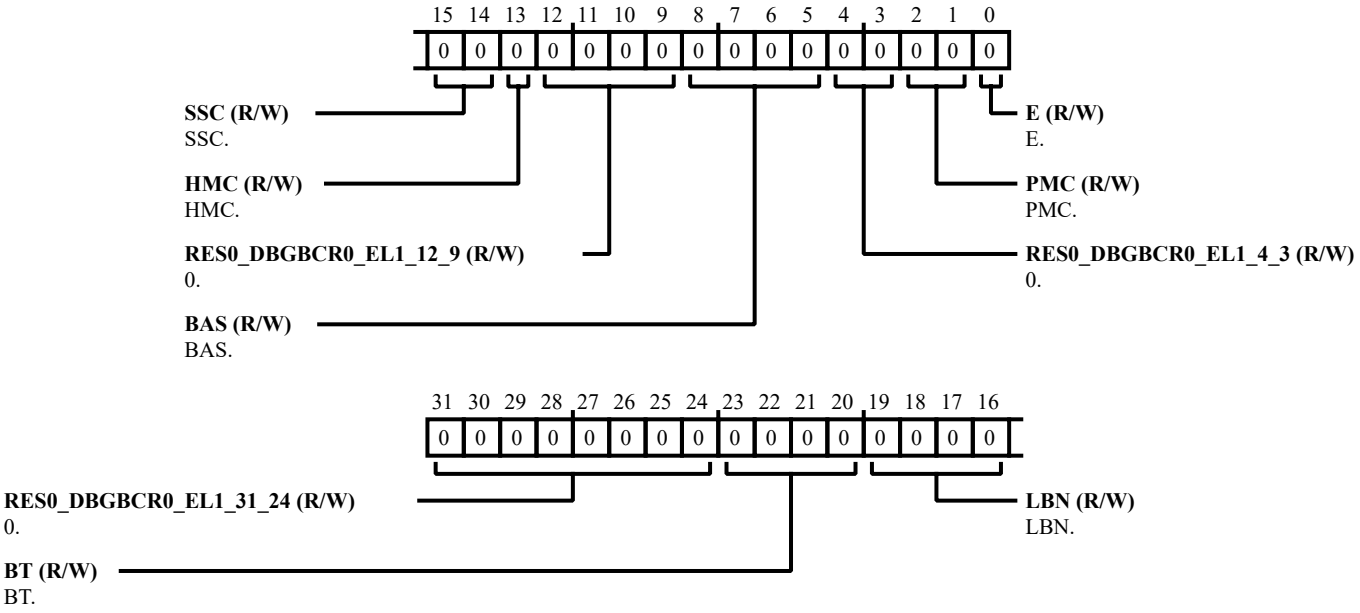


Figure 48-808: A55_C2_DBGBCR0_EL1 Register Diagram

Table 48-816: A55_C2_DBGBCR0_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR0_EL1_31_24	0.. Reserved, RES0.

Table 48-816: A55_C2_DBGBCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-816: A55_C2_DBGBCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR0_EL1_12_9	0.. Reserved, RES0.	

Table 48-816: A55_C2_DBGBCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR0_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR1_EL1.

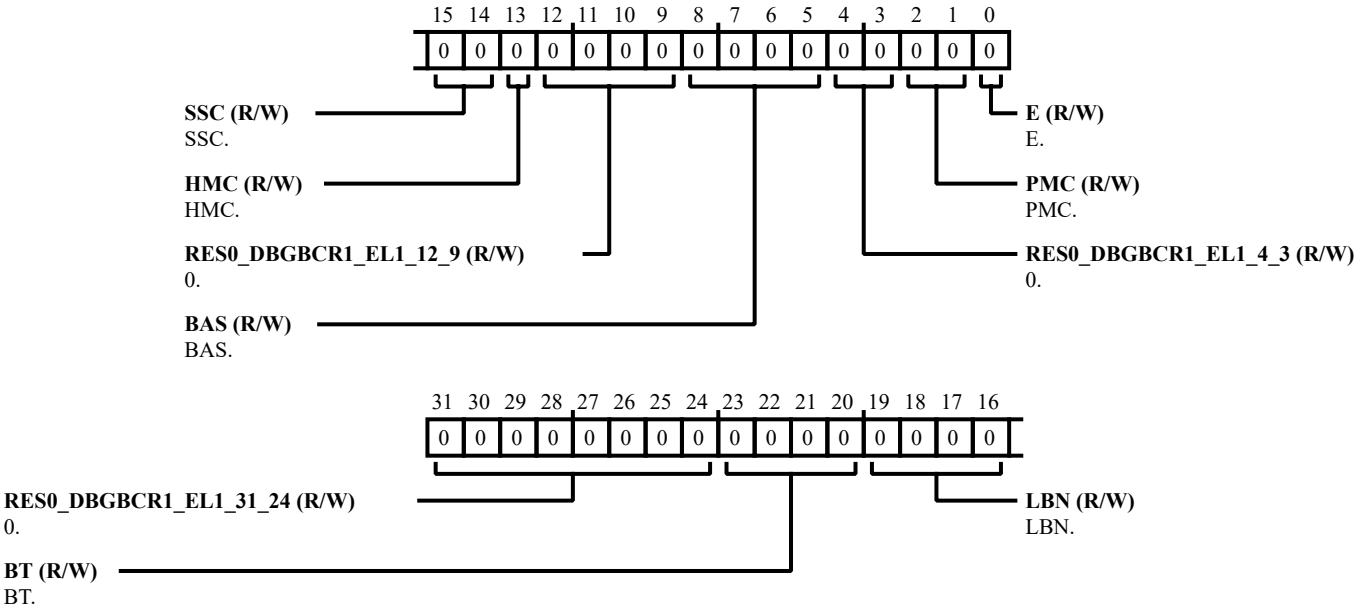


Figure 48-809: A55_C2_DBGBCR1_EL1 Register Diagram

Table 48-817: A55_C2_DBGBCR1_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR1_EL1_31_24	0.. Reserved, RES0.

Table 48-817: A55_C2_DBGBCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-817: A55_C2_DBGBCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR1_EL1_12_9	0.. Reserved, RES0.	

Table 48-817: A55_C2_DBGBCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR1_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR2_EL1.

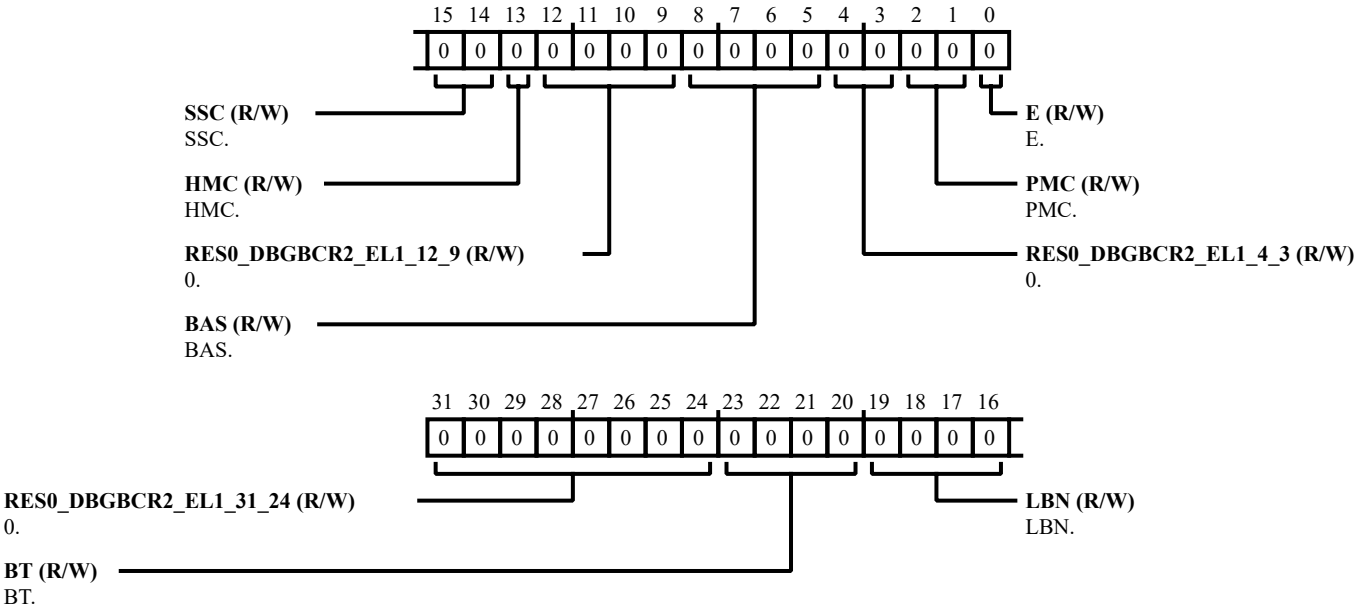


Figure 48-810: A55_C2_DBGBCR2_EL1 Register Diagram

Table 48-818: A55_C2_DBGBCR2_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR2_EL1_31_24	0.. Reserved, RES0.

Table 48-818: A55_C2_DBGBCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-818: A55_C2_DBGBCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR2_EL1_12_9	0.. Reserved, RES0.	

Table 48-818: A55_C2_DBGBCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR2_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR3_EL1.

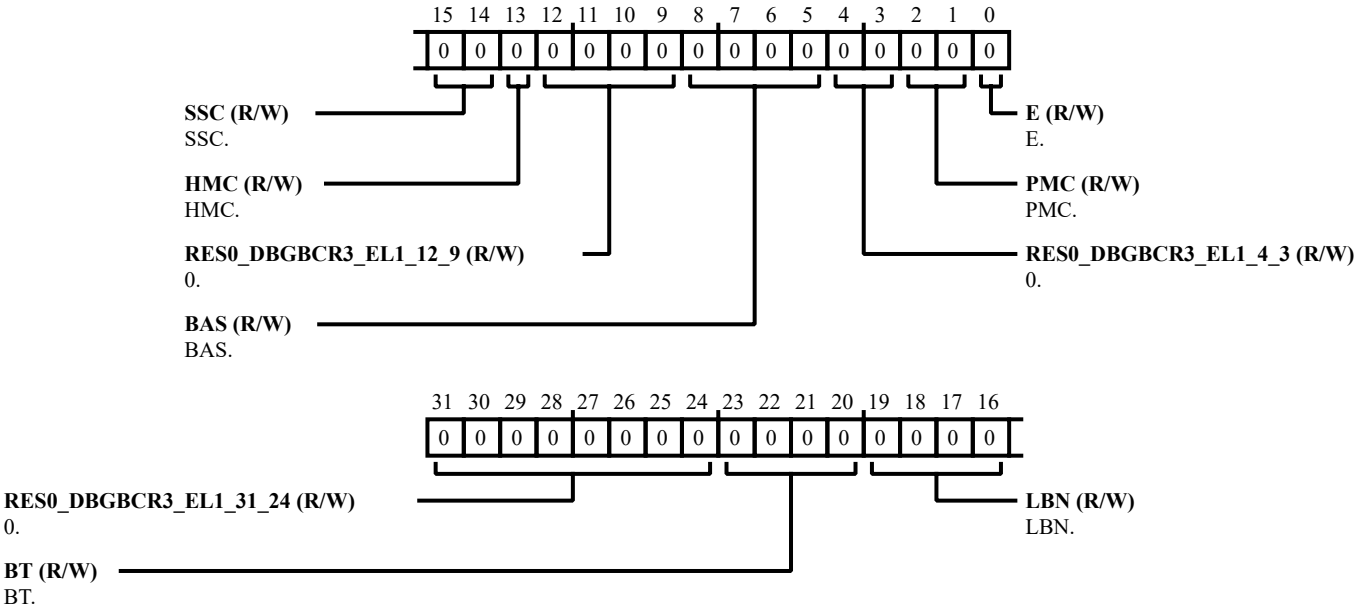


Figure 48-811: A55_C2_DBGBCR3_EL1 Register Diagram

Table 48-819: A55_C2_DBGBCR3_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR3_EL1_31_24	0.. Reserved, RES0.

Table 48-819: A55_C2_DBGBCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-819: A55_C2_DBGBCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR3_EL1_12_9	0.. Reserved, RES0.	

Table 48-819: A55_C2_DBGBCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.	
4:3 (R/W)	RES0_DBGBCR3_EL1_4_3	0.. Reserved, RES0.	
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.	
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled.	
		0	Breakpoint disabled.
		1	Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR4_EL1.

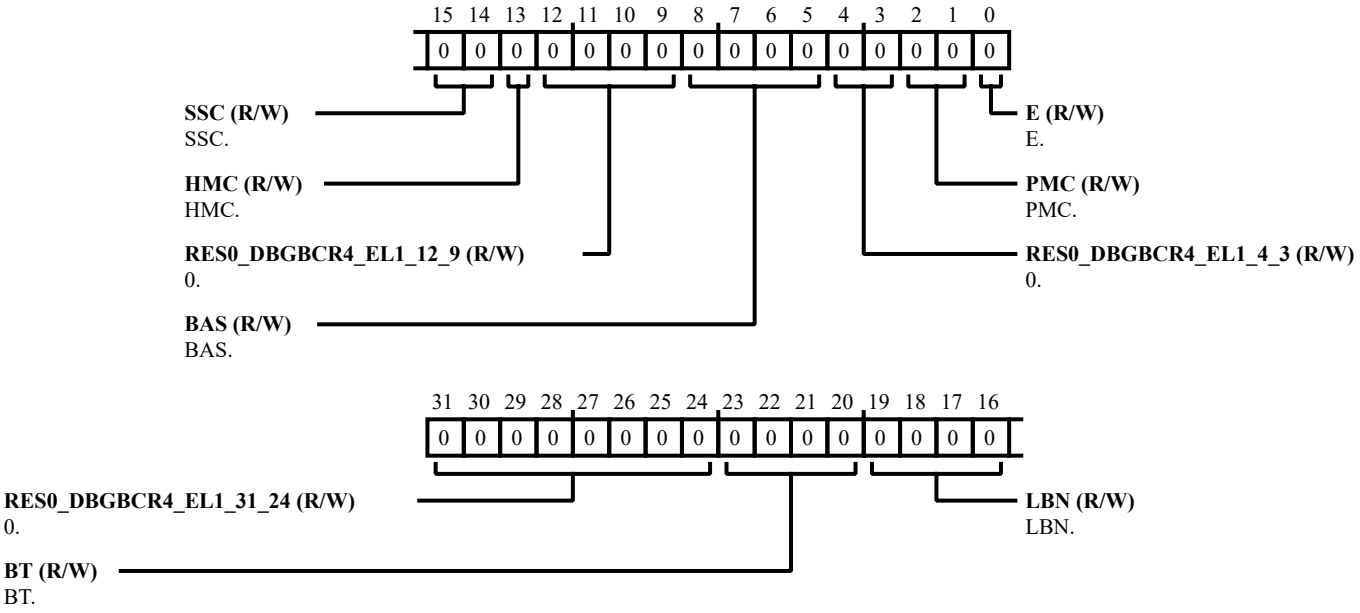


Figure 48-812: A55_C2_DBGBCR4_EL1 Register Diagram

Table 48-820: A55_C2_DBGBCR4_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR4_EL1_31_24	0.. Reserved, RES0.

Table 48-820: A55_C2_DBGBCR4_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-820: A55_C2_DBGBCR4_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR4_EL1_12_9	0.. Reserved, RES0.	

Table 48-820: A55_C2_DBGBCR4_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.
4:3 (R/W)	RES0_DBGBCR4_EL1_4_3	0.. Reserved, RES0.
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled. 0 Breakpoint disabled. 1 Breakpoint enabled.

Debug Breakpoint Control Registers.

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR5_EL1.

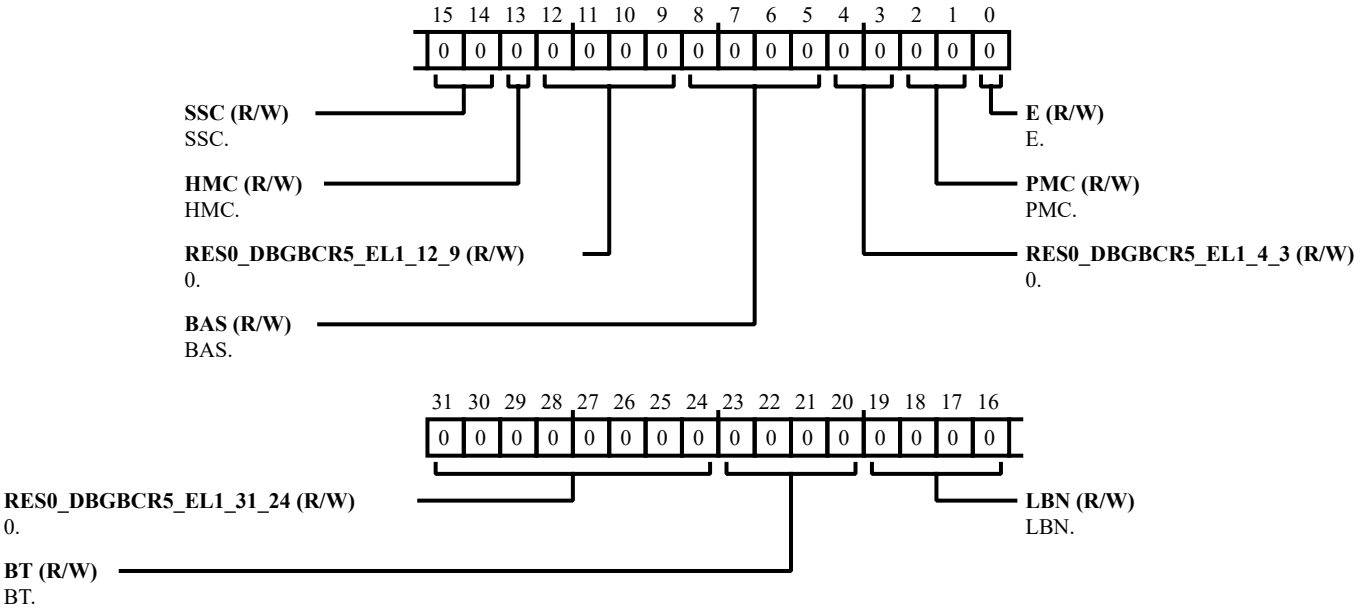


Figure 48-813: A55_C2_DBGBCR5_EL1 Register Diagram

Table 48-821: A55_C2_DBGBCR5_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RES0_DBGBCR5_EL1_31_24	0.. Reserved, RES0.

Table 48-821: A55_C2_DBGBCR5_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	BT	BT.. Breakpoint Type. Possible values are: 0000 Unlinked address match. 0001 Linked address match. 0010 Unlinked Context ID match. 0011 Linked Context ID match. 0100 Unlinked instruction address mismatch. 0101 Linked instruction address mismatch. 0110 Unlinked CONTEXTIDR_EL1 match (ARMv8.1). 0111 Linked CONTEXTIDR_EL1 match (ARMv8.1). 1000 Unlinked VMID match. 1001 Linked VMID match. 1010 Unlinked VMID and Context ID match. 1011 Linked VMID and Context ID match. 1100 Unlinked CONTEXTIDR_EL2 match (ARMv8.1). 1101 Linked CONTEXTIDR_EL2 match (ARMv8.1). 1110 Unlinked Full Context ID match (ARMv8.1). 1111 Linked Full Context ID match (ARMv8.1). The field breaks down as follows: BT[3:1]: Base type.000Match address. DBGBVR<n>_EL1 is the address of an instruction.001Match Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 in ARMv8.0, and in ARMv8.1 when not in a Host OS or a Host Application. In ARMv8.1, when in a Host OS or Host Application, the Context ID is compared against CONTEXTIDR_EL1.010Mismatch address. DBGBVR<n>_EL1 is the address of an instruction to be stepped.011Match CONTEXTIDR_EL1. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.100Match VMID. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.101Match VMID and Context ID. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID.110Match CONTEXTIDR_EL2. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.111Match Full Context ID. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. BT[0]: Enable linking. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see .	
		0	Unlinked address match.
		1	Linked address match.
		2	Unlinked Context ID match.
		3	Linked Context ID match.
		4	Unlinked instruction address mismatch.
		5	Linked instruction address mismatch.
		6	Unlinked CONTEXTIDR_EL1 match (ARMv8.1).
		7	Linked CONTEXTIDR_EL1 match (ARMv8.1).
		8	Unlinked VMID match.
		9	Linked VMID match.

Table 48-821: A55_C2_DBGBCR5_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Unlinked VMID and Context ID match.
		11	Linked VMID and Context ID match.
		12	Unlinked CONTEXTIDR_EL2 match (ARMv8.1).
		13	Linked CONTEXTIDR_EL2 match (ARMv8.1).
		14	Unlinked Full Context ID match (ARMv8.1).
		15	Linked Full Context ID match (ARMv8.1).
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see .	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see DBGBCR<n>_EL1.SSC description.	
12:9 (R/W)	RES0_DBGBCR5_EL1_12_9	0.. Reserved, RES0.	

Table 48-821: A55_C2_DBGBCR5_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8:5 (R/W)	BAS	BAS.. Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1. The permitted values depend on the breakpoint type. For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are: BAS Match instruction at Constraint for debuggers 0011 DBGBVR<n>_EL1 Use for T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for T32 instructions. 1111 DBGBVR<n>_EL1 Use for A64 and A32 instructions. All other values are reserved. For more information, see . For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are: BAS Step instruction at Constraint for debuggers 0000 - Use for a match anywhere breakpoint. 0011 DBGBVR<n>_EL1 Use for stepping T32 instructions. 1100 DBGBVR<n>_EL1+2 Use for stepping T32 instructions. 1111 DBGBVR<n>_EL1 Use for stepping A64 and A32 instructions. For more information, see . For Context matching breakpoints, this field is RES1 and ignored.
4:3 (R/W)	RES0_DBGBCR5_EL1_4_3	0.. Reserved, RES0.
2:1 (R/W)	PMC	PMC.. Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.
0 (R/W)	E	E.. Enable breakpoint DBGBVR<n>_EL1. Possible values are: 0 Breakpoint disabled. 1 Breakpoint enabled. 0 Breakpoint disabled. 1 Breakpoint enabled.

Debug Claim Tag Clear register.

Used by software to read the values of the CLAIM tag bits, and to clear these bits to 0.

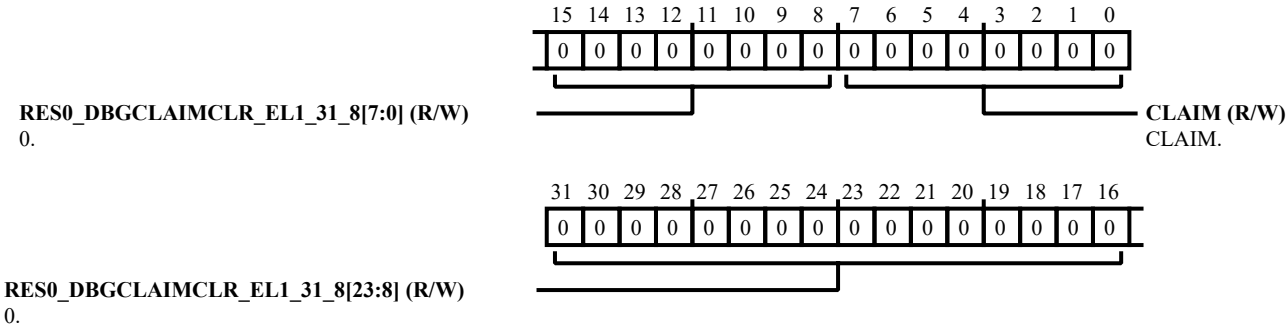


Figure 48-814: A55_C2_DBGCLAIMCLR_EL1 Register Diagram

Table 48-822: A55_C2_DBGCLAIMCLR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_DBGCLAIMCLR_EL1_31_8	0.. Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.
7:0 (R/W)	CLAIM	CLAIM.. Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits. Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0. Writing 0 to one of these bits has no effect. A cold reset clears the CLAIM tag bits to 0.

Debug Claim Tag Set register.

Used by software to set the CLAIM tag bits to 1.

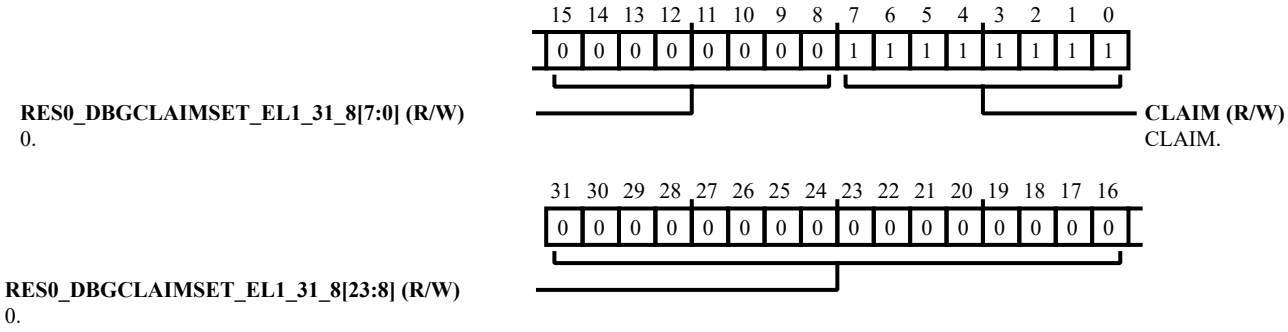


Figure 48-815: A55_C2_DBGCLAIMSET_EL1 Register Diagram

Table 48-823: A55_C2_DBGCLAIMSET_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_DBGCLAIM-SET_EL1_31_8	0.. Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.
7:0 (R/W)	CLAIM	CLAIM.. Set CLAIM tag bits. RAO. Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1. Writing 0 to one of these bits has no effect. A cold reset clears the CLAIM tag bits to 0.

Debug Data Transfer Register, Receive.

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. It is a component of the Debug Communications Channel.

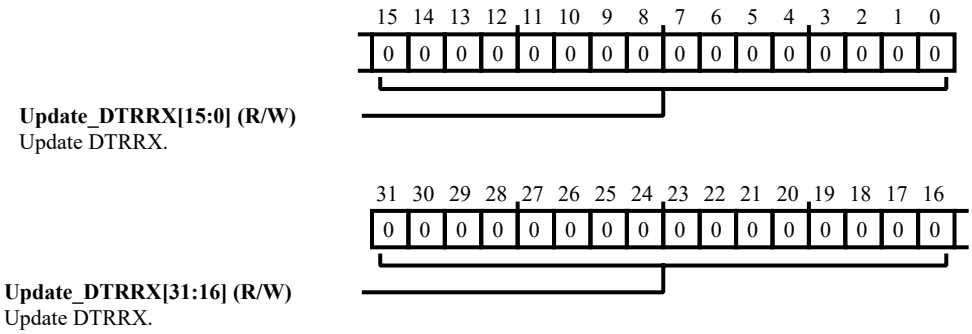


Figure 48-816: A55_C2_DBGDTRRX_EL0 Register Diagram

Table 48-824: A55_C2_DBGDTRRX_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	UPDATE_DTRRX	Update DTRRX.. Update DTRRX. If RXfull is set to 0, then writes to this register update the value in DTRRX and set RXfull to 1. Reads of this register return the last value written to DTRRX and do not change RXfull. For the full behavior of the Debug Communications Channel, see .

Debug Data Transfer Register, Transmit.

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. It is a component of the Debug Communication Channel.

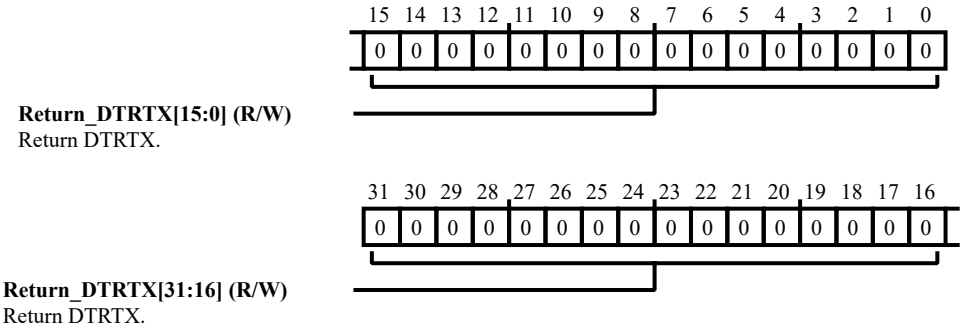


Figure 48-817: A55_C2_DBGDTRTX_EL0 Register Diagram

Table 48-825: A55_C2_DBGDTRTX_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RETURN_DTRTX	Return DTRTX.. Return DTRTX. If TXfull is set to 1, then reads of this register return the value in DTRTX and clear TXfull to 0. Writes of this register update the value in DTRTX and do not change TXfull. For the full behavior of the Debug Communications Channel, see .

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR0_EL1.

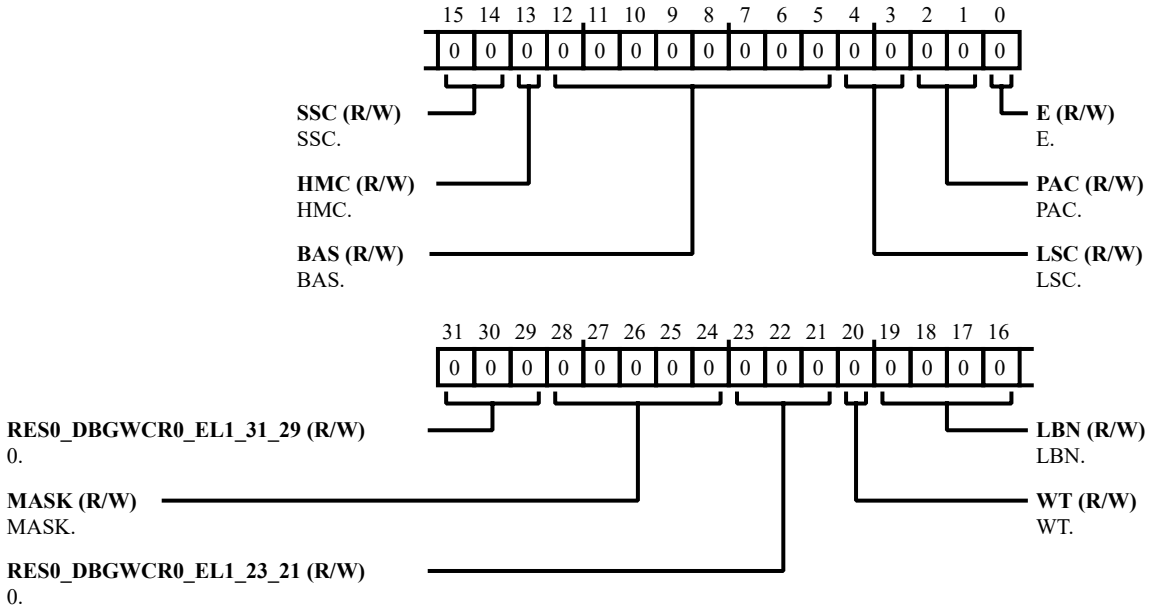


Figure 48-818: A55_C2_DBGWCR0_EL1 Register Diagram

Table 48-826: A55_C2_DBGWCR0_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR0_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-826: A55_C2_DBGWCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR0_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-826: A55_C2_DBGWCR0_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR1_EL1.

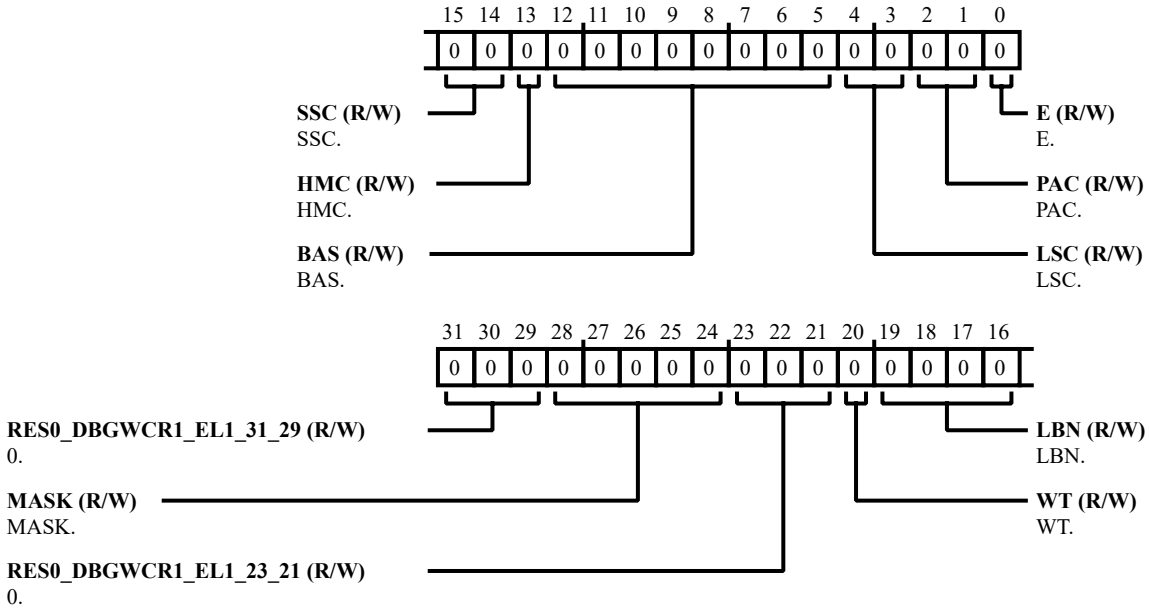


Figure 48-819: A55_C2_DBGWCR1_EL1 Register Diagram

Table 48-827: A55_C2_DBGWCR1_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR1_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-827: A55_C2_DBGWCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR1_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-827: A55_C2_DBGWCR1_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR2_EL1.

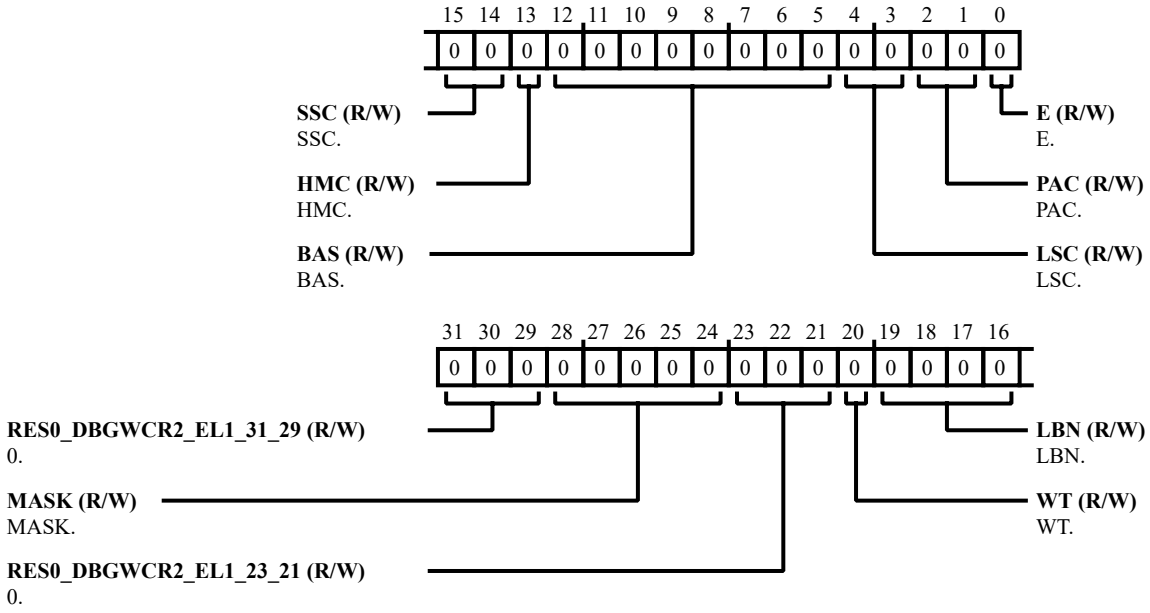


Figure 48-820: A55_C2_DBGWCR2_EL1 Register Diagram

Table 48-828: A55_C2_DBGWCR2_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR2_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-828: A55_C2_DBGWCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR2_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-828: A55_C2_DBGWCR2_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Control Registers.

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR3_EL1.

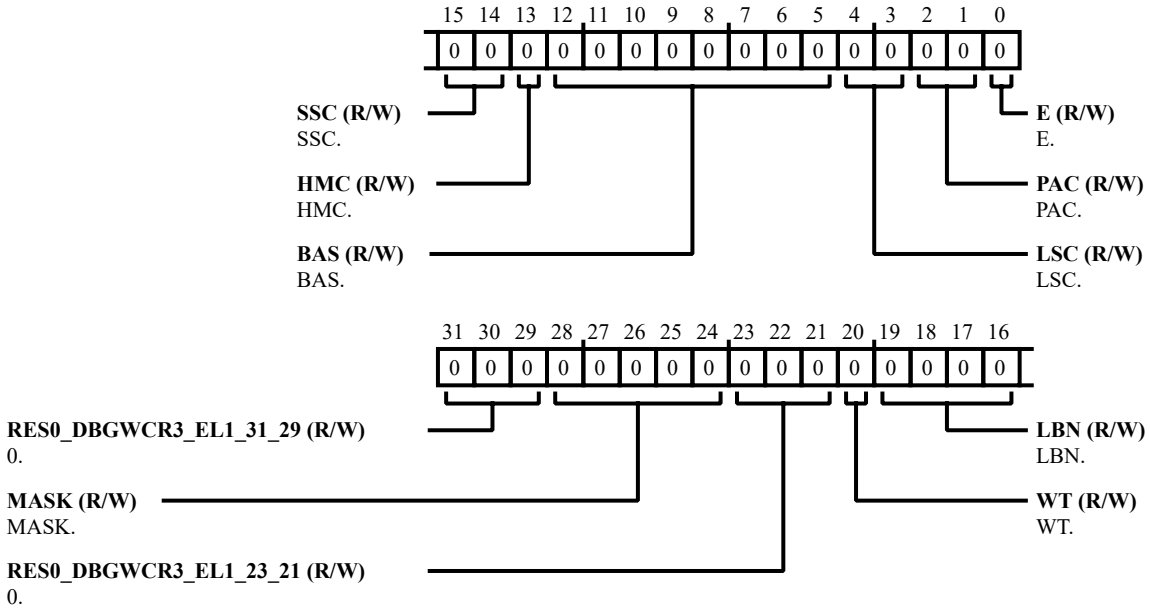


Figure 48-821: A55_C2_DBGWCR3_EL1 Register Diagram

Table 48-829: A55_C2_DBGWCR3_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RES0_DBGWCR3_EL1_31_29	0.. Reserved, RES0.	
28:24 (R/W)	MASK	MASK.. Address mask. Only objects up to 2GB can be watched using a single mask. 00000 No mask. 00001 Reserved. 00010 Reserved. If programmed with a reserved value, a watchpoint must behave as if either: MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).	
		0	No mask.
		1	Reserved.
		2	Reserved.

Table 48-829: A55_C2_DBGWCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	RES0_DBGWCR3_EL1_23_21	0.. Reserved, RES0.	
20 (R/W)	WT	WT.. Watchpoint type. Possible values are: 0 Unlinked data address match. 1 Linked data address match.	
		0	Unlinked data address match.
		1	Linked data address match.
19:16 (R/W)	LBN	LBN.. Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	
15:14 (R/W)	SSC	SSC.. Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.	
13 (R/W)	HMC	HMC.. Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.	
12:5 (R/W)	BAS	BAS.. Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched. BAS Description xxxxxxx1 Match byte at DBGWVR<n>_EL1 xxxxxx1x Match byte at DBGWVR<n>_EL1+1 xxxxx1xx Match byte at DBGWVR<n>_EL1+2 xxxx1xxx Match byte at DBGWVR<n>_EL1+3 In cases where DBGWVR<n>_EL1 addresses a double-word: BAS Description, if DBGWVR<n>_EL1[2] == 0 xxx1xxxx Match byte at DBGWVR<n>_EL1+4 xx1xxxxx Match byte at DBGWVR<n>_EL1+5 x1xxxxxx Match byte at DBGWVR<n>_EL1+6 1xxxxxxx Match byte at DBGWVR<n>_EL1+7 If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See.	

Table 48-829: A55_C2_DBGWCR3_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	LSC	LSC.. Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: 01 Match instructions that load from a watchpointed address. 10 Match instructions that store to a watchpointed address. 11 Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	
		1	Match instructions that load from a watchpointed address.
		2	Match instructions that store to a watchpointed address.
		3	Match instructions that load from or store to a watchpointed address.
2:1 (R/W)	PAC	PAC.. Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.	
0 (R/W)	E	E.. Enable watchpoint n. Possible values are: 0 Watchpoint disabled. 1 Watchpoint enabled.	
		0	Watchpoint disabled.
		1	Watchpoint enabled.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR0_EL1.

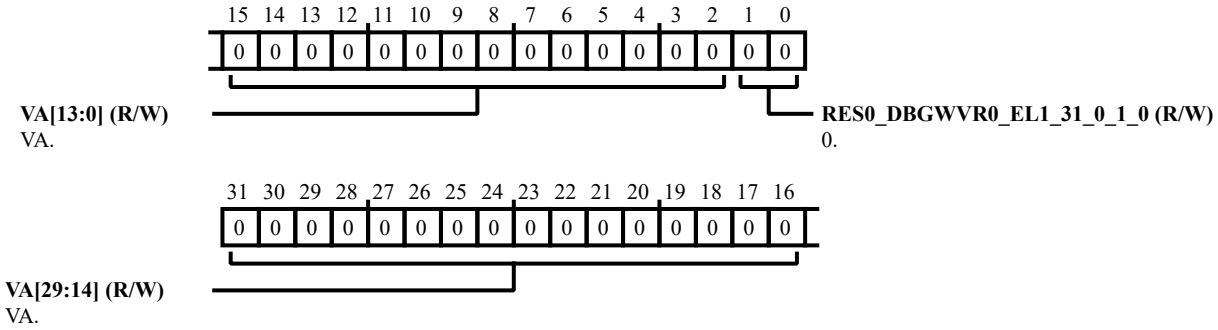


Figure 48-822: A55_C2_DBGWVR0_EL1_31_0 Register Diagram

Table 48-830: A55_C2_DBGWVR0_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR0_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR0_EL1.

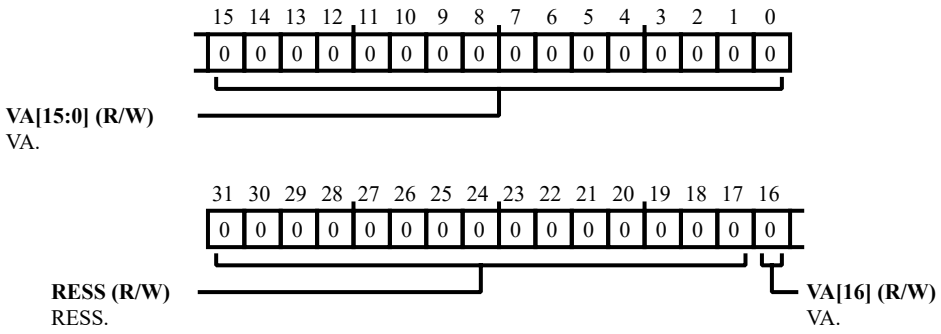


Figure 48-823: A55_C2_DBGWVR0_EL1_63_32 Register Diagram

Table 48-831: A55_C2_DBGWVR0_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR1_EL1.

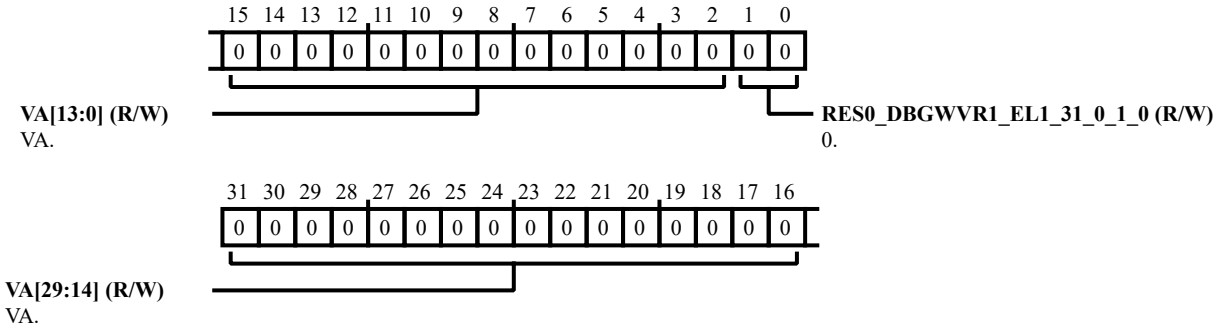


Figure 48-824: A55_C2_DBGWVR1_EL1_31_0 Register Diagram

Table 48-832: A55_C2_DBGWVR1_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR1_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR1_EL1.

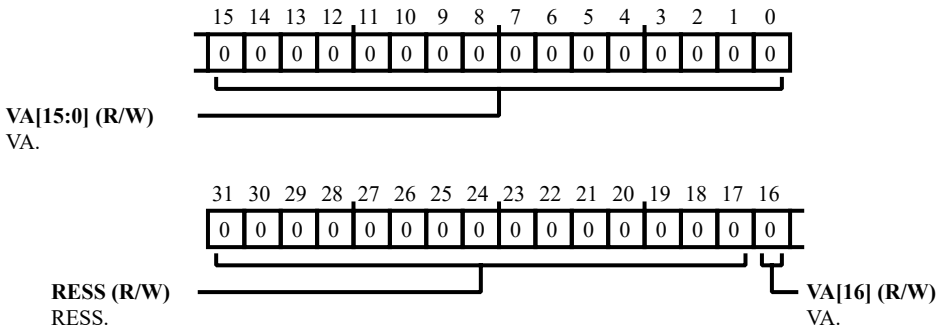


Figure 48-825: A55_C2_DBGWVR1_EL1_63_32 Register Diagram

Table 48-833: A55_C2_DBGWVR1_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR2_EL1.

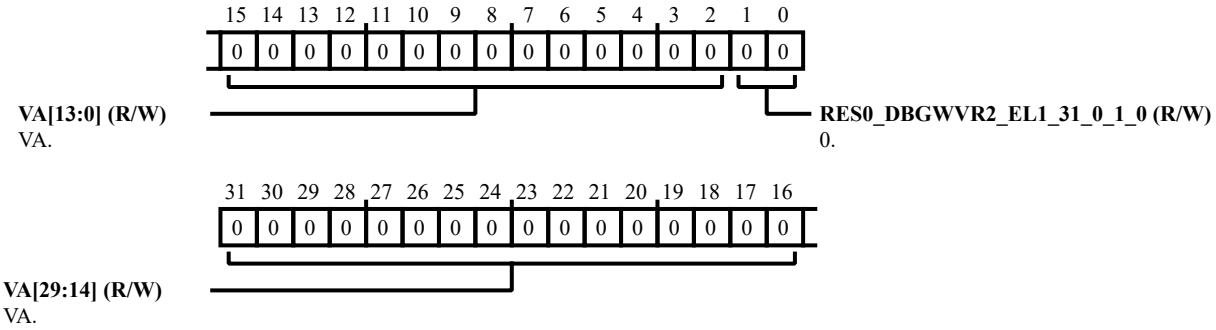


Figure 48-826: A55_C2_DBGWVR2_EL1_31_0 Register Diagram

Table 48-834: A55_C2_DBGWVR2_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR2_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR2_EL1.

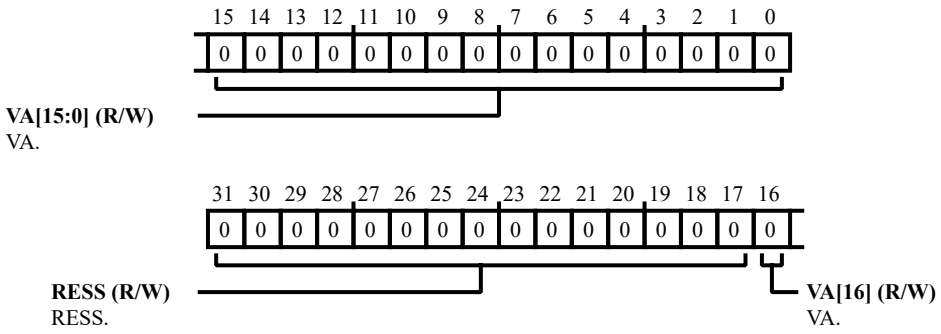


Figure 48-827: A55_C2_DBGWVR2_EL1_63_32 Register Diagram

Table 48-835: A55_C2_DBGWVR2_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR3_EL1.

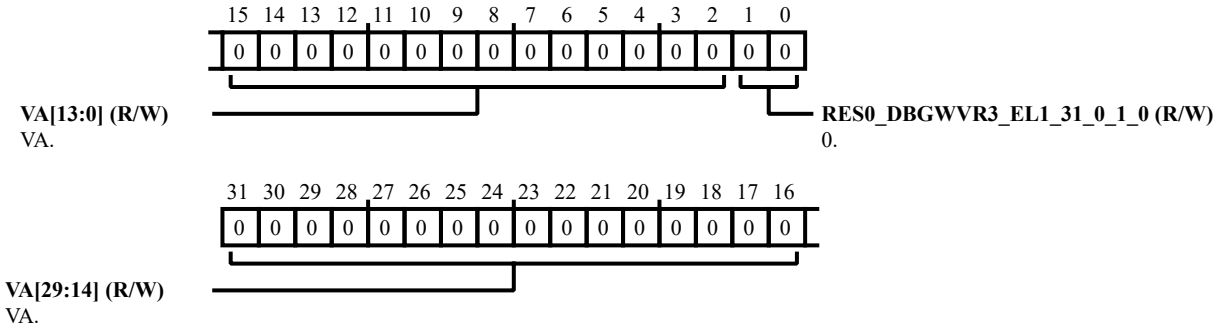


Figure 48-828: A55_C2_DBGWVR3_EL1_31_0 Register Diagram

Table 48-836: A55_C2_DBGWVR3_EL1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.
1:0 (R/W)	RES0_DBGWVR3_EL1_31_0_1_0	0.. Reserved, RES0.

Debug Watchpoint Value Registers.

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register DBGWCR3_EL1.

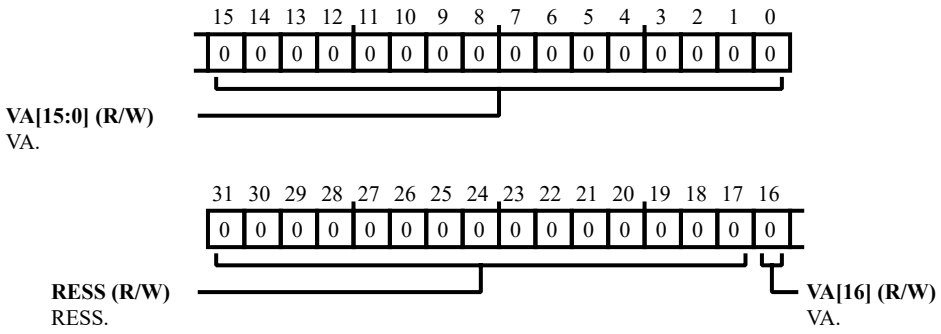


Figure 48-829: A55_C2_DBGWVR3_EL1_63_32 Register Diagram

Table 48-837: A55_C2_DBGWVR3_EL1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:17 (R/W)	RESS	RESS.. Reserved, Sign extended. Hardware and software must treat this field as RES0 if bit[48] is 0 or RES0, and as RES1 if bit[48] is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: The bits are hardwired to a copy of bit [48], meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.
16:0 (R/W)	VA	VA.. Bits[48:2] of the address value for comparison. ARM deprecates setting DBGWVR<n>_EL1[2] == 1.

External Debug Context ID Sample Register.

Contains the sampled value of the Context ID, captured on reading the low half of EDPCSR.

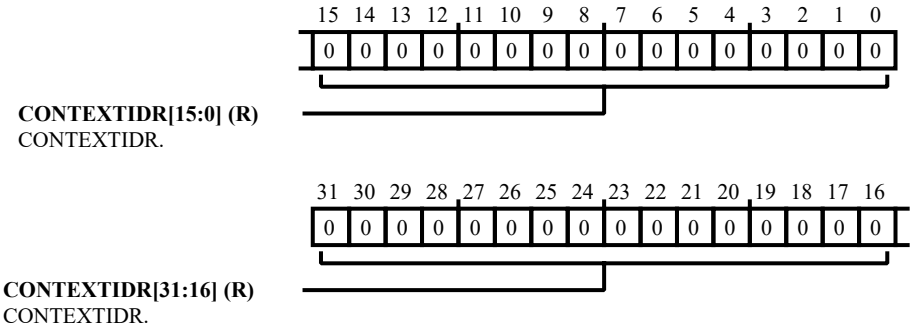


Figure 48-830: A55_C2_EDCIDSR Register Diagram

Table 48-838: A55_C2_EDCIDSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CONTEXTIDR	CONTEXTIDR.. The sampled value of the Context ID, captured on reading the low half of EDPCSR. If EL1 is using AArch64, the Context ID is held in CONTEXTIDR_EL1. If EL1 is using AArch32, the Context ID is held in CONTEXTIDR. If EL3 is implemented and is using AArch32 then CONTEXTIDR is a Banked register, and EDCIDSR samples the current Banked copy of CONTEXTIDR.

External Debug Exception Catch Control Register.

Controls Exception Catch debug events.

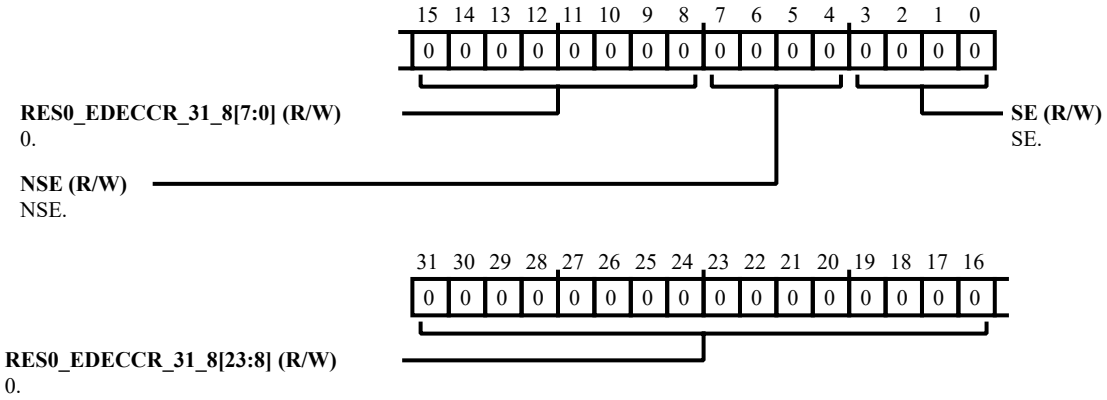


Figure 48-831: A55_C2_EDECCR Register Diagram

Table 48-839: A55_C2_EDECCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	RES0_EDECCR_31_8	0.. Reserved, RES0.	
7:4 (R/W)	NSE	NSE.. Coarse-grained Non-secure exception catch. If EL3 and EL2 are not implemented and the PE behaves as if SCR_EL3.NS is set to 0, this field is reserved, RES0. Otherwise, possible values for this field are: 0000 Exception Catch debug event disabled for Non-secure Exception levels. 0010 Exception Catch debug event enabled for Non-secure EL1. 0100 Exception Catch debug event enabled for Non-secure EL2. 0110 Exception Catch debug event enabled for Non-secure EL1 and EL2. All other values are reserved. A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then: The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR. The value returned for NSE by a read of EDECCR is UNKNOWN.	
		0	Exception Catch debug event disabled for Non-secure Exception levels.
		2	Exception Catch debug event enabled for Non-secure EL1.
		4	Exception Catch debug event enabled for Non-secure EL2.
		6	Exception Catch debug event enabled for Non-secure EL1 and EL2.

Table 48-839: A55_C2_EDECCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	SE	SE.. Coarse-grained Secure exception catch. If EL3 is not implemented and the PE behaves as if SCR_EL3.NS is set to 1, this field is reserved, RES0. Otherwise, possible values for this field are: 0000 Exception Catch debug event disabled for Secure Exception levels. 0010 Exception Catch debug event enabled for Secure EL1. 1000 Exception Catch debug event enabled for Secure EL3. 1010 Exception Catch debug event enabled for Secure EL1 and EL3. All other values are reserved. A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then: The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR. The value returned for SE by a read of EDECCR is UNKNOWN.	
		0	Exception Catch debug event disabled for Secure Exception levels.
		2	Exception Catch debug event enabled for Secure EL1.
		8	Exception Catch debug event enabled for Secure EL3.
		10	Exception Catch debug event enabled for Secure EL1 and EL3.

External Debug Event Status Register.

Indicates the status of internally pending Halting debug events.

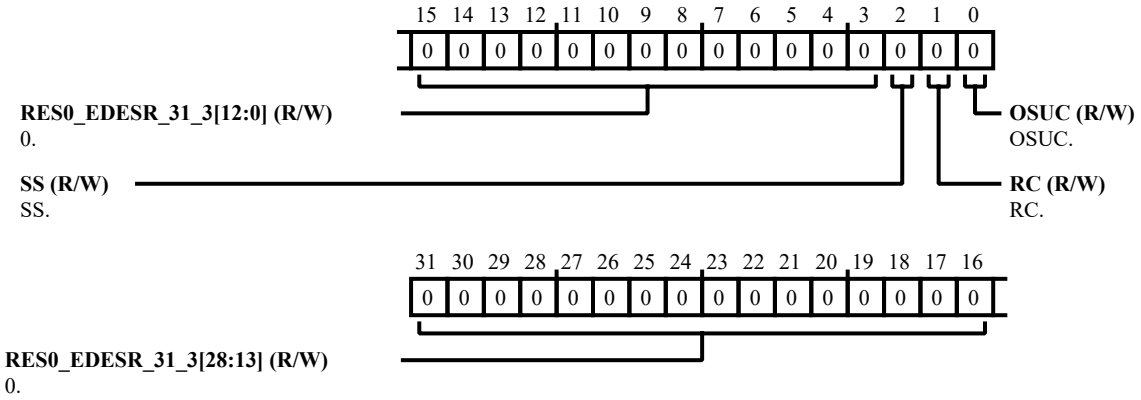


Figure 48-832: A55_C2_EDESR Register Diagram

Table 48-840: A55_C2_EDESR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:3 (R/W)	RES0_EDESR_31_3	0.. Reserved, RES0.	
2 (R/W)	SS	SS.. Halting step debug event pending. Possible values of this field are: 0 Reading this means that a Halting step debug event is not pending. Writing this means no action. 1 Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.	
		0	Reading this means that a Halting step debug event is not pending. Writing this means no action.
		1	Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.
1 (R/W)	RC	RC.. Reset Catch debug event pending. Possible values of this field are: 0 Reading this means that a Reset Catch debug event is not pending. Writing this means no action. 1 Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.	
		0	Reading this means that a Reset Catch debug event is not pending. Writing this means no action.
		1	Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.

Table 48-840: A55_C2_EDESR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	OSUC	OSUC.. OS Unlock Catch debug event pending. Possible values of this field are: 0 Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action. 1 Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.	
		0	Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action.
		1	Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.

External Debug Integration mode Control register.

Enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

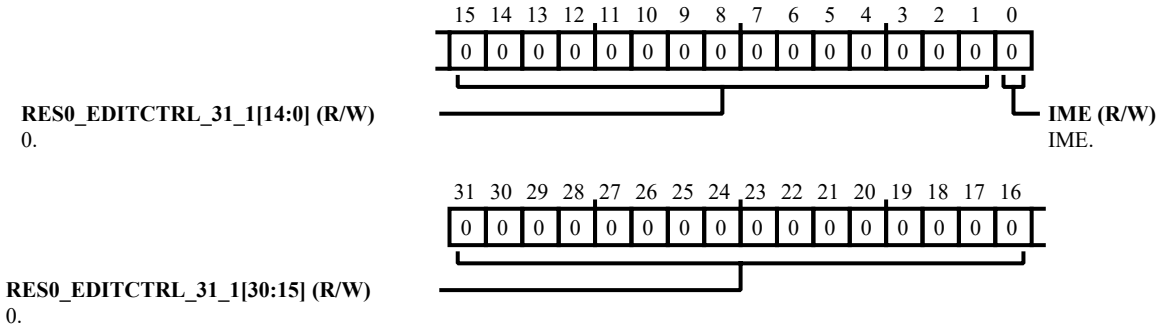


Figure 48-833: A55_C2_EDITCTRL Register Diagram

Table 48-841: A55_C2_EDITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_EDITCTRL_31_1	0.. Reserved, RES0.	
0 (R/W)	IME	IME.. Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED. 0 Normal operation. 1 Integration mode enabled.	
		0	Normal operation.
		1	Integration mode enabled.

External Debug Instruction Transfer Register.

Used in Debug state for passing instructions to the PE for execution.

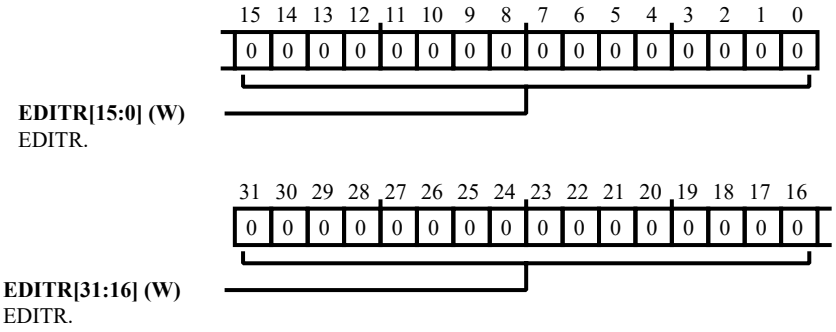


Figure 48-834: A55_C2_EDITR Register Diagram

Table 48-842: A55_C2_EDITR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	EDITR	EDITR.. Used in Debug state for passing instructions to the processor for execution.

External Debug Power/Reset Control Register.

Controls the PE functionality related to powerup, reset, and powerdown.

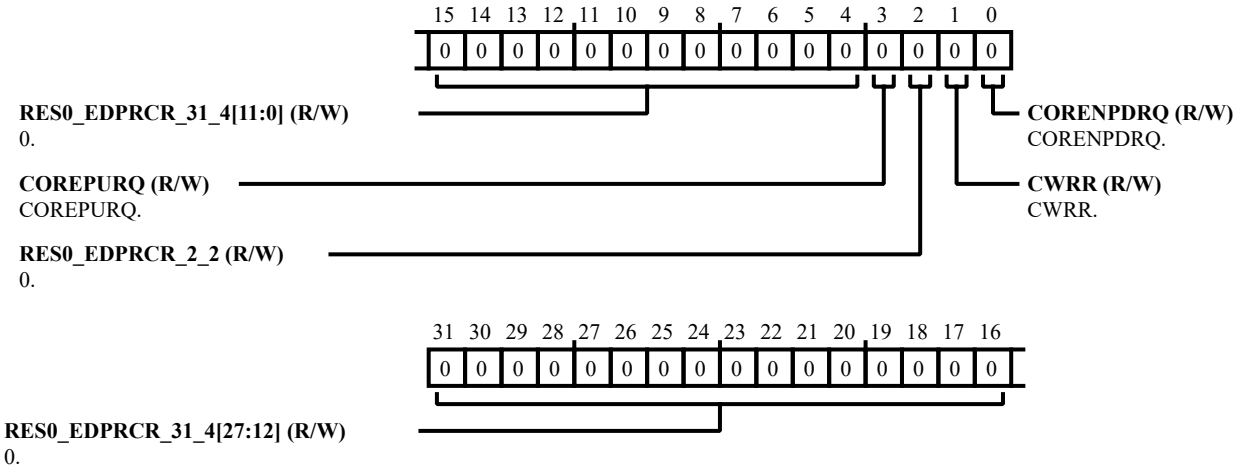


Figure 48-835: A55_C2_EDPRCR Register Diagram

Table 48-843: A55_C2_EDPRCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RES0_EDPRCR_31_4	0.. Reserved, RES0.	
3 (R/W)	COREPURQ	COREPURQ.. Core powerup request. Allows a debugger to request that the power controller power up the core, enabling access to the debug register in the Core power domain. The actions on writing to this bit are: 0 Do not request power up of the Core power domain. 1 Request power up of the Core power domain, and emulation of powerdown. In an implementation that includes the recommended external debug interface, this bit drives the DBGPWRUPREQ signal. Typically, this request is passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. This field is in the Debug power domain and can be read and written when the Core power domain is powered off. On an External debug reset this field resets to 0. The power controller must not allow the Core power domain to switch off while this bit is one.	
		0	Do not request power up of the Core power domain.
		1	Request power up of the Core power domain, and emulation of powerdown.
2 (R/W)	RES0_EDPRCR_2_2	0.. Reserved, RES0.	

Table 48-843: A55_C2_EDPRCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	CWRR	CWRR.. Warm reset request. Write only bit that reads as zero. The actions on writing to this bit are: 0 No action. 1 Request Warm reset. The PE ignores writes to this bit if any of the following are true: ExternalInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Non-Secure state. External-SecureInvasiveDebugEnabled() == FALSE and one of the following is true:EL3 is implemented.The implemented Security state is Secure state. The Core power domain is either off or in a low-power state where the Core power domain registers cannot be accessed. DoubleLockStatus() == TRUE (OS Double Lock is set). OSLSR.OSLK == 1 (OS lock is locked). In an implementation that includes the recommended external debug interface, this bit drives the DBGRSTREQ signal. Whether a write to this bit initiates a Warm Reset, the extent of the reset is IMPLEMENTATION DEFINED, but must be one of: The request is ignored. Only this PE is Warm reset. This PE and other components of the system, possibly including other PEs, are Warm reset. Although the ARM architecture permits the first option from the above list, ARM recommends that either of the other options is implemented.	
		0	No action.
		1	Request Warm reset.
0 (R/W)	CORENPDRQ	CORENPDRQ.. Core no powerdown request. Requests emulation of powerdown. Possible values of this bit are: 0 If the system responds to a powerdown request, it powers down Core power domain. 1 If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. This bit is UNKNOWN, and the PE ignores writes to this bit if any of the following are true: The Core power domain is either off or in a low-power state where the Core power domain registers cannot be accessed. DoubleLockStatus() == TRUE (OS Double Lock is set). OSLSR.OSLK == 1 (OS lock is locked).	
		0	If the system responds to a powerdown request, it powers down Core power domain.
		1	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

External Debug Processor Status Register.

Holds information about the reset and powerdown state of the PE.

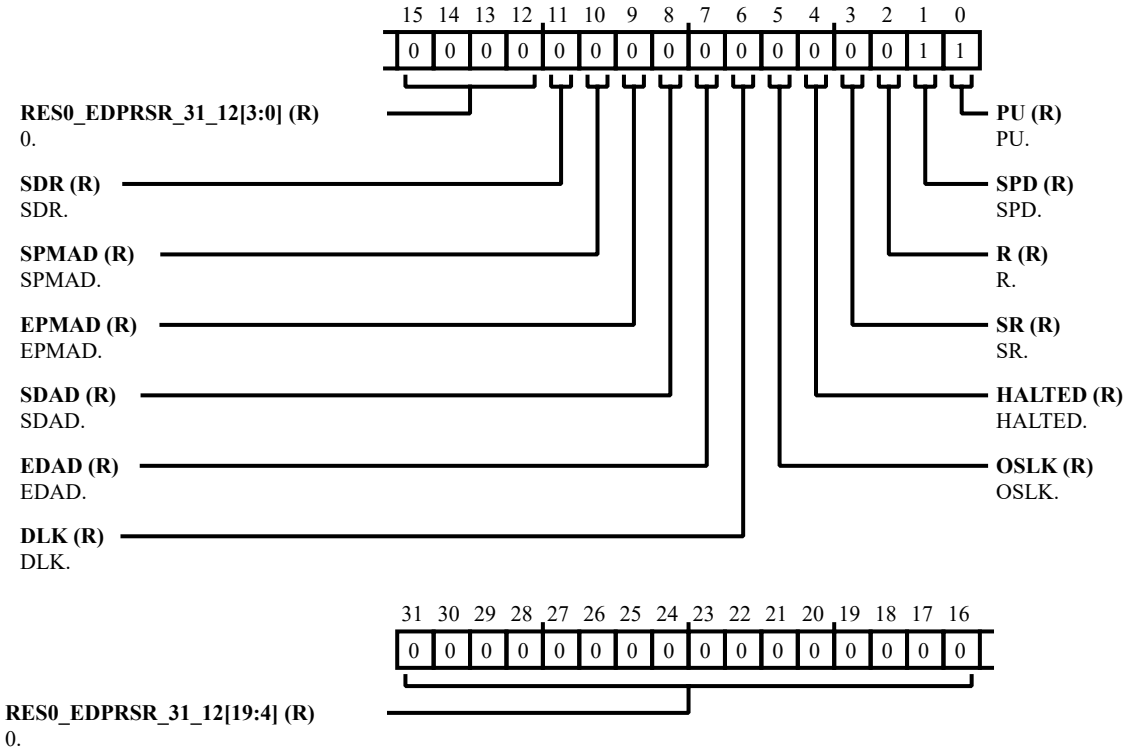


Figure 48-836: A55_C2_EDPRSR Register Diagram

Table 48-844: A55_C2_EDPRSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/NW)	RES0_EDPRSR_31_12	0.. Reserved, RES0.

Table 48-844: A55_C2_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	SDR	SDR.. Sticky debug restart. Set to 1 when the PE exits Debug state. This bit is UNKNOWN on reads if any of the following are true: DoubleLockStatus() == TRUE (EDPRSR.DLK == 1). The OS double-lock is locked. EDPRSR.R == 1. The PE is in Reset state. EDPRSR.PU == 0. The Core power domain is powered down. Otherwise permitted values are: 0 The PE has not restarted since EDPRSR was last read. 1 The PE has restarted since EDPRSR was last read. If a reset occurs when the PE is in Debug state, the PE exits Debug state. SDR is UNKNOWN on Warm reset, meaning a debugger must also use the SR bit to determine whether the PE has left Debug state. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs.	
		0	The PE has not restarted since EDPRSR was last read.
		1	The PE has restarted since EDPRSR was last read.
10 (R/NW)	SPMAD	SPMAD.. Sticky EPMAD error. Set to 1 if an external debug interface access to a Performance Monitors register returns an error because AllowExternalPMUAccess() == FALSE. This bit is UNKNOWN on reads if any of the following are true: EDPRSR.{DLK, OSLK, R} is 1. EDPRSR.PU is 0. Otherwise permitted values are: 0 No accesses to the external Performance Monitors registers have failed since EDPRSR was last read. 1 At least one access to the external Performance Monitors registers has failed since EDPRSR was last read. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs. The write to SPMAD is an indirect write to EDPRSR that is a side effect of the access. The indirect write might not occur for a memory-mapped access to the external debug interface.	
		0	No accesses to the external Performance Monitors registers have failed since EDPRSR was last read.
		1	At least one access to the external Performance Monitors registers has failed since EDPRSR was last read.

Table 48-844: A55_C2_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	EPMAD	EPMAD.. External Performance Monitors access disable status. This bit is UNKNOWN on reads if any of EDPRSR.{DLK, OSLK, R} is 1, or EDPRSR.PU is 0. Otherwise permitted values are: 0 External Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE. 1 External Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE. If external performance monitors access is not implemented, EPMAD is RAO.	
		0	External Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE.
		1	External Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE.
8 (R/NW)	SDAD	SDAD.. Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because AllowExternalDebugAccess() == FALSE. This bit is UNKNOWN on reads if any of the following are true: Either of EDPRSR.{DLK, R} is 1. EDPRSR.PU is 0. EDPRSR.OSLK is 1 and external debug writes to OSLAR_EL1 do not return an error when AllowExternalDebugAccess() == FALSE. Otherwise permitted values are: 0 No accesses to the external debug registers have failed since EDPRSR was last read. 1 At least one access to the external debug registers has failed since EDPRSR was last read. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs. The write to SDAD is an indirect write to EDPRSR that is a side effect of the access. The indirect write might not occur for a memory-mapped access to the external debug interface.	
		0	No accesses to the external debug registers have failed since EDPRSR was last read.
		1	At least one access to the external debug registers has failed since EDPRSR was last read.
7 (R/NW)	EDAD	EDAD.. External debug access disable status. This bit is UNKNOWN on reads if any of the following are true: Either of EDPRSR.{DLK, R} is 1. EDPRSR.PU is 0. EDPRSR.OSLK is 1 and external debug writes to OSLAR_EL1 do not return an error when AllowExternalDebugAccess() == FALSE. Otherwise permitted values are: 0 External debug access enabled. AllowExternalDebugAccess() == TRUE. 1 External debug access disabled. AllowExternalDebugAccess() == FALSE.	
		0	External debug access enabled. AllowExternalDebugAccess() == TRUE.
		1	External debug access disabled. AllowExternalDebugAccess() == FALSE.

Table 48-844: A55_C2_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	DLK	DLK.. OS Double Lock status bit. Returns the result of the pseudocode function DoubleLockStatus(). This bit is UNKNOWN on reads if EDPRSR.PU is 0. Otherwise reads as zero if any of the following are true, that is when DoubleLockStatus() == FALSE: OSDLR_EL1.DLK == 0. DBGPRCR_EL1.CORENPDRQ == 1. The PE is in Debug state. If the Core power domain is powered up and DoubleLockStatus() == TRUE, it is IMPLEMENTATION DEFINED whether: EDPRSR.PU reads as 1, EDPRSR.DLK reads as 1, and EDPRSR.SPD is UNKNOWN. EDPRSR.PU reads as 0, EDPRSR.DLK is UNKNOWN, and EDPRSR.SPD reads as 0. If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see.	
5 (R/NW)	OSLK	OSLK.. OS lock status bit. This bit is UNKNOWN on reads if either of EDPRSR.{DLK, R} is 1 or EDPRSR.PU is 0. A read of this bit returns the value of OSLSR_EL1.OSLK.	
4 (R/NW)	HALTED	HALTED.. Halted status bit. This bit is UNKNOWN on reads if EDPRSR.PU is 0. Otherwise permitted values are: 0 PE in Non-debug state. 1 PE is in Debug state. Because the OS Double Lock is never set when the PE is in Debug state, this bit is always RAZ when EDPRSR.DLK is set to 1.	
		0	PE in Non-debug state.
		1	PE is in Debug state.
3 (R/NW)	SR	SR.. Sticky core reset status bit. This bit is UNKNOWN on reads if EDPRSR.DLK is 1 or EDPRSR.PU is 0. Otherwise permitted values are: 0 The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read. 1 The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read. If DoubleLockStatus() == FALSE then following a read of EDPRSR, this bit clears to 0 if the non-debug logic of the PE is not in reset state. Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs.	
		0	The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read.
		1	The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read.

Table 48-844: A55_C2_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	R	R.. PE reset status bit. This bit is UNKNOWN on reads if either EDPRSR.DLK is 1 or EDPRSR.PU is 0. Otherwise permitted values are: 0 The non-debug logic of the PE is not in reset state. 1 The non-debug logic of the PE is in reset state. If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see.	
		0	The non-debug logic of the PE is not in reset state.
		1	The non-debug logic of the PE is in reset state.
1 (R/NW)	SPD	SPD.. Sticky core powerdown status bit. This bit is UNKNOWN on reads if both EDPRSR.DLK and EDPRSR.PU are 1. Otherwise, permitted values are: 0 If EDPRSR.PU is 0, the debugger does not know whether the state of the debug registers in the Core power domain is lost. If the Core power domain is in a powerup state (EDPRSR.PU is 1), the state of the debug registers in the Core power domain has not been lost. 1 The EDPRSR.PU is 1, the state of the debug registers in the Core power domain has been lost. If DoubleLockStatus() == FALSE and the PE is not in the powerdown state, then following a read of EDPRSR, this bit clears to 0. If DoubleLockStatus() == TRUE and the PE is not in the powerdown state, it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs. When the value of EDPRSR.PU is 0 indicating that the Core power domain is in either retention or powerdown state, EDPRSR.SPD is not cleared following a read of EDPRSR, for the IMPLEMENTATION DEFINED behavior see EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see.	
		0	If EDPRSR.PU is 0, the debugger does not know whether the state of the debug registers in the Core power domain is lost. If the Core power domain is in a powerup state (EDPRSR.PU is 1), the state of the debug registers in the Core power domain has not been lost.
		1	The EDPRSR.PU is 1, the state of the debug registers in the Core power domain has been lost.

Table 48-844: A55_C2_EDPRSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	PU	PU.. Core powerup status bit. Indicates whether the Core power domain debug registers can be accessed. The value of EDPRSR.PU is IMPLEMENTATION DEFINED when the Core power domain is powered-up and OS double-lock is locked. See the description of DLK for more information. Otherwise, permitted values are: 0 Core is in a low-power or powerdown state where the debug registers cannot be accessed. 1 Core is in a powerup state where the debug registers can be accessed. If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see.	
		0	Core is in a low-power or powerdown state where the debug registers cannot be accessed.
		1	Core is in a powerup state where the debug registers can be accessed.

External Debug Reserve Control Register.

This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

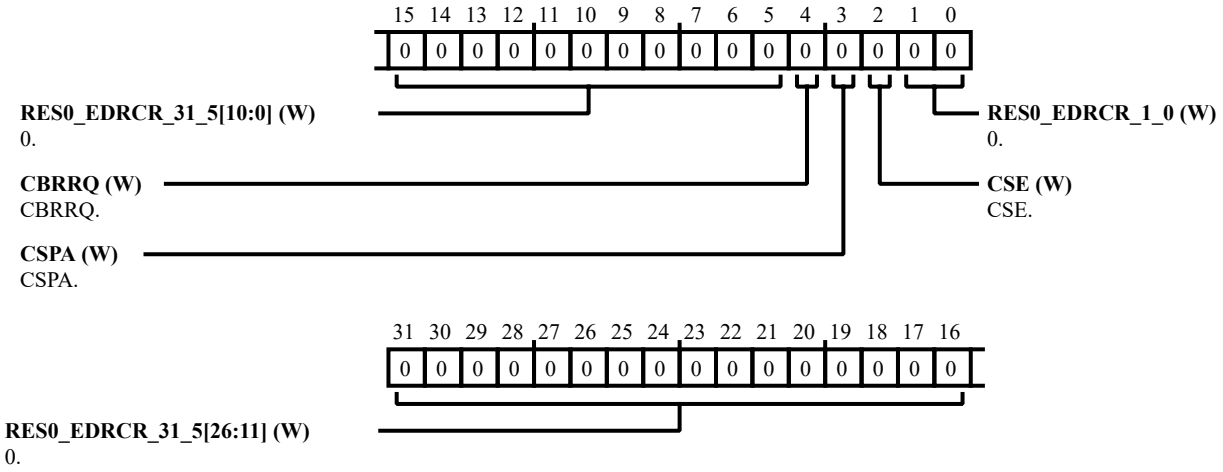


Figure 48-837: A55_C2_EDRCR Register Diagram

Table 48-845: A55_C2_EDRCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (RX/W)	RES0_EDRCR_31_5	0.. Reserved, RES0.	
4 (RX/W)	CBRRQ	CBRRQ.. Allow imprecise entry to Debug state. The actions on writing to this bit are: 0 No action. 1 Allow imprecise entry to Debug state, for example by canceling pending bus accesses. Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1. This feature is optional. If this feature is not implemented, writes to this bit are ignored.	
		0	No action.
		1	Allow imprecise entry to Debug state, for example by canceling pending bus accesses.
3 (RX/W)	CSPA	CSPA.. Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The actions on writing to this bit are: 0 No action. 1 Clear the EDSCR.PipeAdv bit to 0.	
		0	No action.
		1	Clear the EDSCR.PipeAdv bit to 0.

Table 48-845: A55_C2_EDRCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (RX/W)	CSE	CSE.. Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The actions on writing to this bit are: 0 No action. 1 Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the EDSCR.ITO bit, to 0.	
		0	No action.
		1	Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the EDSCR.ITO bit, to 0.
1:0 (RX/W)	RES0_EDRCR_1_0	0.. Reserved, RES0.	

External Debug Status and Control Register.

Main control register for the debug implementation.

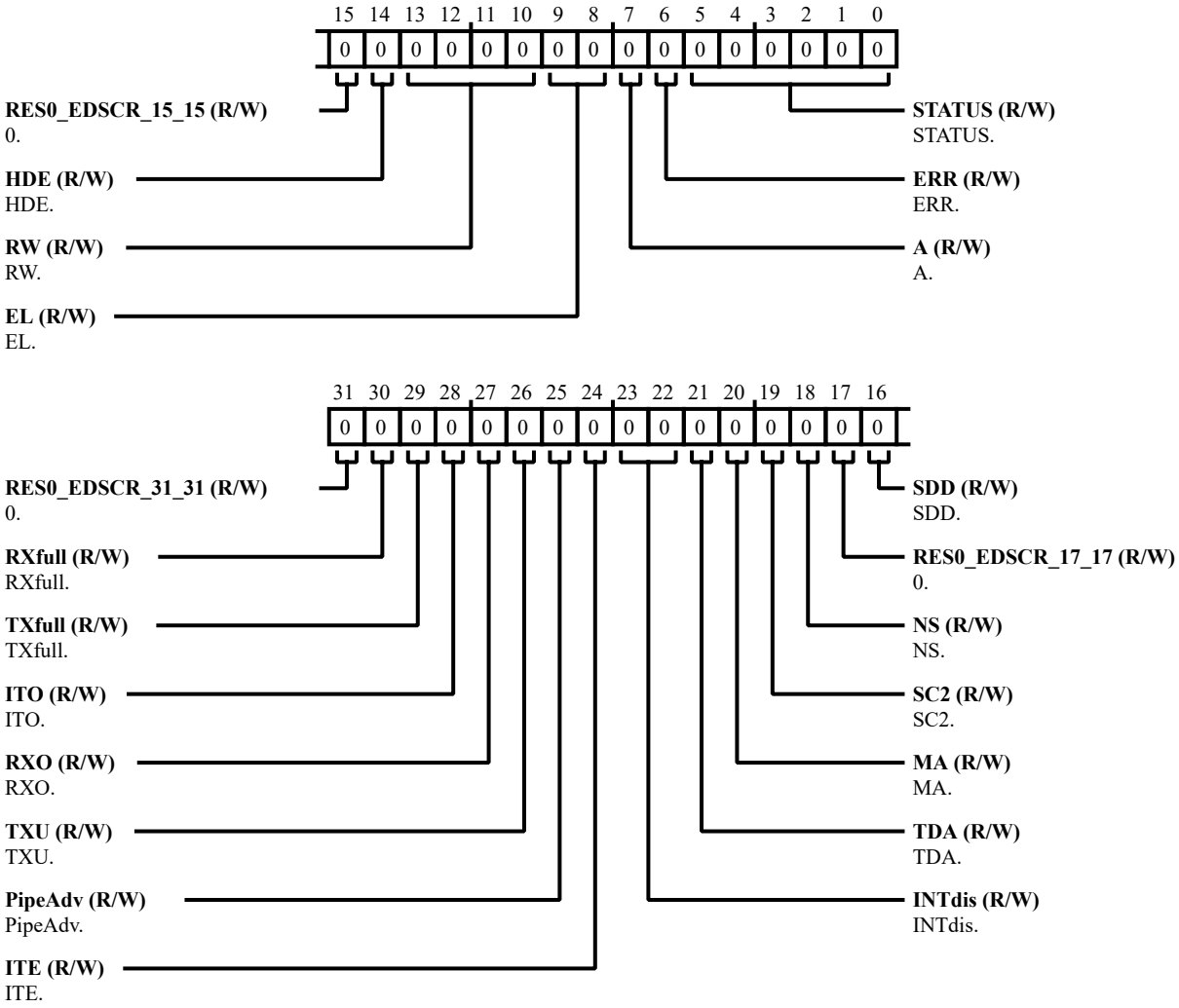


Figure 48-838: A55_C2_EDSCR Register Diagram

Table 48-846: A55_C2_EDSCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	RES0_EDSCR_31_31	0.. Reserved, RES0.
30 (R/W)	RXFULL	RXfull.. DTRRX full. This bit is RO.

Table 48-846: A55_C2_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	TXFULL	TXfull.. DTRTX full. This bit is RO.
28 (R/W)	ITO	ITO.. ITR overrun. This bit is RO. If the PE is in Non-debug state, this bit is UNKNOWN. ITO is set to 0 on entry to Debug state.
27 (R/W)	RXO	RXO.. DTRRX overrun. This bit is RO.
26 (R/W)	TXU	TXU.. DTRTX underrun. This bit is RO.
25 (R/W)	PIPEADV	PipeAdv.. Pipeline advance. This bit is RO. Set to 1 every time the PE pipeline retires one or more instructions. Cleared to 0 by a write to EDRCR.CSPA. The architecture does not define precisely when this bit is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing.
24 (R/W)	ITE	ITE.. ITR empty. This bit is RO. If the PE is in Non-debug state, this bit is UNKNOWN. It is always valid in Debug state.

Table 48-846: A55_C2_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/W)	INTDIS	INTdis.. Interrupt disable. Disables taking interrupts (including virtual interrupts and System Error interrupts) in Non-Debug state. If external invasive debug is disabled, the value of this field is ignored. If external invasive debug is enabled, the possible values of this field are: 00 Do not disable interrupts. 01 Disable interrupts taken to Non-secure EL1. 10 Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable interrupts taken to Secure EL1. 11 Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable all other interrupts. The value of INTdis does not affect whether an interrupt is a WFI wake-up event, but can mask an interrupt as a WFE wake-up event. If EL3 and EL2 are not implemented, the values 0b01 and 0b10 are reserved. If programmed with a reserved value the PE behaves as if INTdis has been programmed with a defined value, other than for a direct read of EDSCR, and the value returned by a read of EDSCR.INTdis is UNKNOWN.	
		0	Do not disable interrupts.
		1	Disable interrupts taken to Non-secure EL1.
		2	Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable interrupts taken to Secure EL1.
		3	Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable all other interrupts.
21 (R/W)	TDA	TDA.. Traps accesses to the following Debug System registers: AArch64: DBGBCR<n>_EL1, DBGBVR<n>_EL1, DBGWCR<n>_EL1, DBGWVR<n>_EL1. AArch32: DBGBCR<n>, DBGBVR<n>, DBGBXVR<n>, DBGWCR<n>, DBGWVR<n>. The possible values of this field are: 0 Accesses to Debug System registers do not generate a Software Access debug event. 1 Accesses to Debug System registers generate a Software Access debug event, if OSLSR.OSLK is 0 and if halting is allowed.	
		0	Accesses to Debug System registers do not generate a Software Access debug event.
		1	Accesses to Debug System registers generate a Software Access debug event, if OSLSR.OSLK is 0 and if halting is allowed.

Table 48-846: A55_C2_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	MA	MA.. Memory access mode. Controls use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state. Possible values of this field are: 0 Normal access mode. 1 Memory access mode.	
		0	Normal access mode.
		1	Memory access mode.
19 (R/W)	SC2	SC2.. Sample CONTEXTIDR_EL2. Controls whether the Sample-based Profiling Extension samples CONTEXTIDR_EL2 or VTTBR_EL2.VMID. 0 Sample VTTBR_EL2.VMID. 1 Sample CONTEXTIDR_EL2.	
		0	Sample VTTBR_EL2.VMID.
		1	Sample CONTEXTIDR_EL2.
18 (R/W)	NS	NS.. Non-secure status. Read-only. When in Debug state, gives the current Security state: 0 Secure state, IsSecure() == TRUE. 1 Non-secure state, IsSecure() == FALSE. In Non-debug state, this bit is UNKNOWN.	
		0	Secure state, IsSecure() == TRUE.
		1	Non-secure state, IsSecure() == FALSE.
17 (R/W)	RES0_EDSCR_17_17	0.. Reserved, RES0.	
16 (R/W)	SDD	SDD.. Secure debug disabled. This bit is RO. On entry to Debug state: If entering in Secure state, SDD is set to 0. If entering in Non-secure state, SDD is set to the inverse of ExternalSecureInvasiveDebugEnabled(). In Debug state, the value of the SDD bit does not change, even if ExternalSecureInvasiveDebugEnabled() changes. In Non-debug state: SDD returns the inverse of ExternalSecureInvasiveDebugEnabled(). If the authentication signals that control ExternalSecureInvasiveDebugEnabled() change, a context synchronization operation is required to guarantee their effect. This bit is unaffected by the Security state of the PE. If EL3 is not implemented and the implementation is Non-secure, this bit is RES1.	
15 (R/W)	RES0_EDSCR_15_15	0.. Reserved, RES0.	

Table 48-846: A55_C2_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	HDE	HDE.. Halting debug enable. The possible values of this field are: 0 Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events. 1 Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.	
		0	Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events.
		1	Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.
13:10 (R/W)	RW	RW.. Exception level Execution state status. Read-only. In Debug state, each bit gives the current Execution state of each EL: RW Meaning 1111 All Exception levels are using AArch64. 1110 EL0 is using AArch32. All other Exception levels are using AArch64. 110x EL0 and EL1 are using AArch32. All other Exception levels are using AArch64. Never seen if EL2 is not implemented in the current Security state. 10xx EL0, EL1, and, if implemented in the current Security state, EL2 are using AArch32. All other Exception levels are using AArch64. 0xxx All Exception levels are using AArch32. However: The value of 1110 is only permitted at EL0. The values 110x are not permitted if either:EL2 is not implemented.EL3 is implemented and SCR_EL3.NS/ SCR.NS == 0. The values 10xx are not permitted if EL3 is not implemented. In Non-debug state, this field is RAO.	
9:8 (R/W)	EL	EL.. Exception level. Read-only. In Debug state, this gives the current EL of the PE. In Non-debug state, this field is RAZ.	
7 (R/W)	A	A.. System Error interrupt pending. Read-only. In Debug state, indicates whether a SError interrupt is pending: If HCR_EL2.{AMO, TGE} = {1, 0} and in Non-secure EL0 or EL1, a virtual SError interrupt. Otherwise, a physical SError interrupt. 0 No SError interrupt pending. 1 SError interrupt pending. A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted. UNKNOWN in Non-debug state.	
		0	No SError interrupt pending.
		1	SError interrupt pending.
6 (R/W)	ERR	ERR.. Cumulative error flag. This field is RO. It is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR.	

Table 48-846: A55_C2_EDSCR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:0 (R/W)	STATUS	STATUS.. Debug status flags. This field is RO. The possible values of this field are: 000010 PE is in Non-debug state. 000001 PE is restarting, exiting Debug state. 000111 Breakpoint. 010011 External debug request. 011011 Halting step, normal. 011111 Halting step, exclusive. 100011 OS Unlock Catch. 100111 Reset Catch. 101011 Watchpoint. 101111 HLT instruction. 110011 Software access to debug register. 110111 Exception Catch. 111011 Halting step, no syndrome. All other values of STATUS are reserved.	
		1	PE is restarting, exiting Debug state.
		2	PE is in Non-debug state.
		7	Breakpoint.
		19	External debug request.
		27	Halting step, normal.
		31	Halting step, exclusive.
		35	OS Unlock Catch.
		39	Reset Catch.
		43	Watchpoint.
		47	HLT instruction.
		51	Software access to debug register.
		55	Exception Catch.
		59	Halting step, no syndrome.

External Debug Watchpoint Address Register.

Returns the virtual data address being accessed when a watchpoint debug event was triggered.

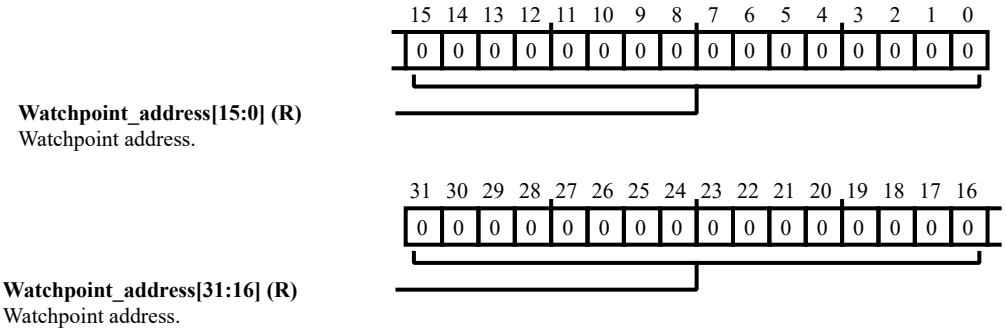


Figure 48-839: A55_C2_EDWAR_31_0 Register Diagram

Table 48-847: A55_C2_EDWAR_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	WATCHPOINT_ADDRESS	Watchpoint address.. Watchpoint address. The data virtual address being accessed when a watchpoint debug event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the DC ZVA block size. The value of this register is UNKNOWN if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event. The value of EDWAR[63:32] is UNKNOWN if Debug state was entered for a Watchpoint debug event taken from AArch32 state. The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See.

External Debug Watchpoint Address Register.

Returns the virtual data address being accessed when a watchpoint debug event was triggered.

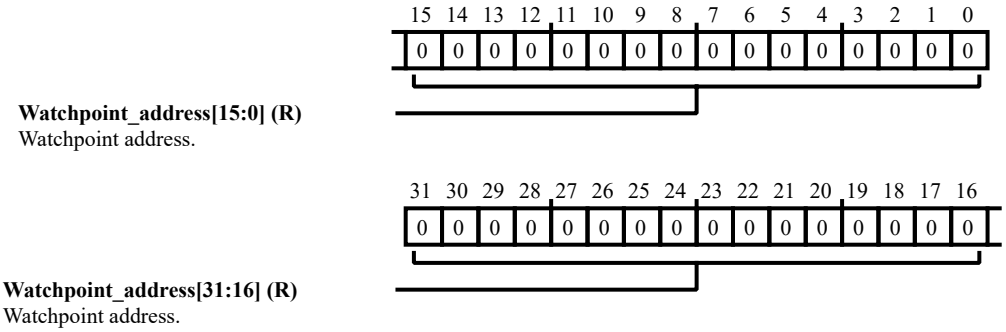


Figure 48-840: A55_C2_EDWAR_63_32 Register Diagram

Table 48-848: A55_C2_EDWAR_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	WATCHPOINT_ADDRESS	Watchpoint address.. Watchpoint address. The data virtual address being accessed when a watchpoint debug event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the DC ZVA block size. The value of this register is UNKNOWN if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event. The value of EDWAR[63:32] is UNKNOWN if Debug state was entered for a Watchpoint debug event taken from AArch32 state. The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See.

OS Lock Access Register.

Used to lock or unlock the OS lock.

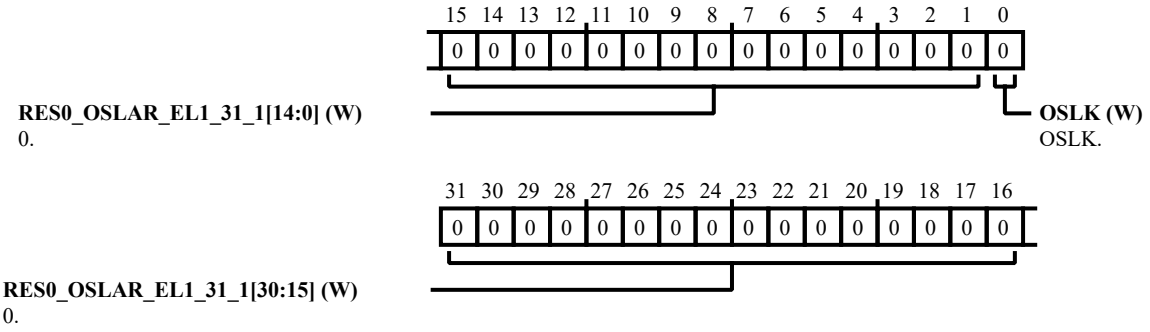


Figure 48-841: A55_C2_OSLAR_EL1 Register Diagram

Table 48-849: A55_C2_OSLAR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:1 (RX/W)	RES0_OSLAR_EL1_31_1	0.. Reserved, RES0.
0 (RX/W)	OSLK	OSLK.. On writes to OSLAR_EL1, bit[0] is copied to the OS lock. Use EDPRSR.OSLK to check the current status of the lock.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

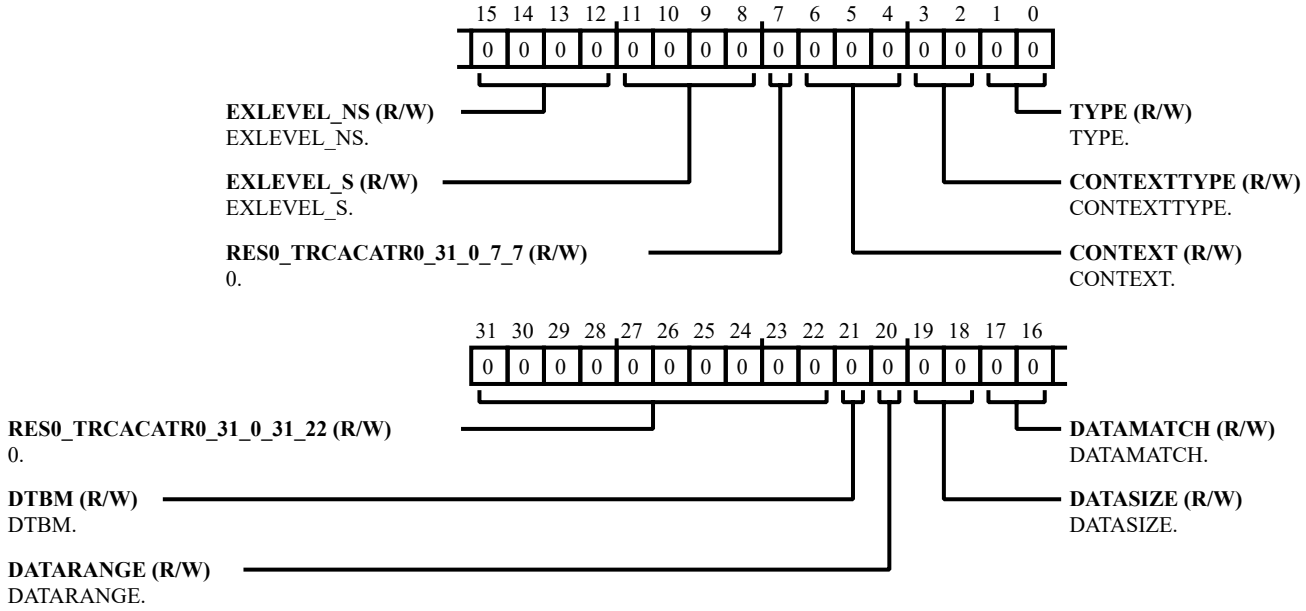


Figure 48-842: A55_C2_TRCACATR0_31_0 Register Diagram

Table 48-850: A55_C2_TRCACATR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR0_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-850: A55_C2_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-850: A55_C2_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR0_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-850: A55_C2_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-850: A55_C2_TRCACATR0_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

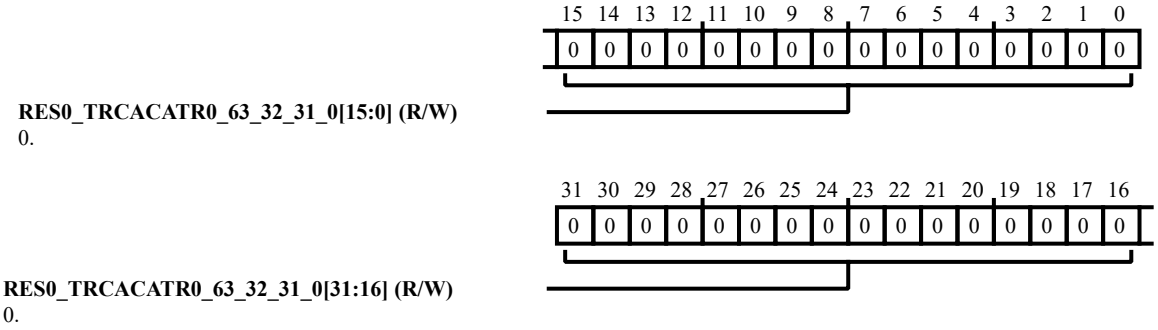


Figure 48-843: A55_C2_TRCACATR0_63_32 Register Diagram

Table 48-851: A55_C2_TRCACATR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR0_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

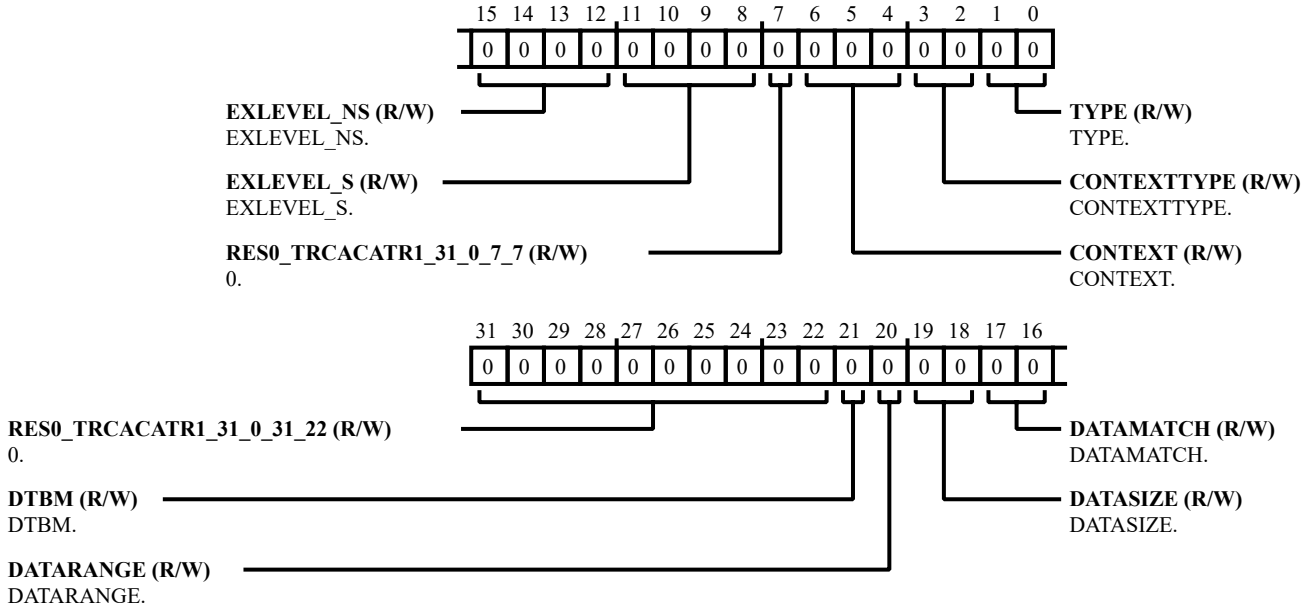


Figure 48-844: A55_C2_TRCACATR1_31_0 Register Diagram

Table 48-852: A55_C2_TRCACATR1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR1_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-852: A55_C2_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-852: A55_C2_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR1_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-852: A55_C2_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-852: A55_C2_TRCACATR1_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

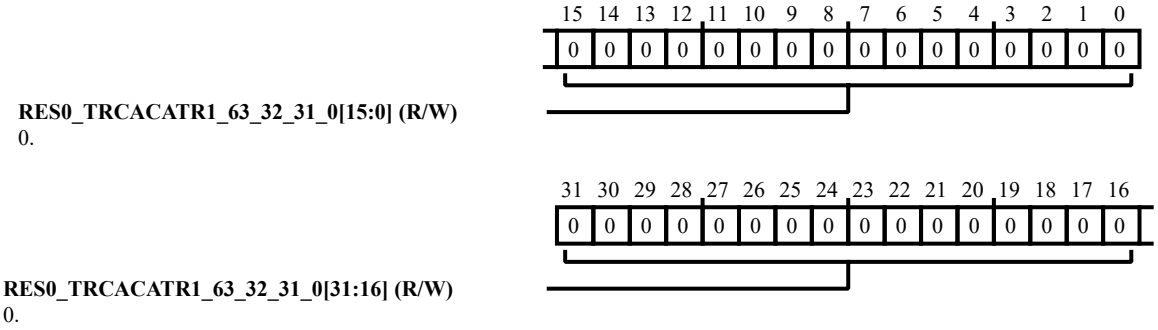


Figure 48-845: A55_C2_TRCACATR1_63_32 Register Diagram

Table 48-853: A55_C2_TRCACATR1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR1_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

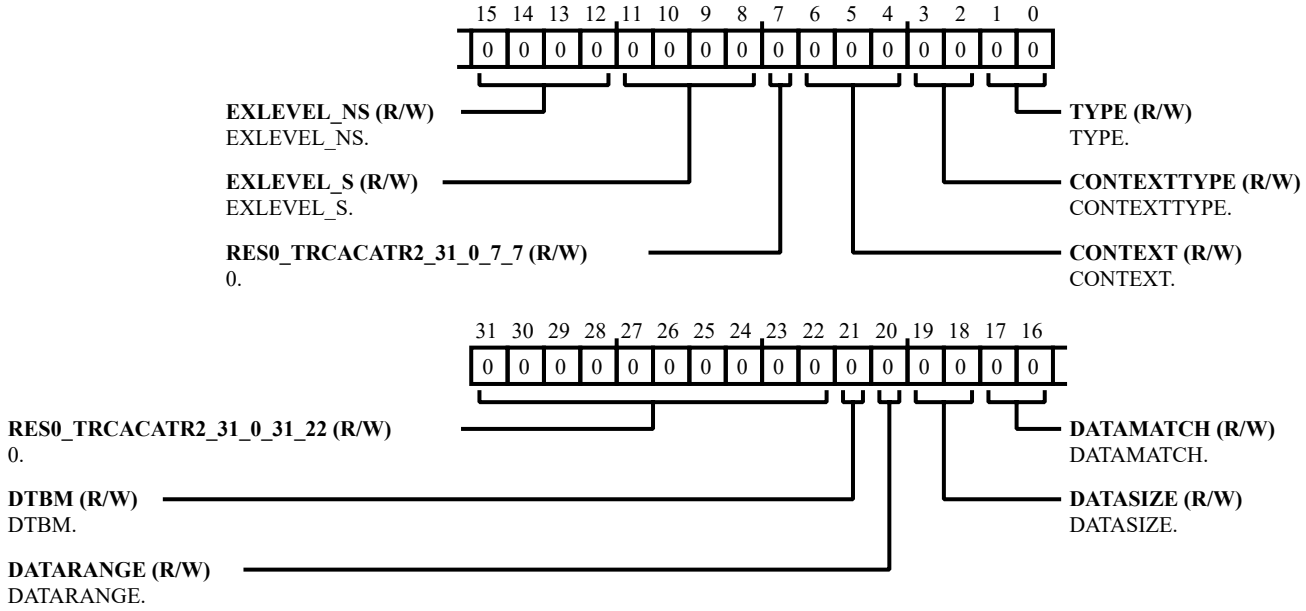


Figure 48-846: A55_C2_TRCACATR2_31_0 Register Diagram

Table 48-854: A55_C2_TRCACATR2_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR2_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-854: A55_C2_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-854: A55_C2_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR2_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-854: A55_C2_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-854: A55_C2_TRCACATR2_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

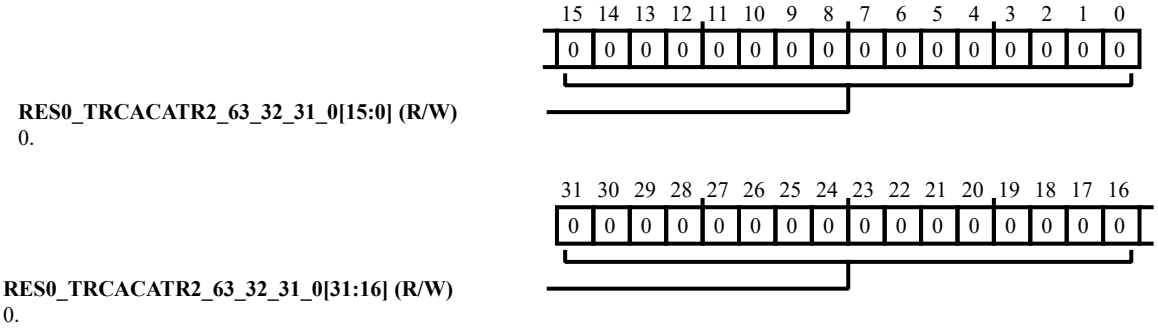


Figure 48-847: A55_C2_TRCACATR2_63_32 Register Diagram

Table 48-855: A55_C2_TRCACATR2_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR2_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

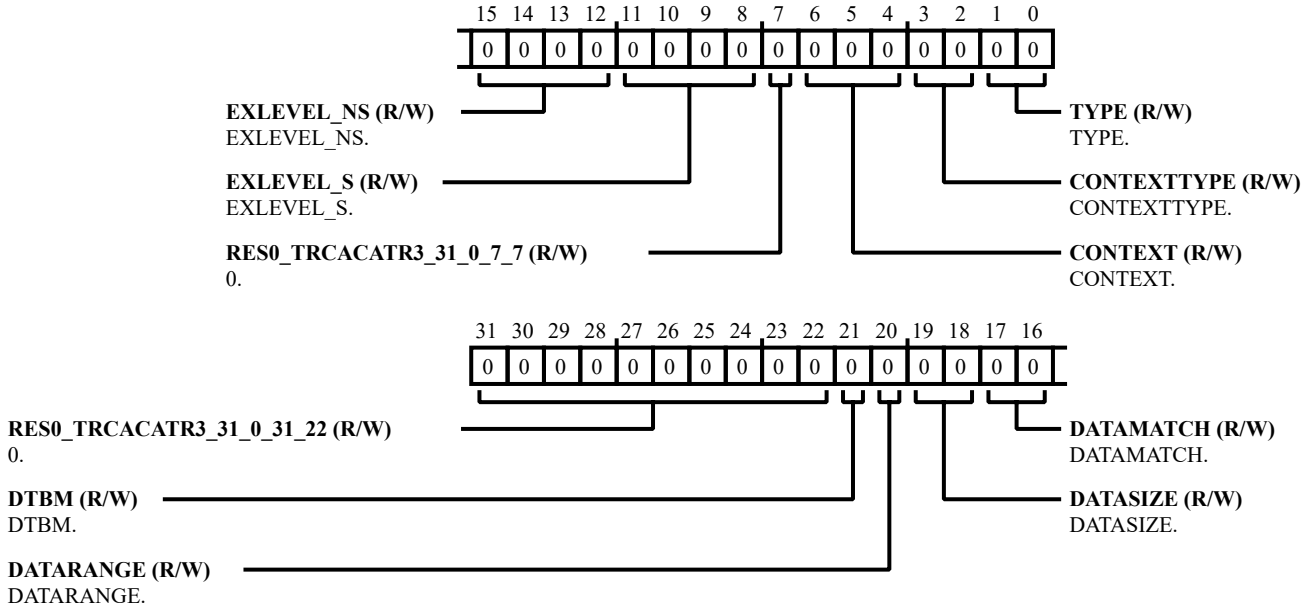


Figure 48-848: A55_C2_TRCACATR3_31_0 Register Diagram

Table 48-856: A55_C2_TRCACATR3_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR3_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-856: A55_C2_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-856: A55_C2_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR3_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-856: A55_C2_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-856: A55_C2_TRCACATR3_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

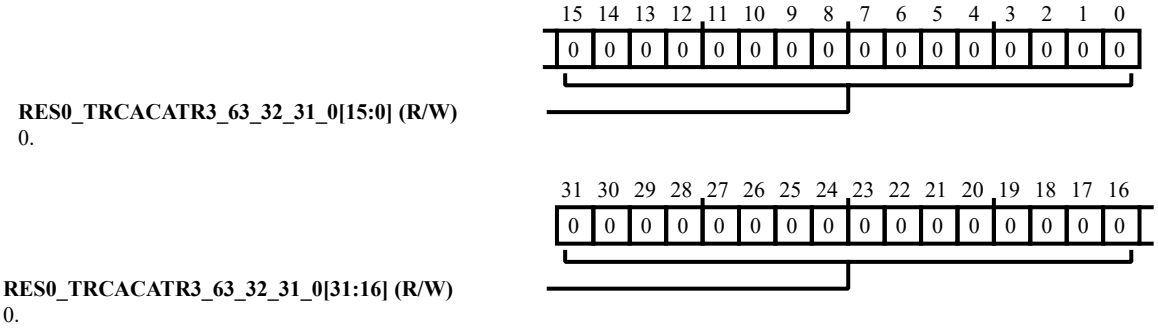


Figure 48-849: A55_C2_TRCACATR3_63_32 Register Diagram

Table 48-857: A55_C2_TRCACATR3_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR3_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

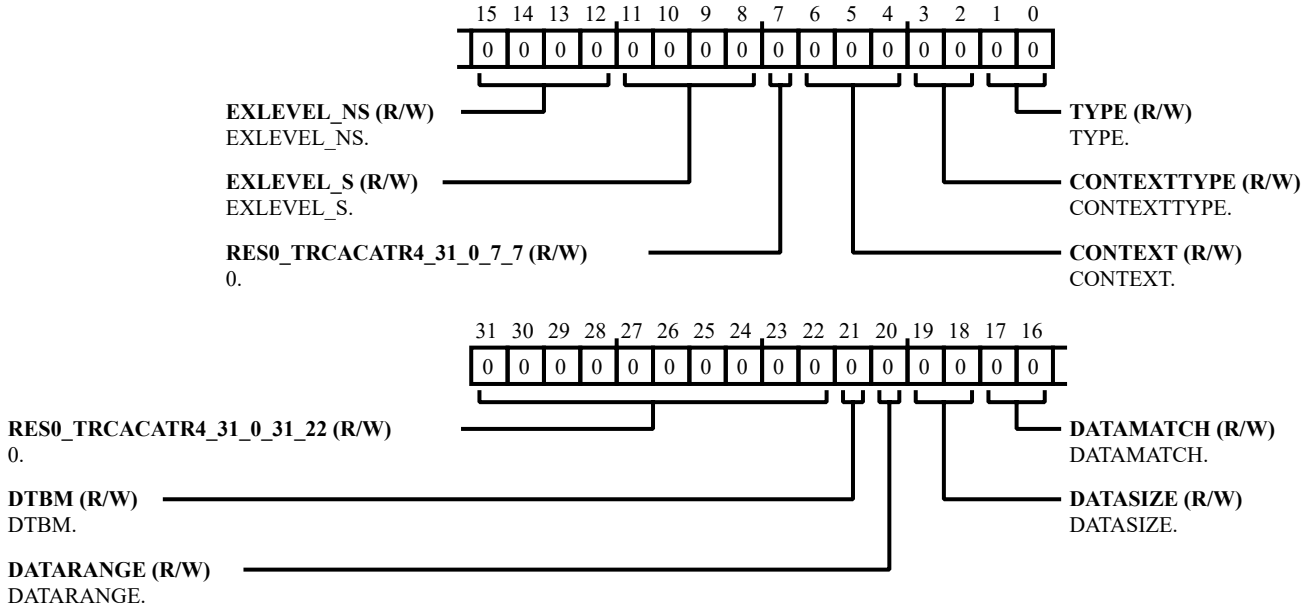


Figure 48-850: A55_C2_TRCACATR4_31_0 Register Diagram

Table 48-858: A55_C2_TRCACATR4_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR4_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-858: A55_C2_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-858: A55_C2_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR4_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-858: A55_C2_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-858: A55_C2_TRCACATR4_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

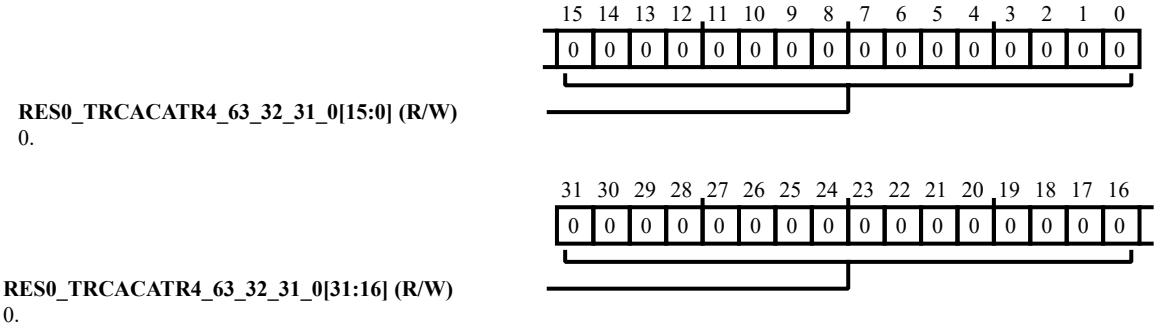


Figure 48-851: A55_C2_TRCACATR4_63_32 Register Diagram

Table 48-859: A55_C2_TRCACATR4_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR4_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

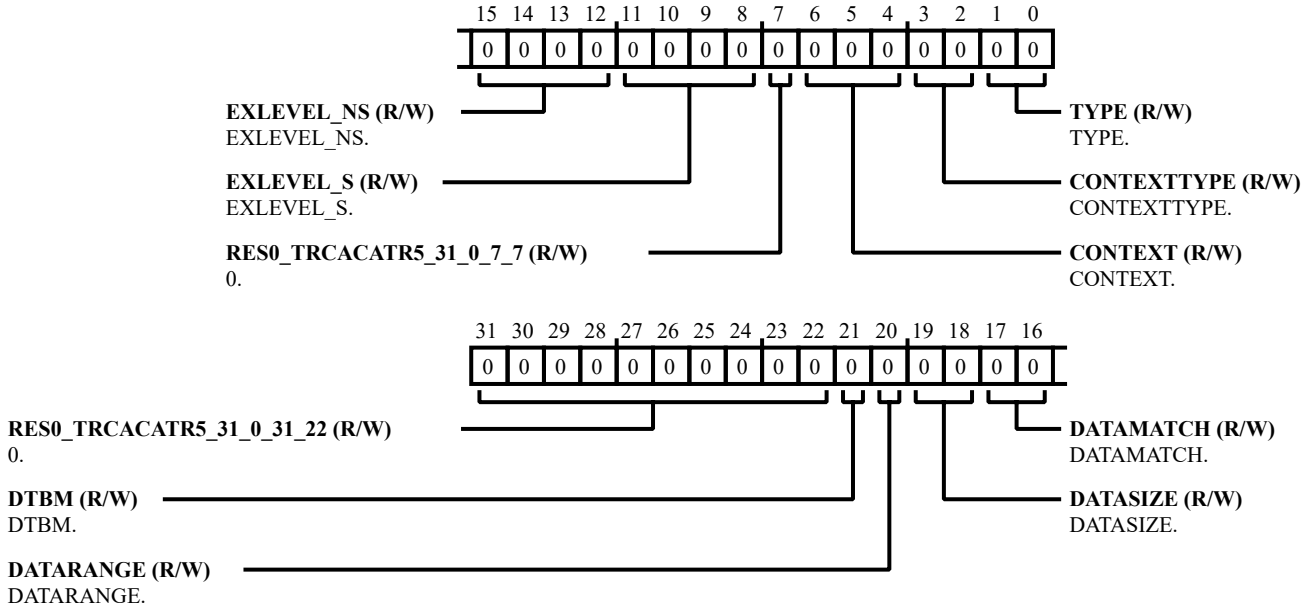


Figure 48-852: A55_C2_TRCACATR5_31_0 Register Diagram

Table 48-860: A55_C2_TRCACATR5_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR5_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-860: A55_C2_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-860: A55_C2_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR5_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-860: A55_C2_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-860: A55_C2_TRCACATR5_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

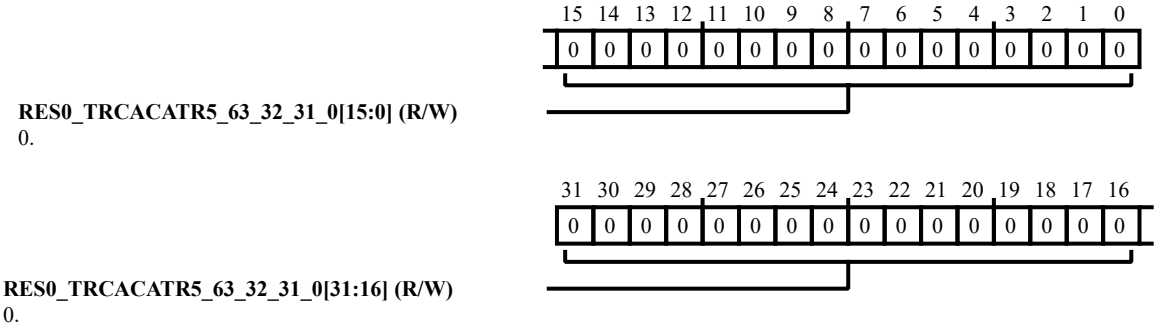


Figure 48-853: A55_C2_TRCACATR5_63_32 Register Diagram

Table 48-861: A55_C2_TRCACATR5_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR5_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

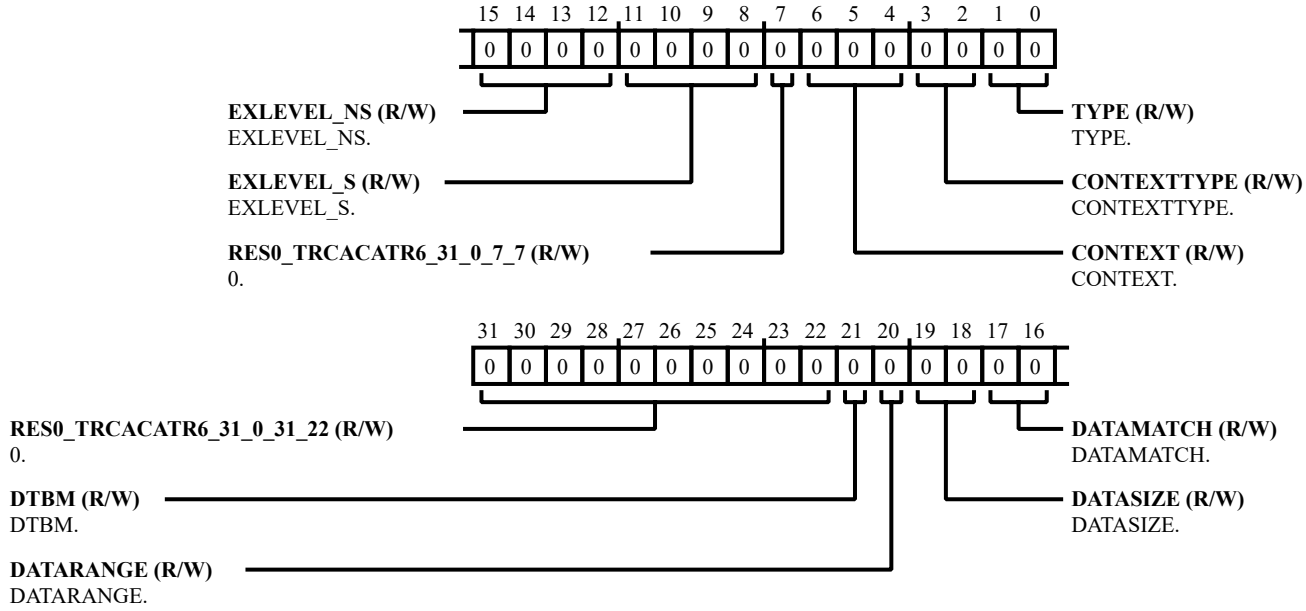


Figure 48-854: A55_C2_TRCACATR6_31_0 Register Diagram

Table 48-862: A55_C2_TRCACATR6_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR6_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-862: A55_C2_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-862: A55_C2_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR6_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-862: A55_C2_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-862: A55_C2_TRCACATR6_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

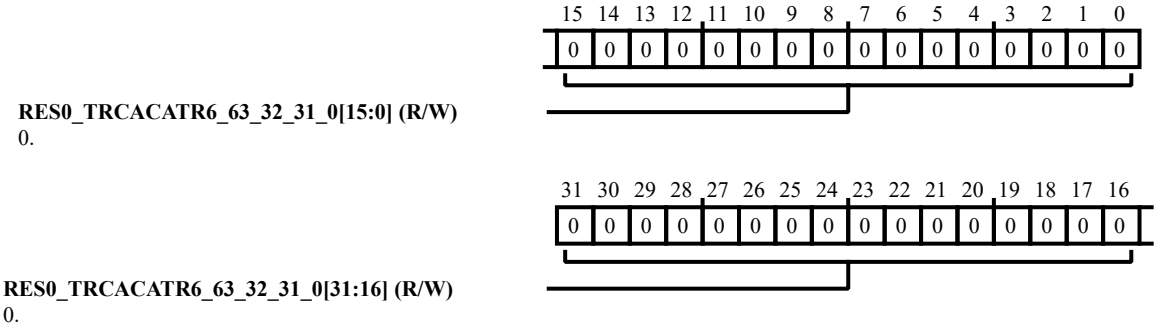


Figure 48-855: A55_C2_TRCACATR6_63_32 Register Diagram

Table 48-863: A55_C2_TRCACATR6_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR6_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

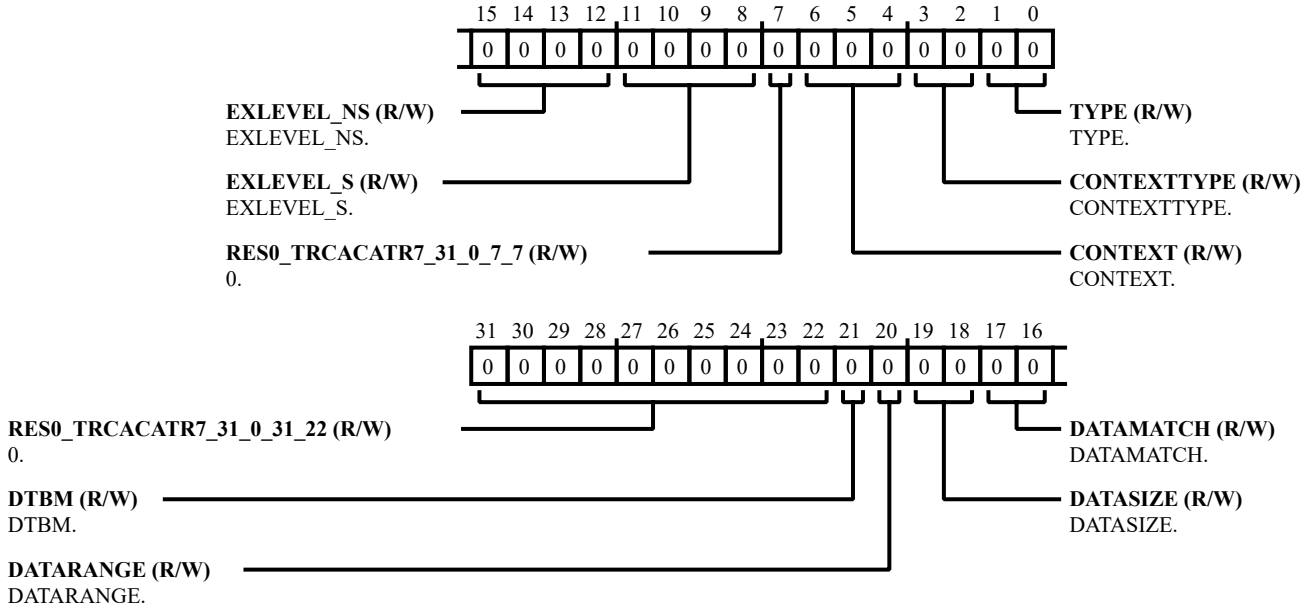


Figure 48-856: A55_C2_TRCACATR7_31_0 Register Diagram

Table 48-864: A55_C2_TRCACATR7_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCA-CATR7_31_0_31_22	0.. Reserved, RES0.	
21 (R/W)	DTBM	DTBM.. Controls whether data address comparisons use the data address [63:56] bits: 0 The trace unit ignores the data address [63:56] bits for data address comparisons. 1 The trace unit uses the data address [63:56] bits for data address comparisons. Supported only if TRCIDR2.DASIZE indicates that the data address size is 64 bits, otherwise this bit is RES0.	
		0	The trace unit ignores the data address [63:56] bits for data address comparisons.
		1	The trace unit uses the data address [63:56] bits for data address comparisons.

Table 48-864: A55_C2_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	DATARANGE	DATARANGE.. Controls whether a data value comparison uses the single address comparator or the address range comparator: 0 The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE. 1 The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE. The trace unit ignores this field when DATAMATCH==0b00.Supported only if the corresponding data value comparator is supported, otherwise this bit is RES0.	
		0	The trace unit uses the single address comparator for data value comparisons. The behavior of the address range comparator is UNPREDICTABLE.
		1	The trace unit uses the address range comparator for data value comparisons. The behavior of the single address comparators in this pair is UNPREDICTABLE.
19:18 (R/W)	DATASIZE	DATASIZE.. Controls the width of the data value comparison: 00 Byte. 01 Halfword. 10 Word. 11 Doubleword. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.The doubleword width is supported only if TRCIDR2.DVSIZE indicates that 64-bit values are supported. If 64-bit values are not supported, 0b11 is reserved.	
		0	Byte.
		1	Halfword.
		2	Word.
		3	Doubleword.
17:16 (R/W)	DATAMATCH	DATAMATCH.. Controls how the trace unit performs a data value comparison: 00 The trace unit does not perform a data value comparison. 01 The trace unit performs a data value comparison and signals a match if both values are identical. 10 Reserved. 11 The trace unit performs a data value comparison and signals a match if both values are different. Supported only if the corresponding data value comparator is supported, otherwise this field is RES0.	
		0	The trace unit does not perform a data value comparison.
		1	The trace unit performs a data value comparison and signals a match if both values are identical.
		2	Reserved.
		3	The trace unit performs a data value comparison and signals a match if both values are different.

Table 48-864: A55_C2_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Non-secure state, for exception level n. 1 The trace unit does not perform a comparison, in Non-secure state, for exception level n. The exception levels are:Bit[12]Exception level 0.Bit[13]Exception level 1.Bit[14]Exception level 2.Bit[15]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Non-secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Non-secure state, for exception level n.
11:8 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether a comparison can occur for the corresponding exception level: 0 The trace unit can perform a comparison, in Secure state, for exception level n. 1 The trace unit does not perform a comparison, in Secure state, for exception level n. The exception levels are:Bit[8]Exception level 0.Bit[9]Exception level 1.Bit[10]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[11]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. Unimplemented bits are RAZ/WI.	
		0	The trace unit can perform a comparison, in Secure state, for exception level n.
		1	The trace unit does not perform a comparison, in Secure state, for exception level n.
7 (R/W)	RES0_TRCACATR7_31_0_7_7	0.. Reserved, RES0.	
6:4 (R/W)	CONTEXT	CONTEXT.. If TRCIDR4.NUMCIDFC > 0 or TRCIDR4.NUMVMIDC > 0, selects a Context ID comparator or VMID comparator: 000 Comparator 0. 001 Comparator 1. 010 Comparator 2. and so on up to 0b111, which indicates comparator 7.The implemented width of this field is determined by the size of whichever of TRCIDR4.NUMVMIDC and TRCIDR4.NUMCIDC is larger. If the largest field is one bit long, then this field implements one bit, bit[4]. If the largest field is four bits long, then this field implements two bits, bits[5:4]. Unimplemented bits within the field are RAZ/WI.If TRCIDR4.NUMCIDFC==0 and TRCIDR4.NUMVMIDC==0, this field is RES0.	
		0	Comparator 0.
		1	Comparator 1.
		2	Comparator 2.

Table 48-864: A55_C2_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	CONTEXTTYPE	CONTEXTTYPE.. If TRCIDR4.NUMVMIDC>0 and TRCIDR4.NUMCIDC>0, this field controls whether the trace unit performs a Context ID comparison, a virtual machine identifier (VMID) comparison, or both comparisons: 00 The trace unit does not perform a Context ID or VMID comparison. 01 The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match. 10 The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match. 11 The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches. If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUMCIDC>0, bit [3] is RES0 and bit[2] controls whether the trace unit performs a Context ID comparison, as with cases 0b00 and 0b01 above.If TRCIDR4.NUMVMIDC==0 and TRCIDR4.NUM-CIDC==0, both bits are RES0.	
		0	The trace unit does not perform a Context ID or VMID comparison.
		1	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
		2	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
		3	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Table 48-864: A55_C2_TRCACATR7_31_0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	TYPE	TYPE.. Controls what type of comparison the trace unit performs: 00 Instruction address. 01 Data load address. 10 Data store address. 11 Data load address or data store address. If TRCIDR4.SUPPDAC does not indicate that data address comparisons are implemented, then this field is RES0. This means that any comparison performed by this address comparator is an instruction address comparison.	
		0	Instruction address.
		1	Data load address.
		2	Data store address.
		3	Data load address or data store address.

Trace Address Comparator Access Type Registers.

Defines the type of access for the corresponding TRCACVRn register. This register configures the context type, exception levels, alignment and masking that is applied by the address comparator, as well as how the address comparator behaves when it is acting as part of an address range comparator.

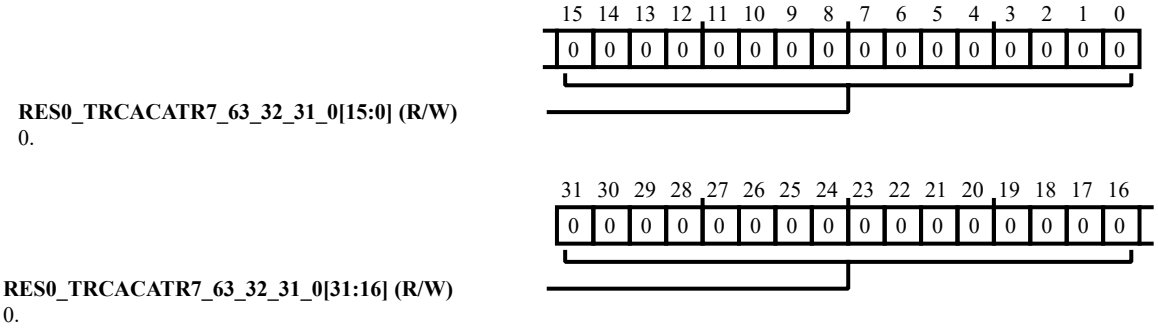


Figure 48-857: A55_C2_TRCACATR7_63_32 Register Diagram

Table 48-865: A55_C2_TRCACATR7_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RES0_TRCA-CATR7_63_32_31_0	0.. Reserved, RES0.

Trace Address Comparator Value Registers.

Contains an address value.

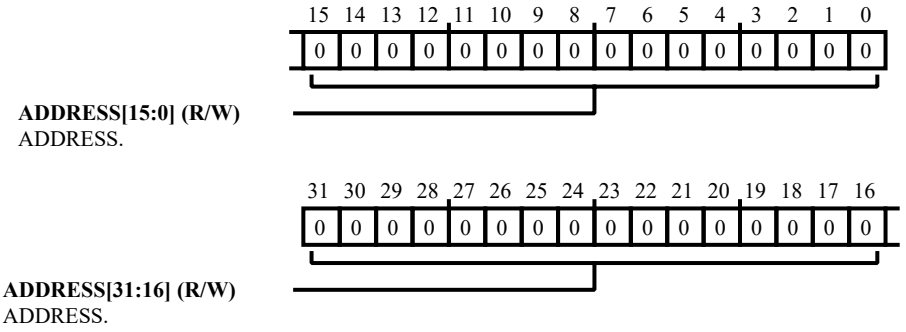


Figure 48-858: A55_C2_TRCACVR0_31_0 Register Diagram

Table 48-866: A55_C2_TRCACVR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

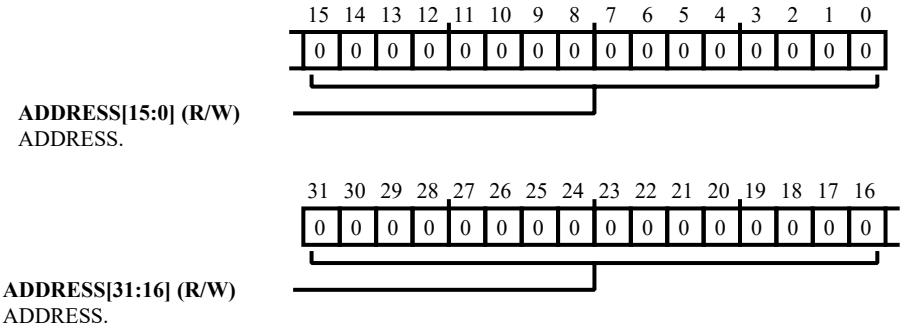


Figure 48-859: A55_C2_TRCACVR0_63_32 Register Diagram

Table 48-867: A55_C2_TRCACVR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

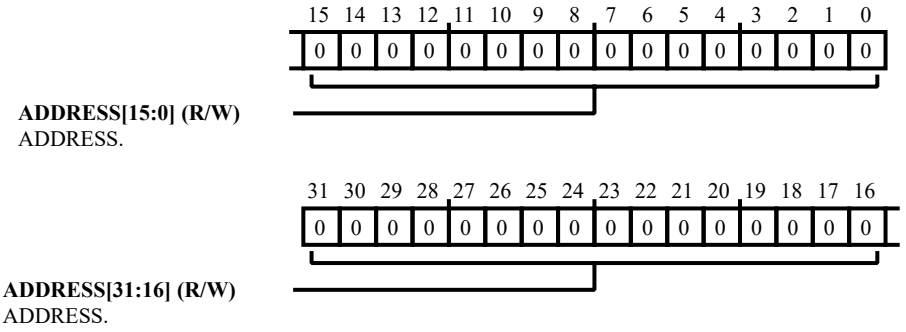


Figure 48-860: A55_C2_TRCACVR1_31_0 Register Diagram

Table 48-868: A55_C2_TRCACVR1_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

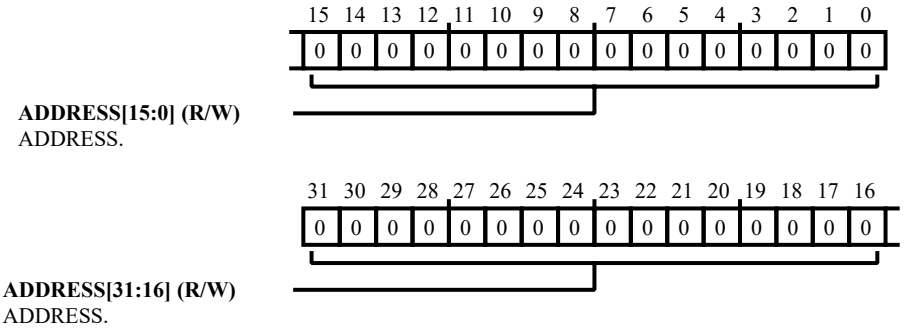


Figure 48-861: A55_C2_TRCACVR1_63_32 Register Diagram

Table 48-869: A55_C2_TRCACVR1_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

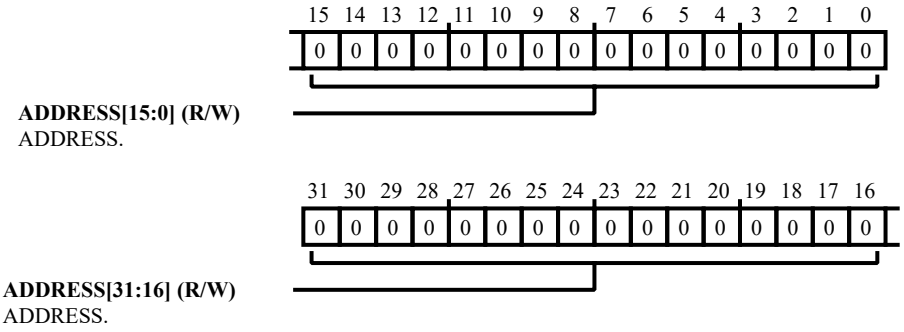


Figure 48-862: A55_C2_TRCACVR2_31_0 Register Diagram

Table 48-870: A55_C2_TRCACVR2_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

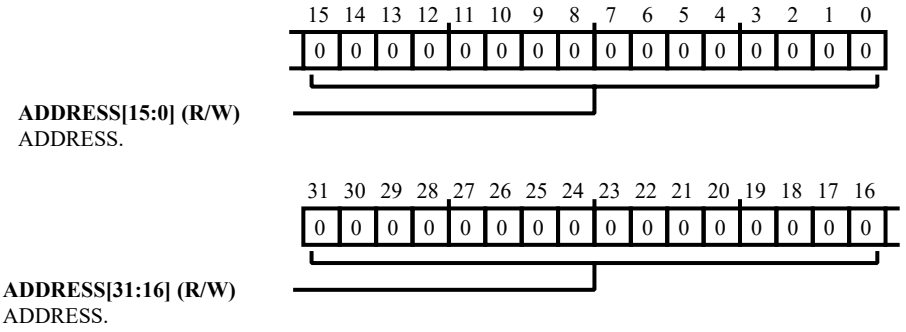


Figure 48-863: A55_C2_TRCACVR2_63_32 Register Diagram

Table 48-871: A55_C2_TRCACVR2_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

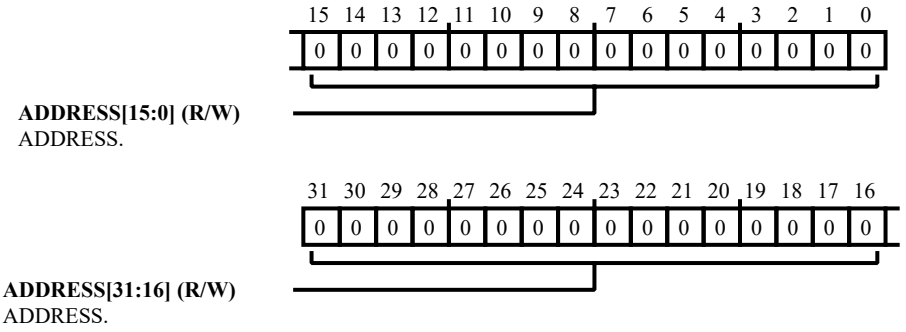


Figure 48-864: A55_C2_TRCACVR3_31_0 Register Diagram

Table 48-872: A55_C2_TRCACVR3_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

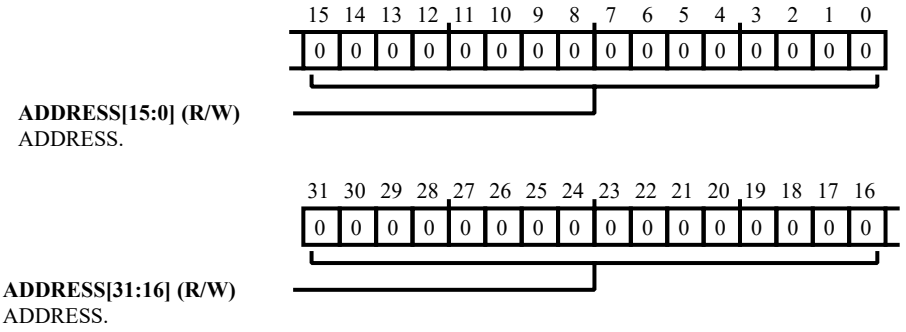


Figure 48-865: A55_C2_TRCACVR3_63_32 Register Diagram

Table 48-873: A55_C2_TRCACVR3_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

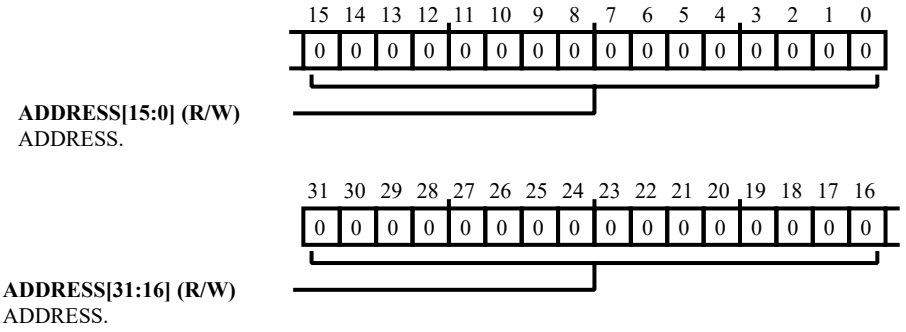


Figure 48-866: A55_C2_TRCACVR4_31_0 Register Diagram

Table 48-874: A55_C2_TRCACVR4_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

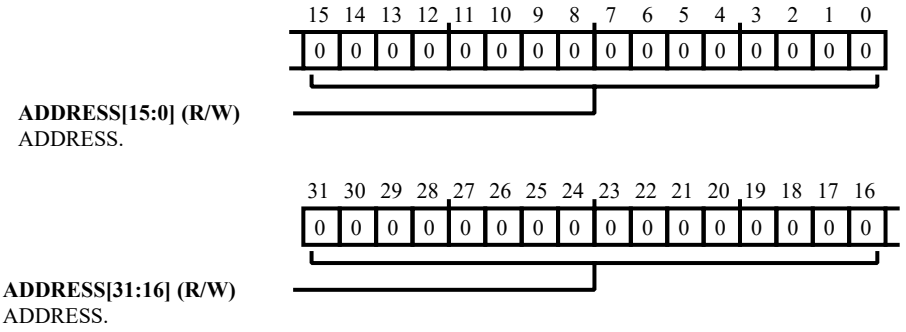


Figure 48-867: A55_C2_TRCACVR4_63_32 Register Diagram

Table 48-875: A55_C2_TRCACVR4_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

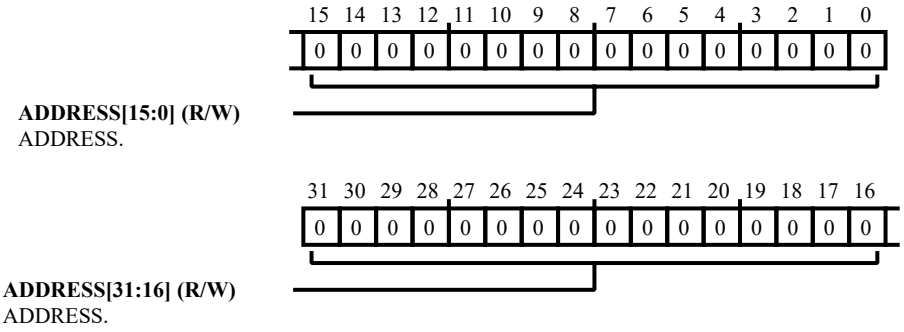


Figure 48-868: A55_C2_TRCACVR5_31_0 Register Diagram

Table 48-876: A55_C2_TRCACVR5_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

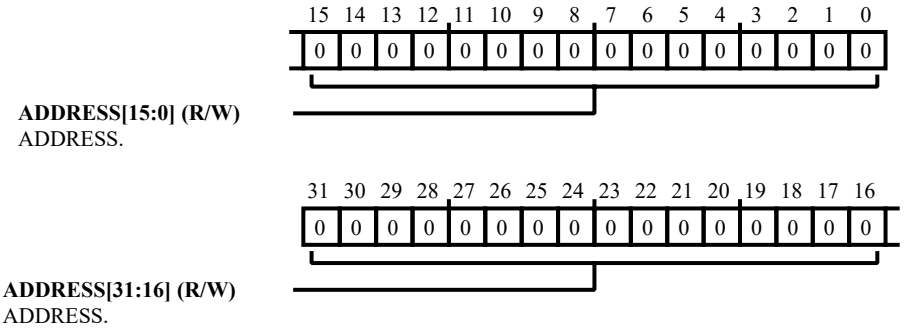


Figure 48-869: A55_C2_TRCACVR5_63_32 Register Diagram

Table 48-877: A55_C2_TRCACVR5_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

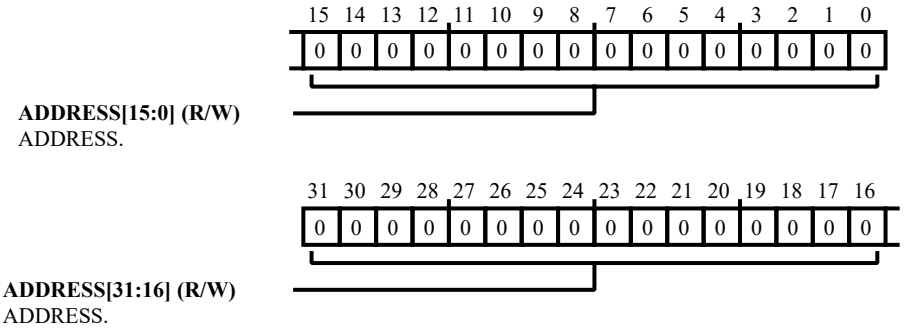


Figure 48-870: A55_C2_TRCACVR6_31_0 Register Diagram

Table 48-878: A55_C2_TRCACVR6_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

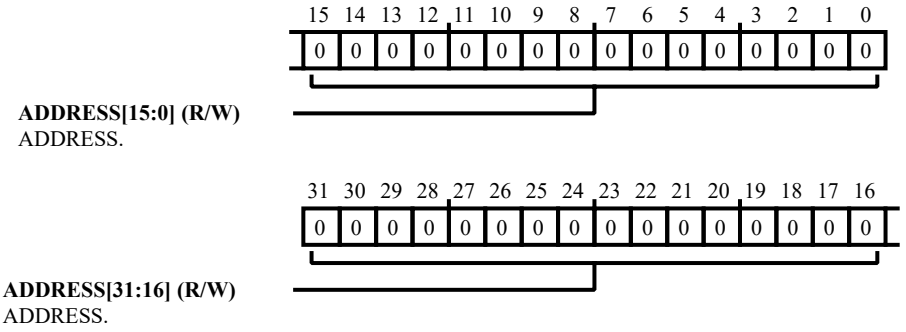


Figure 48-871: A55_C2_TRCACVR6_63_32 Register Diagram

Table 48-879: A55_C2_TRCACVR6_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

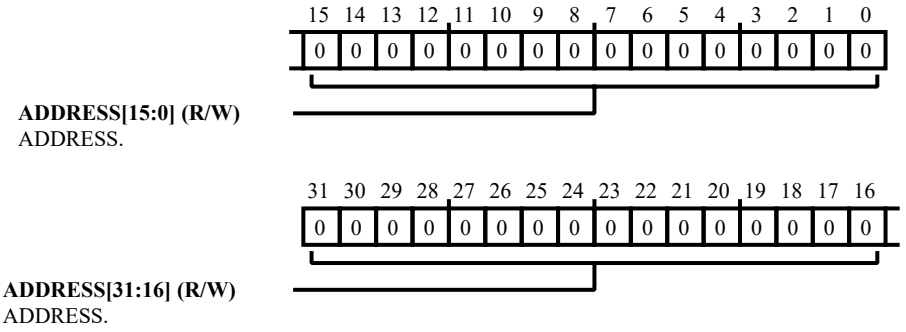


Figure 48-872: A55_C2_TRCACVR7_31_0 Register Diagram

Table 48-880: A55_C2_TRCACVR7_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Address Comparator Value Registers.

Contains an address value.

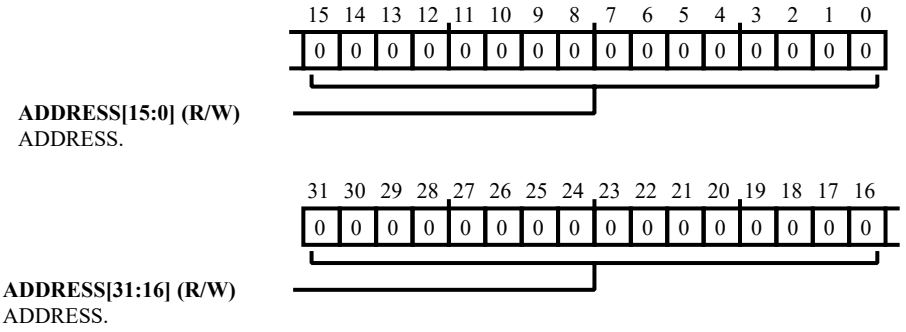


Figure 48-873: A55_C2_TRCACVR7_63_32 Register Diagram

Table 48-881: A55_C2_TRCACVR7_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRESS	ADDRESS.. Address value.The address comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then it must ignore those upper bits in the comparison. For example, in a system that supports both 32-bit and 64-bit addresses, when the processor is in 32-bit state the comparator must ignore the ADDRESS[63:32] bits.

Trace Auxiliary Control Register.

The function of this register is IMPLEMENTATION DEFINED.

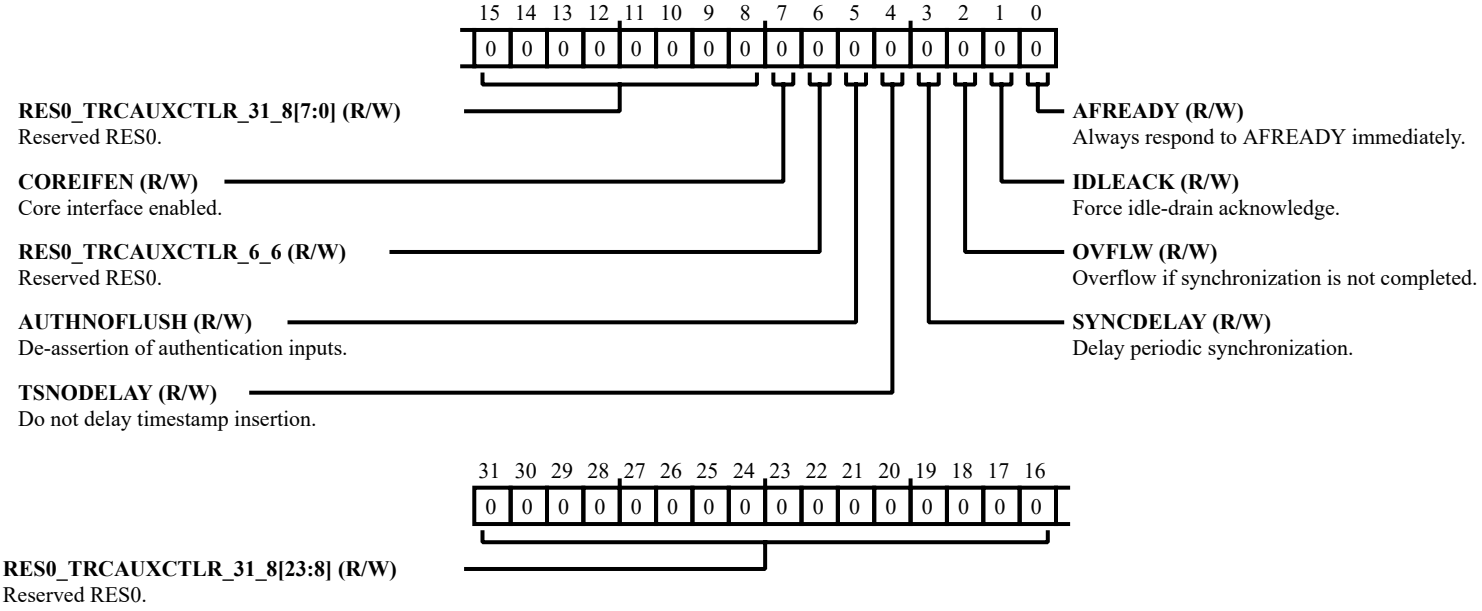


Figure 48-874: A55_C2_TRCAUXCTLR Register Diagram

Table 48-882: A55_C2_TRCAUXCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	RES0_TRCAUXCTLR_31_8	Reserved RES0.. Reserved RES0.	
		0	Reserved.
7 (R/W)	COREIFEN	Core interface enabled.. Keep core interface enabled regardless of trace enable register state.	
		0	Core interface enabled is set by trace enable register state.
		1	Enable core interface regardless of trace enable register state.
6 (R/W)	RES0_TRCAUXCTLR_6_6	Reserved RES0.. Reserved RES0.	
		0	Reserved.

Table 48-882: A55_C2_TRCAUXCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	AUTHNOFLUSH	De-assertion of authentication inputs..	
		Do not flush trace on de-assertion of authentication inputs. When this bit is set to 1 the trace unit behavior deviates from architecturally-specified behavior.	
		0	ETM trace unit FIFO is flushed and ETM trace unit enters idle state when DBGEN or NIDEN is LOW.
		1	ETM trace unit FIFO is not flushed and ETM trace unit does not enter idle state when DBGEN or NIDEN is LOW.
4 (R/W)	TSNODELAY	Do not delay timestamp insertion..	
		Do not delay timestamp insertion based on FIFO depth.	
		0	Timestamp packets are inserted into FIFO only when trace activity is LOW.
		1	Timestamp packets are inserted into FIFO irrespective of trace activity.
3 (R/W)	SYNCDELAY	Delay periodic synchronization..	
		Delay periodic synchronization if FIFO is more than half-full.	
		0	ETM trace unit AFREADYM output is asserted only when the ETM trace unit is in idle state or when all the trace bytes in FIFO before a flush request are output.
		1	ETM trace unit AFREADYM output is always asserted HIGH. When this bit is set to 1 trace unit behavior deviates from architecturally-specified behavior.
2 (R/W)	OVFLW	Overflow if synchronization is not completed..	
		Force an overflow if synchronization is not completed when second synchronization becomes due. When this bit is set to 1 the trace unit behavior deviates from architecturally-specified behavior.	
		0	ETM trace unit idle acknowledge is asserted only when the ETM trace unit is in idle state.
		1	ETM trace unit idle acknowledge is asserted irrespective of the ETM trace unit idle state.
1 (R/W)	IDLEACK	Force idle-drain acknowledge..	
		Force idle-drain acknowledge high CPU does not wait for trace to drain before entering WFX state. When this bit is set to 1 trace unit behavior deviates from architecturally-specified behavior.	
		0	No FIFO overflow when SYNC packets are delayed.
		1	Forces FIFO overflow when SYNC packets are delayed.

Table 48-882: A55_C2_TRCAUXCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	AFREADY	Always respond to AFREADY immediately.. Always respond to AFREADY immediately. Does not have any interaction with FIFO draining even in WFI state.	
		0	SYNC packets are inserted into FIFO only when trace activity is low.
		1	SYNC packets are inserted into FIFO irrespective of trace activity.

Trace Branch Broadcast Control Register.

Controls which regions in the memory map are enabled to use branch broadcasting.

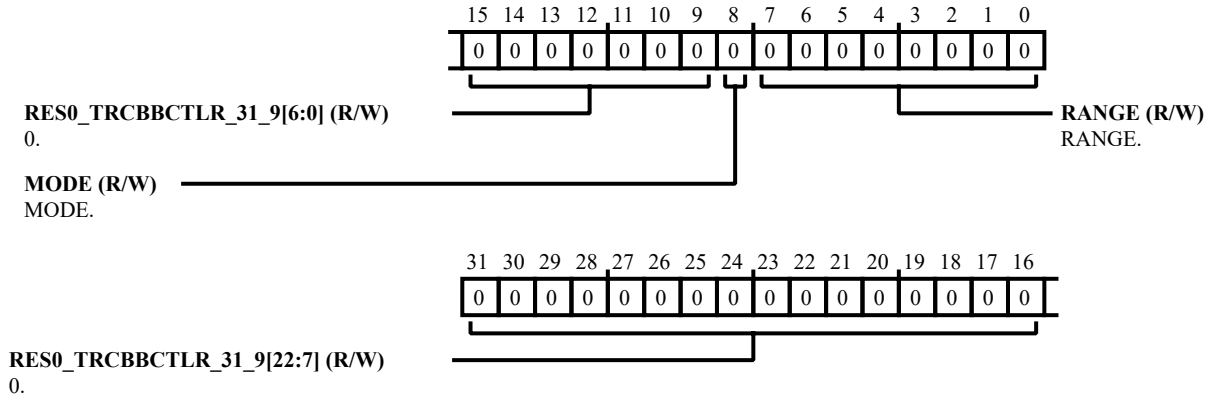


Figure 48-875: A55_C2_TRCBBCTLR Register Diagram

Table 48-883: A55_C2_TRCBBCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:9 (R/W)	RES0_TRCBBCTLR_31_9	0.. Reserved, RES0.	
8 (R/W)	MODE	MODE.. Mode bit: 0 Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines. If RANGE==0 then branch broadcasting is enabled for the entire memory map. 1 Include mode. Branch broadcasting is enabled in the address range that RANGE defines. If RANGE==0 then the branch broadcasting behavior is UNPREDICTABLE.	
		0	Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines. If RANGE==0 then branch broadcasting is enabled for the entire memory map.
		1	Include mode. Branch broadcasting is enabled in the address range that RANGE defines. If RANGE==0 then the branch broadcasting behavior is UNPREDICTABLE.

Table 48-883: A55_C2_TRCBBCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W)	RANGE	RANGE.. Address range field. Selects which address range comparator pairs are in use with branch broadcasting. Each bit represents an address range comparator pair, so bit[n] controls the selection of address range comparator pair n. If bit[n] is: 0 The address range that address range comparator pair n defines is not selected. 1 The address range that address range comparator pair n defines is selected.	
		0	The address range that address range comparator pair n defines is not selected.
		1	The address range that address range comparator pair n defines is selected.

Trace Cycle Count Control Register.

Sets the threshold value for cycle counting.

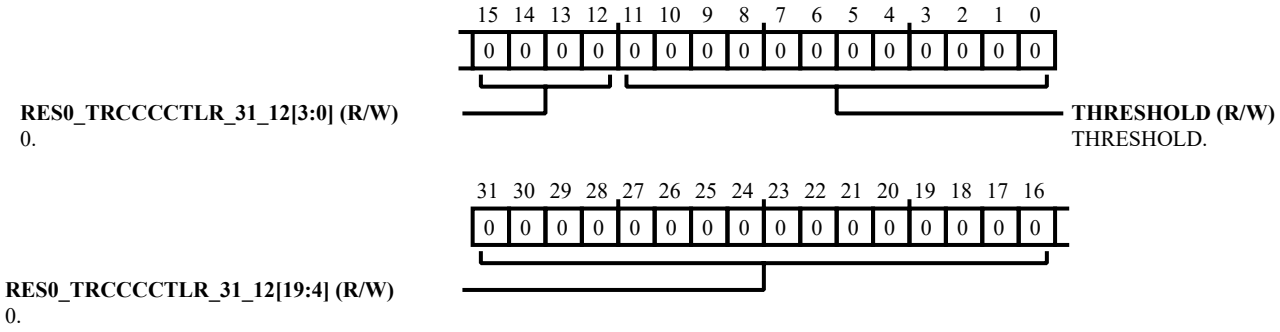


Figure 48-876: A55_C2_TRCCCCTLR Register Diagram

Table 48-884: A55_C2_TRCCCCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/W)	RES0_TRCCCCTLR_31_12	0.. Reserved, RES0.
11:0 (R/W)	THRESHOLD	THRESHOLD.. Sets the threshold value for instruction trace cycle counting.The minimum threshold value that can be programmed into THRESHOLD is given in TRCIDR3.CCITMIN.Writing a value of zero might cause UNPREDICTABLE behaviour.

Trace Context ID Comparator Control Register 0.

Contains Context ID mask values for the TRCCIDCVRn registers, where n=0-3.

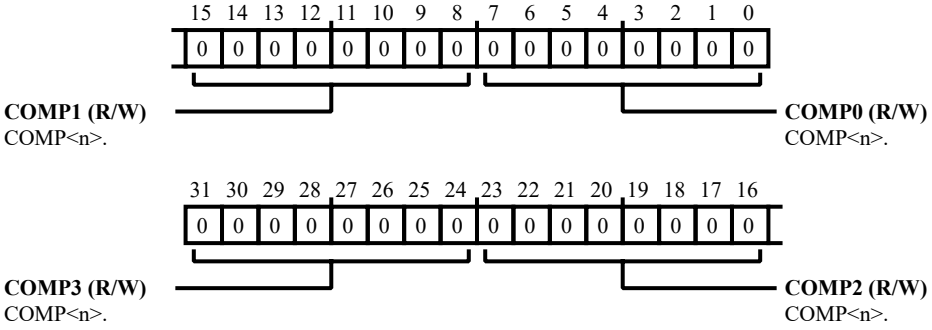


Figure 48-877: A55_C2_TRCCIDCCTLR0 Register Diagram

Table 48-885: A55_C2_TRCCIDCCTLR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	COMP3	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.

Table 48-885: A55_C2_TRCCIDCCTLR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	COMP2	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
15:8 (R/W)	COMP1	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
7:0 (R/W)	COMP0	COMP<n>.. Controls the mask value that the trace unit applies to TRCCIDCVRn. Each bit in this field corresponds to a byte in TRCCIDCVRn. When a bit is: 0 The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. 1 The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison. Supported only if TRCIDR4.NUMCIDC > n, otherwise the field is RES0.	
		0	The trace unit includes the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.
		1	The trace unit ignores the relevant byte in TRCCIDCVRn when it performs the Context ID comparison.

Trace Context ID Comparator Value Registers.

Contains a Context ID value.

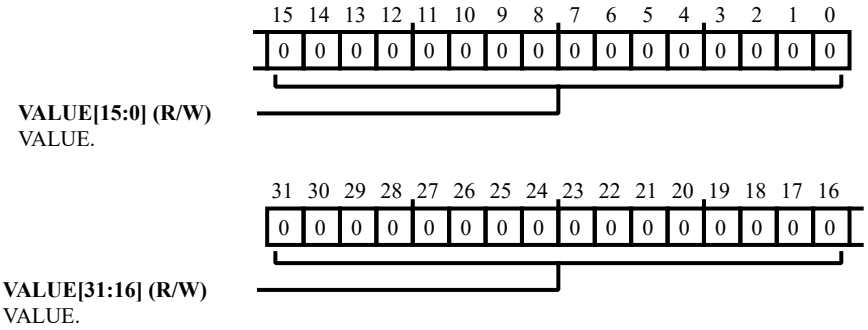


Figure 48-878: A55_C2_TRCCIDCVR0_31_0 Register Diagram

Table 48-886: A55_C2_TRCCIDCVR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. Context ID value. The implemented width of this field is IMPLEMENTATION DEFINED and is set by TRCIDR2.CIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the Context ID is zero until the processor updates the Context ID.

Trace Context ID Comparator Value Registers.

Contains a Context ID value.

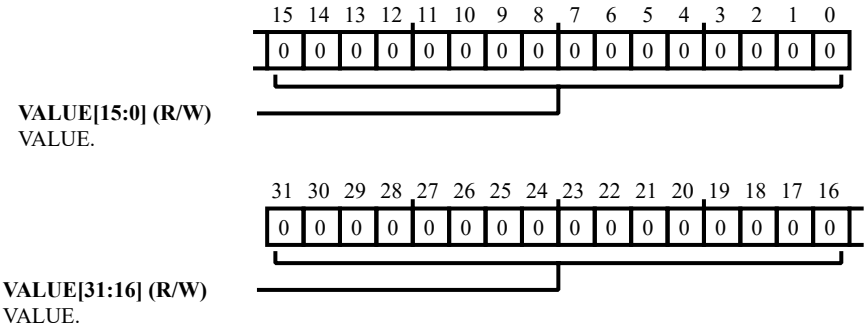


Figure 48-879: A55_C2_TRCCIDCVR0_63_32 Register Diagram

Table 48-887: A55_C2_TRCCIDCVR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. Context ID value. The implemented width of this field is IMPLEMENTATION DEFINED and is set by TRCIDR2.CIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the Context ID is zero until the processor updates the Context ID.

Trace Claim Tag Clear Register.

Software can use this to clear a claim tag bit to 0, and to read the values of the claim tag.

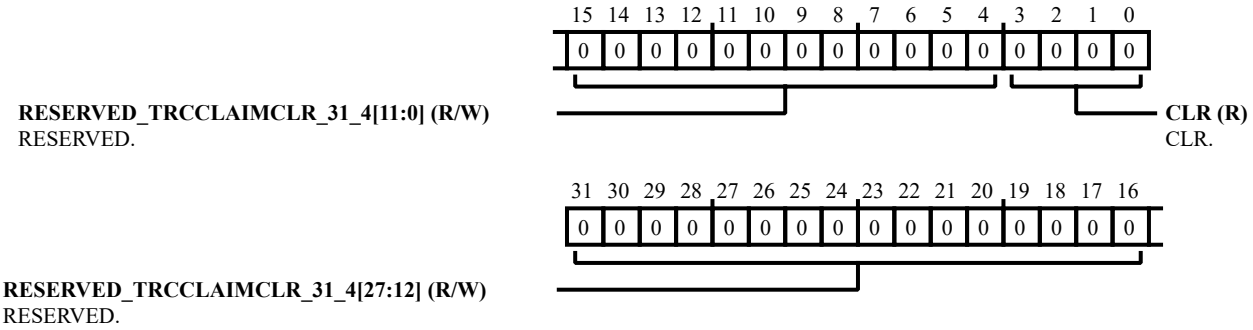


Figure 48-880: A55_C2_TRCCLAIMCLR Register Diagram

Table 48-888: A55_C2_TRCCLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_TRCCLAIMCLR_31_4	RESERVED.. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	CLR	CLR.. Clears bits in the claim tag and determines the current value of the claim tag.	
		0	On reads for each bit Claim tag bit is set.
		1	On reads for each bit Claim tag bit is not set.

Trace Claim Tag Set Register.

Software can use this to set bits in the claim tag to 1, and discover the number of bits that the claim tag supports.

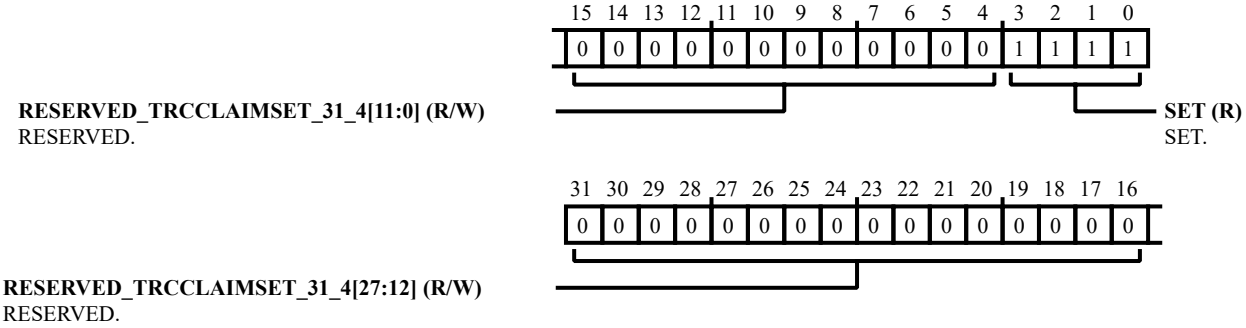


Figure 48-881: A55_C2_TRCCLAIMSET Register Diagram

Table 48-889: A55_C2_TRCCLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/W)	RESERVED_TRCCLAIM-SET_31_4	RESERVED.. Reserved RES0.	
		0	Reserved.
3:0 (R/NW)	SET	SET.. Sets bits in the claim tag and determines the number of claim tag bits implemented.	
		0	On reads for each bit Claim tag bit is implemented.
		1	On reads for each bit Claim tag bit is not implemented.

Trace Counter Control Registers.

Trace Counter Control Registers.

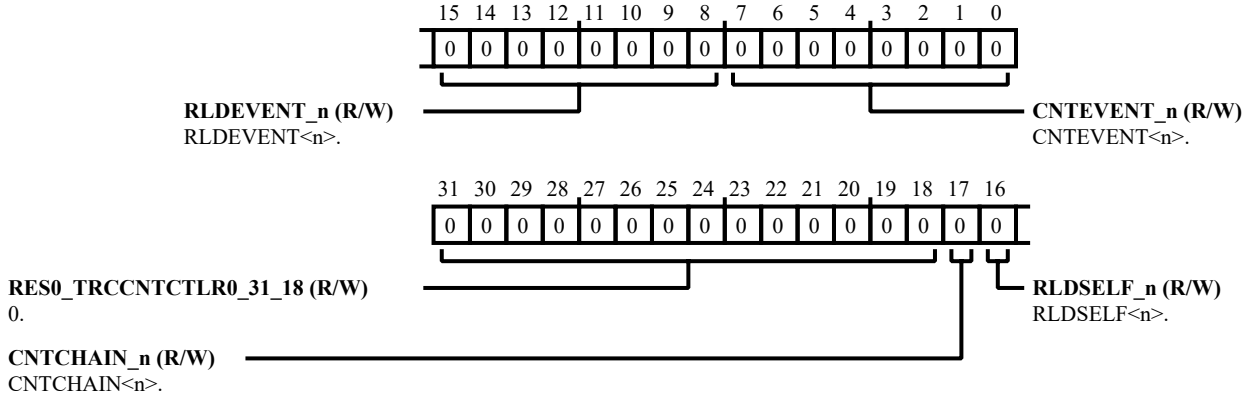


Figure 48-882: A55_C2_TRCCNTCTLR0 Register Diagram

Table 48-890: A55_C2_TRCCNTCTLR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/W)	RES0_TRCCNTCTLR0_31_18	0.. Reserved, RES0.	
17 (R/W)	CNTCHAIN_N	CNTCHAIN<n>.. For TRCCNTCTLR3 and TRCCNTCTLR1, controls whether counter <n> decrements when a reload event occurs for counter <n-1>: 0 1 For TRCCNTCTLR2 and TRCCNTCTLR0, this bit is RES0.	
		0	
		1	
16 (R/W)	RLDSELF_N	RLDSELF<n>.. Controls whether a reload event occurs for counter <n>, when counter <n> reaches zero: 0 The trace unit does not generate a reload event. 1 The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.	
		0	The trace unit does not generate a reload event.
		1	The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.
15:8 (R/W)	RLDEVENT_N	RLDEVENT<n>.. Selects an event, that when it occurs causes a reload event for counter <n>.	

Table 48-890: A55_C2_TRCCNTCTLR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	CNTEVENT_N	CNTEVENT<n>.. Selects an event, that when it occurs causes counter <n> to decrement.

Trace Counter Control Registers.

Trace Counter Control Registers.

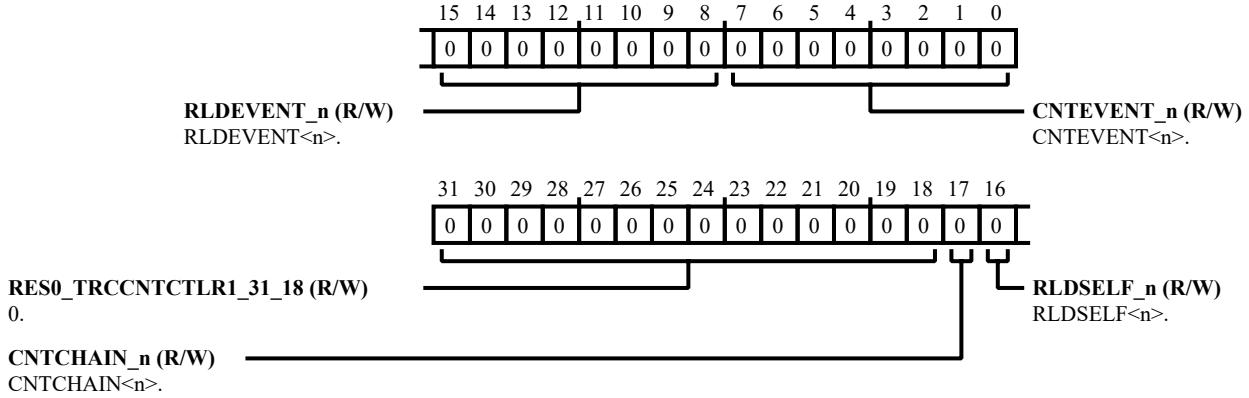


Figure 48-883: A55_C2_TRCCNTCTLR1 Register Diagram

Table 48-891: A55_C2_TRCCNTCTLR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/W)	RES0_TRCCNTCTLR1_31_18	0.. Reserved, RES0.	
17 (R/W)	CNTCHAIN_N	CNTCHAIN<n>.. For TRCCNTCTLR3 and TRCCNTCTLR1, controls whether counter <n> decrements when a reload event occurs for counter <n-1>: 0 1 For TRCCNTCTLR2 and TRCCNTCTLR0, this bit is RES0.	
		0	
		1	
16 (R/W)	RLDSELF_N	RLDSELF<n>.. Controls whether a reload event occurs for counter <n>, when counter <n> reaches zero: 0 The trace unit does not generate a reload event. 1 The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.	
		0	The trace unit does not generate a reload event.
		1	The trace unit generates a reload event for counter <n>, provided that the event resource that CNTEVENT<n> specifies is active.
15:8 (R/W)	RLDEVENT_N	RLDEVENT<n>.. Selects an event, that when it occurs causes a reload event for counter <n>.	

Table 48-891: A55_C2_TRCCNTCTLR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	CNTEVENT_N	CNTEVENT<n>.. Selects an event, that when it occurs causes counter <n> to decrement.

Trace Counter Reload Value Registers.

This sets or returns the reload count value for counter 0.

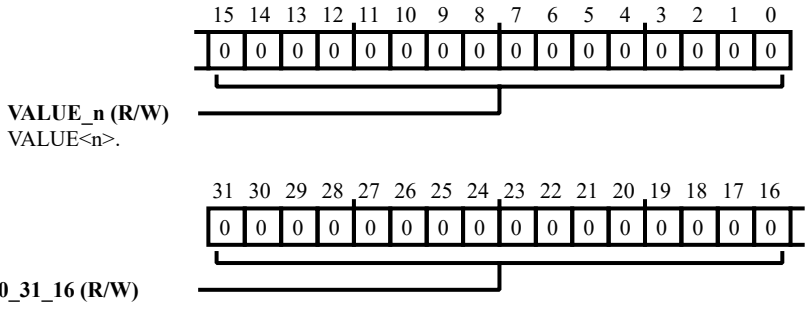


Figure 48-884: A55_C2_TRCCNTRLDVR0 Register Diagram

Table 48-892: A55_C2_TRCCNTRLDVR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTRLDVR0_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the reload value for counter <n>. When a reload event occurs for counter <n> then the trace unit copies the VALUE<n> field into counter <n>.

Trace Counter Reload Value Registers.

This sets or returns the reload count value for counter 1.

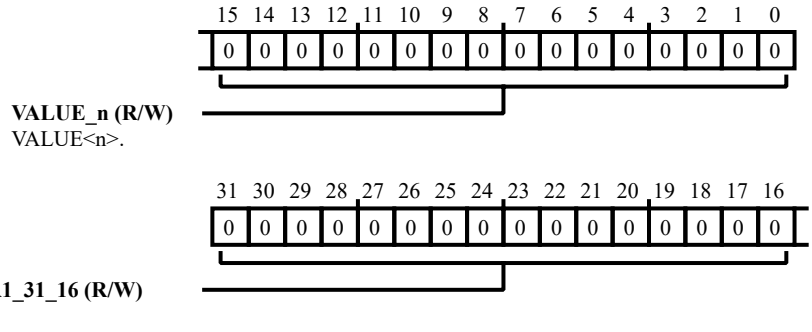


Figure 48-885: A55_C2_TRCCNTRLDVR1 Register Diagram

Table 48-893: A55_C2_TRCCNTRLDVR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTRLDVR1_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the reload value for counter <n>. When a reload event occurs for counter <n> then the trace unit copies the VALUE<n> field into counter <n>.

Trace Counter Value Registers.

This sets or returns the value of counter 0.

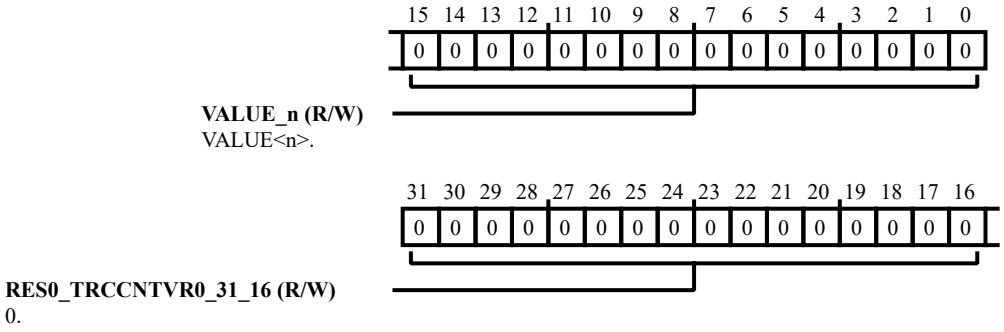


Figure 48-886: A55_C2_TRCCNTVR0 Register Diagram

Table 48-894: A55_C2_TRCCNTVR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTVR0_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the count value of counter <n>.

Trace Counter Value Registers.

This sets or returns the value of counter 1.

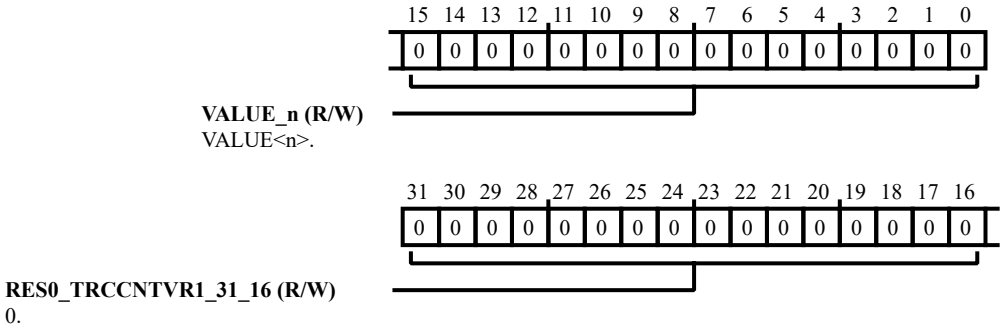


Figure 48-887: A55_C2_TRCCNTVR1 Register Diagram

Table 48-895: A55_C2_TRCCNTVR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCCNTVR1_31_16	0.. Reserved, RES0.
15:0 (R/W)	VALUE_N	VALUE<n>.. Contains the count value of counter <n>.

Trace Configuration Register.

Controls the tracing options.

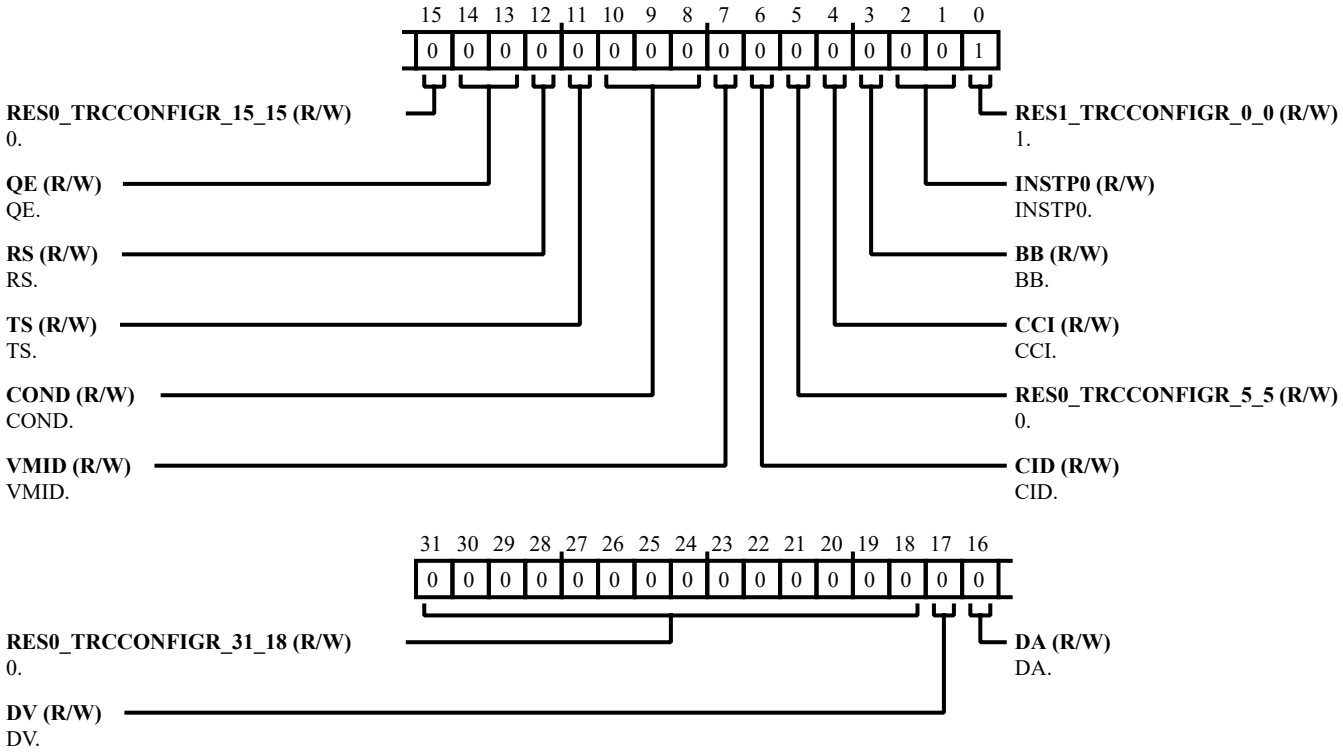


Figure 48-888: A55_C2_TRCCONFIGR Register Diagram

Table 48-896: A55_C2_TRCCONFIGR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:18 (R/W)	RES0_TRCCON-FIGR_31_18	0.. Reserved, RES0.	
17 (R/W)	DV	DV.. Data value tracing bit: 0 Data value tracing is disabled. 1 Data value tracing is enabled when INSTP0 is not 0b00. TRCIDR0.TRCDATA indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Data value tracing is disabled.
		1	Data value tracing is enabled when INSTP0 is not 0b00.

Table 48-896: A55_C2_TRCCONFIGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	DA	DA.. Data address tracing bit: 0 Data address tracing is disabled. 1 Data address tracing is enabled when INSTP0 is not 0b00. TRCIDR0.TRCDATA indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Data address tracing is disabled.
		1	Data address tracing is enabled when INSTP0 is not 0b00.
15 (R/W)	RES0_TRCCON-FIGR_15_15	0.. Reserved, RES0.	
14:13 (R/W)	QE	QE.. Q element enable field: 00 Q elements are disabled. 01 Q elements with instruction counts are enabled. Q elements without instruction counts are disabled. 11 Q elements with and without instruction counts are enabled. The value 0b10 is reserved.TRCIDR0.QSUPP indicates which values of this field are implemented.TRCCONFIGR.QE must be set to 0b00 if any of the following are true:TRCCONFIGR.INSTP0 is not 0b00.TRCCONFIGR.COND is not 0b000.TRCCONFIGR.BB is not 0.	
		0	Q elements are disabled.
		1	Q elements with instruction counts are enabled. Q elements without instruction counts are disabled.
		3	Q elements with and without instruction counts are enabled.
12 (R/W)	RS	RS.. Return stack enable bit. 0 Return stack is disabled. 1 Return stack is enabled. TRCIDR0.RETSTACK indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Return stack is disabled.
		1	Return stack is enabled.
11 (R/W)	TS	TS.. Global timestamp tracing bit: 0 Global timestamp tracing is disabled. 1 Global timestamp tracing is enabled. TRCTSCTLR controls the insertion of timestamps in the trace. TRCIDR0.TSSIZE indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Global timestamp tracing is disabled.
		1	Global timestamp tracing is enabled. TRCTSCTLR controls the insertion of timestamps in the trace.

Table 48-896: A55_C2_TRCCONFIGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:8 (R/W)	COND	COND.. Conditional instruction tracing bit. The permitted values are: 000 Conditional instruction tracing is disabled. 001 Conditional load instructions are traced. 010 Conditional store instructions are traced. 011 Conditional load and store instructions are traced. 111 All conditional instructions are traced. All other values are reserved.TRCIDR0.TRCCOND indicates whether this field is supported. If it is not supported then this field is RES0.	
		0	Conditional instruction tracing is disabled.
		1	Conditional load instructions are traced.
		2	Conditional store instructions are traced.
		3	Conditional load and store instructions are traced.
		7	All conditional instructions are traced.
7 (R/W)	VMID	VMID.. VMID tracing bit: 0 VMID tracing is disabled. 1 VMID tracing is enabled. TRCIDR2.VMIDSIZE indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	VMID tracing is disabled.
		1	VMID tracing is enabled.
6 (R/W)	CID	CID.. Context ID tracing bit: 0 Context ID tracing is disabled. 1 Context ID tracing is enabled. TRCIDR2.CIDSIZE indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Context ID tracing is disabled.
		1	Context ID tracing is enabled.
5 (R/W)	RES0_TRCCONFIGR_5_5	0.. Reserved, RES0.	
4 (R/W)	CCI	CCI.. Cycle counting instruction trace bit: 0 Cycle counting in the instruction trace is disabled. 1 Cycle counting in the instruction trace is enabled. TRCCCCTLR controls the threshold value for cycle counting. TRCIDR0.TRCCCI indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Cycle counting in the instruction trace is disabled.
		1	Cycle counting in the instruction trace is enabled. TRCCCCTLR controls the threshold value for cycle counting.

Table 48-896: A55_C2_TRCCONFIGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	BB	BB.. Branch broadcast mode bit: 0 Branch broadcast mode is disabled. 1 Branch broadcast mode is enabled. TRCBBCTLR controls which regions of memory are enabled to use branch broadcasting. TRCIDR0.TRCBB indicates whether this bit is supported. If it is not supported then this bit is RES0.	
		0	Branch broadcast mode is disabled.
		1	Branch broadcast mode is enabled. TRCBBCTLR controls which regions of memory are enabled to use branch broadcasting.
2:1 (R/W)	INSTP0	INSTP0.. Instruction P0 bit. This field controls whether load and store instructions are traced as P0 instructions: 00 Do not trace load and store instructions as P0 instructions. 01 Trace load instructions as P0 instructions. 10 Trace store instructions as P0 instructions. 11 Trace load and store instructions as P0 instructions. TRCIDR0.INSTP0 indicates whether this field is supported. If it is not supported then this field is RES0.	
		0	Do not trace load and store instructions as P0 instructions.
		1	Trace load instructions as P0 instructions.
		2	Trace store instructions as P0 instructions.
		3	Trace load and store instructions as P0 instructions.
0 (R/W)	RES1_TRCCONFIGR_0_0	1.. Reserved, RES1.	

Trace Event Control 0 Register.

Trace Event Control 0 Register.

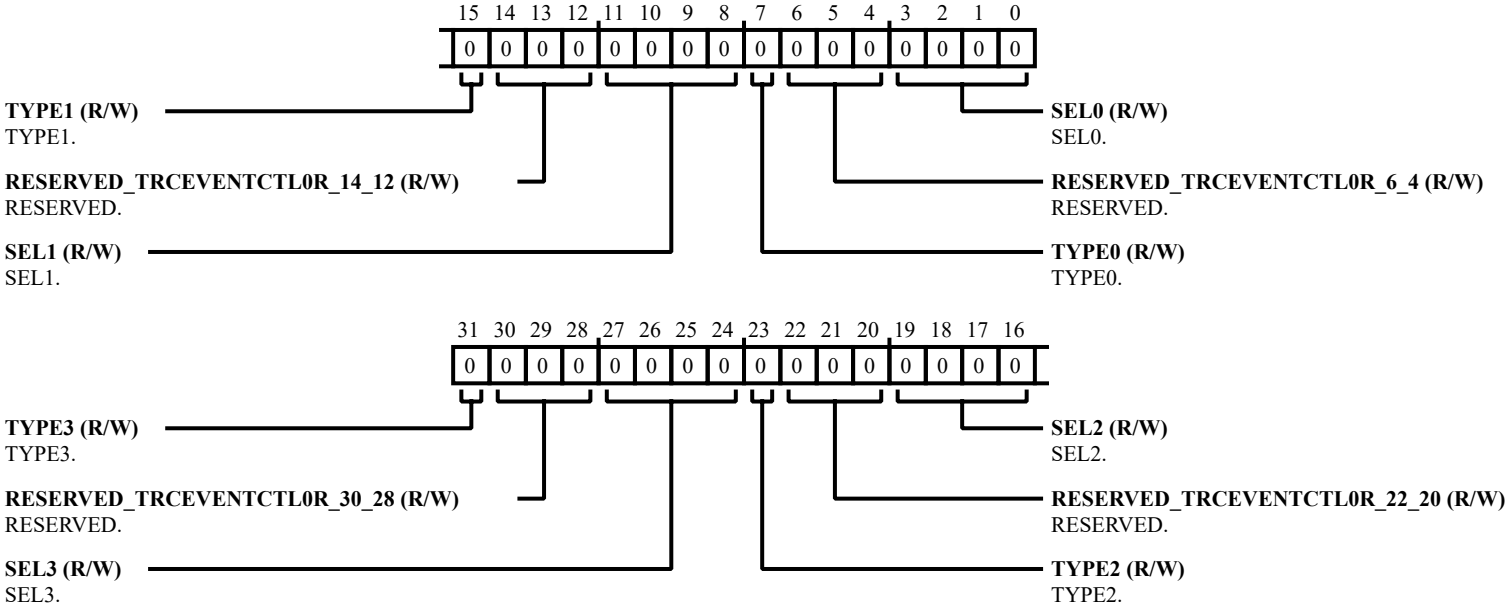


Figure 48-889: A55_C2_TRCEVENTCTL0R Register Diagram

Table 48-897: A55_C2_TRCEVENTCTL0R Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TYPE3	TYPE3.. Selects the resource type for trace event 3.	
		0	Single selected resource.
		1	Boolean combined resource pair.
30:28 (R/W)	RESERVED_TRCE-VENTCTL0R_30_28	RESERVED.. Reserved RES0.	
		0	Reserved.
27:24 (R/W)	SEL3	SEL3.. Selects the resource number based on the value of TYPE3.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Table 48-897: A55_C2_TRCEVENTCTL0R Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	TYPE2	TYPE2.. Selects the resource type for trace event 2.	
		0	Single selected resource.
		1	Boolean combined resource pair.
22:20 (R/W)	RESERVED_TRCE-VENTCTL0R_22_20	RESERVED.. Reserved RES0.	
		0	Reserved.
19:16 (R/W)	SEL2	SEL2.. Selects the resource number based on the value of TYPE2.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
15 (R/W)	TYPE1	TYPE1.. Selects the resource type for trace event 1.	
		0	Single selected resource.
		1	Boolean combined resource pair.
14:12 (R/W)	RESERVED_TRCE-VENTCTL0R_14_12	RESERVED.. Reserved RES0.	
		0	Reserved.
11:8 (R/W)	SEL1	SEL1.. Selects the resource number based on the value of TYPE1.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
7 (R/W)	TYPE0	TYPE0.. Selects the resource type for trace event 0.	
		0	Single selected resource.
		1	Boolean combined resource pair.
6:4 (R/W)	RESERVED_TRCE-VENTCTL0R_6_4	RESERVED.. Reserved RES0.	
		0	Reserved.

Table 48-897: A55_C2_TRCEVENTCTL0R Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	SEL0	SEL0.. Selects the resource number based on the value of TYPE0.	
		0	Selects a single selected resource from 0-15 defined by bits[3:0].
		1	Selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Trace Event Control 1 Register.

Trace Event Control 1 Register.

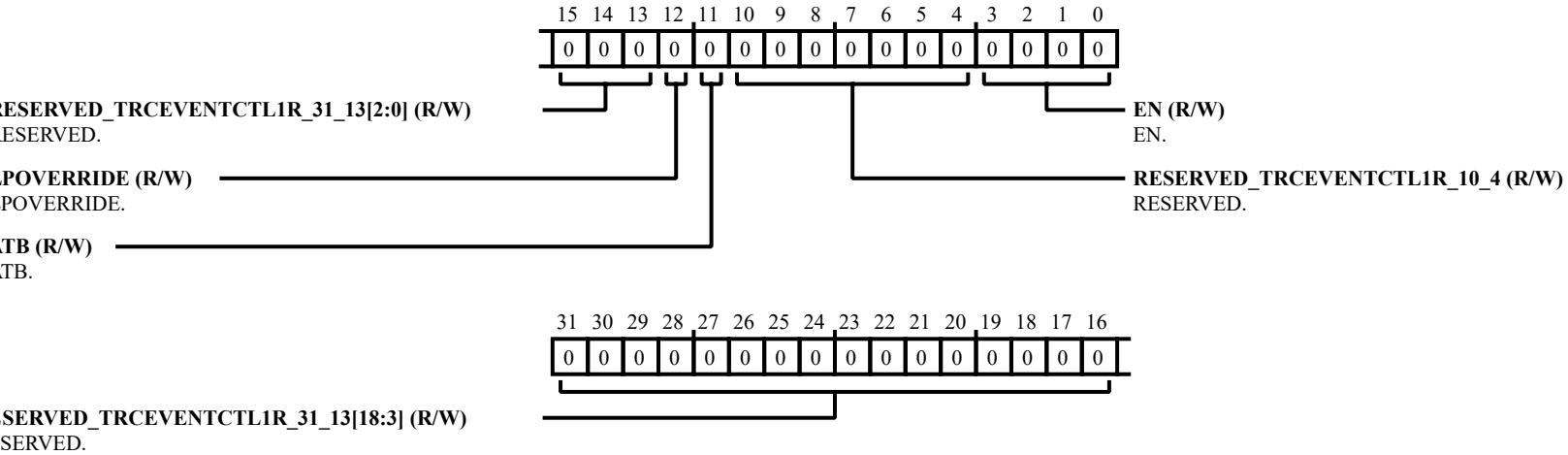


Figure 48-890: A55_C2_TRCEVENTCTL1R Register Diagram

Table 48-898: A55_C2_TRCEVENTCTL1R Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:13 (R/W)	RESERVED_TRCE-VENTCTL1R_31_13	RESERVED.. Reserved RES0.	
		0	Reserved.
12 (R/W)	LPOVERRIDE	LPOVERRIDE.. Low power state behavior override.	
		0	Low power state behavior unaffected.
		1	Low power state behavior overridden. The resources and Event trace generation are unaffected by entry to a low power state.
11 (R/W)	ATB	ATB.. ATB trigger enable.	
		0	ATB trigger disabled.
		1	ATB trigger enabled.
10:4 (R/W)	RESERVED_TRCE-VENTCTL1R_10_4	RESERVED.. Reserved RES0.	
		0	Reserved.

Table 48-898: A55_C2_TRCEVENTCTL1R Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	EN	EN.. One bit per event to enable generation of an event element in the instruction trace stream when the selected event occurs.	
		0	Event does not cause an event element.
		1	Event causes an event element.

Trace External Input Select Register.

Use this to set, or read, which external inputs are resources to the trace unit.

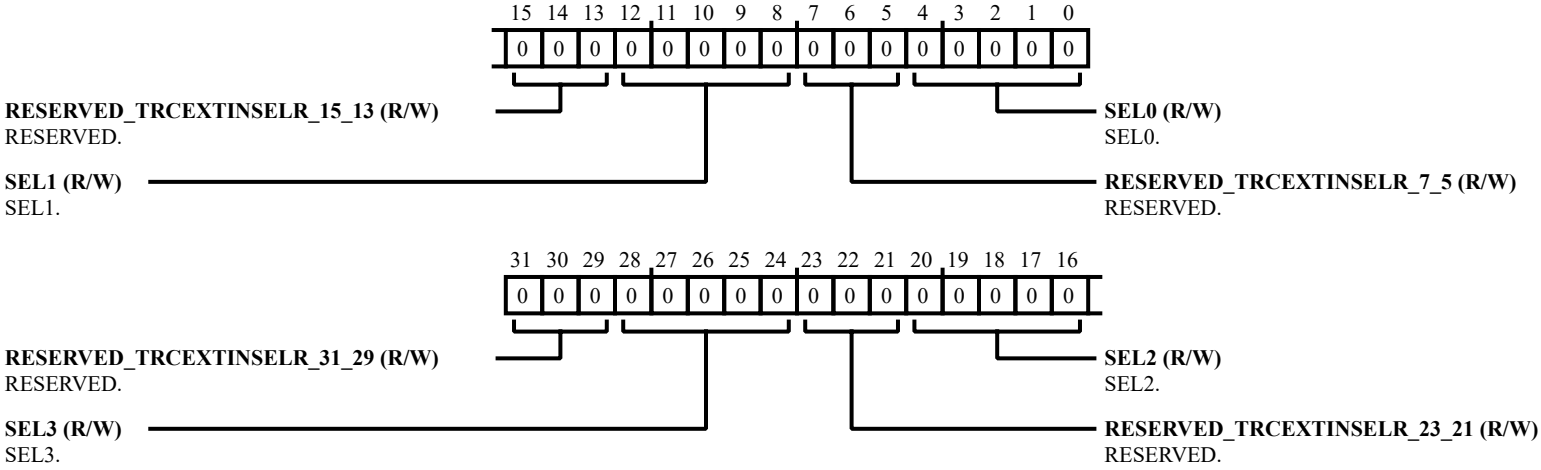


Figure 48-891: A55_C2_TRCEXTINSELR Register Diagram

Table 48-899: A55_C2_TRCEXTINSELR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/W)	RESERVED_TRCEXTIN-SELR_31_29	RESERVED.. Reserved RES0.	
		0	Reserved.
28:24 (R/W)	SEL3	SEL3.. Selects an event from the external input bus for External Input Resource 3.	
23:21 (R/W)	RESERVED_TRCEXTIN-SELR_23_21	RESERVED.. Reserved RES0.	
		0	Reserved.
20:16 (R/W)	SEL2	SEL2.. Selects an event from the external input bus for External Input Resource 2.	
15:13 (R/W)	RESERVED_TRCEXTIN-SELR_15_13	RESERVED.. Reserved RES0.	
		0	Reserved.
12:8 (R/W)	SEL1	SEL1.. Selects an event from the external input bus for External Input Resource 1.	

Table 48-899: A55_C2_TRCEXTINSELR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:5 (R/W)	RESERVED_TRCEXTIN-SELR_7_5	RESERVED.. Reserved RES0.	
		0	Reserved.
4:0 (R/W)	SEL0	SEL0.. Selects an event from the external input bus for External Input Resource 0.	

Trace ID Register 0.

Returns the tracing capabilities of the trace unit.

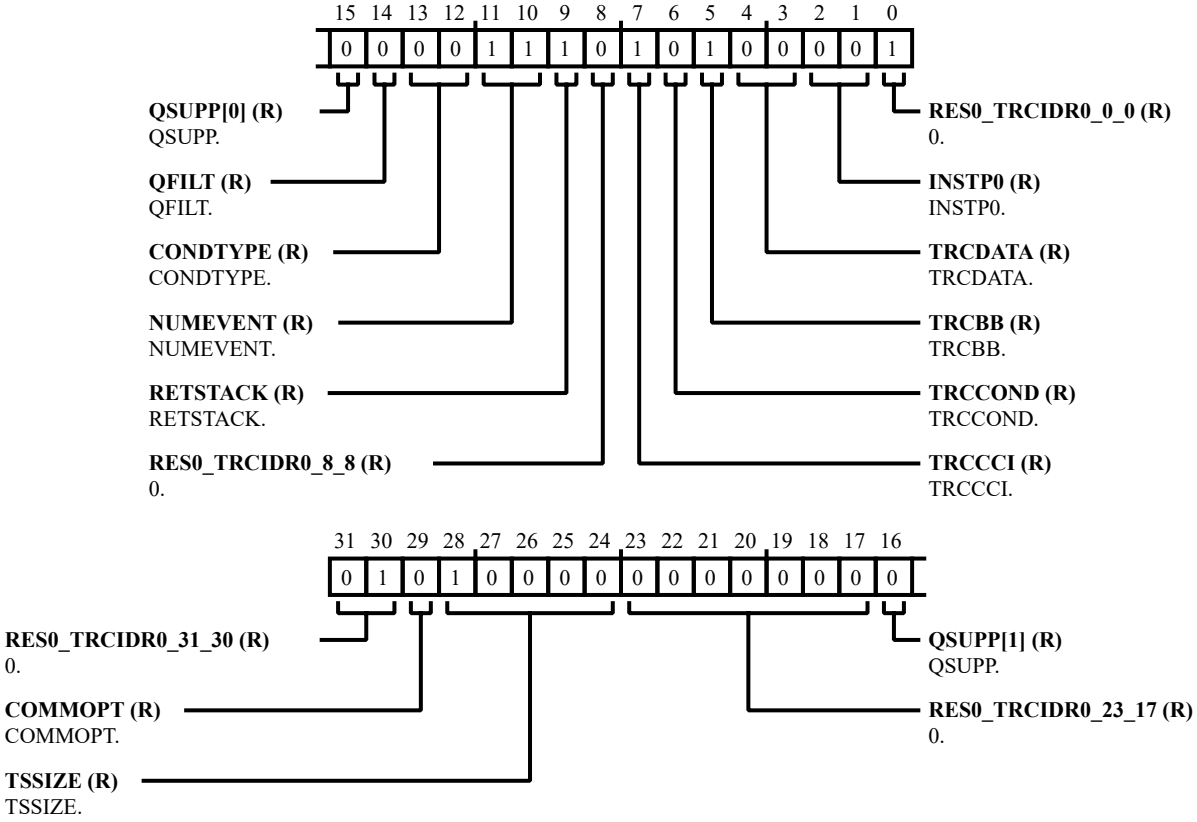


Figure 48-892: A55_C2_TRCIDR0 Register Diagram

Table 48-900: A55_C2_TRCIDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	RES0_TRCIDR0_31_30	0.. Reserved, RES0.	
29 (R/NW)	COMMOPT	COMMOPT.. Conditional instruction tracing support bit. Indicates if the trace unit supports conditional instruction tracing: 0 Conditional instruction tracing is not supported. 1 Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.	
		0	Conditional instruction tracing is not supported.
		1	Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.

Table 48-900: A55_C2_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28:24 (R/NW)	TSSIZE	TSSIZE.. Global timestamp size field. The permitted values are: 00000 Global timestamping is not implemented. 00110 Implementation supports a maximum global timestamp of 48bits. 01000 Implementation supports a maximum global timestamp of 64bits. All other values are reserved.When global timestamping is implemented then TRCCONFIGR.TS and TRCTSCTLR are supported.	
		0	Global timestamping is not implemented.
		6	Implementation supports a maximum global timestamp of 48bits.
		8	Implementation supports a maximum global timestamp of 64bits.
23:17 (R/NW)	RES0_TRCIDR0_23_17	0.. Reserved, RES0.	
16:15 (R/NW)	QSUPP	QSUPP.. Q element support field. The permitted values are: 00 Q element support is not implemented. TRCCONFIGR.QE is RES0. 01 Q element support is implemented, and only supports Q elements with instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b01. 10 Q element support is implemented, and only supports Q elements without instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b11. 11 Q element support is implemented, and supports both Q elements with instruction counts and Q elements without instruction counts. TRCCONFIGR.QE is fully implemented.	
		0	Q element support is not implemented. TRCCONFIGR.QE is RES0.
		1	Q element support is implemented, and only supports Q elements with instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b01.
		2	Q element support is implemented, and only supports Q elements without instruction counts. TRCCONFIGR.QE can only take the values 0b00 or 0b11.
		3	Q element support is implemented, and supports both Q elements with instruction counts and Q elements without instruction counts. TRCCONFIGR.QE is fully implemented.

Table 48-900: A55_C2_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/NW)	QFILT	QFILT.. Q element filtering support field. The permitted values are: 0 Q element filtering is not implemented. 1 Q element filtering is implemented. TRCQCTLR is implemented. When QSUPP==0b00, this field is RES0.	
		0	Q element filtering is not implemented.
		1	Q element filtering is implemented. TRCQCTLR is implemented.
13:12 (R/NW)	CONDTYPE	CONDTYPE.. Conditional tracing field. The permitted values are: 00 The trace unit indicates only if a conditional instruction is a pass or fail. 01 The trace unit provides the Current Program Status Register (CPSR) status, for a conditional instruction. All other values are reserved.	
		0	The trace unit indicates only if a conditional instruction is a pass or fail.
		1	The trace unit provides the Current Program Status Register (CPSR) status, for a conditional instruction.
11:10 (R/NW)	NUMEVENT	NUMEVENT.. Number of events field. Indicates how many events the trace unit supports: 00 The trace unit supports 1 event. 01 The trace unit supports 2 events. 10 The trace unit supports 3 events. 11 The trace unit supports 4 events. This field controls how many fields are supported in TRCEVENTCTL0R, and indicates the size of TRCEVENTCTL1R.INSTEN.	
		0	The trace unit supports 1 event.
		1	The trace unit supports 2 events.
		2	The trace unit supports 3 events.
		3	The trace unit supports 4 events.
9 (R/NW)	RETSTACK	RETSTACK.. Return stack bit. Indicates if the implementation supports a return stack: 0 Return stack is not implemented. 1 Return stack is implemented, so TRCCONFIGR.RS is supported.	
		0	Return stack is not implemented.
		1	Return stack is implemented, so TRCCONFIGR.RS is supported.
8 (R/NW)	RES0_TRCIDR0_8_8	0.. Reserved, RES0.	

Table 48-900: A55_C2_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	TRCCCI	TRCCCI.. Cycle counting instruction bit. Indicates if the trace unit supports cycle counting for instructions: 0 Cycle counting in the instruction trace is not implemented. 1 Cycle counting in the instruction trace is implemented, so TRCCONFIGR.CCI and TRCCCCTLR are supported.	
		0	Cycle counting in the instruction trace is not implemented.
		1	Cycle counting in the instruction trace is implemented, so TRCCONFIGR.CCI and TRCCCCTLR are supported.
6 (R/NW)	TRCCOND	TRCCOND.. Conditional instruction tracing support bit. Indicates if the trace unit supports conditional instruction tracing: 0 Conditional instruction tracing is not supported. 1 Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.	
		0	Conditional instruction tracing is not supported.
		1	Conditional instruction tracing is supported, so TRCCONFIGR.COND is supported.
5 (R/NW)	TRCBB	TRCBB.. Branch broadcast tracing support bit. Indicates if the trace unit supports branch broadcast tracing: 0 Branch broadcast tracing is not supported. 1 Branch broadcast tracing is supported, so TRCCONFIGR.BB and TRCBBCTLR are supported.	
		0	Branch broadcast tracing is not supported.
		1	Branch broadcast tracing is supported, so TRCCONFIGR.BB and TRCBBCTLR are supported.
4:3 (R/NW)	TRCDATA	TRCDATA.. Conditional tracing field. The permitted values are: 00 Data tracing is not supported. 11 Tracing of data addresses and data values is supported, so TRCCONFIGR.DA, TRCCONFIGR.DV, TRCSTALLCTLR.DATADISCARD, TRCSTALLCTLR.INSTPRIORITY, TRCSTALLCTLR.DSTALL, and TRCEVENTCTL1R.DATAEN are supported. All other values are reserved.	
		0	Data tracing is not supported.
		3	Tracing of data addresses and data values is supported, so TRCCONFIGR.DA, TRCCONFIGR.DV, TRCSTALLCTLR.DATADISCARD, TRCSTALLCTLR.INSTPRIORITY, TRCSTALLCTLR.DSTALL, and TRCEVENTCTL1R.DATAEN are supported.

Table 48-900: A55_C2_TRCIDR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:1 (R/NW)	INSTP0	INSTP0.. P0 tracing support field. The permitted values are: 00 Tracing of load and store instructions as P0 elements is not supported. 11 Tracing of load and store instructions as P0 elements is supported, so TRCCONFIGR.INSTP0 is supported. All other values are reserved.	
		0	Tracing of load and store instructions as P0 elements is not supported.
		3	Tracing of load and store instructions as P0 elements is supported, so TRCCONFIGR.INSTP0 is supported.
0 (R/NW)	RES0_TRCIDR0_0_0	0.. Reserved, RES0.	

Trace ID Register 1.

Returns the base architecture of the trace unit.

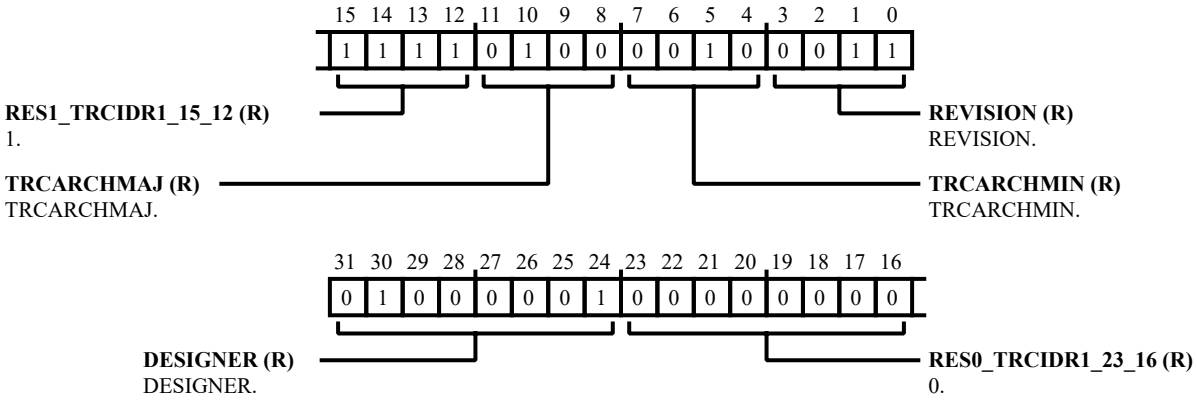


Figure 48-893: A55_C2_TRCIDR1 Register Diagram

Table 48-901: A55_C2_TRCIDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/NW)	DESIGNER	DESIGNER.. Indicates which company designed the trace unit. The permitted values are: 01000001 ARM Limited. 01000100 Digital Equipment Corporation. 01001101 Motorola, Freescale Semiconductor Inc. 01010001 QUALCOMM Inc. 01010110 Marvell Semiconductor Inc. 01101001 Intel Corporation. All other values are reserved.	
		65	ARM Limited.
		68	Digital Equipment Corporation.
		77	Motorola, Freescale Semiconductor Inc.
		81	QUALCOMM Inc.
		86	Marvell Semiconductor Inc.
		105	Intel Corporation.
23:16 (R/NW)	RES0_TRCIDR1_23_16	0.. Reserved, RES0.	
15:12 (R/NW)	RES1_TRCIDR1_15_12	1.. Reserved, RES1.	
11:8 (R/NW)	TRCARCHMAJ	TRCARCHMAJ.. Indicates the major version of the ETM architecture. The permitted value is: 100 ETMv4. All other values are reserved.	
		4	ETMv4.

Table 48-901: A55_C2_TRCIDR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	TRCARCHMIN	TRCARCHMIN.. Indicates the minor version of the ETM architecture. The permitted value is: 0 ETMv4 minor version 0. All other values are reserved.	
		0	ETMv4 minor version 0.
3:0 (R/NW)	REVISION	REVISION.. Returns an IMPLEMENTATION DEFINED value that identifies the revision of the trace registers and the OS Save and Restore registers.ARM recommends:That the initial implementation sets REVISION==0x0 and the field then increments for any subsequent implementations. However, it is acceptable to omit some values or use another scheme to identify the revision number.That TRCPIDR2.REVISION==TRCIDR1.REVISION. However, in situations where it is difficult to align these fields, such as with a metal layer fix, then it is acceptable to change the REVISION fields independently.	

Trace ID Register 2.

Returns the maximum size of the data value, data address, VMID, Context ID and instruction address in the trace unit.

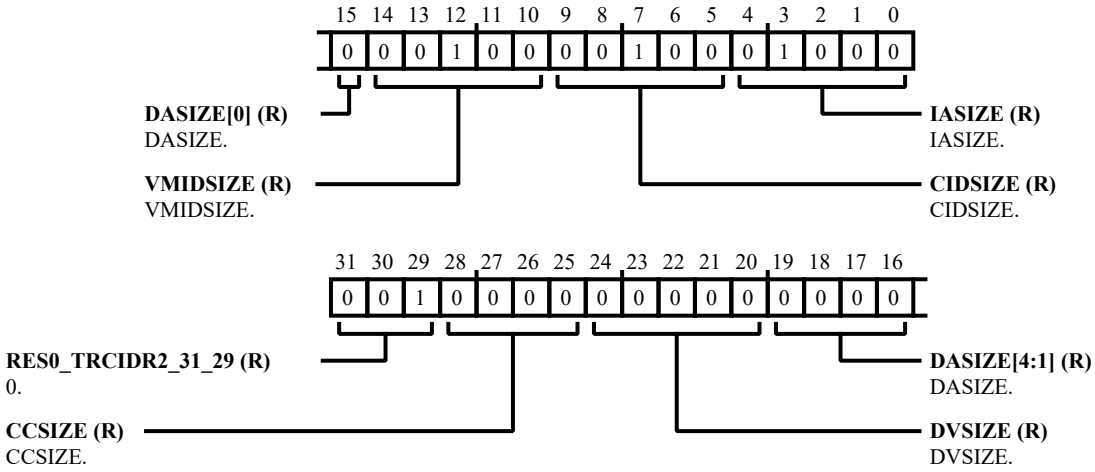


Figure 48-894: A55_C2_TRCIDR2 Register Diagram

Table 48-902: A55_C2_TRCIDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:29 (R/NW)	RES0_TRCIDR2_31_29	0.. Reserved, RES0.	
28:25 (R/NW)	CCSIZE	CCSIZE.. Indicates the size of the cycle counter in bits minus 12. 0000 The cycle counter is 12 bits in length. 0001 The cycle counter is 13 bits in length. and so on up to 0b1000, indicating the cycle counter is 20 bits in length.All other values are reserved.If cycle counting is not implemented, as indicated by TRCIDR0.TRCCCI, this field is 0b0000.	
		0	The cycle counter is 12 bits in length.
		1	The cycle counter is 13 bits in length.

Table 48-902: A55_C2_TRCIDR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24:20 (R/NW)	DVSIZE	DVSIZE.. Indicates the data value size in bytes. The permitted values are: 00000 Data value tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00. 00100 Maximum of 32-bit data value size. 01000 Maximum of 64-bit data value size. All other values are reserved.	
		0	Data value tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00.
		4	Maximum of 32-bit data value size.
		8	Maximum of 64-bit data value size.
19:15 (R/NW)	DASIZE	DASIZE.. Indicates the data address size in bytes. The permitted values are: 00000 Data address tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00. 00100 Maximum of 32-bit data address size. 01000 Maximum of 64-bit data address size. All other values are reserved.	
		0	Data address tracing is not supported. Therefore, an implementation must also set TRCIDR0.TRCDATA==0b00.
		4	Maximum of 32-bit data address size.
		8	Maximum of 64-bit data address size.
14:10 (R/NW)	VMIDSIZE	VMIDSIZE.. Indicates the VMID size. The permitted values are: 00000 VMID tracing is not supported. 00001 Maximum of 8-bit VMID size, so TRCCONFIGR.VMID is supported. All other values are reserved.	
		0	VMID tracing is not supported.
		1	Maximum of 8-bit VMID size, so TRCCONFIGR.VMID is supported.
9:5 (R/NW)	CIDSIZE	CIDSIZE.. Indicates the Context ID size. The permitted values are: 00000 Context ID tracing is not supported. 00100 Maximum of 32-bit Context ID size, so TRCCONFIGR.CID is supported. All other values are reserved.	
		0	Context ID tracing is not supported.
		4	Maximum of 32-bit Context ID size, so TRCCONFIGR.CID is supported.

Table 48-902: A55_C2_TRCIDR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:0 (R/NW)	IASIZE	IASIZE.. Indicates the instruction address size. The permitted values are: 00100 Maximum of 32-bit address size. 01000 Maximum of 64-bit address size. All other values are reserved.	
		4	Maximum of 32-bit address size.
		8	Maximum of 64-bit address size.

Trace ID Register 3.

Indicates the features that the trace unit supports.

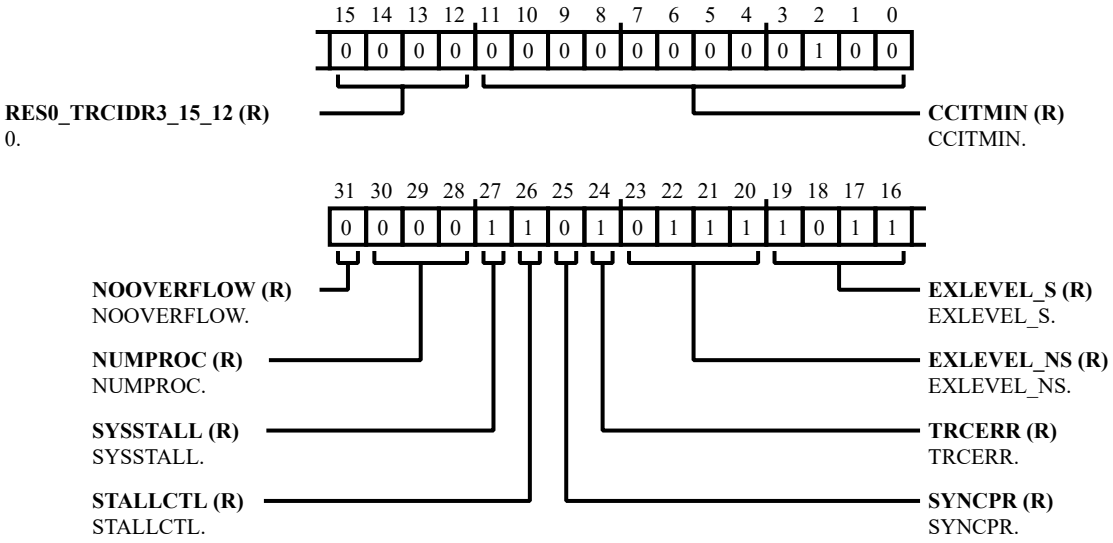


Figure 48-895: A55_C2_TRCIDR3 Register Diagram

Table 48-903: A55_C2_TRCIDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	NOOVERFLOW	NOOVERFLOW.. Indicates if TRCSTALLCTLR.NOOVERFLOW is supported: 0 TRCSTALLCTLR.NOOVERFLOW is not supported, or STALLCTL==0. 1 TRCSTALLCTLR.NOOVERFLOW is supported.	
		0	TRCSTALLCTLR.NOOVERFLOW is not supported, or STALLCTL==0.
		1	TRCSTALLCTLR.NOOVERFLOW is supported.
30:28 (R/NW)	NUMPROC	NUMPROC.. Indicates the number of processors available for tracing. The possible values are: 000 The trace unit can trace one processor. 001 The trace unit can trace two processors. 010 The trace unit can trace three processors. and so on up to 0b111, which indicates the trace unit can trace eight processors.This field sets the maximum value of TRCPROCSELR.PROCSEL.	
		0	The trace unit can trace one processor.
		1	The trace unit can trace two processors.
		2	The trace unit can trace three processors.

Table 48-903: A55_C2_TRCIDR3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	SYSSTALL	SYSSTALL.. Indicates if the implementation can support stall control: 0 The system does not support stall control of the processor. 1 The system can support stall control of the processor. The system supports stalling of the processor only when SYSSTALL==1 and STALLCTL==1.	
		0	The system does not support stall control of the processor.
		1	The system can support stall control of the processor.
26 (R/NW)	STALLCTL	STALLCTL.. Indicates if TRCSTALLCTLR is supported: 0 TRCSTALLCTLR is not supported. 1 TRCSTALLCTLR is supported.	
		0	TRCSTALLCTLR is not supported.
		1	TRCSTALLCTLR is supported.
25 (R/NW)	SYNCPR	SYNCPR.. Indicates if an implementation has a fixed synchronization period: 0 TRCSYNCPR is read-write so software can change the synchronization period. 1 TRCSYNCPR is read-only so the synchronization period is fixed.	
		0	TRCSYNCPR is read-write so software can change the synchronization period.
		1	TRCSYNCPR is read-only so the synchronization period is fixed.
24 (R/NW)	TRCERR	TRCERR.. Indicates if TRCVICTLR.TRCERR is supported: 0 TRCVICTLR.TRCERR is not supported 1 TRCVICTLR.TRCERR is supported.	
		0	TRCVICTLR.TRCERR is not supported.
		1	TRCVICTLR.TRCERR is supported.

Table 48-903: A55_C2_TRCIDR3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/NW)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit indicates whether instruction tracing is supported for the corresponding exception level: 0 In Non-secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are not supported. 1 In Non-secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are supported. The exception levels are:Bit[20]Exception level 0.Bit[21]Exception level 1.Bit[22]Exception level 2.Bit[23]SBZ. EXLEVEL_NS[3] is never implemented.	
		0	In Non-secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are not supported.
		1	In Non-secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_NS and TRCVICTLR.EXLEVEL_NS are supported.
19:16 (R/NW)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit indicates whether instruction tracing is supported for the corresponding exception level: 0 In Secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are not supported. 1 In Secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are supported. The exception levels are:Bit[16]Exception level 0.Bit[17]Exception level 1.Bit[18]SBZ. EXLEVEL_S[2] is never implemented.Bit[19]Exception level 3.	
		0	In Secure state, exception level n is not supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are not supported.
		1	In Secure state, exception level n is supported so the corresponding bits in TRCACATRn.EXLEVEL_S and TRCVICTLR.EXLEVEL_S are supported.
15:12 (R/NW)	RES0_TRCIDR3_15_12	0.. Reserved, RES0.	
11:0 (R/NW)	CCITMIN	CCITMIN.. Indicates the minimum value that can be programmed in TRCCCCTLR.THRESHOLD.When cycle counting in the instruction trace is supported, that is TRCIDR0.TRCCCI==1, then the minimum value of this field is 0x001, otherwise it is 0x000.	

Trace ID Register 4.

Returns how many resources the trace unit supports.

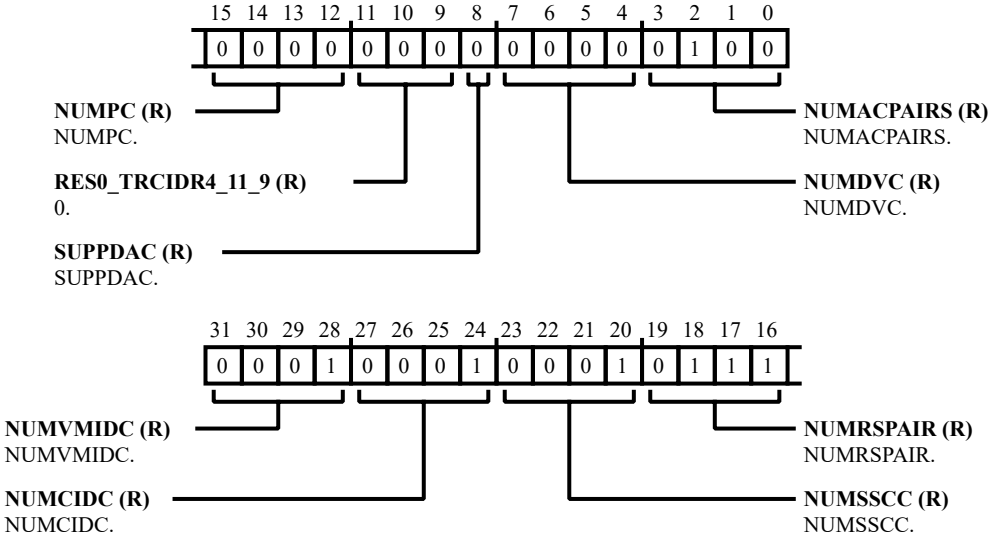


Figure 48-896: A55_C2_TRCIDR4 Register Diagram

Table 48-904: A55_C2_TRCIDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	NUMVMIDC	NUMVMIDC.. Indicates the number of VMID comparators that are available for tracing. The permitted values are: 0000 No VMID comparators are available. 0001 The implementation has one VMID comparator. 0010 The implementation has two VMID comparators. and so on up to 0b1000, which indicates that the implementation has eight VMID comparators.All other values are reserved.	
		0	No VMID comparators are available.
		1	The implementation has one VMID comparator.
		2	The implementation has two VMID comparators.
27:24 (R/NW)	NUMCIDC	NUMCIDC.. Indicates the number of Context ID comparators that are available for tracing. The permitted values are: 0000 No Context ID comparators are available. 0001 The implementation has one Context ID comparator. 0010 The implementation has two Context ID comparators. and so on up to 0b1000, which indicates that the implementation has eight Context ID comparators.All other values are reserved.	
		0	No Context ID comparators are available.
		1	The implementation has one Context ID comparator.
		2	The implementation has two Context ID comparators.

Table 48-904: A55_C2_TRCIDR4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/NW)	NUMSSCC	NUMSSCC.. Indicates the number of single-shot comparator controls that are available for tracing. The permitted values are: 0000 No single-shot comparator controls are available. 0001 The implementation has one single-shot comparator control. 0010 The implementation has two single-shot comparator controls. and so on up to 0b1000, which indicates that the implementation has eight single-shot comparator controls.All other values are reserved.	
		0	No single-shot comparator controls are available.
		1	The implementation has one single-shot comparator control.
		2	The implementation has two single-shot comparator controls.
19:16 (R/NW)	NUMRSPAIR	NUMRSPAIR.. Indicates the number of resource selection pairs that are available for tracing. The permitted values are: 0000 The implementation has one resource selection pair. 0001 The implementation has two resource selection pairs. 0010 The implementation has three resource selection pairs. and so on up to 0b1111, which indicates that the implementation has 16 resource selection pairs.Implementations always have at least one resource selection pair so that they can support the FALSE and TRUE resource selectors, 0 and 1.	
		0	The implementation has one resource selection pair.
		1	The implementation has two resource selection pairs.
		2	The implementation has three resource selection pairs.
15:12 (R/NW)	NUMPC	NUMPC.. Indicates the number of processor comparator inputs that are available for tracing. The permitted values are: 0000 No processor comparator inputs are available. 0001 The implementation has one processor comparator input. 0010 The implementation has two processor comparator inputs. and so on up to 0b1000, which indicates that the implementation has eight processor comparator inputs.All other values are reserved.	
		0	No processor comparator inputs are available.
		1	The implementation has one processor comparator input.
		2	The implementation has two processor comparator inputs.
11:9 (R/NW)	RES0_TRCIDR4_11_9	0.. Reserved, RES0.	

Table 48-904: A55_C2_TRCIDR4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	SUPPDAC	SUPPDAC.. Indicates if the implementation can support data address comparisons: 0 The implementation does not support data address comparisons. 1 The implementation can support data address comparisons.	
		0	The implementation does not support data address comparisons.
		1	The implementation can support data address comparisons.
7:4 (R/NW)	NUMDVC	NUMDVC.. Indicates the number of data value comparators that are available for tracing. The permitted values are: 0000 No data value comparators are available. 0001 The implementation has one data value comparator. 0010 The implementation has two data value comparators. and so on up to 0b1000, which indicates that the implementation has eight data value comparators.All other values are reserved.	
		0	No data value comparators are available.
		1	The implementation has one data value comparator.
		2	The implementation has two data value comparators.
3:0 (R/NW)	NUMACPAIRS	NUMACPAIRS.. Indicates the number of address comparator pairs that are available for tracing. The permitted values are: 0000 No address comparator pairs are available. 0001 The implementation has one address comparator pair. 0010 The implementation has two address comparator pairs. and so on up to 0b1000, which indicates that the implementation has eight address comparator pairs.All other values are reserved.	
		0	No address comparator pairs are available.
		1	The implementation has one address comparator pair.
		2	The implementation has two address comparator pairs.

Trace ID Register 5.

Indicates the features that the trace unit supports.

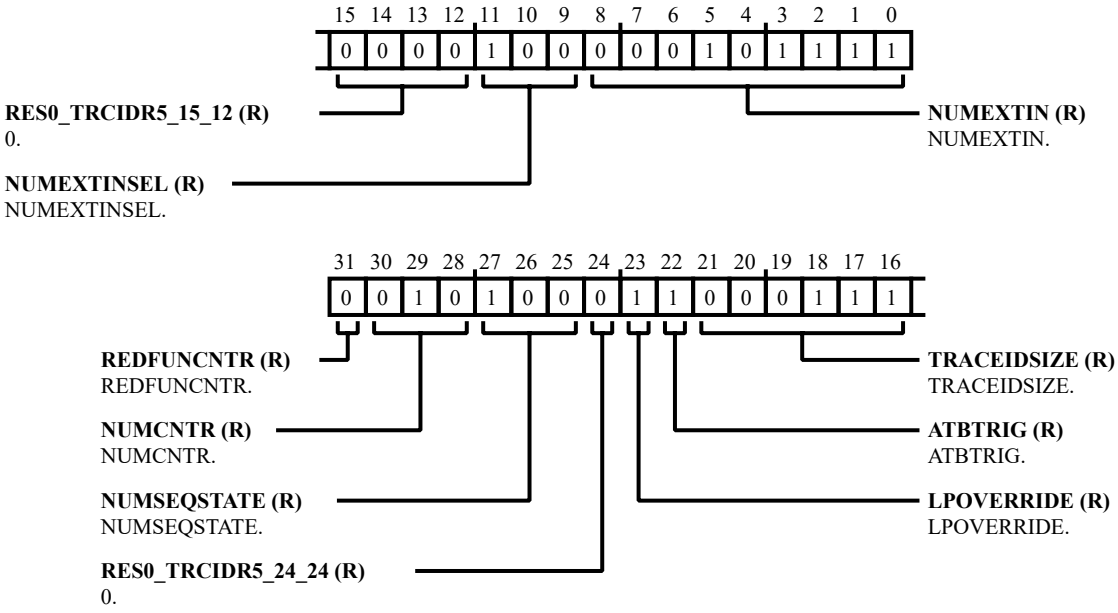


Figure 48-897: A55_C2_TRCIDR5 Register Diagram

Table 48-905: A55_C2_TRCIDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	REDFUNCNTR	REDFUNCNTR.. Indicates if the reduced function counter is implemented: 0 The reduced function counter is not supported. 1 Counter 0 is implemented as a reduced function counter.	
		0	The reduced function counter is not supported.
		1	Counter 0 is implemented as a reduced function counter.
30:28 (R/NW)	NUMCNTR	NUMCNTR.. Indicates the number of counters that are available for tracing. The permitted values are: 000 No counters are available. 001 The implementation has one counter. 010 The implementation has two counters. 011 The implementation has three counters. 100 The implementation has four counters. All other values are reserved.	
		0	No counters are available.
		1	The implementation has one counter.
		2	The implementation has two counters.
		3	The implementation has three counters.

Table 48-905: A55_C2_TRCIDR5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	The implementation has four counters.
27:25 (R/NW)	NUMSEQSTATE	NUMSEQSTATE.. Indicates the number of sequencer states that are implemented. The permitted values are: 000 No sequencer states are implemented. 100 The implementation has four sequencer states. All other values are reserved.	
		0	No sequencer states are implemented.
		4	The implementation has four sequencer states.
24 (R/NW)	RES0_TRCIDR5_24_24	0.. Reserved, RES0.	
23 (R/NW)	LPOVERRIDE	LPOVERRIDE.. Indicates if the implementation can support low-power state override: 0 The implementation does not support low-power state override. 1 The implementation supports low-power state override, and the TRCEVENTCTL1R.LPOVERRIDE field is implemented. The trace unit must support low-power state override if it can enter a low-power mode where the resources and event trace generation are disabled.	
		0	The implementation does not support low-power state override.
		1	The implementation supports low-power state override, and the TRCEVENTCTL1R.LPOVERRIDE field is implemented.
22 (R/NW)	ATBTRIG	ATBTRIG.. Indicates if the implementation can support ATB triggers: 0 The implementation does not support ATB triggers. 1 The implementation supports ATB triggers, and the TRCEVENTCTL1R.ATBTRIG field is implemented.	
		0	The implementation does not support ATB triggers.
		1	The implementation supports ATB triggers, and the TRCEVENTCTL1R.ATBTRIG field is implemented.
21:16 (R/NW)	TRACEIDSIZE	TRACEIDSIZE.. Indicates the trace ID width. The permitted value is: 111 The implementation supports a 7-bit trace ID. This sets the width of the TRCTRACEIDR.TRACEID field. All other values are reserved.The CoreSight ATB requires a 7-bit trace ID width.	
		7	The implementation supports a 7-bit trace ID. This sets the width of the TRCTRACEIDR.TRACEID field.
15:12 (R/NW)	RES0_TRCIDR5_15_12	0.. Reserved, RES0.	

Table 48-905: A55_C2_TRCIDR5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:9 (R/NW)	NUMEXTINSEL	NUMEXTINSEL.. Indicates how many external input select resources are implemented. The permitted values are: 000 No external input select resources are available. If NUMEXTINSEL is zero, NUMEXTIN must also be zero. 001 The implementation has one external input select resource. 010 The implementation has two external input select resources. 011 The implementation has three external input select resources. 100 The implementation has four external input select resources. All other values are reserved.See TRCEXTINSELR for how to select an input select resource.	
		0	No external input select resources are available. If NUMEXTINSEL is zero, NUMEXTIN must also be zero.
		1	The implementation has one external input select resource.
		2	The implementation has two external input select resources.
		3	The implementation has three external input select resources.
		4	The implementation has four external input select resources.
8:0 (R/NW)	NUMEXTIN	NUMEXTIN.. Indicates how many external inputs are implemented. The permitted values are: 000000000 No external inputs are available. If NUMEXTIN is zero, NUMEXTINSEL must also be zero. 000000001 The implementation has one external input. 000000010 The implementation has two external inputs. and so on up to 0b100000000, which indicates that the implementation has 256 external inputs.All other values >0b100000000 are reserved.See TRCEXTINSELR for how to select an external input.	
		0	No external inputs are available. If NUMEXTIN is zero, NUMEXTINSEL must also be zero.
		1	The implementation has one external input.
		2	The implementation has two external inputs.

Trace IMPLEMENTATION DEFINED Register 0.

TRCIMSPEC0 shows the presence of any IMPLEMENTATION DEFINED features, and provides an interface to enable the features that are provided. The TRCIMSPECn registers are reserved for the future implementation of up to eight IMPLEMENTATION DEFINED registers. When a trace unit does not implement these registers, TRCIMSPEC0 must still be partially implemented so that a debugger can implement a general mechanism for detecting the IMPLEMENTATION DEFINED registers.

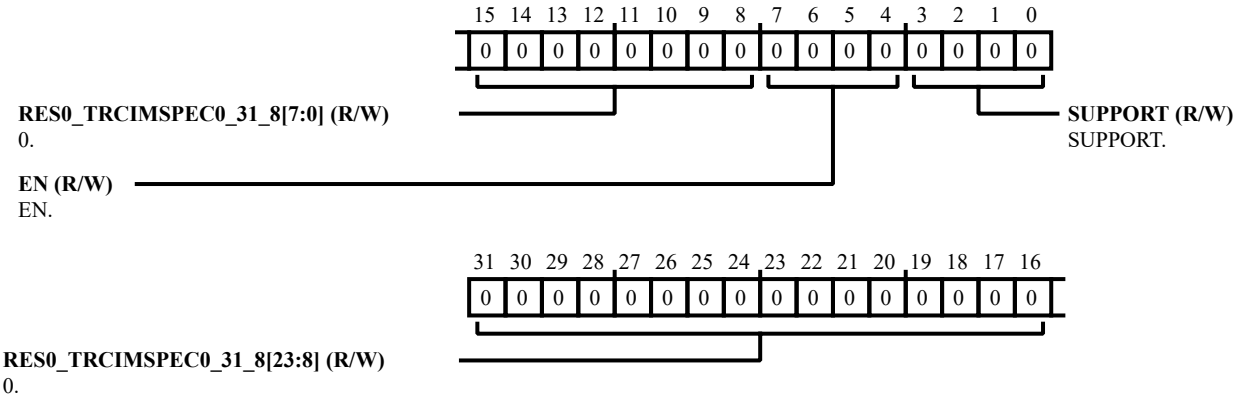


Figure 48-898: A55_C2_TRCIMSPEC0 Register Diagram

Table 48-906: A55_C2_TRCIMSPEC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	RES0_TRCIM-SPEC0_31_8	0.. Reserved, RES0.	
7:4 (R/W)	EN	EN.. If SUPPORT is not 0b0000, controls whether the IMPLEMENTATION DEFINED features are enabled. The permitted values are: 0000 The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported. and any other value, which indicates that the trace unit behavior is IMPLEMENTATION DEFINED.If SUPPORT is 0b0000, this field is RES0.	
		0	The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported.

Table 48-906: A55_C2_TRCIMSPEC0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	SUPPORT	SUPPORT.. Indicates whether the implementation supports IMPLEMENTATION DEFINED features. The permitted values are: 0000 No IMPLEMENTATION DEFINED features are supported. The EN field is RES0. and any other value, which indicates that IMPLEMENTATION DEFINED features are supported. Use of these values requires written permission from ARM.	
		0	No IMPLEMENTATION DEFINED features are supported. The EN field is RES0.

Integration ATB Identification Register.

Integration ATB Identification Register.

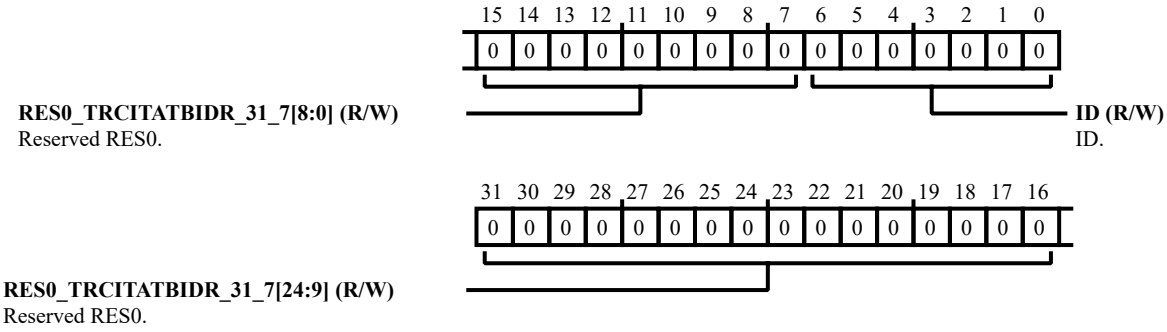


Figure 48-899: A55_C2_TRCITATBIDR Register Diagram

Table 48-907: A55_C2_TRCITATBIDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:7 (R/W)	RES0_TRCITAT-BIDR_31_7	Reserved RES0.. Reserved RES0.
		0
6:0 (R/W)	ID	ID.. Drives the ATIDMn[6:0] output pins.

Trace Integration Mode Control Register.

Controls whether the trace unit is in integration mode.

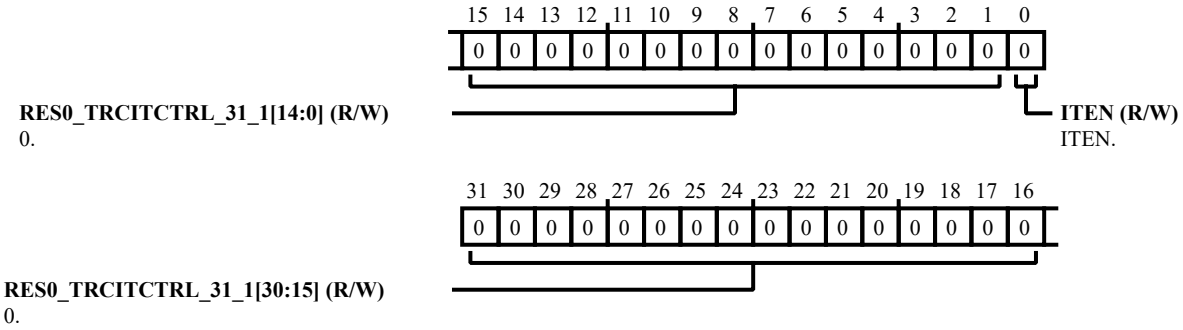


Figure 48-900: A55_C2_TRCITCTRL Register Diagram

Table 48-908: A55_C2_TRCITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_TRCITCTRL_31_1	0.. Reserved, RES0.	
0 (R/W)	ITEN	ITEN.. Integration mode enable bit: 0 The trace unit is not in integration mode. 1 The trace unit is in integration mode. This mode enables a debug agent to perform topology detection, and System-on-Chip (SoC) test software to perform integration testing.	
		0	The trace unit is not in integration mode.
		1	The trace unit is in integration mode. This mode enables a debug agent to perform topology detection, and System-on-Chip (SoC) test software to perform integration testing.

Integration Instruction ATB In Register.

Integration Instruction ATB In Register.

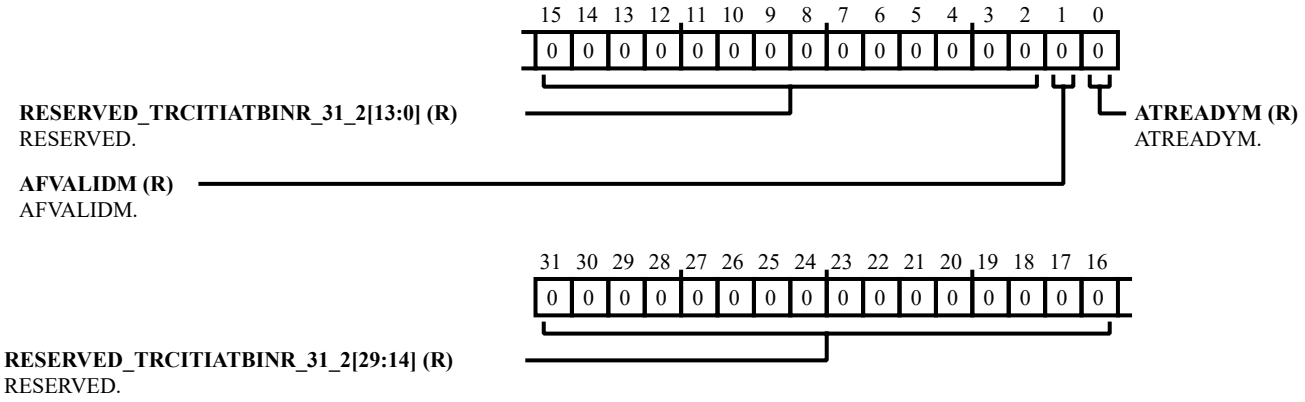


Figure 48-901: A55_C2_TRCITIATBINR Register Diagram

Table 48-909: A55_C2_TRCITIATBINR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/NW)	RESERVED_TRCITIAT-BINR_31_2	RESERVED.. Reserved. Read undefined.
1 (R/NW)	AFVALIDM	AFVALIDM.. Returns the value of the AFVALIDMn input pin.
0 (R/NW)	ATREADYM	ATREADYM.. Returns the value of the ATREADYMn input pin.

Integration Instruction ATB Out Register.

Integration Instruction ATB Out Register.

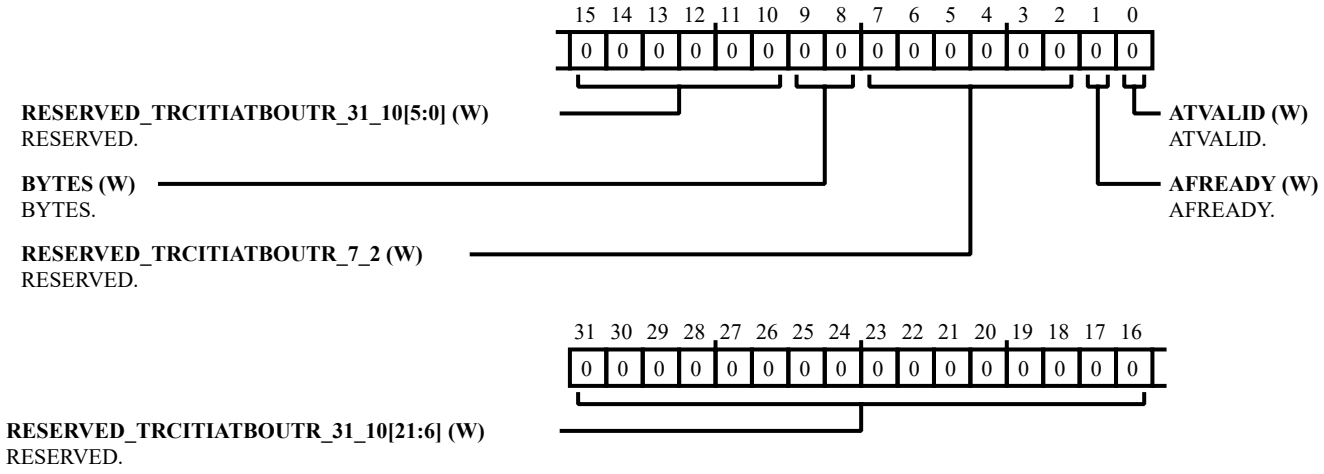


Figure 48-902: A55_C2_TRCITIATBOUTR Register Diagram

Table 48-910: A55_C2_TRCITIATBOUTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:10 (RX/W)	RESERVED_TRCITIAT-BOUTR_31_10	RESERVED.. Reserved. Read undefined.
9:8 (RX/W)	BYTES	BYTES.. Drives the ATBYTESMn[1:0] output pins.
7:2 (RX/W)	RESERVED_TRCITIAT-BOUTR_7_2	RESERVED.. Reserved. Read undefined.
1 (RX/W)	AFREADY	AFREADY.. Drives the AFREADYMn output pin.
0 (RX/W)	ATVALID	ATVALID.. Drives the ATVALIDMn output pin.

Integration Instruction ATB Data Register.

Integration Instruction ATB Data Register.

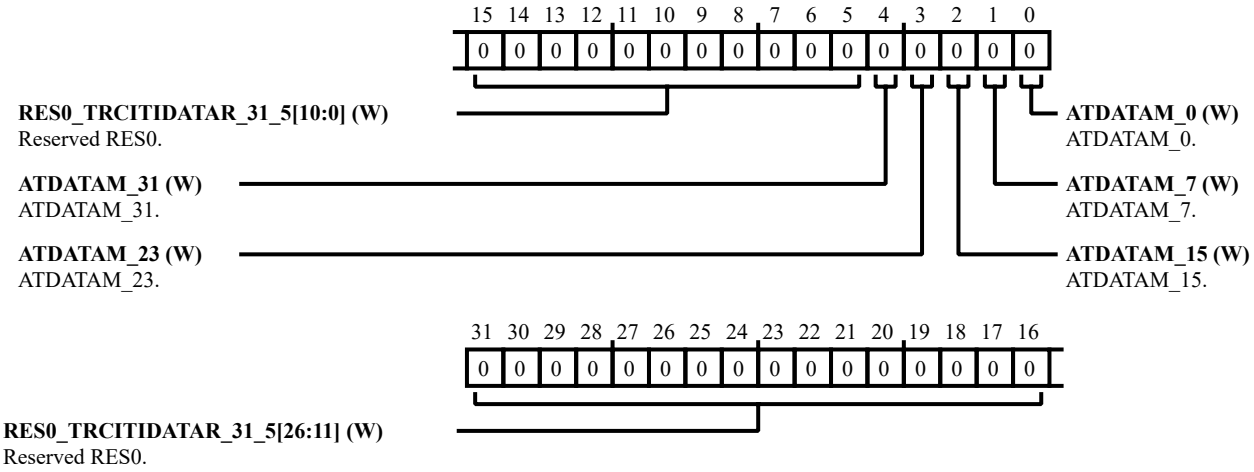


Figure 48-903: A55_C2_TRCITIDATAR Register Diagram

Table 48-911: A55_C2_TRCITIDATAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (RX/W)	RES0_TRCITIDA-TAR_31_5	Reserved RES0.. Reserved RES0.	
		0	Reserved.
4 (RX/W)	ATDATAM_31	ATDATAM_31.. Drives the ATDATAM[31] output.	
3 (RX/W)	ATDATAM_23	ATDATAM_23.. Drives the ATDATAM[23] output.	
2 (RX/W)	ATDATAM_15	ATDATAM_15.. Drives the ATDATAM[15] output.	
1 (RX/W)	ATDATAM_7	ATDATAM_7.. Drives the ATDATAM[7] output.	
0 (RX/W)	ATDATAM_0	ATDATAM_0.. Drives the ATDATAM[0] output.	

Trace OS Lock Access Register.

Controls whether the OS Lock is locked.

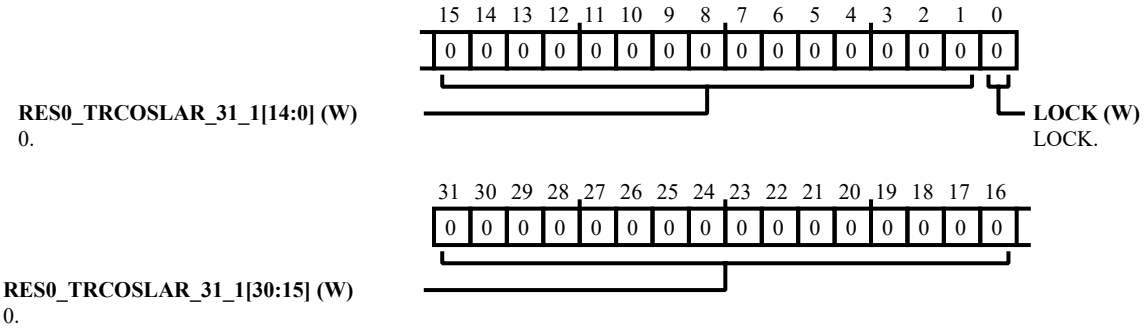


Figure 48-904: A55_C2_TRCOSLAR Register Diagram

Table 48-912: A55_C2_TRCOSLAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (RX/W)	RES0_TRCOSLAR_31_1	0.. Reserved, RES0.	
0 (RX/W)	LOCK	LOCK.. OS Lock control bit: 0 Unlocks the OS Lock. 1 Locks the OS Lock. This setting disables the trace unit.	
		0	Unlocks the OS Lock.
		1	Locks the OS Lock. This setting disables the trace unit.

Trace OS Lock Status Register.

Returns the status of the OS Lock.

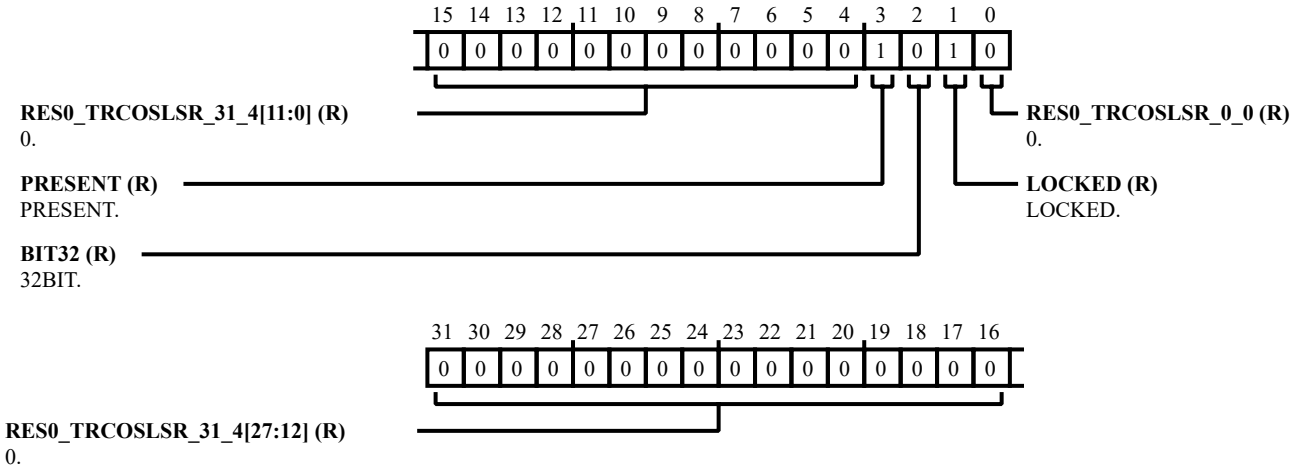


Figure 48-905: A55_C2_TRCOSLSR Register Diagram

Table 48-913: A55_C2_TRCOSLSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:4 (R/NW)	RES0_TRCOSLSR_31_4	0.. Reserved, RES0.	
3 (R/NW)	PRESENT	PRESENT.. Indicates whether the OS Lock is implemented.This bit is RES1, which indicates that the OS Lock is always implemented.	
2 (R/NW)	BIT32	32BIT.. This bit is RES0, which indicates that software must perform a 32-bit write to update the TRCOSLAR.	
1 (R/NW)	LOCKED	LOCKED.. OS Lock status bit: 0 The OS Lock is unlocked. 1 The OS Lock is locked. When the trace unit core power domain is powered down the value is UNKNOWN. The TRCPDSR indicates if the trace unit core power domain is powered down.	
		0	The OS Lock is unlocked.
		1	The OS Lock is locked.
0 (R/NW)	RES0_TRCOSLSR_0_0	0.. Reserved, RES0.	

Trace PowerDown Status Register.

Returns information about the OS Lock status, core power domain status and power interruption status for the trace unit.

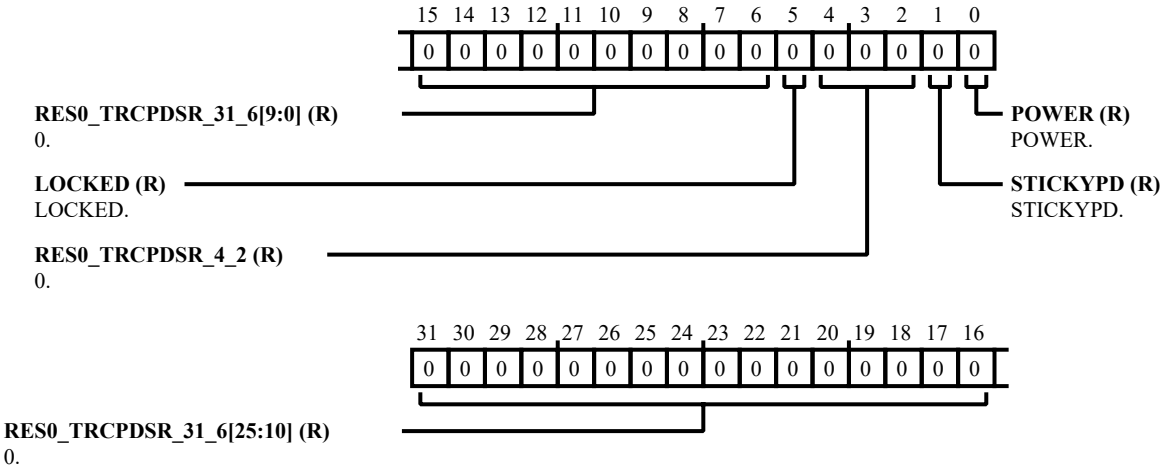


Figure 48-906: A55_C2_TRCPDSR Register Diagram

Table 48-914: A55_C2_TRCPDSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:6 (R/NW)	RES0_TRCPDSR_31_6	0.. Reserved, RES0.	
5 (R/NW)	LOCKED	LOCKED.. OS Lock status bit: 0 The OS Lock is unlocked. 1 The OS Lock is locked. The value is UNKNOWN when the trace unit core power domain is powered down, that is, when POWER==0.	
		0	The OS Lock is unlocked.
		1	The OS Lock is locked.
4:2 (R/NW)	RES0_TRCPDSR_4_2	0.. Reserved, RES0.	

Table 48-914: A55_C2_TRCPDSR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	STICKYPD	STICKYPD.. Sticky powerdown status bit. Indicates whether the trace register state is valid: 0 If POWER==1 then the state of TRCOSLSR and the trace registers are valid. If POWER==0 then it is UNKNOWN whether the state of TRCOSLSR and the trace registers are valid. 1 The state of TRCOSLSR and the trace registers might not be valid. The trace unit sets this bit to 1 if either the trace unit is reset, or the power to the trace unit core power domain is removed and the trace register state is not valid. After this register is read, if the Software Lock is unlocked and the trace unit core power domain is powered up, then the trace unit sets this bit to 0. The TRCLAR controls whether the Software Lock is locked.	
		0	If POWER==1 then the state of TRCOSLSR and the trace registers are valid. If POWER==0 then it is UNKNOWN whether the state of TRCOSLSR and the trace registers are valid.
		1	The state of TRCOSLSR and the trace registers might not be valid. The trace unit sets this bit to 1 if either the trace unit is reset, or the power to the trace unit core power domain is removed and the trace register state is not valid.
0 (R/NW)	POWER	POWER.. Power status bit: 0 The trace unit core power domain is not powered. The trace registers are not accessible and they all return an error response. 1 The trace unit core power domain is powered. The trace registers are accessible.	
		0	The trace unit core power domain is not powered. The trace registers are not accessible and they all return an error response.
		1	The trace unit core power domain is powered. The trace registers are accessible.

Trace Programming Control Register.

Enables the trace unit.

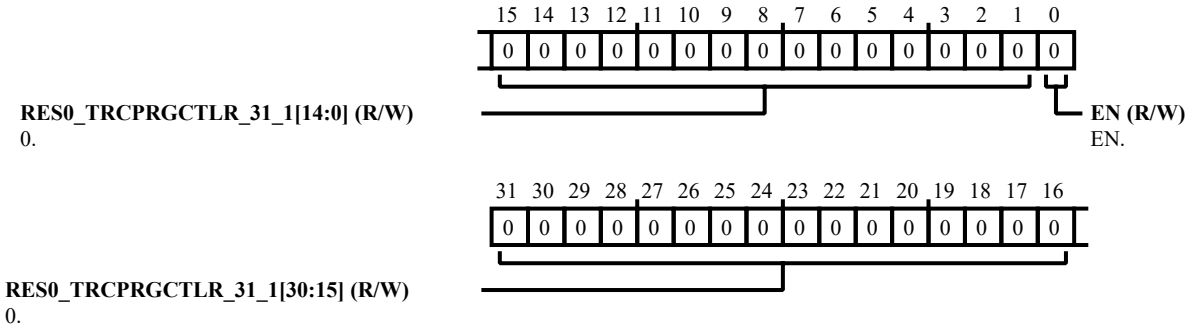


Figure 48-907: A55_C2_TRCPRGCTLR Register Diagram

Table 48-915: A55_C2_TRCPRGCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_TRCPRGCTLR_31_1	0.. Reserved, RES0.	
0 (R/W)	EN	EN.. Trace unit enable bit. Possible values are: 0 The trace unit is disabled. All trace resources are inactive and no trace is generated. 1 The trace unit is enabled.	
		0	The trace unit is disabled. All trace resources are inactive and no trace is generated.
		1	The trace unit is enabled.

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

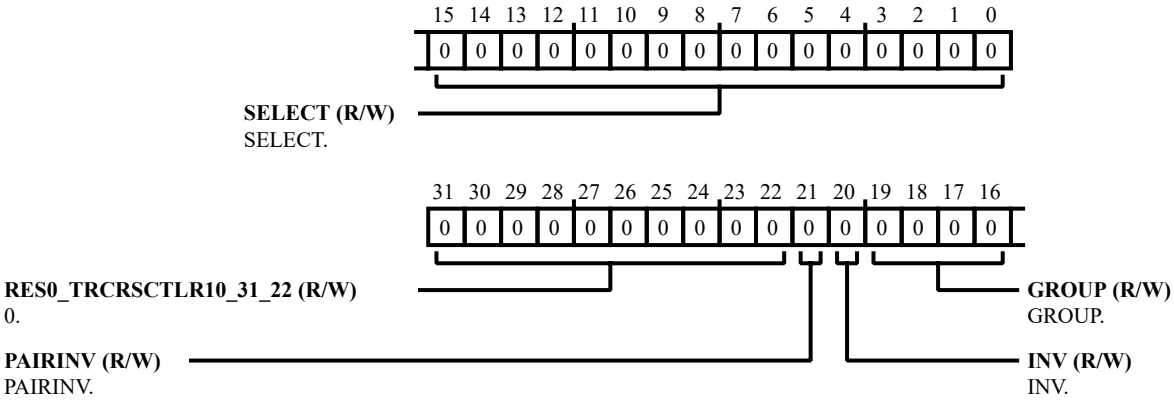


Figure 48-908: A55_C2_TRCRSCTLR10 Register Diagram

Table 48-916: A55_C2_TRCRSCTLR10 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR10_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-916: A55_C2_TRCRSCTLR10 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

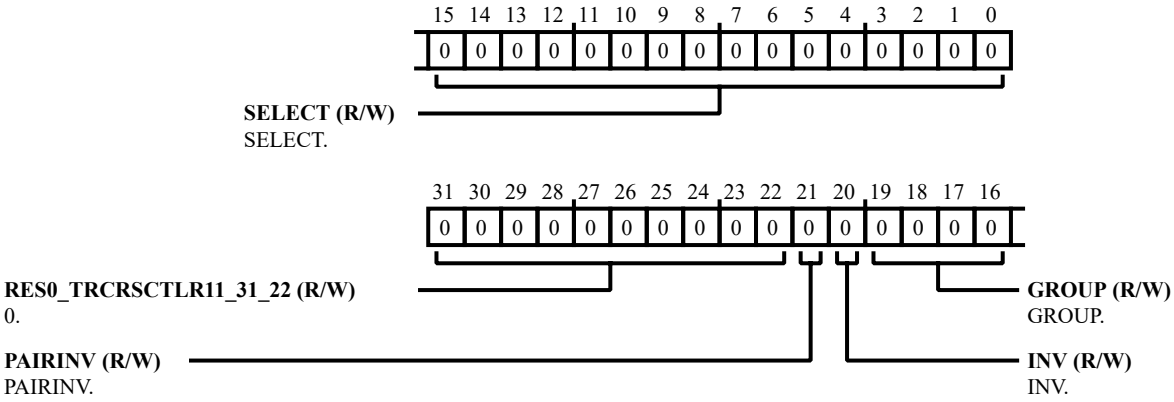


Figure 48-909: A55_C2_TRCRSCTLR11 Register Diagram

Table 48-917: A55_C2_TRCRSCTLR11 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR11_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-917: A55_C2_TRCRSCTLR11 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

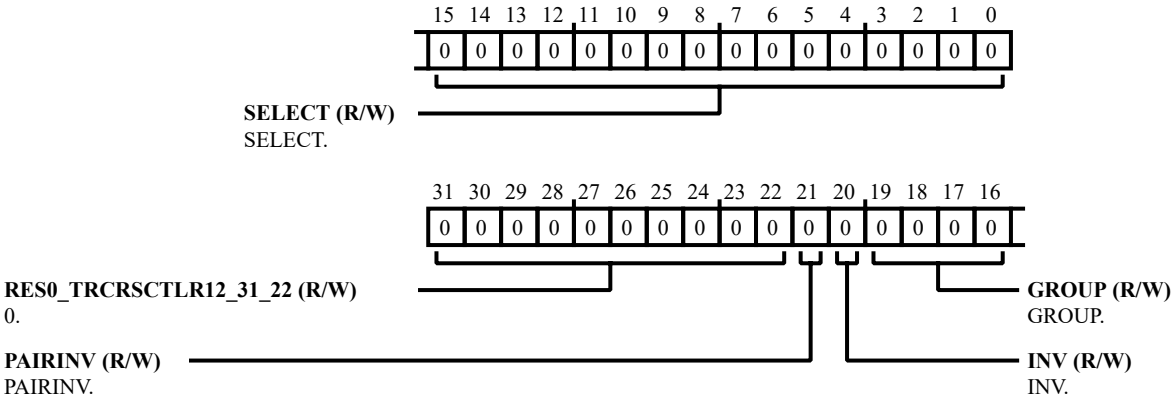


Figure 48-910: A55_C2_TRCRSCTLR12 Register Diagram

Table 48-918: A55_C2_TRCRSCTLR12 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR12_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-918: A55_C2_TRCRSCTLR12 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

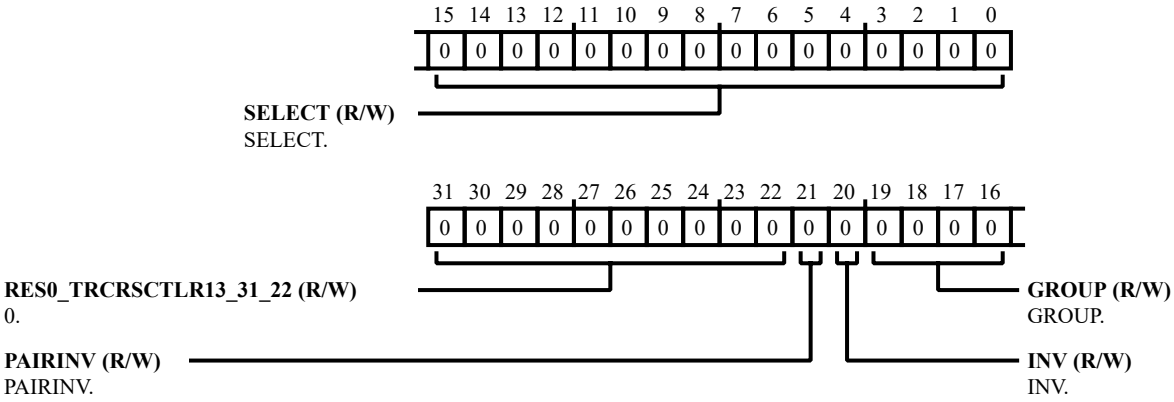


Figure 48-911: A55_C2_TRCRSCTLR13 Register Diagram

Table 48-919: A55_C2_TRCRSCTLR13 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR13_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-919: A55_C2_TRCRSCTLR13 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

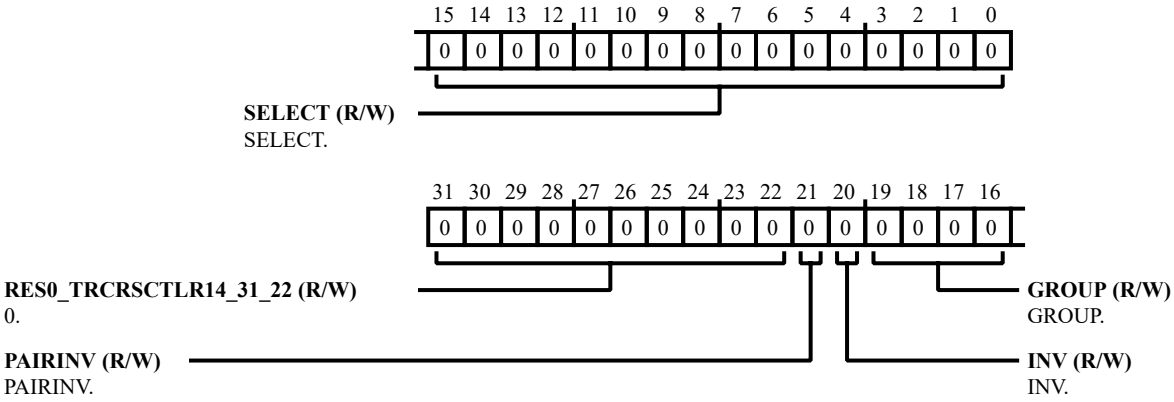


Figure 48-912: A55_C2_TRCRSCTLR14 Register Diagram

Table 48-920: A55_C2_TRCRSCTLR14 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR14_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-920: A55_C2_TRCRSCTLR14 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

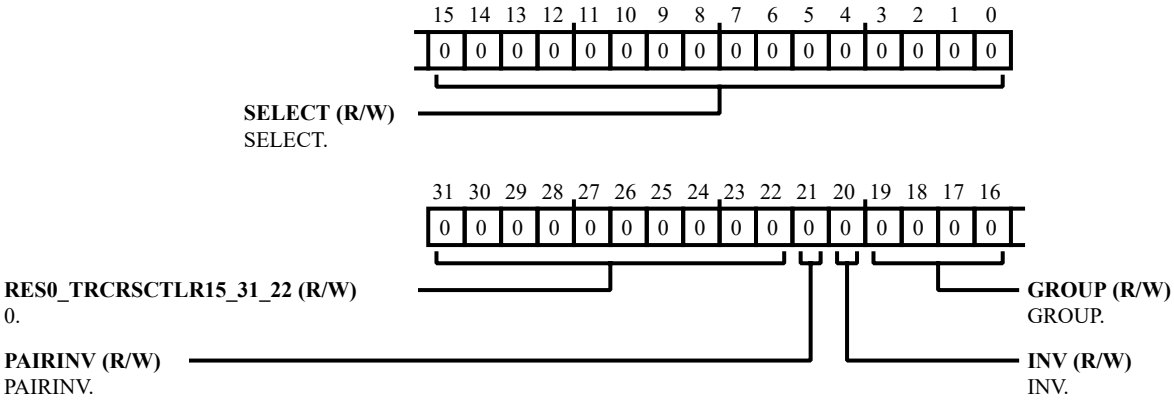


Figure 48-913: A55_C2_TRCRSCTLR15 Register Diagram

Table 48-921: A55_C2_TRCRSCTLR15 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR15_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-921: A55_C2_TRCRSCTLR15 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

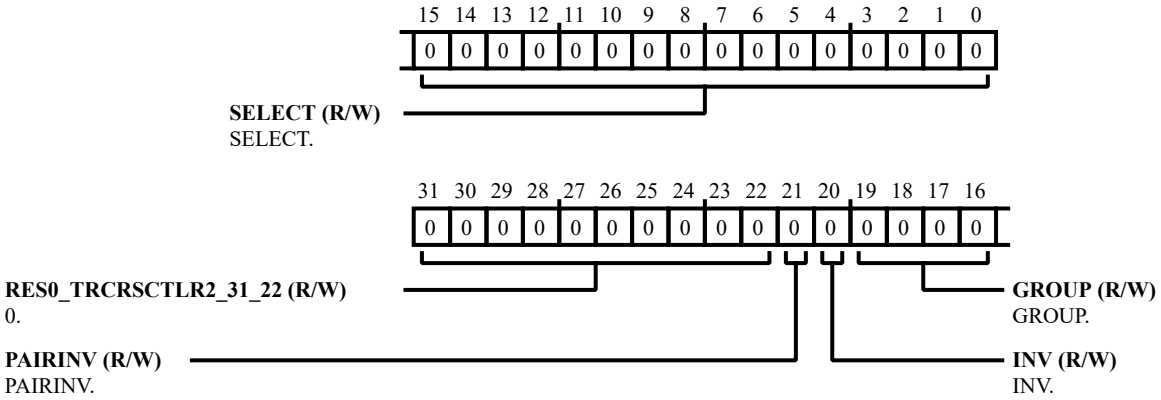


Figure 48-914: A55_C2_TRCRSCTLR2 Register Diagram

Table 48-922: A55_C2_TRCRSCTLR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR2_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-922: A55_C2_TRCRSCTLR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

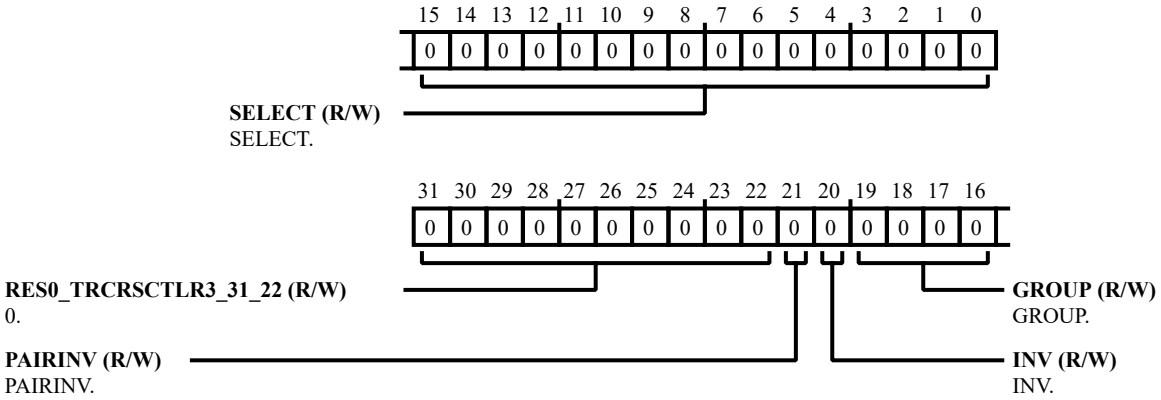


Figure 48-915: A55_C2_TRCRSCTLR3 Register Diagram

Table 48-923: A55_C2_TRCRSCTLR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR3_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-923: A55_C2_TRCRSCTLR3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

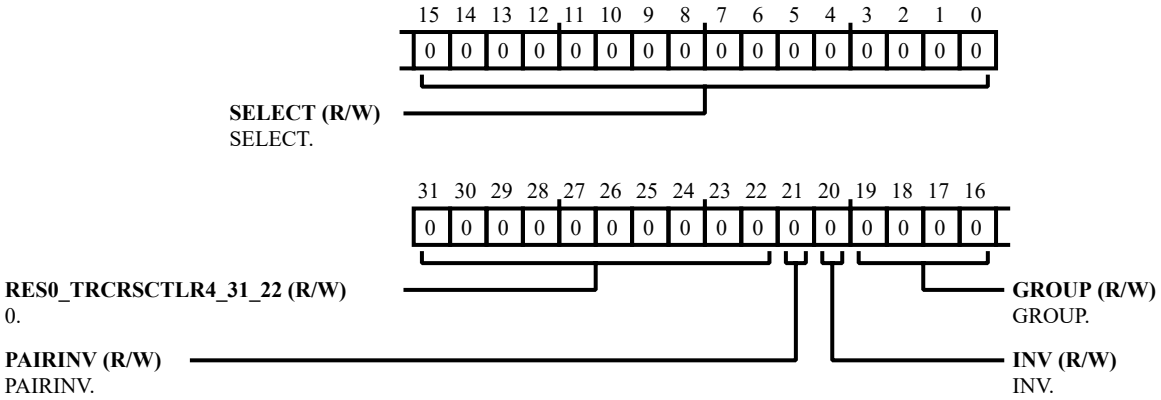


Figure 48-916: A55_C2_TRCRSCTLR4 Register Diagram

Table 48-924: A55_C2_TRCRSCTLR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR4_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-924: A55_C2_TRCRSCTLR4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

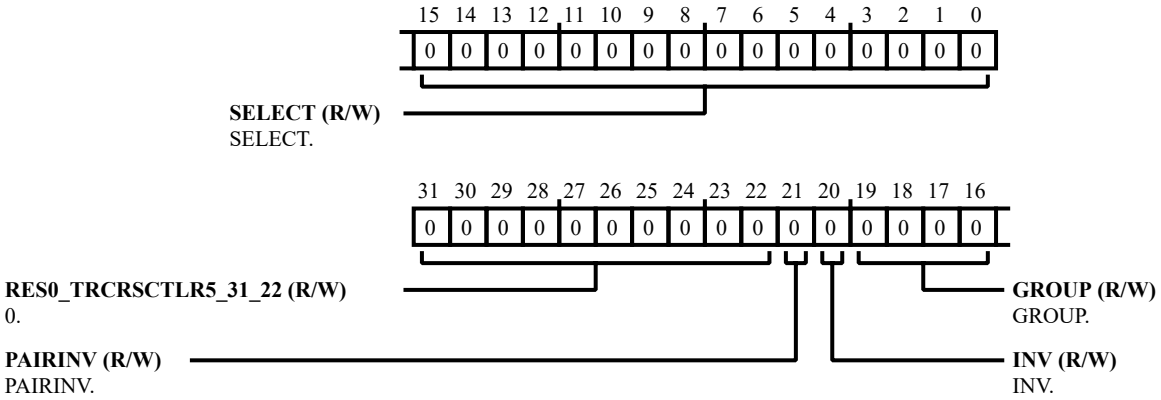


Figure 48-917: A55_C2_TRCRSCTLR5 Register Diagram

Table 48-925: A55_C2_TRCRSCTLR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR5_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-925: A55_C2_TRCRSCTLR5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

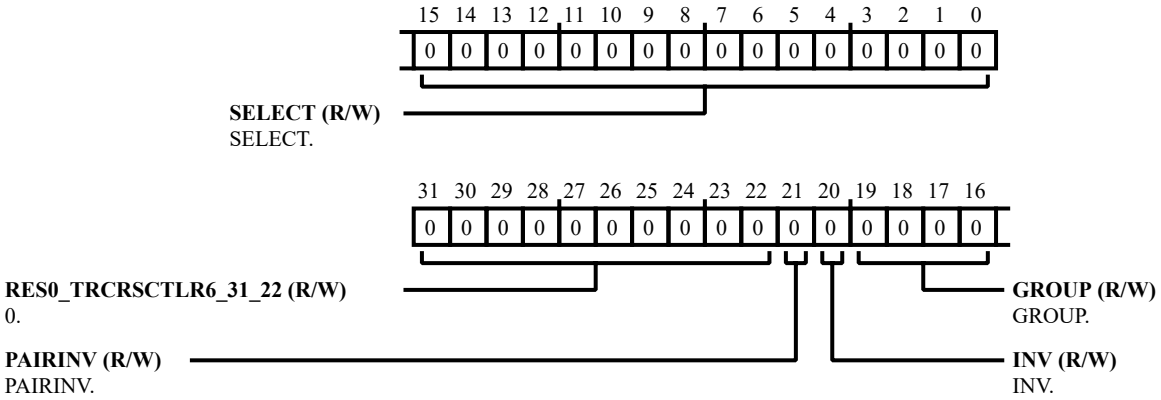


Figure 48-918: A55_C2_TRCRSCTLR6 Register Diagram

Table 48-926: A55_C2_TRCRSCTLR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR6_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-926: A55_C2_TRCRSCTLR6 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

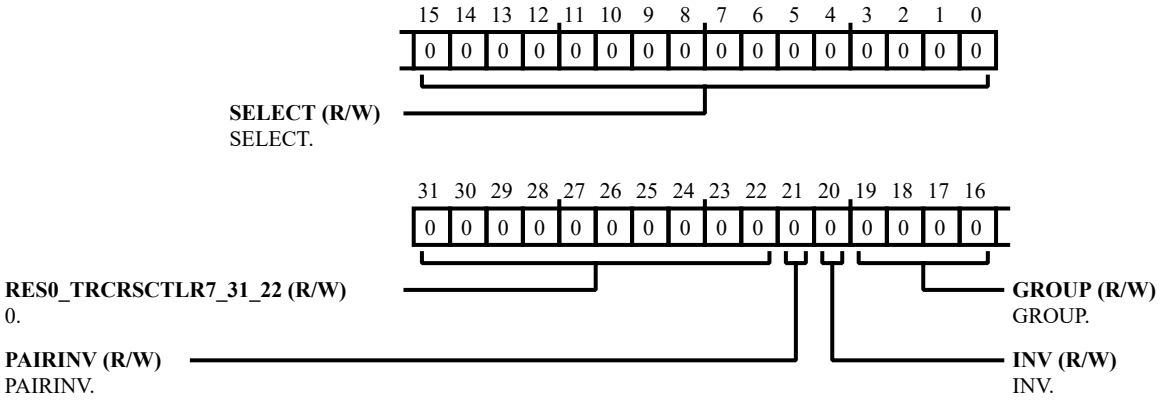


Figure 48-919: A55_C2_TRCRSCTLR7 Register Diagram

Table 48-927: A55_C2_TRCRSCTLR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR7_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-927: A55_C2_TRCRSCTLR7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

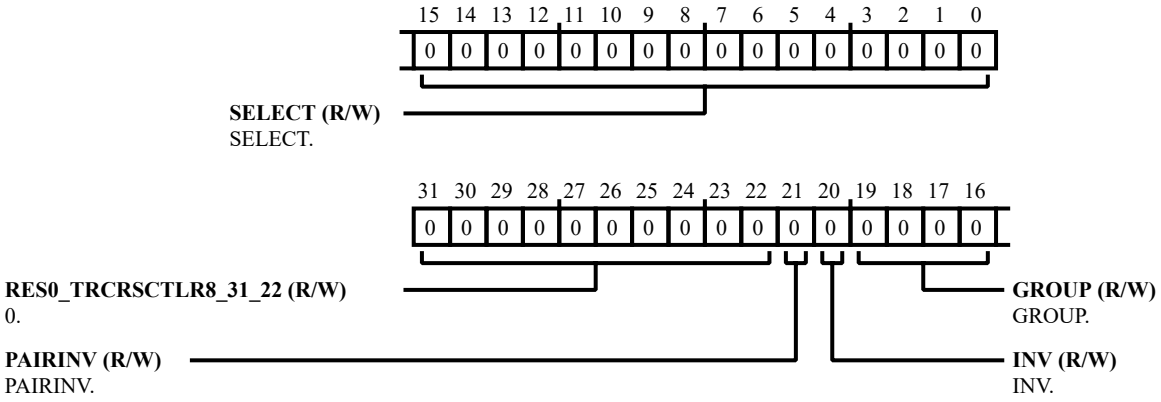


Figure 48-920: A55_C2_TRCRSCTLR8 Register Diagram

Table 48-928: A55_C2_TRCRSCTLR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR8_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-928: A55_C2_TRCRSCTLR8 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Resource Selection Control Registers.

Controls the selection of the resources in the trace unit.

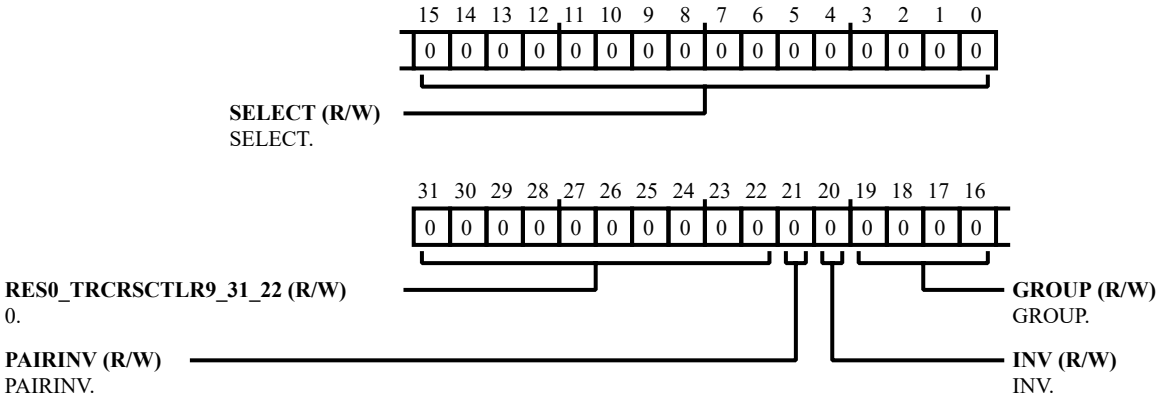


Figure 48-921: A55_C2_TRCRSCTLR9 Register Diagram

Table 48-929: A55_C2_TRCRSCTLR9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:22 (R/W)	RES0_TRCRSCTLR9_31_22	0.. Reserved, RES0.	
21 (R/W)	PAIRINV	PAIRINV.. If n is an even number, controls whether the combined result from a resource pair is inverted: 0 The combined result is not inverted. 1 The combined result is inverted. If n is an odd number, this field is RES0.	
		0	The combined result is not inverted.
		1	The combined result is inverted.
20 (R/W)	INV	INV.. Controls whether the resource that GROUP and SELECT selects is inverted: 0 The selected resource is not inverted. 1 The selected resource is inverted.	
		0	The selected resource is not inverted.
		1	The selected resource is inverted.

Table 48-929: A55_C2_TRCRSCTLR9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GROUP	GROUP.. Selects a group of resources. Possible values are: 0000 For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved. 0001 For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved. 0010 For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved. 0011 For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved. 0100 For SELECT bits 0 to 15, selects single address comparator 0 to 15. 0101 For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved. 0110 For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved. 0111 For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved. All other values are reserved.	
		0	For SELECT bits 0 to 3, selects external input selector 0 to 3; other bits are reserved.
		1	For SELECT bits 0 to 7, selects processor comparator inputs 0 to 7; other bits are reserved.
		2	For SELECT bits 0 to 3, selects counter at zero 0 to 3; for SELECT bits 4 to 7, selects sequencer states 0 to 3; other bits are reserved.
		3	For SELECT bits 0 to 7, selects single-shot comparator control 0 to 7; other bits are reserved.
		4	For SELECT bits 0 to 15, selects single address comparator 0 to 15.
		5	For SELECT bits 0 to 7, selects address range comparator 0 to 7; other bits are reserved.
		6	For SELECT bits 0 to 7, selects Context ID comparator 0 to 7; other bits are reserved.
		7	For SELECT bits 0 to 7, selects VMID comparator 0 to 7; other bits are reserved.
15:0 (R/W)	SELECT	SELECT.. Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group.See the GROUP field description for details.	

Trace Sequencer State Transition Control Registers.

Moves the sequencer state backwards (from state n+1 to state n) when a programmed event occurs, and forwards (from state n to state n+1) when a programmed event occurs.

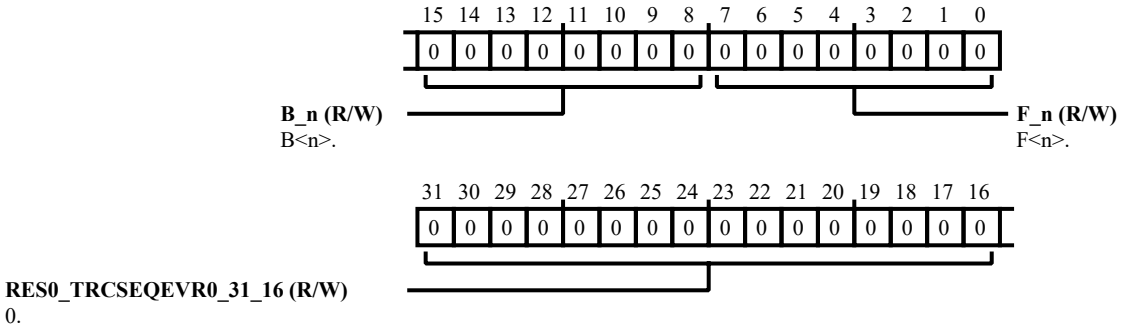


Figure 48-922: A55_C2_TRCSEQEVR0 Register Diagram

Table 48-930: A55_C2_TRCSEQEVR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCSE-QEVR0_31_16	0.. Reserved, RES0.
15:8 (R/W)	B_N	B<n>.. Backward field. Contains an event number. When the event occurs then the sequencer state moves from state n+1 to state n.For example, for TRCSEQEVR2, if B2==0x14 then when event 0x14 occurs, the sequencer moves from state 3 to state 2.
7:0 (R/W)	F_N	F<n>.. Forward field. Contains an event number. When the event occurs then the sequencer state moves from state n to state n+1.For example, for TRCSEQEVR1, if F1==0x12 then when event 0x12 occurs, the sequencer moves from state 1 to state 2.

Trace Sequencer State Transition Control Registers.

Moves the sequencer state backwards (from state n+1 to state n) when a programmed event occurs, and forwards (from state n to state n+1) when a programmed event occurs.

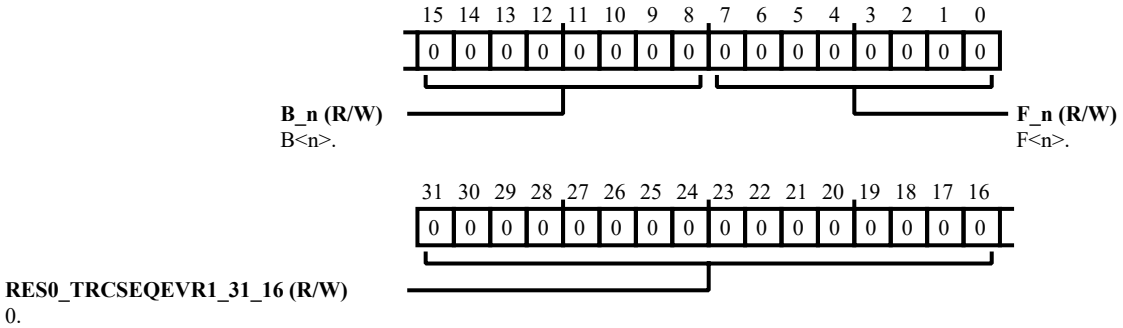


Figure 48-923: A55_C2_TRCSEQEVR1 Register Diagram

Table 48-931: A55_C2_TRCSEQEVR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCSE-QEVR1_31_16	0.. Reserved, RES0.
15:8 (R/W)	B_N	B<n>.. Backward field. Contains an event number. When the event occurs then the sequencer state moves from state n+1 to state n.For example, for TRCSEQEVR2, if B2==0x14 then when event 0x14 occurs, the sequencer moves from state 3 to state 2.
7:0 (R/W)	F_N	F<n>.. Forward field. Contains an event number. When the event occurs then the sequencer state moves from state n to state n+1.For example, for TRCSEQEVR1, if F1==0x12 then when event 0x12 occurs, the sequencer moves from state 1 to state 2.

Trace Sequencer State Transition Control Registers.

Moves the sequencer state backwards (from state n+1 to state n) when a programmed event occurs, and forwards (from state n to state n+1) when a programmed event occurs.

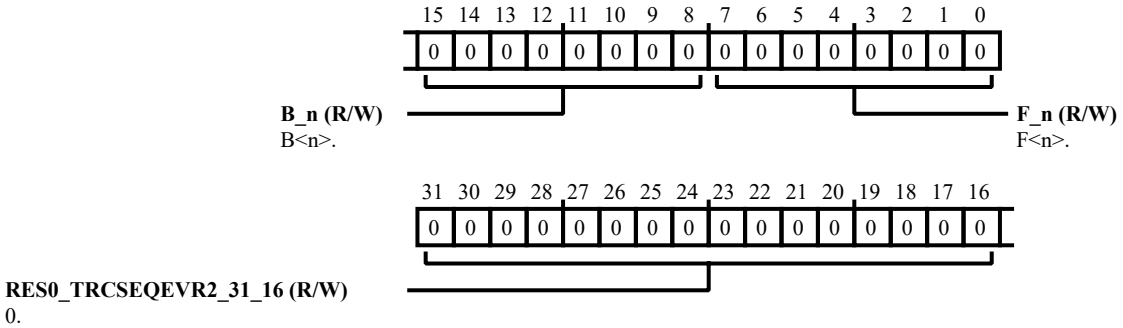


Figure 48-924: A55_C2_TRCSEQEVR2 Register Diagram

Table 48-932: A55_C2_TRCSEQEVR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RES0_TRCSE-QEVR2_31_16	0.. Reserved, RES0.
15:8 (R/W)	B_N	B<n>.. Backward field. Contains an event number. When the event occurs then the sequencer state moves from state n+1 to state n.For example, for TRCSEQEVR2, if B2==0x14 then when event 0x14 occurs, the sequencer moves from state 3 to state 2.
7:0 (R/W)	F_N	F<n>.. Forward field. Contains an event number. When the event occurs then the sequencer state moves from state n to state n+1.For example, for TRCSEQEVR1, if F1==0x12 then when event 0x12 occurs, the sequencer moves from state 1 to state 2.

Trace Sequencer Reset Control Register.

Moves the sequencer to state 0 when a programmed event occurs.

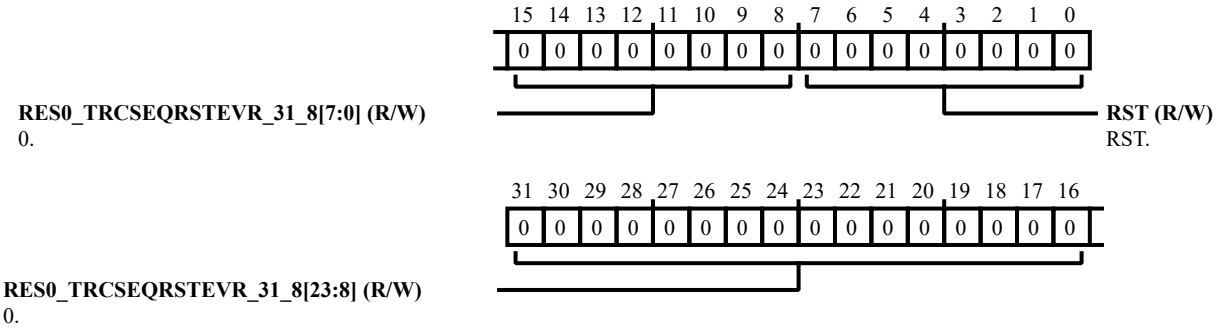


Figure 48-925: A55_C2_TRCSEQRSTEVR Register Diagram

Table 48-933: A55_C2_TRCSEQRSTEVR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_TRCSEQR-STEVR_31_8	0.. Reserved, RES0.
7:0 (R/W)	RST	RST.. Contains an event number. When the event occurs then the sequencer state moves to state 0.

Trace Sequencer State Register.

Use this to set, or read, the sequencer state.

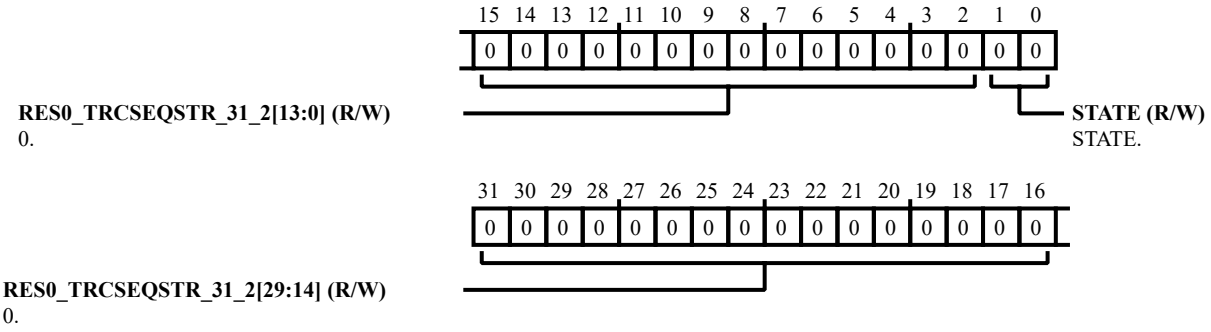


Figure 48-926: A55_C2_TRCSEQSTR Register Diagram

Table 48-934: A55_C2_TRCSEQSTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/W)	RES0_TRCSEQSTR_31_2	0.. Reserved, RES0.	
1:0 (R/W)	STATE	STATE.. Sets or returns the state of the sequencer: 00 State 0. 01 State 1. 10 State 2. 11 State 3.	
		0	State 0.
		1	State 1.
		2	State 2.
		3	State 3.

Trace Single-Shot Comparator Control Registers.

Controls the corresponding single-shot comparator resource.

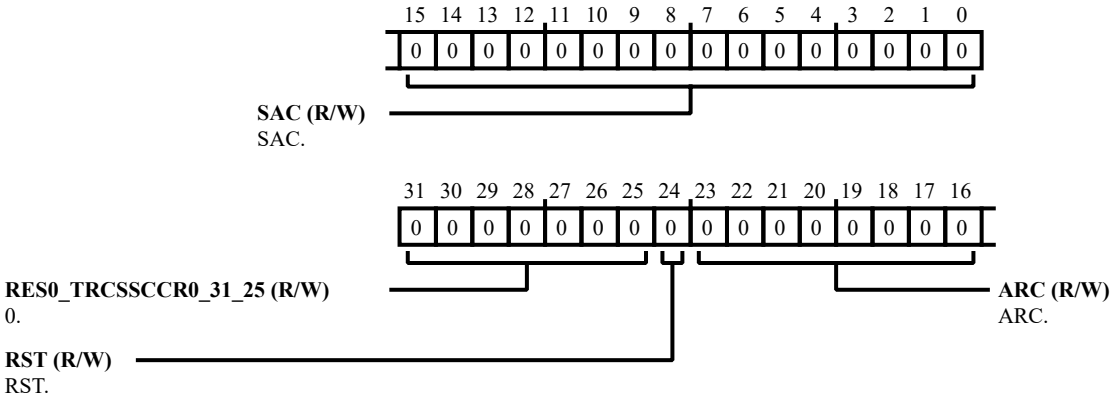


Figure 48-927: A55_C2_TRCSSCCR0 Register Diagram

Table 48-935: A55_C2_TRCSSCCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:25 (R/W)	RES0_TRCSSCCR0_31_25	0.. Reserved, RES0.	
24 (R/W)	RST	RST.. Controls whether the single-shot comparator resource is reset when it fires. 0 When the single-shot comparator resource fires, it is not reset. 1 When the single-shot comparator resource fires, it is reset. This enables the single-shot comparator resource to fire multiple times.	
		0	When the single-shot comparator resource fires, it is not reset.
		1	When the single-shot comparator resource fires, it is reset. This enables the single-shot comparator resource to fire multiple times.

Table 48-935: A55_C2_TRCSSCCR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	ARC	ARC.. Selects one or more address range comparators for single-shot control.Each bit represents an address range comparator pair, so bit[n-16] controls the selection of address range comparator pair n-16. If bit[n-16] is: 0 The address range comparator pair n-16 is not selected for single-shot control. 1 The address range comparator pair n-16 is selected for single-shot control. The width of this field is IMPLEMENTATION DEFINED. The field contains a number of implemented bits equal to TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The address range comparator pair n-16 is not selected for single-shot control.
		1	The address range comparator pair n-16 is selected for single-shot control.
15:0 (R/W)	SAC	SAC.. Selects one or more single address comparators for single-shot control.Each bit represents a single address comparator, so bit[n] controls the selection of single address comparator n. If bit[n] is: 0 The single address comparator n, is not selected for single-shot control. 1 The single address comparator n, is selected for single-shot control. The width of this field is IMPLEMENTATION DEFINED. The field contains a number of implemented bits equal to 2 x TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The single address comparator n, is not selected for single-shot control.
		1	The single address comparator n, is selected for single-shot control.

Trace Single-Shot Comparator Status Registers.

Returns the status of the corresponding single-shot comparator.

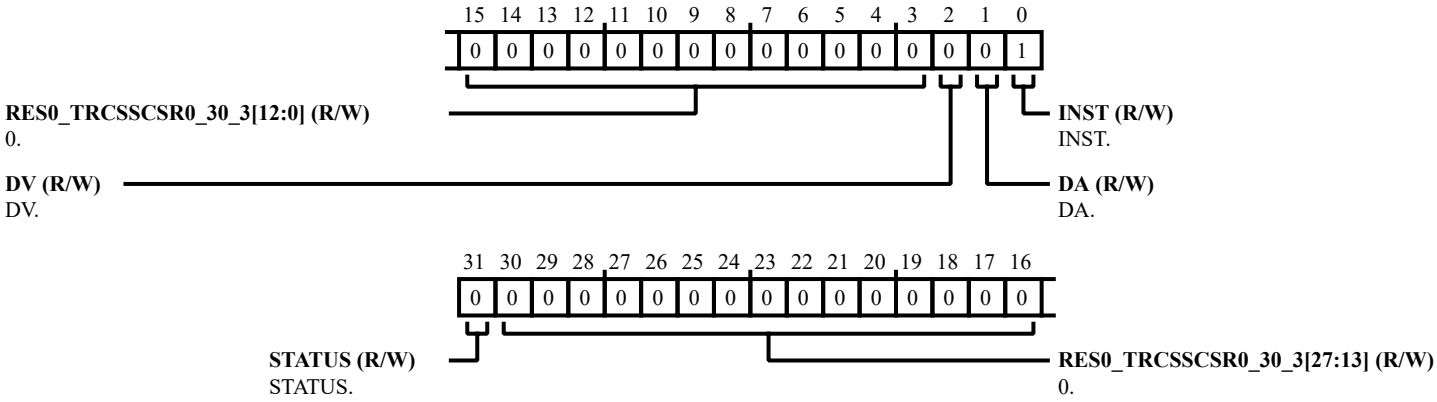


Figure 48-928: A55_C2_TRCSSCSR0 Register Diagram

Table 48-936: A55_C2_TRCSSCSR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	STATUS	STATUS.. Single-shot status bit. Indicates if any of the comparators that TRCSSCCRn.SAC or TRCSSCCRn.ARC selects have matched: 0 No match has occurred. 1 One or more matches has occurred. If TRCSSCCRn.RST==0 then there is only one match and no more matches are possible, and software must reset this bit to 0 to re-enable the single-shot control. STATUS must be written to set an initial state when configuring the trace unit, if the single-shot comparator is to be used.	
		0	No match has occurred.
		1	One or more matches has occurred. If TRCSSCCRn.RST==0 then there is only one match and no more matches are possible, and software must reset this bit to 0 to re-enable the single-shot control.
30:3 (R/W)	RES0_TRCSSCSR0_30_3	0.. Reserved, RES0.	

Table 48-936: A55_C2_TRCSSCSR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	DV	DV.. Data value comparator support bit. Indicates if the trace unit supports data address with data value comparisons: 0 Single-shot data address with data value comparisons are not supported. 1 Single-shot data address with data value comparisons are supported.	
		0	Single-shot data address with data value comparisons are not supported.
		1	Single-shot data address with data value comparisons are supported.
1 (R/W)	DA	DA.. Data address comparator support bit. Indicates if the trace unit supports data address comparisons: 0 Single-shot data address comparisons are not supported. 1 Single-shot data address comparisons are supported.	
		0	Single-shot data address comparisons are not supported.
		1	Single-shot data address comparisons are supported.
0 (R/W)	INST	INST.. Instruction address comparator support bit. Indicates if the trace unit supports instruction address comparisons: 0 Single-shot instruction address comparisons are not supported. 1 Single-shot instruction address comparisons are supported.	
		0	Single-shot instruction address comparisons are not supported.
		1	Single-shot instruction address comparisons are supported.

Trace Stall Control Register.

Trace Stall Control Register.

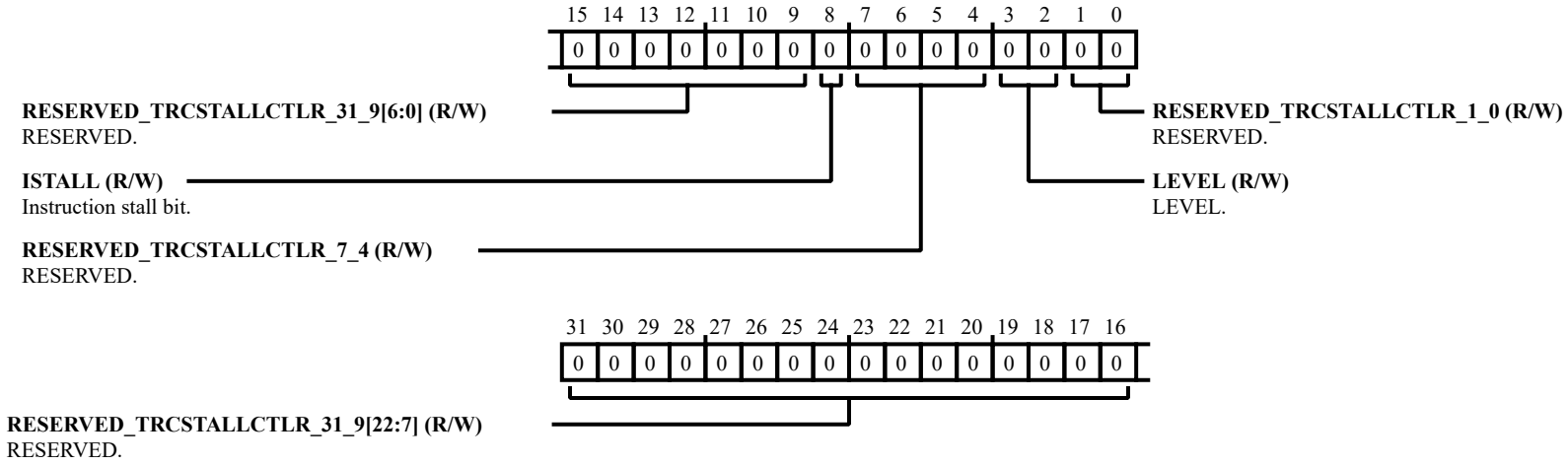


Figure 48-929: A55_C2_TRCSTALLCTLR Register Diagram

Table 48-937: A55_C2_TRCSTALLCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:9 (R/W)	RESERVED_TRCSTALLCTLR_31_9	RESERVED.. Reserved RES0.	
		0	Reserved.
8 (R/W)	ISTALL	Instruction stall bit.. Controls if the trace unit can stall the processor when the instruction trace buffer space is less than LEVEL.	
		0	The trace unit does not stall the processor.
		1	The trace unit can stall the processor.
7:4 (R/W)	RESERVED_TRCSTALLCTLR_7_4	RESERVED.. Reserved RES0.	
		0	Reserved.
3:2 (R/W)	LEVEL	LEVEL.. The field can support 4 monotonic levels from 0b00 to 0b11.	
		0	Minimum invasion. This setting has a greater risk of an ETM trace unit FIFO overflow.
		3	Maximum invasion occurs but there is less risk of a FIFO overflow.

Table 48-937: A55_C2_TRCSTALLCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	RESERVED_TRCSTALLCTLR_1_0	RESERVED.. Reserved RES0.	
		0	Reserved.

Trace Status Register.

Returns the trace unit status.

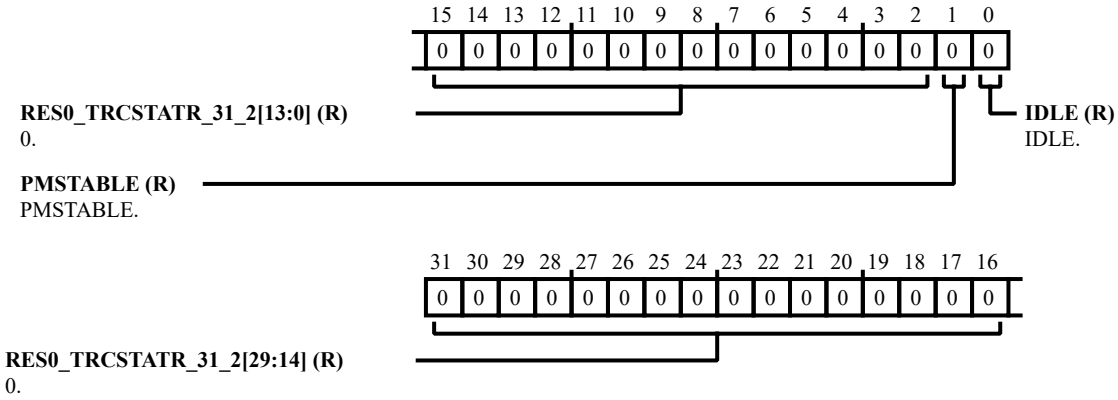


Figure 48-930: A55_C2_TRCSTATR Register Diagram

Table 48-938: A55_C2_TRCSTATR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	RES0_TRCSTATR_31_2	0.. Reserved, RES0.	
1 (R/NW)	PMSTABLE	PMSTABLE.. Programmer's model stable bit: 0 The programmer's model is not stable. 1 The programmer's model is stable. When polled, the trace unit trace registers return stable data. The programmer's model is stable when all of the following are true:TRCPRGCTLR.EN==0 or the OS Lock is locked.Reads from trace unit registers return stable data.	
		0	The programmer's model is not stable.
		1	The programmer's model is stable. When polled, the trace unit trace registers return stable data.
0 (R/NW)	IDLE	IDLE.. Idle status bit: 0 The trace unit is not idle. 1 The trace unit is idle. The trace unit is idle when all of the following are true:TRCPRGCTLR.EN==0 or the OS Lock is locked.The trace unit is drained of any trace.With the exception of the programming interfaces, all external interfaces on the trace unit are quiescent.	
		0	The trace unit is not idle.
		1	The trace unit is idle.

Trace Synchronization Period Register.

Controls how often periodic trace synchronization requests occur.

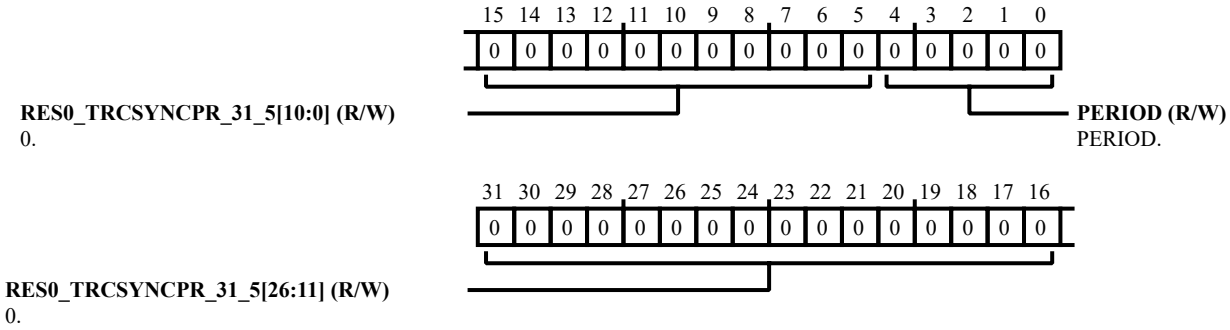


Figure 48-931: A55_C2_TRCSYNCPR Register Diagram

Table 48-939: A55_C2_TRCSYNCPR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:5 (R/W)	RES0_TRCSYNCPR_31_5	0.. Reserved, RES0.	
4:0 (R/W)	PERIOD	PERIOD.. Controls how many bytes of trace, the sum of instruction and data, that a trace unit can generate before a periodic trace synchronization request occurs. The number of bytes is always a power of two and the permitted values are: 00000 Periodic trace synchronization requests are disabled. This setting does not disable other types of trace synchronization request. 01000 Periodic trace synchronization request occurs after 2^8, or 256, bytes of trace. 01001 Periodic trace synchronization request occurs after 2^9, or 512, bytes of trace. 01010 Periodic trace synchronization request occurs after 2^10, or 1024, bytes of trace. and so on up to 0b10100, for which the request occurs after 2^20, or 1048576, bytes of trace.Values between 0b00001 and 0b001111 are reserved, as are values from 0b10101 onwards.	
		0	Periodic trace synchronization requests are disabled. This setting does not disable other types of trace synchronization request.
		8	Periodic trace synchronization request occurs after 2^8, or 256, bytes of trace.
		9	Periodic trace synchronization request occurs after 2^9, or 512, bytes of trace.
		10	Periodic trace synchronization request occurs after 2^10, or 1024, bytes of trace.

Trace Trace ID Register.

Sets the trace ID for instruction trace. If data trace is enabled then it also sets the trace ID for data trace, to (trace ID for instruction trace) + 1.

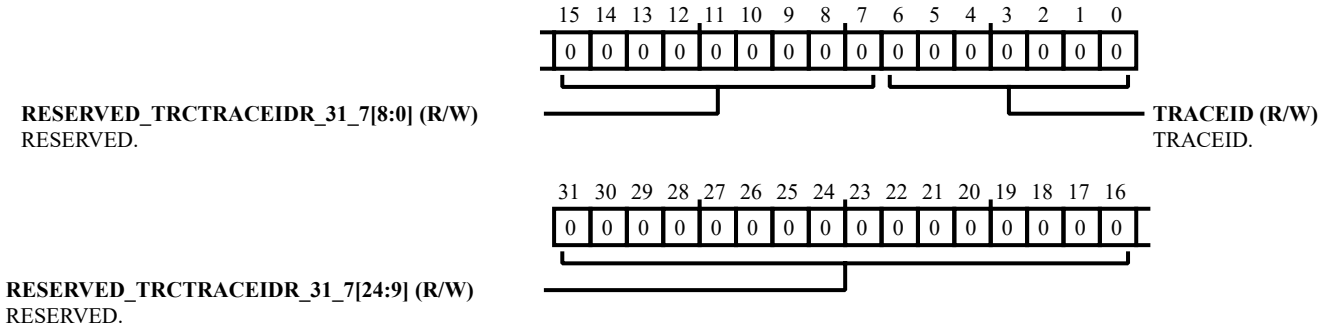


Figure 48-932: A55_C2_TRCTRACEIDR Register Diagram

Table 48-940: A55_C2_TRCTRACEIDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:7 (R/W)	RESERVED_TRCTRA-CEIDR_31_7	RESERVED.. Reserved RES0.	
		0	Reserved.
6:0 (R/W)	TRACEID	TRACEID.. Trace ID value. When only instruction tracing is enabled this provides the trace ID.	

Trace Global Timestamp Control Register.

Controls the insertion of global timestamps in the trace streams.

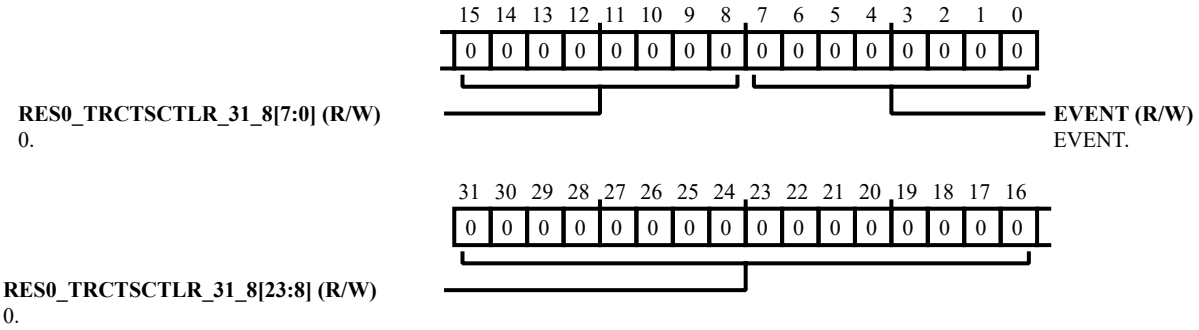


Figure 48-933: A55_C2_TRCTSCTLR Register Diagram

Table 48-941: A55_C2_TRCTSCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:8 (R/W)	RES0_TRCTSCTLR_31_8	0.. Reserved, RES0.
7:0 (R/W)	EVENT	EVENT.. An event selector. When the selected event is triggered, the trace unit inserts a global timestamp into the trace streams.

Trace ViewInst Main Control Register.

Controls instruction trace filtering.

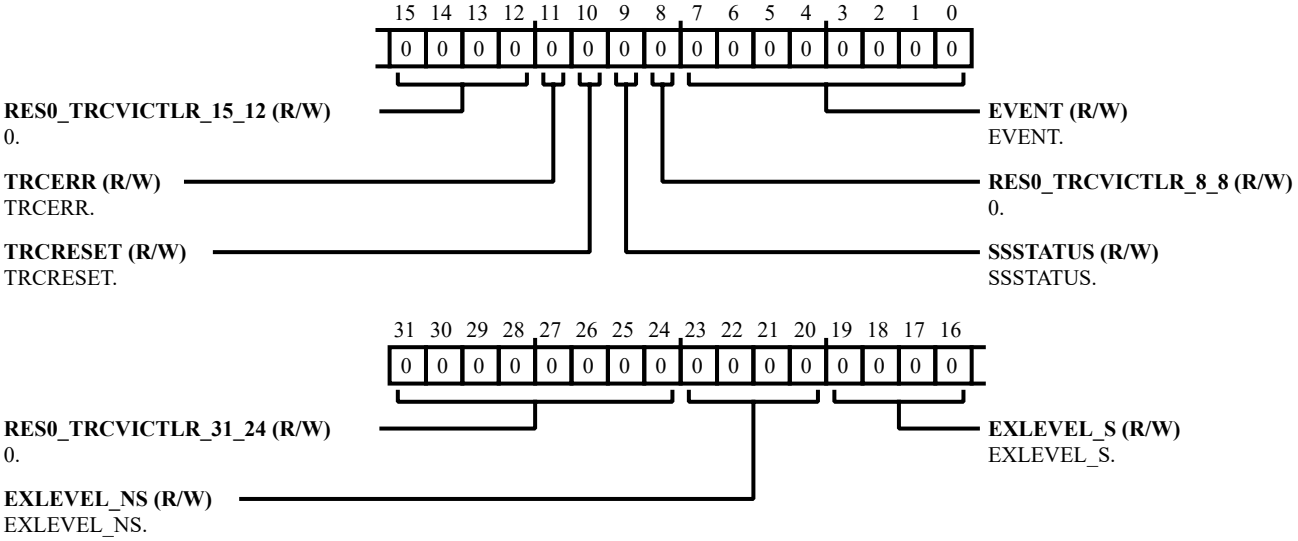


Figure 48-934: A55_C2_TRCVICTLR Register Diagram

Table 48-942: A55_C2_TRCVICTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	RES0_TRCVICTLR_31_24	0.. Reserved, RES0.	
23:20 (R/W)	EXLEVEL_NS	EXLEVEL_NS.. In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding exception level: 0 The trace unit generates instruction trace, in Non-secure state, for exception level n. 1 The trace unit does not generate instruction trace, in Non-secure state, for exception level n. The exception levels are:Bit[20]Exception level 0.Bit[21]Exception level 1.Bit[22]Exception level 2.Bit[23]RAZ/WI. EXLEVEL_NS[3] is never implemented.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_NS. If instruction tracing is not implemented for a given exception level, the corresponding bit in this field is not implemented. Unimplemented bits are RAZ/WI.	
		0	The trace unit generates instruction trace, in Non-secure state, for exception level n.
		1	The trace unit does not generate instruction trace, in Non-secure state, for exception level n.

Table 48-942: A55_C2_TRCVICTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	EXLEVEL_S	EXLEVEL_S.. In Secure state, each bit controls whether instruction tracing is enabled for the corresponding exception level: 0 The trace unit generates instruction trace, in Secure state, for exception level n. 1 The trace unit does not generate instruction trace, in Secure state, for exception level n. The exception levels are:Bit[16]Exception level 0.Bit[17]Exception level 1.Bit[18]RAZ/WI. EXLEVEL_S[2] is never implemented.Bit[19]Exception level 3.The content of the field is IMPLEMENTATION DEFINED and is set by the value of TRCIDR3.EXLEVEL_S. If instruction tracing is not implemented for a given exception level, the corresponding bit in this field is not implemented. Unimplemented bits are RAZ/WI.	
		0	The trace unit generates instruction trace, in Secure state, for exception level n.
		1	The trace unit does not generate instruction trace, in Secure state, for exception level n.
15:12 (R/W)	RES0_TRCVICTLR_15_12	0.. Reserved, RES0.	
11 (R/W)	TRCERR	TRCERR.. If TRCIDR3.TRCERR==1, this bit controls whether a trace unit must trace a system error exception: 0 The trace unit does not trace a system error exception unless it traces the exception or instruction immediately prior to the system error exception. 1 The trace unit always traces a system error exception. If TRCIDR3.TRCERR==0, this bit is RES0.	
		0	The trace unit does not trace a system error exception unless it traces the exception or instruction immediately prior to the system error exception.
		1	The trace unit always traces a system error exception.
10 (R/W)	TRCRESET	TRCRESET.. Controls whether a trace unit must trace a Reset exception: 0 The trace unit does not trace a Reset exception unless it traces the exception or instruction immediately prior to the Reset exception. 1 The trace unit always traces a Reset exception.	
		0	The trace unit does not trace a Reset exception unless it traces the exception or instruction immediately prior to the Reset exception.
		1	The trace unit always traces a Reset exception.

Table 48-942: A55_C2_TRCVICTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SSSTATUS	SSSTATUS.. IF TRCIDR4.NUMACPAIRS>0 or TRCIDR.NUMPC>0, this bit returns the status of the start-stop logic: 0 The start-stop logic is in the stopped state. 1 The start-stop logic is in the started state. The bit only returns stable data when TRCSTATR.PMSTABLE==1.Before software enables the trace unit, TRCPRGCTLR.EN==1, it must write to this bit to set the initial state of the start-stop logic. If the start-stop logic is not used then set this bit to 1. ARM recommends that the value of this bit is set before each trace run begins.If TRCIDR4.NUMACPAIRS==0 and TRCIDR4.NUMPC==0, this bit is RES0, indicating that the start-stop logic is not implemented.	
		0	The start-stop logic is in the stopped state.
		1	The start-stop logic is in the started state.
8 (R/W)	RES0_TRCVICTLR_8_8	0.. Reserved, RES0.	
7:0 (R/W)	EVENT	EVENT.. An event selector. [TODO: Add the bit assignments for EVENT fields into the descriptions directly?].	

Trace ViewInst Include/Exclude Control Register.

Use this to set or read the address range comparators for the ViewInst include and exclude controls.

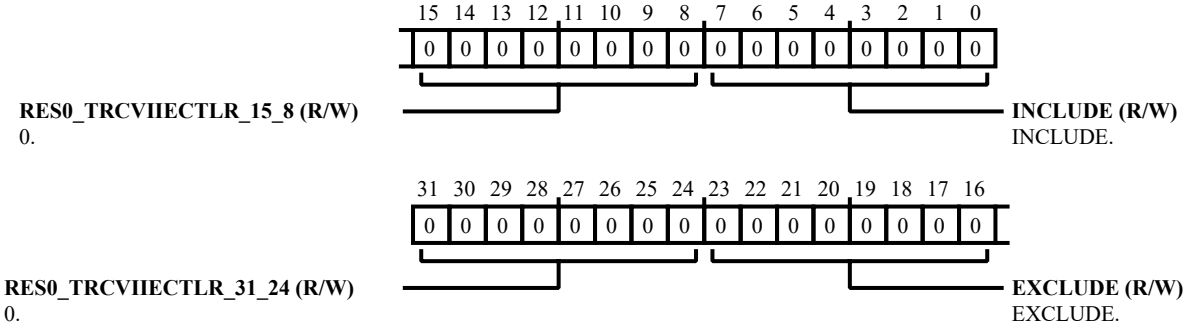


Figure 48-935: A55_C2_TRCVIIECTLR Register Diagram

Table 48-943: A55_C2_TRCVIIECTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	RES0_TRCVIIECTLR_31_24	0.. Reserved, RES0.	
23:16 (R/W)	EXCLUDE	EXCLUDE.. 0 1 The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	
		1	
15:8 (R/W)	RES0_TRCVIIECTLR_15_8	0.. Reserved, RES0.	

Table 48-943: A55_C2_TRCVIIECTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W)	INCLUDE	INCLUDE.. Include range field. Selects which address range comparator pairs are in use with ViewInst include control. Each bit represents an address range comparator pair, so bit[m] controls the selection of address range comparator pair m. If bit[m] is: 0 The address range that address range comparator pair m defines is not selected for ViewInst include control. 1 The address range that address range comparator pair m defines is selected for ViewInst include control. The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.Selecting no include comparators indicates that all instructions are included by default. The exclude control then indicates which ranges are excluded.	
		0	The address range that address range comparator pair m defines is not selected for ViewInst include control.
		1	The address range that address range comparator pair m defines is selected for ViewInst include control.

Trace ViewInst Start/Stop Control Register.

Use this to set, or read, the single address comparators that control the ViewInst start-stop logic. The start-stop logic is active for an instruction which causes a start and remains active up to and including an instruction which causes a stop, and then the start-stop logic becomes inactive.

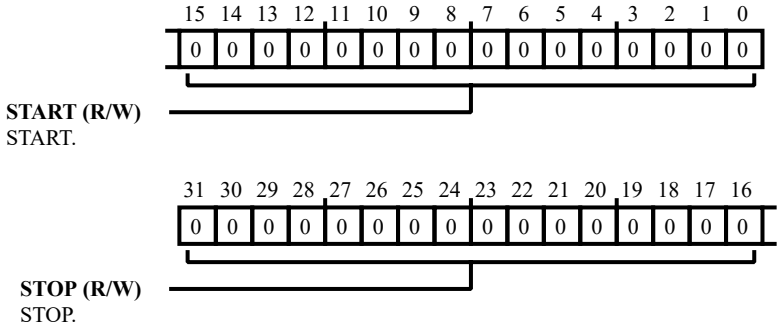


Figure 48-936: A55_C2_TRCVISSCTLR Register Diagram

Table 48-944: A55_C2_TRCVISSCTLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/W)	STOP	STOP.. Selects which single address comparators are in use with ViewInst start-stop control, for the purpose of stopping trace. Each bit represents a single address comparator, so bit[m] controls the selection of single address comparator m-16. If bit[m] is: 0 The single address comparator m-16 is not selected as a stop resource. 1 The single address comparator m-16 is selected as a stop resource. The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of 2 x TRCIDR4.NUMACPAIRS. Unimplemented bits are RAZ/WI.	
		0	The single address comparator m-16 is not selected as a stop resource.
		1	The single address comparator m-16 is selected as a stop resource.

Table 48-944: A55_C2_TRCVISSCTLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:0 (R/W)	START	START.. Selects which single address comparators are in use with ViewInst start-stop control, for the purpose of starting trace. Each bit represents a single address comparator, so bit[n] controls the selection of single address comparator n. If bit[n] is: 0 The single address comparator n is not selected as a start resource. 1 The single address comparator n is selected as a start resource. The implemented width of the field, n, is IMPLEMENTATION DEFINED and is set by the value of 2 x TRCIDR4.NUMAC-PAIRS. Unimplemented bits are RAZ/WI.	
		0	The single address comparator n is not selected as a start resource.
		1	The single address comparator n is selected as a start resource.

Trace VMID Comparator Value Registers.

Contains a VMID value.

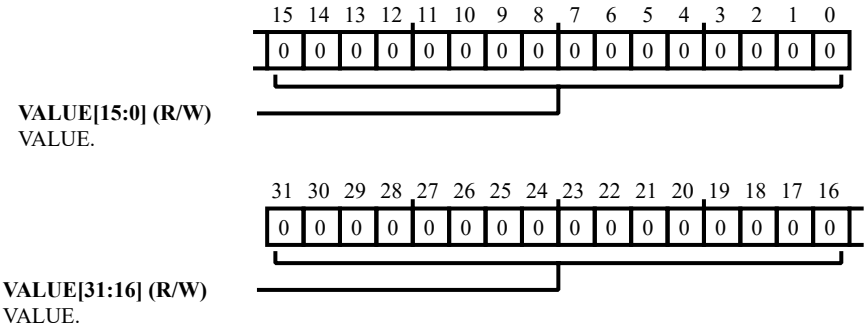


Figure 48-937: A55_C2_TRCVMIDCVR0_31_0 Register Diagram

Table 48-945: A55_C2_TRCVMIDCVR0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. VMID value. The implemented width of this field is IMPLEMENTATION DEFINED, and is set by TRCIDR2.VMIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the VMID is zero until the processor updates the VMID.

Trace VMID Comparator Value Registers.

Contains a VMID value.

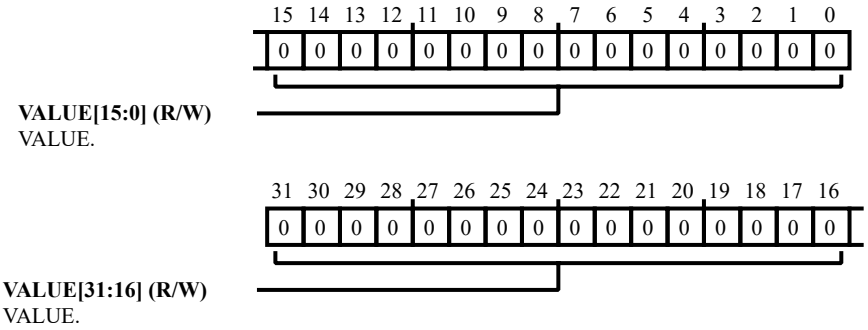


Figure 48-938: A55_C2_TRCVMIDCVR0_63_32 Register Diagram

Table 48-946: A55_C2_TRCVMIDCVR0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	VALUE.. VMID value. The implemented width of this field is IMPLEMENTATION DEFINED, and is set by TRCIDR2.VMIDSIZE. Unimplemented bits are RAZ/WI.After a processor reset, the ETM architecture assumes that the VMID is zero until the processor updates the VMID.

Performance Monitors Cycle Counter Filter Register.

Determines the modes in which the Cycle Counter, PMCCNTR_EL0, increments.

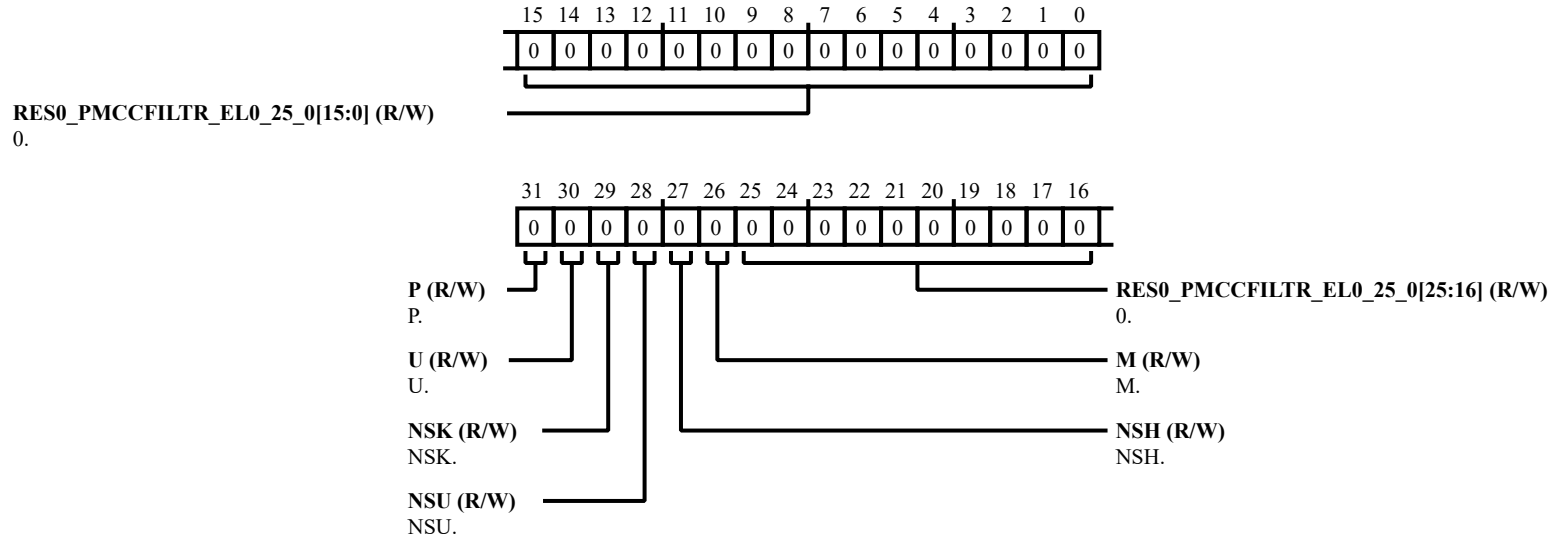


Figure 48-939: A55_C2_PMCCFILTR_EL0 Register Diagram

Table 48-947: A55_C2_PMCCFILTR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged modes filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count cycles in EL1. 1 Do not count cycles in EL1.	
		0	Count cycles in EL1.
		1	Do not count cycles in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count cycles in EL0. 1 Do not count cycles in EL0.	
		0	Count cycles in EL0.
		1	Do not count cycles in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Non-secure EL1 are counted. Otherwise, cycles in Non-secure EL1 are not counted.	

Table 48-947: A55_C2_PMCCFILTR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, cycles in Non-secure EL0 are counted. Otherwise, cycles in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering bit. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count cycles in EL2. 1 Count cycles in EL2.	
		0	Do not count cycles in EL2.
		1	Count cycles in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMCCFILTR System register.	
25:0 (R/W)	RES0_PMCCFILTR_EL0_25_0	0.. Reserved, RES0.	

Performance Monitors Cycle Counter.

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See for more information.

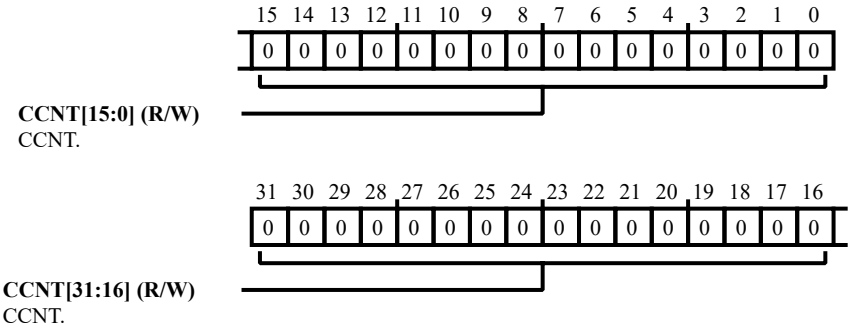


Figure 48-940: A55_C2_PMCCNTR_EL0_31_0 Register Diagram

Table 48-948: A55_C2_PMCCNTR_EL0_31_0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CCNT	CCNT.. Cycle count. Depending on the values of PMCR_EL0.{LC,D}, the cycle count increments in one of the following ways: Every processor clock cycle. Every 64th processor clock cycle. Writing 1 to PMCR_EL0.C sets this field to 0.

Performance Monitors Cycle Counter.

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See for more information.

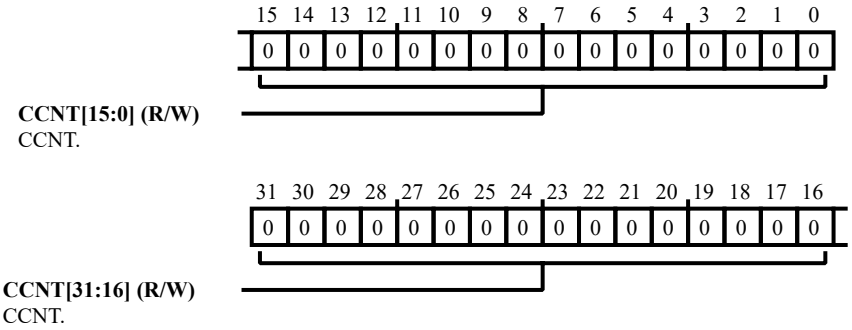


Figure 48-941: A55_C2_PMCCNTR_EL0_63_32 Register Diagram

Table 48-949: A55_C2_PMCCNTR_EL0_63_32 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CCNT	CCNT.. Cycle count. Depending on the values of PMCR_EL0.{LC,D}, the cycle count increments in one of the following ways: Every processor clock cycle. Every 64th processor clock cycle. Writing 1 to PMCR_EL0.C sets this field to 0.

Performance Monitors Common Event Identification register 0.

Defines which common architectural and common microarchitectural feature events in the range 0x000 to 0x01F are implemented. If a particular bit is set to 1, then the event for that bit is implemented. This view of the register has previously been called PMCEID0_EL0.

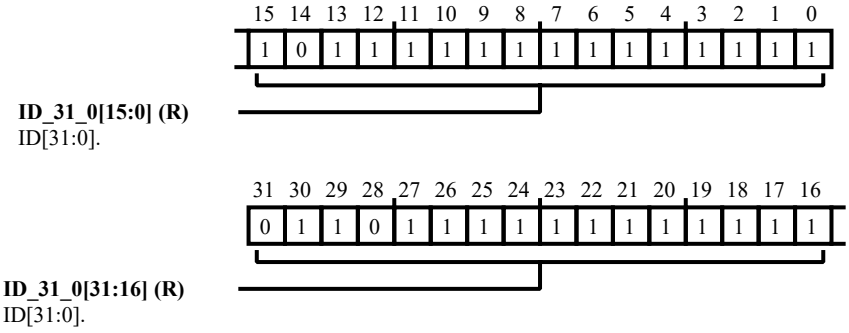


Figure 48-942: A55_C2_PMCEID0 Register Diagram

Table 48-950: A55_C2_PMCEID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_31_0	ID[31:0].. PMCEID0[n] maps to event n. For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Common Event Identification register 1.

Defines which common architectural and common microarchitectural feature events in the range 0x020 to 0x03F are implemented. If a particular bit is set to 1, then the event for that bit is implemented. This view of the register has previously been called PMCEID1_EL0.

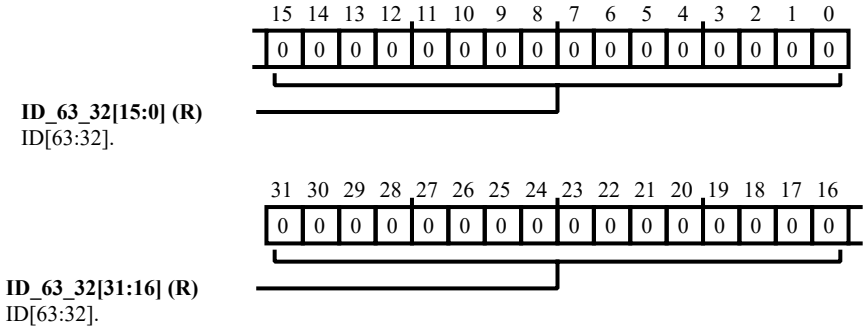


Figure 48-943: A55_C2_PMCEID1 Register Diagram

Table 48-951: A55_C2_PMCEID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_63_32	ID[63:32].. PMCEID1[n] maps to event (n + 32). For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Common Event Identification register 2.

Defines which common architectural and common microarchitectural feature events in the range 0x4000 to 0x401F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

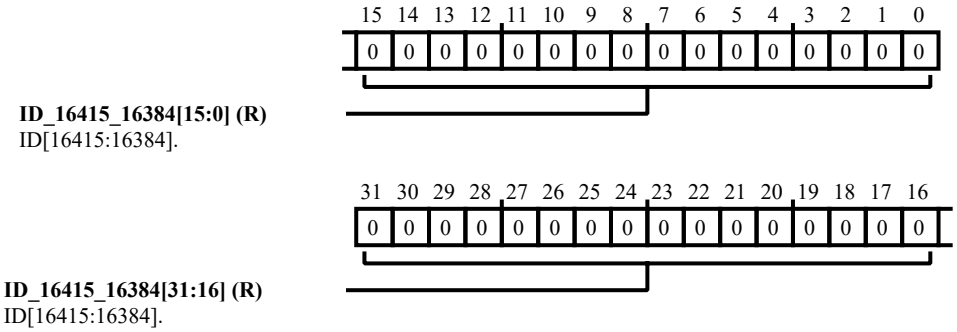


Figure 48-944: A55_C2_PMCEID2 Register Diagram

Table 48-952: A55_C2_PMCEID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_16415_16384	ID[16415:16384].. PMCEID2[31:0] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Common Event Identification register 3.

Defines which common architectural and common microarchitectural feature events in the range 0x4020 to 0x403F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

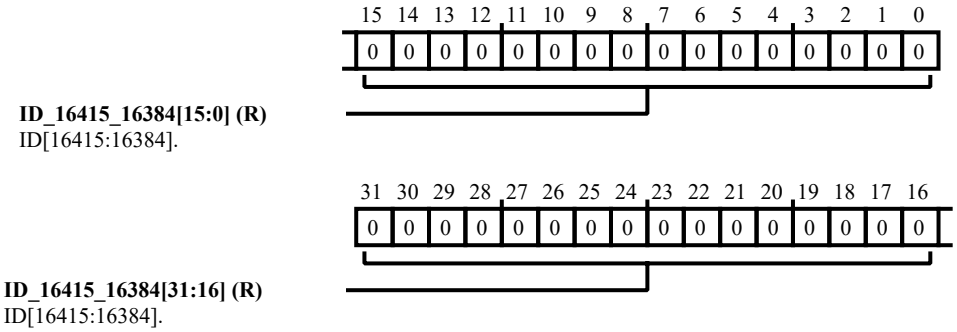


Figure 48-945: A55_C2_PMCEID3 Register Diagram

Table 48-953: A55_C2_PMCEID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	ID_16415_16384	ID[16415:16384].. PMCEID2[31:0] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see . For each bit: 0 The common event is not implemented. 1 The common event is implemented. Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture. Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.	
		0	The common event is not implemented.
		1	The common event is implemented.

Performance Monitors Configuration Register.

Contains PMU-specific configuration data.

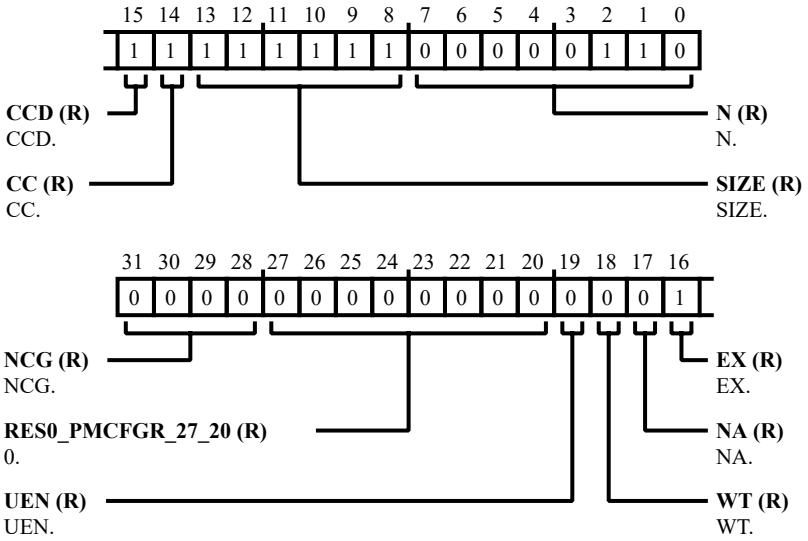


Figure 48-946: A55_C2_PMCFGR Register Diagram

Table 48-954: A55_C2_PMCFGR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	NCG	NCG.. This feature is not supported, so this field is RAZ.	
27:20 (R/NW)	RES0_PMCFGR_27_20	0.. Reserved, RES0.	
19 (R/NW)	UEN	UEN.. User-mode Enable Register supported. PMUSERENR_EL0 is not visible in the external debug interface, so this bit is RAZ.	
18 (R/NW)	WT	WT.. This feature is not supported, so this bit is RAZ.	
17 (R/NW)	NA	NA.. This feature is not supported, so this bit is RAZ.	
16 (R/NW)	EX	EX.. Export supported. Value is IMPLEMENTATION DEFINED. 0 PMCR_EL0.X is RES0. 1 PMCR_EL0.X is read/write.	
		0	PMCR_EL0.X is RES0.
		1	PMCR_EL0.X is read/write.

Table 48-954: A55_C2_PMCFGR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	CCD	CCD.. Cycle counter has prescale. This is RES1 if AArch32 is supported at any EL, and RAZ otherwise. 0 PMCR_EL0.D is RES0. 1 PMCR_EL0.D is read/write.	
		0	PMCR_EL0.D is RES0.
		1	PMCR_EL0.D is read/write.
14 (R/NW)	CC	CC.. Dedicated cycle counter (counter 31) supported. This bit is RAO.	
13:8 (R/NW)	SIZE	SIZE.. Size of counters. This field determines the spacing of counters in the memory-map. In ARMv8 the counters are at doubleword-aligned addresses, and the largest counter is 64-bits, so this field is 0b111111.	
7:0 (R/NW)	N	N.. Number of counters implemented in addition to the cycle counter, PMCCNTR_EL0. The maximum number of event counters is 31. 00000000 Only PMCCNTR_EL0 implemented. 00000001 PMCCNTR_EL0 plus one event counter implemented. and so on up to 0b00011111, which indicates PMCCNTR_EL0 and 31 event counters implemented.	
		0	Only PMCCNTR_EL0 implemented.
		1	PMCCNTR_EL0 plus one event counter implemented.

Performance Monitors Count Enable Clear register.

Disables the Cycle Count Register, PMCCNTR_EL0, and any implemented event counters PMEVCNTR0. Reading this register shows which counters are enabled.

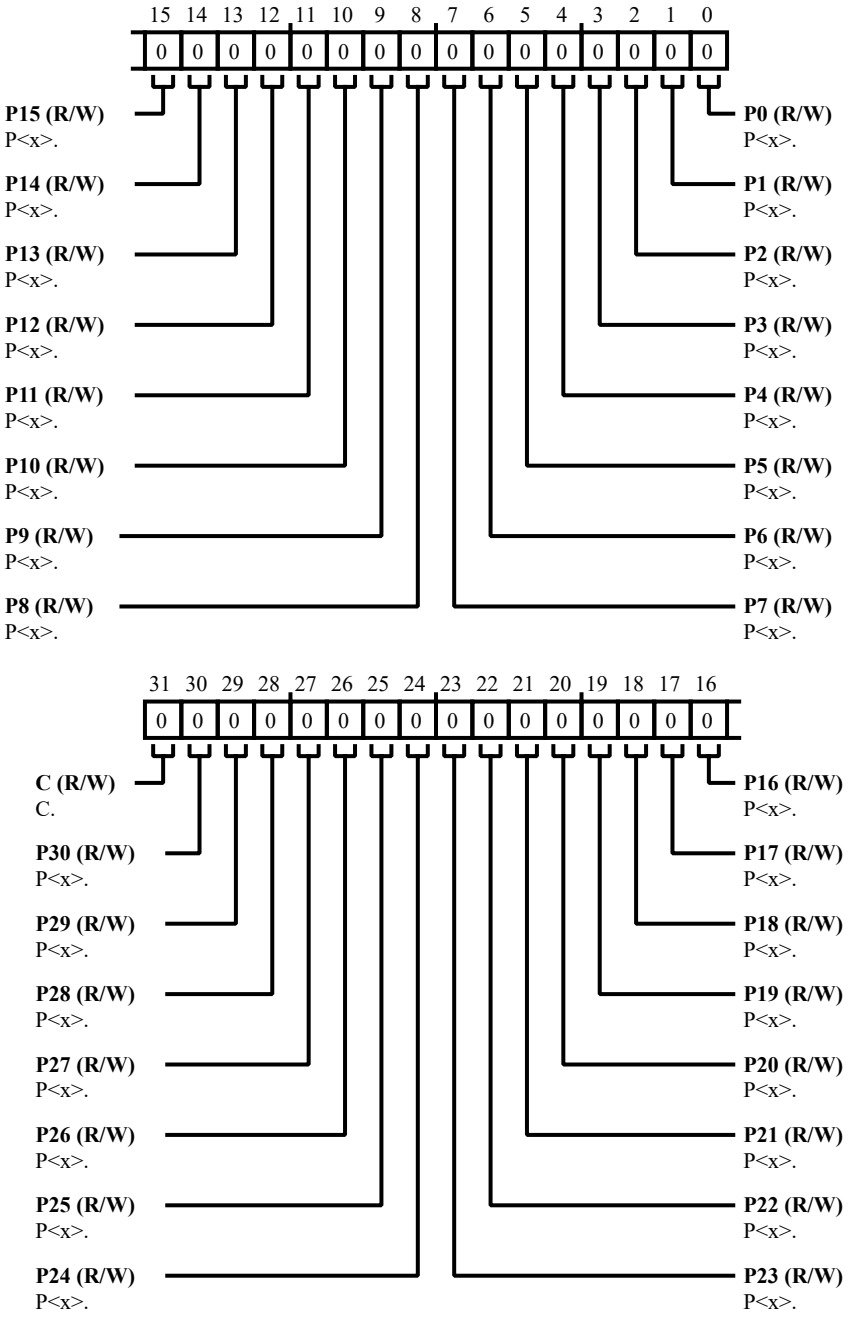


Figure 48-947: A55_C2_PMCNTENCLR_EL0 Register Diagram

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 disable bit. Disables the cycle counter register. Possible values are: 0 When read, means the cycle counter is disabled. When written, has no effect. 1 When read, means the cycle counter is enabled. When written, disables the cycle counter.	
		0	When read, means the cycle counter is disabled. When written, has no effect.
		1	When read, means the cycle counter is enabled. When written, disables the cycle counter.
30 (R/W)	P30	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
29 (R/W)	P29	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
28 (R/W)	P28	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	P27	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
26 (R/W)	P26	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
25 (R/W)	P25	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
24 (R/W)	P24	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	P23	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
22 (R/W)	P22	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
21 (R/W)	P21	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
20 (R/W)	P20	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
18 (R/W)	P18	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
17 (R/W)	P17	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
16 (R/W)	P16	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	P15	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
14 (R/W)	P14	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
13 (R/W)	P13	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
12 (R/W)	P12	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	P11	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
10 (R/W)	P10	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
9 (R/W)	P9	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
8 (R/W)	P8	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
6 (R/W)	P6	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
5 (R/W)	P5	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
4 (R/W)	P4	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Table 48-955: A55_C2_PMCNTENCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	P3	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
2 (R/W)	P2	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
1 (R/W)	P1	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.
0 (R/W)	P0	P<x>.. Event counter disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> is enabled. When written, disables PMEVCNTR<x>.

Performance Monitors Count Enable Set register.

Enables the Cycle Count Register, PMCCNTR_EL0, and any implemented event counters PMEVCNTR0. Reading this register shows which counters are enabled.

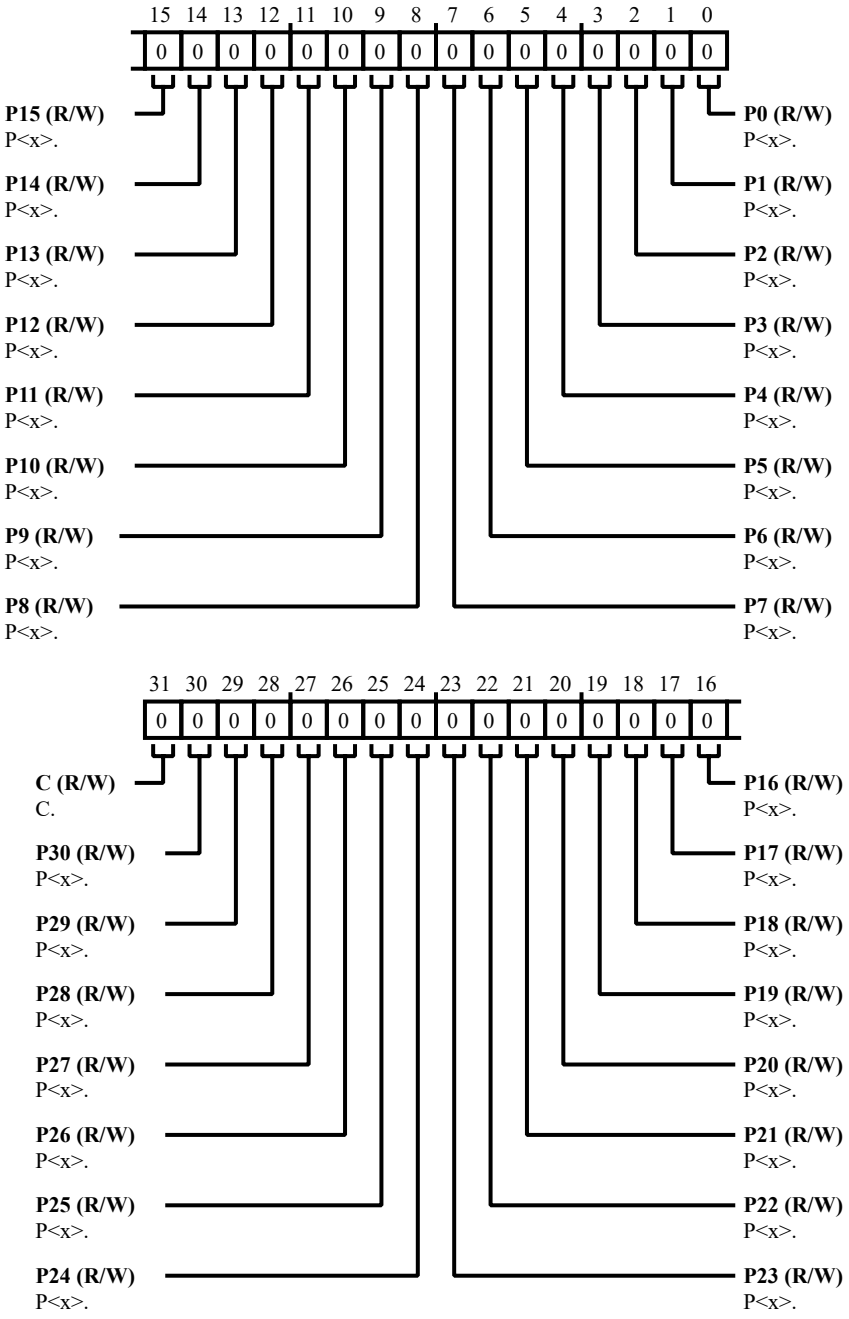


Figure 48-948: A55_C2_PMCNTENSET_EL0 Register Diagram

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 enable bit. Enables the cycle counter register. Possible values are: 0 When read, means the cycle counter is disabled. When written, has no effect. 1 When read, means the cycle counter is enabled. When written, enables the cycle counter.	
		0	When read, means the cycle counter is disabled. When written, has no effect.
		1	When read, means the cycle counter is enabled. When written, enables the cycle counter.
30 (R/W)	P30	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
29 (R/W)	P29	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
28 (R/W)	P28	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	P27	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
26 (R/W)	P26	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
25 (R/W)	P25	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
24 (R/W)	P24	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	P23	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
22 (R/W)	P22	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
21 (R/W)	P21	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
20 (R/W)	P20	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
18 (R/W)	P18	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
17 (R/W)	P17	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
16 (R/W)	P16	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	P15	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
14 (R/W)	P14	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
13 (R/W)	P13	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
12 (R/W)	P12	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	P11	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
10 (R/W)	P10	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
9 (R/W)	P9	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
8 (R/W)	P8	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
6 (R/W)	P6	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
5 (R/W)	P5	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
4 (R/W)	P4	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Table 48-956: A55_C2_PMCNTENSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	P3	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
2 (R/W)	P2	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
1 (R/W)	P1	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.
0 (R/W)	P0	P<x>.. Event counter enable bit for PMEVCNTR<x>. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<x> is disabled. When written, has no effect. 1 When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.	
		0	When read, means that PMEVCNTR<x> is disabled. When written, has no effect.
		1	When read, means that PMEVCNTR<x> event counter is enabled. When written, enables PMEVCNTR<x>.

Performance Monitors Control Register.

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

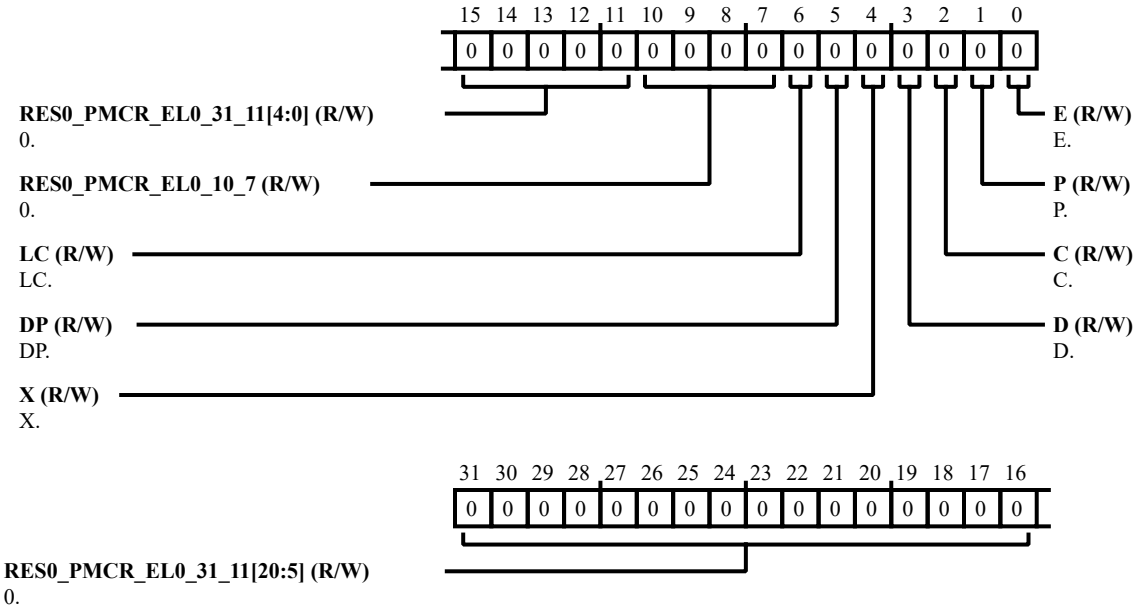


Figure 48-949: A55_C2_PMCR_EL0 Register Diagram

Table 48-957: A55_C2_PMCR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:11 (R/W)	RES0_PMCR_EL0_31_11	0.. Reserved, RAZ/WI.	
10:7 (R/W)	RES0_PMCR_EL0_10_7	0.. Reserved, RES0.	
6 (R/W)	LC	LC.. Long cycle counter enable. Determines which PMCCNTR_EL0 bit generates an overflow recorded by PMOVSR[31]. 0 Cycle counter overflow on increment that changes PMCCNTR_EL0[31] from 1 to 0. 1 Cycle counter overflow on increment that changes PMCCNTR_EL0[63] from 1 to 0. ARM deprecates use of PMCR_EL0.LC = 0. In an AArch64-only implementation, this field is RES1.	
		0	Cycle counter overflow on increment that changes PMCCNTR_EL0[31] from 1 to 0.
		1	Cycle counter overflow on increment that changes PMCCNTR_EL0[63] from 1 to 0.

Table 48-957: A55_C2_PMCR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DP	DP.. Disable cycle counter when event counting is prohibited. The possible values of this bit are: 0 PMCCNTR_EL0, if enabled, counts when event counting is prohibited. 1 PMCCNTR_EL0 does not count when event counting is prohibited. Event counting is prohibited when ProfilingProhibited(IsSecure(),PSTATE.EL) == TRUE. When EL3 is not implemented, this field is RES0: In ARMv8.0. In ARMv8.1, only if EL2 is not implemented. Otherwise this field is RW.	
		0	PMCCNTR_EL0, if enabled, counts when event counting is prohibited.
		1	PMCCNTR_EL0 does not count when event counting is prohibited.
4 (R/W)	X	X.. Enable export of events in an IMPLEMENTATION DEFINED event stream. The possible values of this bit are: 0 Do not export events. 1 Export events where not prohibited. This field enables the exporting of events over an event bus to another device, for example to an OPTIONAL trace macrocell. If the implementation does not include such an event bus then this field is RAZ/WI, otherwise it is an RW field. In an implementation that includes an event bus, no events are exported when counting is prohibited. This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.	
		0	Do not export events.
		1	Export events where not prohibited.
3 (R/W)	D	D.. Clock divider. The possible values of this bit are: 0 When enabled, PMCCNTR_EL0 counts every clock cycle. 1 When enabled, PMCCNTR_EL0 counts once every 64 clock cycles. In an AArch64-only implementation this field is RES0, otherwise it is an RW field.If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.	
		0	When enabled, PMCCNTR_EL0 counts every clock cycle.
		1	When enabled, PMCCNTR_EL0 counts once every 64 clock cycles.

Table 48-957: A55_C2_PMCR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	C	C.. Cycle counter reset. This bit is WO. The effects of writing to this bit are: 0 No action. 1 Reset PMCCNTR_EL0 to zero. This bit is always RAZ. Resetting PMCCNTR_EL0 does not clear the PMCCNTR_EL0 overflow bit to 0.	
		0	No action.
		1	Reset PMCCNTR_EL0 to zero.
1 (R/W)	P	P.. Event counter reset. This bit is WO. The effects of writing to this bit are: 0 No action. 1 Reset all event counters, not including PMCCNTR_EL0, to zero. This bit is always RAZ. Resetting the event counters does not clear any overflow bits to 0.	
		0	No action.
		1	Reset all event counters, not including PMCCNTR_EL0, to zero.
0 (R/W)	E	E.. Enable. The possible values of this bit are: 0 All counters, including PMCCNTR_EL0, are disabled. 1 All counters are enabled by PMCNTENSET_EL0. This bit is RW.	
		0	All counters, including PMCCNTR_EL0, are disabled.
		1	All counters are enabled by PMCNTENSET_EL0.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

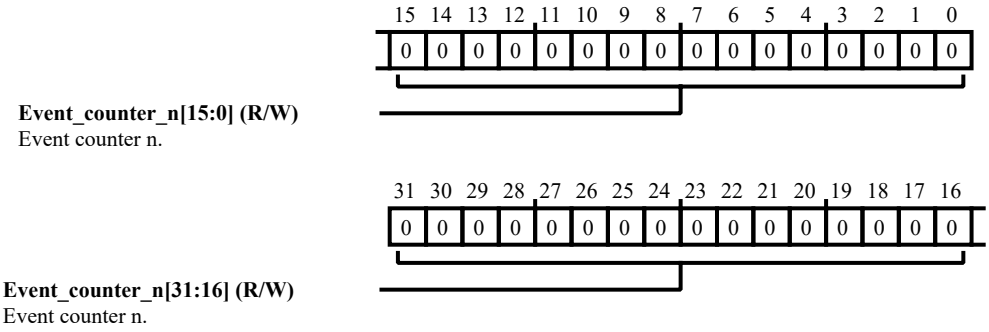


Figure 48-950: A55_C2_PMEVCNTR0_EL0 Register Diagram

Table 48-958: A55_C2_PMEVCNTR0_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

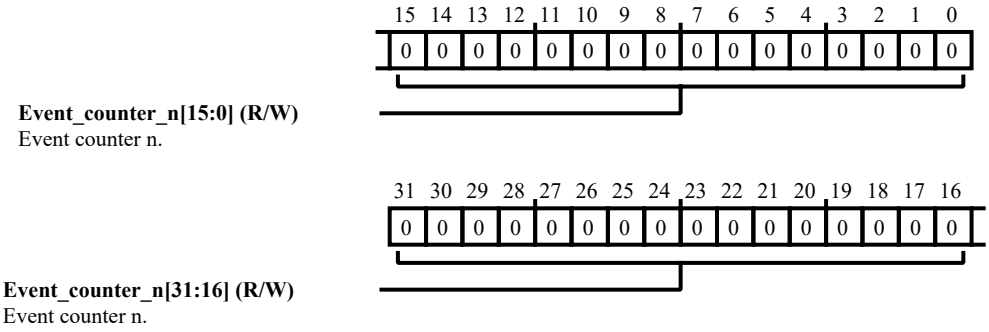


Figure 48-951: A55_C2_PMEVCNTR1_EL0 Register Diagram

Table 48-959: A55_C2_PMEVCNTR1_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

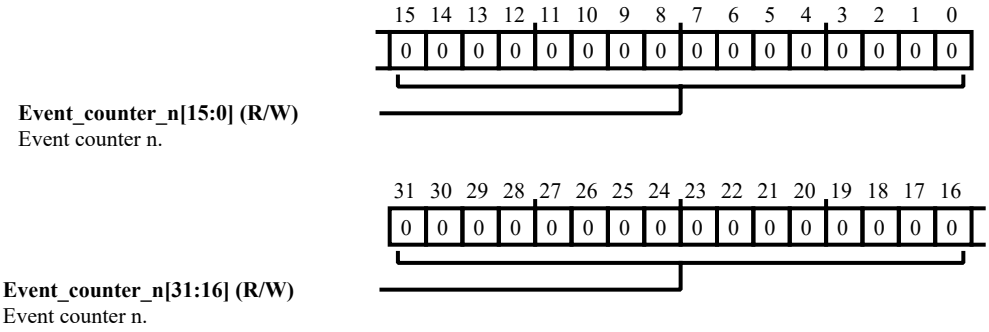


Figure 48-952: A55_C2_PMEVCNTR2_EL0 Register Diagram

Table 48-960: A55_C2_PMEVCNTR2_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

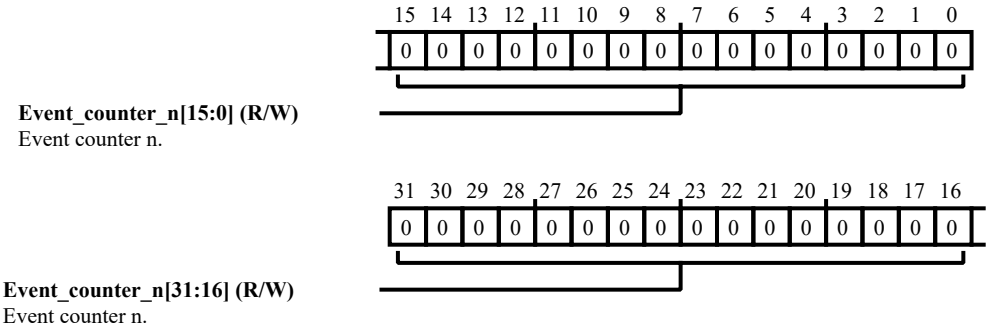


Figure 48-953: A55_C2_PMEVCNTR3_EL0 Register Diagram

Table 48-961: A55_C2_PMEVCNTR3_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

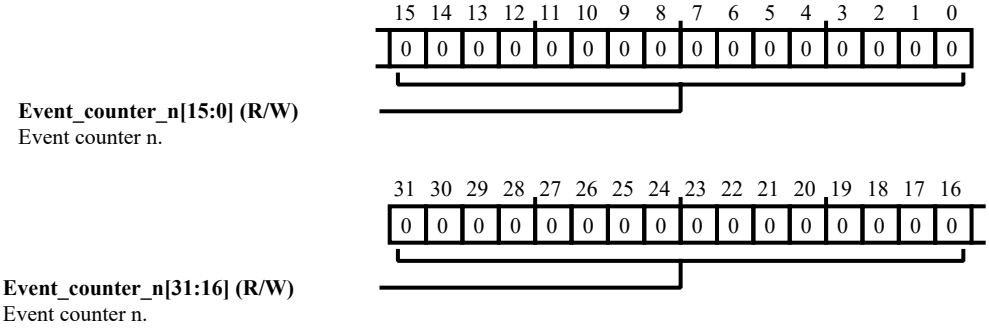


Figure 48-954: A55_C2_PMEVCNTR4_EL0 Register Diagram

Table 48-962: A55_C2_PMEVCNTR4_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Count Registers.

Holds event counter n, which counts events, where n is 0 to 30.

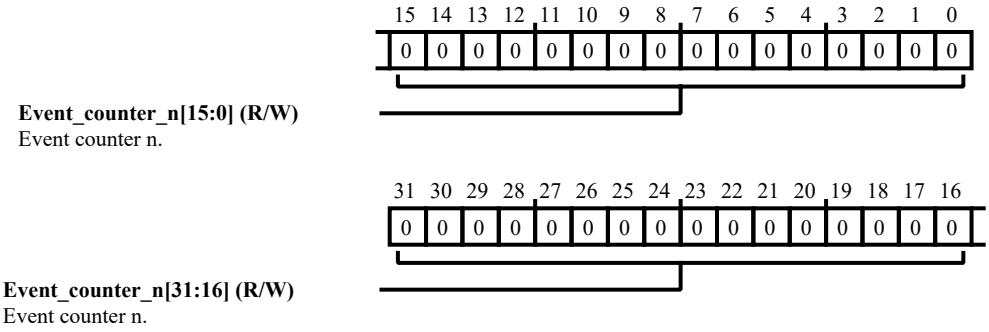


Figure 48-955: A55_C2_PMEVCNTR5_EL0 Register Diagram

Table 48-963: A55_C2_PMEVCNTR5_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	EVENT_COUNTER_N	Event counter n.. Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

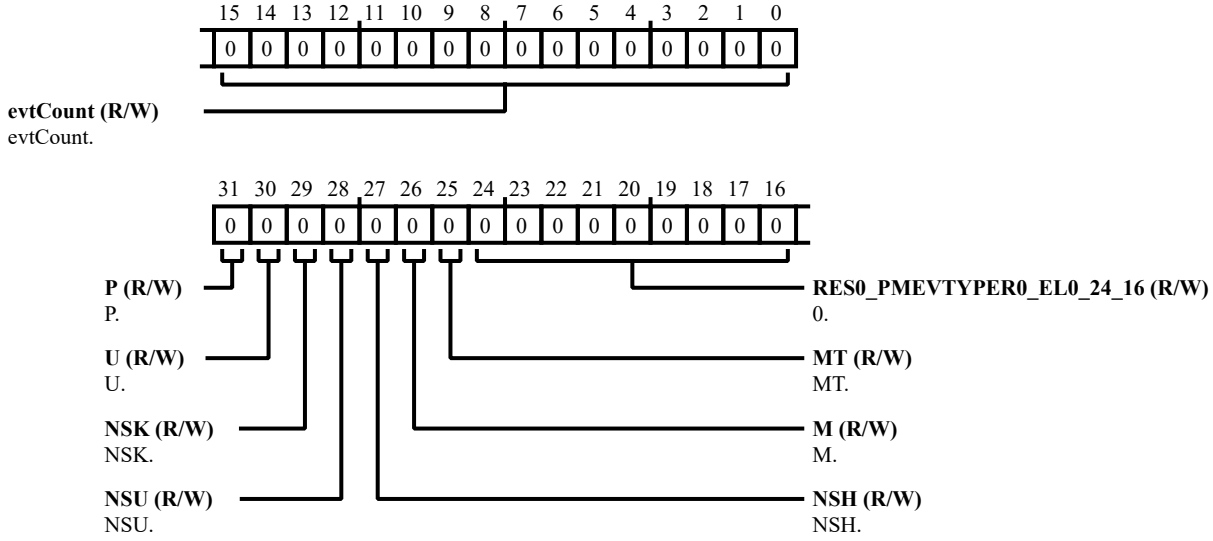


Figure 48-956: A55_C2_PMEVTYPER0_EL0 Register Diagram

Table 48-964: A55_C2_PMEVTYPER0_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-964: A55_C2_PMEVTYPER0_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER0_EL0_24_16	0.. Reserved, RES0.	

Table 48-964: A55_C2_PMEVTYPER0_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

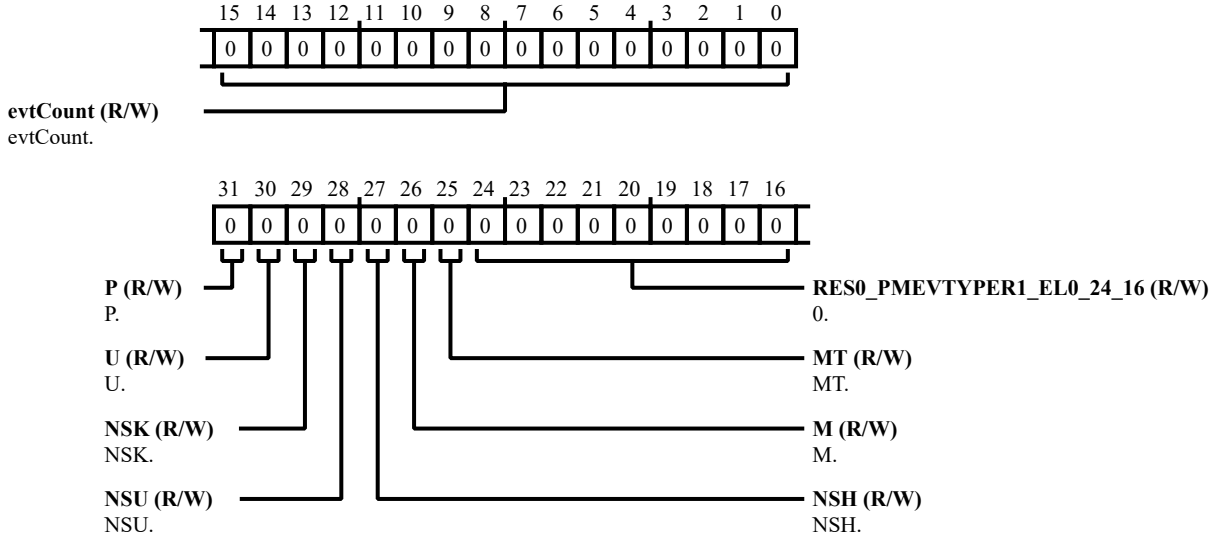


Figure 48-957: A55_C2_PMEVTYPER1_EL0 Register Diagram

Table 48-965: A55_C2_PMEVTYPER1_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-965: A55_C2_PMEVTYPER1_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER1_EL0_24_16	0.. Reserved, RES0.	

Table 48-965: A55_C2_PMEVTYPER1_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

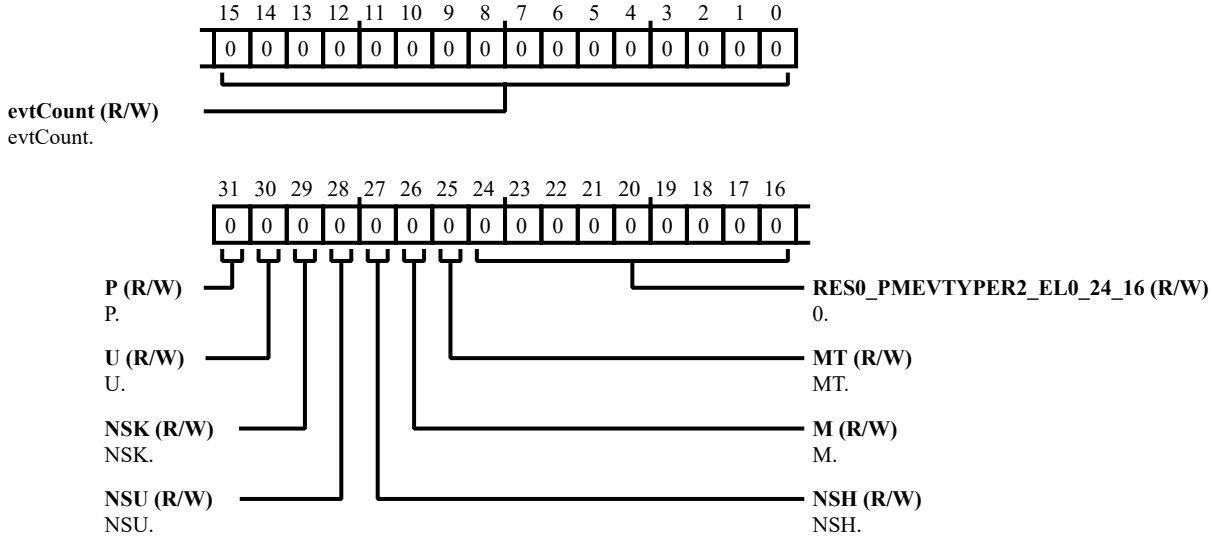


Figure 48-958: A55_C2_PMEVTYPER2_EL0 Register Diagram

Table 48-966: A55_C2_PMEVTYPER2_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-966: A55_C2_PMEVTYPER2_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER2_EL0_24_16	0.. Reserved, RES0.	

Table 48-966: A55_C2_PMEVTYPER2_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

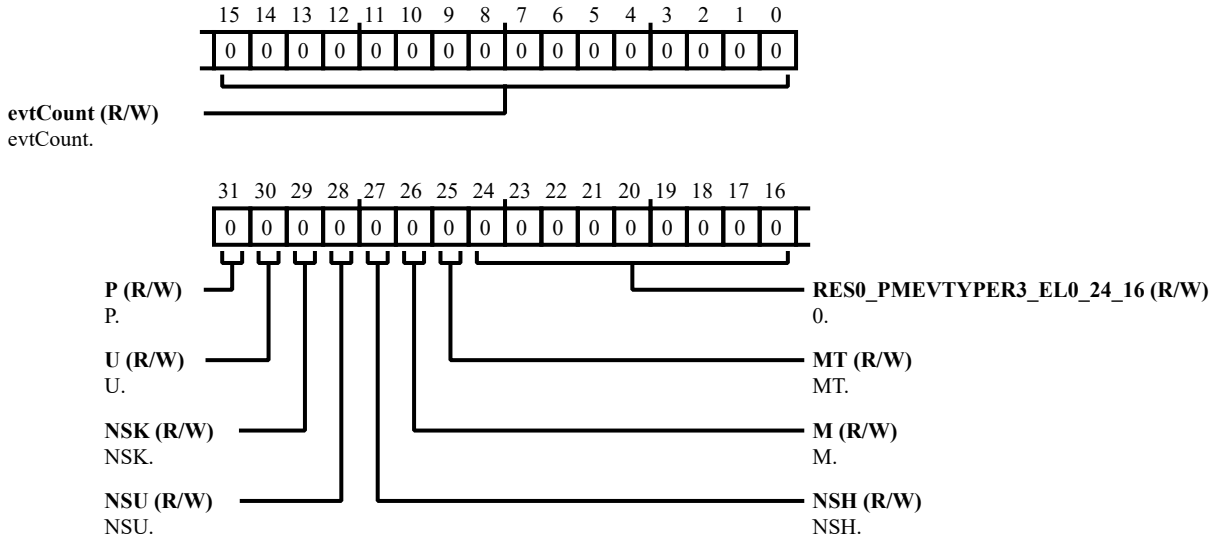


Figure 48-959: A55_C2_PMEVTYPER3_EL0 Register Diagram

Table 48-967: A55_C2_PMEVTYPER3_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-967: A55_C2_PMEVTYPER3_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER3_EL0_24_16	0.. Reserved, RES0.	

Table 48-967: A55_C2_PMEVTYPER3_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

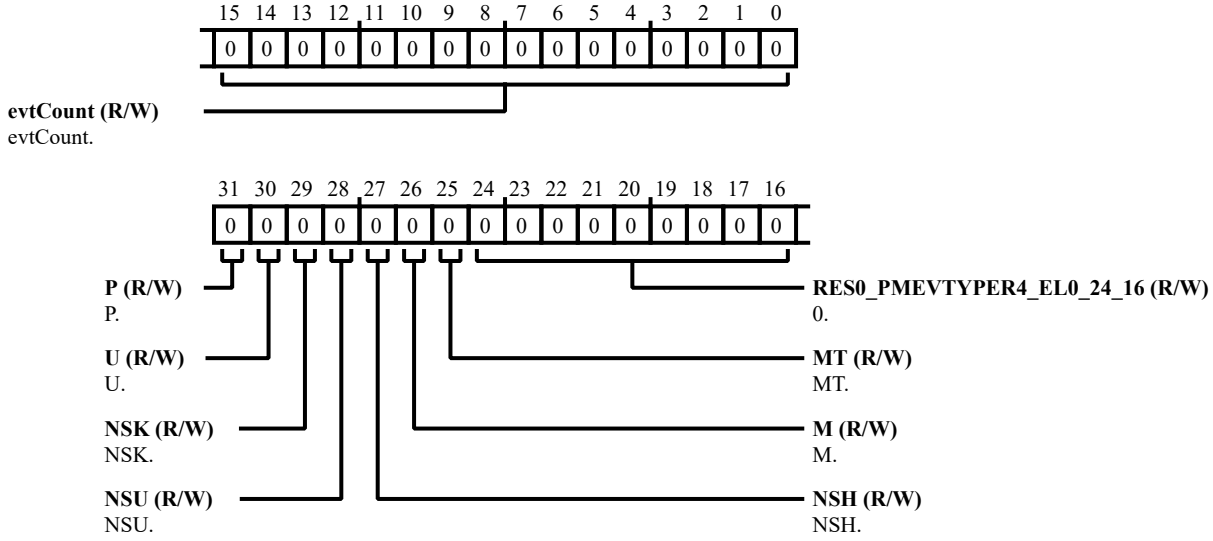


Figure 48-960: A55_C2_PMEVTYPER4_EL0 Register Diagram

Table 48-968: A55_C2_PMEVTYPER4_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-968: A55_C2_PMEVTYPER4_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER4_EL0_24_16	0.. Reserved, RES0.	

Table 48-968: A55_C2_PMEVTYPER4_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Event Type Registers.

Configures event counter n, where n is 0 to 30.

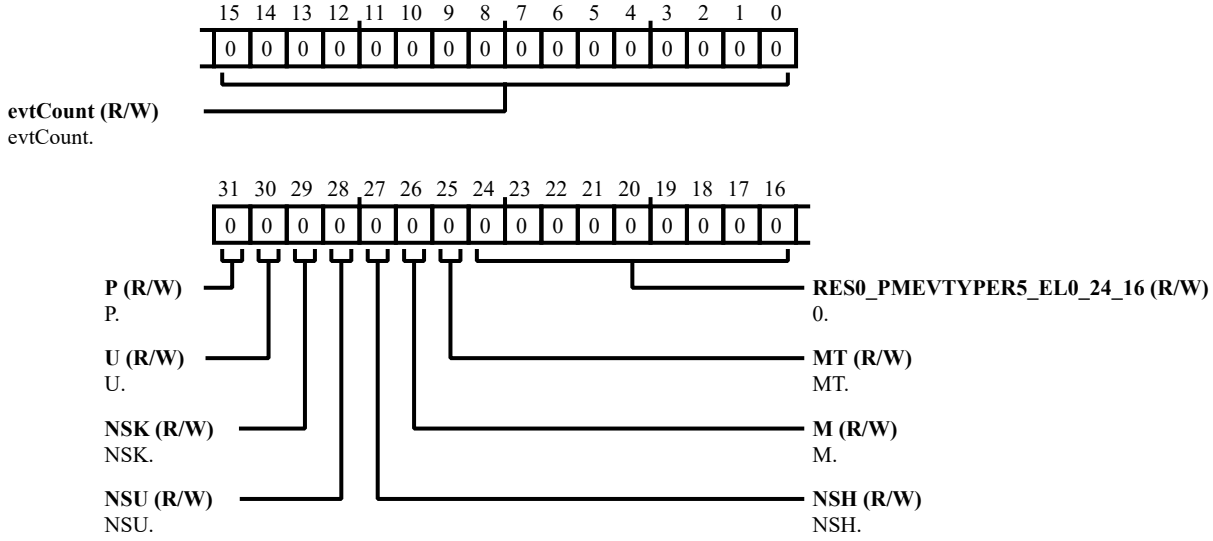


Figure 48-961: A55_C2_PMEVTYPER5_EL0 Register Diagram

Table 48-969: A55_C2_PMEVTYPER5_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	P	P.. Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are: 0 Count events in EL1. 1 Do not count events in EL1.	
		0	Count events in EL1.
		1	Do not count events in EL1.
30 (R/W)	U	U.. User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are: 0 Count events in EL0. 1 Do not count events in EL0.	
		0	Count events in EL0.
		1	Do not count events in EL0.
29 (R/W)	NSK	NSK.. Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	

Table 48-969: A55_C2_PMEVTYPER5_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	NSU	NSU.. Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	
27 (R/W)	NSH	NSH.. Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0. 0 Do not count events in EL2. 1 Count events in EL2.	
		0	Do not count events in EL2.
		1	Count events in EL2.
26 (R/W)	M	M.. Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0. If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0. This field is not visible in the AArch32 PMEVTYPER System register.	
25 (R/W)	MT	MT.. Multithreading. When the implementation is multi-threaded, the valid values for this bit are: 0 Count events only on controlling PE. 1 Count events from any PE with the same affinity at level 1 and above as this PE. When the implementation is not multi-threaded, this bit is RES0. An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1. Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.	
		0	Count events only on controlling PE.
		1	Count events from any PE with the same affinity at level 1 and above as this PE.
24:16 (R/W)	RES0_PMEVTYP-ER5_EL0_24_16	0.. Reserved, RES0.	

Table 48-969: A55_C2_PMEVTYPER5_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	EVTCOUNT	evtCount.. Event to count. The event number of the event that is counted by event counter PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. There are three ranges of event numbers: Event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events. Event numbers in the range 0x0040 to 0x00BF are ARM recommended common architectural and microarchitectural events. Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events. If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type: For the range 0x000 to 0x03F, no events are counted, and the value returned by a read of the evtCount field is the value written to the field. For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a read of the evtCount field is UNKNOWN. UNPREDICTABLE in this case means the event must not expose privileged information. UNPREDICTABLE means the event must not expose privileged information. For IMPLEMENTATION DEFINED events: It is UNPREDICTABLE what event, if any, is counted. UNPREDICTABLE in this case means the event must not expose privileged information. The value read back from evtCount is UNKNOWN. ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

Performance Monitors Interrupt Enable Clear register.

Disables the generation of interrupt requests on overflows from the Cycle Count Register, PMCCNTR_EL0, and the event counters PMEVCNTR0_EL0. Reading the register shows which overflow interrupt requests are enabled.

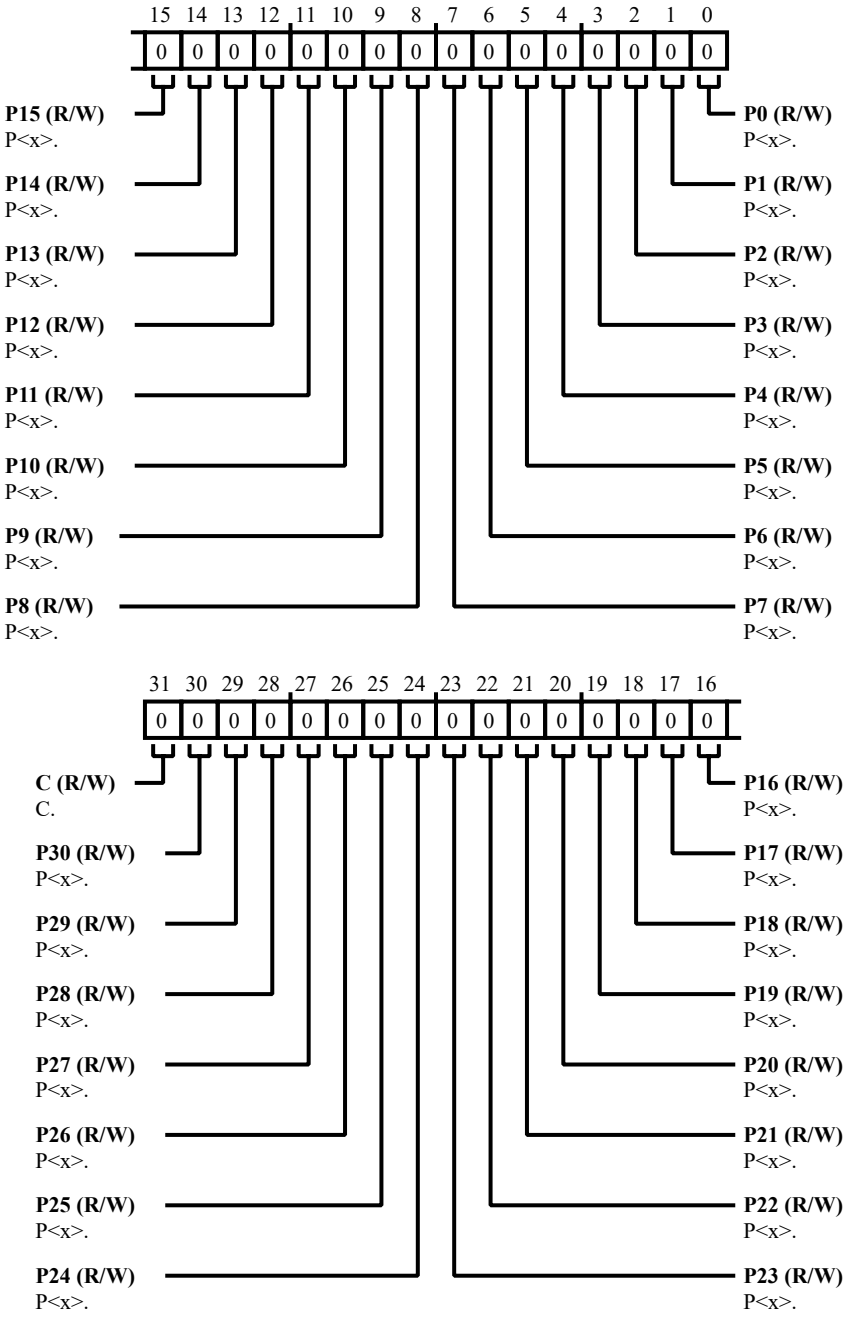


Figure 48-962: A55_C2_PMINTENCLR_EL1 Register Diagram

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow interrupt request disable bit. Possible values are: 0 When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. 1 When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.	
		0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
		1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.
30 (R/W)	P30	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
29 (R/W)	P29	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
27 (R/W)	P27	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	P26	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
25 (R/W)	P25	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	P24	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
23 (R/W)	P23	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
21 (R/W)	P21	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	P20	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
19 (R/W)	P19	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	P18	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
17 (R/W)	P17	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
15 (R/W)	P15	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	P14	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
13 (R/W)	P13	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	P12	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
11 (R/W)	P11	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
9 (R/W)	P9	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	P8	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
7 (R/W)	P7	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	P6	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
5 (R/W)	P5	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
3 (R/W)	P3	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	P2	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.
1 (R/W)	P1	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-970: A55_C2_PMINTENCLR_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	P0	P<x>.. Event counter overflow interrupt request disable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<x>_EL0 interrupt request.

Performance Monitors Interrupt Enable Set register.

Enables the generation of interrupt requests on overflows from the Cycle Count Register, PMCCNTR_EL0, and the event counters PMEVCNTR0_EL0. Reading the register shows which overflow interrupt requests are enabled.

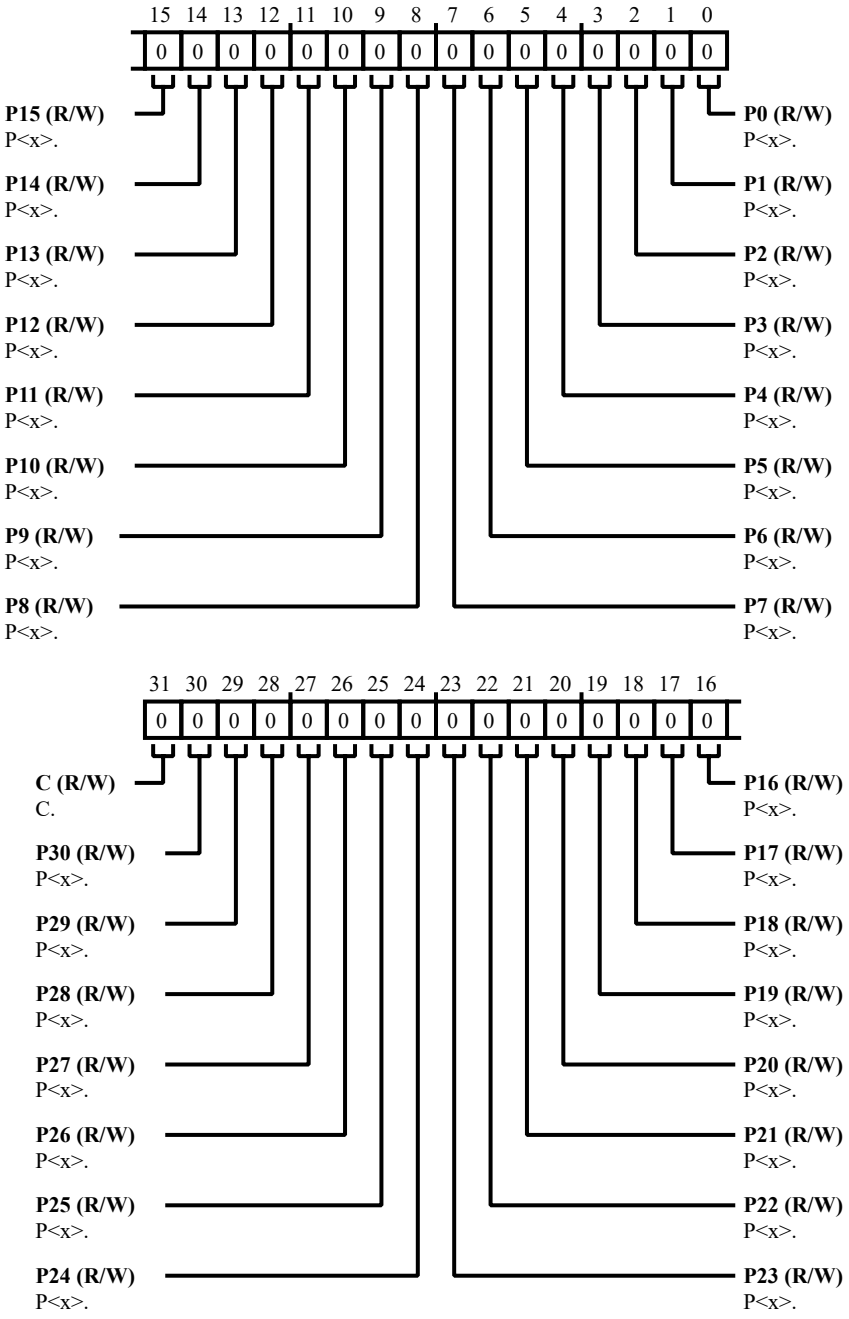


Figure 48-963: A55_C2_PMINTENSET_EL1 Register Diagram

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow interrupt request enable bit. Possible values are: 0 When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. 1 When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.	
		0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
		1	When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.
30 (R/W)	P30	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
29 (R/W)	P29	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
27 (R/W)	P27	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	P26	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
25 (R/W)	P25	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	P24	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
23 (R/W)	P23	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
21 (R/W)	P21	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	P20	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
19 (R/W)	P19	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	P18	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
17 (R/W)	P17	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
15 (R/W)	P15	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	P14	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
13 (R/W)	P13	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	P12	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
11 (R/W)	P11	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
9 (R/W)	P9	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	P8	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
7 (R/W)	P7	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	P6	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
5 (R/W)	P5	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
3 (R/W)	P3	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	P2	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.
1 (R/W)	P1	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Table 48-971: A55_C2_PMINTENSET_EL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	P0	P<x>.. Event counter overflow interrupt request enable bit for PMEVCNTR<x>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect. 1 When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.	
		0	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is disabled. When written, has no effect.
		1	When read, means that the PMEVCNTR<x>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<x>_EL0 interrupt request.

Performance Monitors Integration mode Control register.

Enables the Performance Monitors to switch from default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

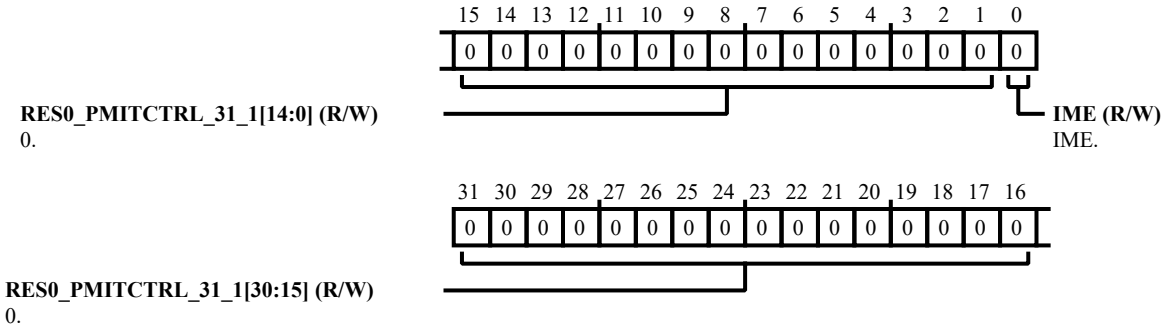


Figure 48-964: A55_C2_PMITCTRL Register Diagram

Table 48-972: A55_C2_PMITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:1 (R/W)	RES0_PMITCTRL_31_1	0.. Reserved, RES0.	
0 (R/W)	IME	IME.. Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED. 0 Normal operation. 1 Integration mode enabled.	
		0	Normal operation.
		1	Integration mode enabled.

Performance Monitors Overflow Flag Status Clear register.

Contains the state of the overflow bit for the Cycle Count Register, PMCCNTR_EL0, and each of the implemented event counters PMEVCNTR0. Writing to this register clears these bits.

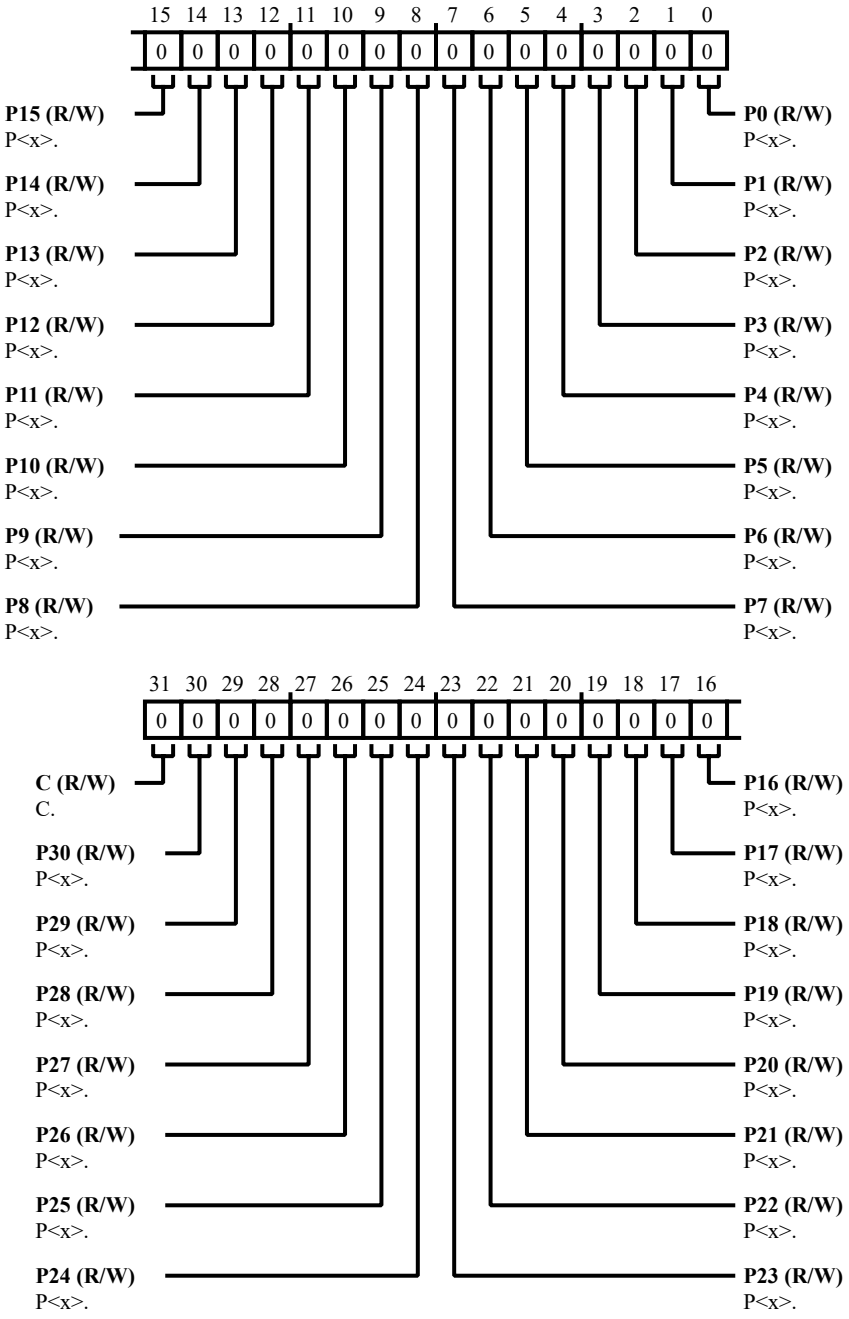


Figure 48-965: A55_C2_PMOVSCLR_EL0 Register Diagram

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow bit. Possible values are: 0 When read, means the cycle counter has not overflowed. When written, has no effect. 1 When read, means the cycle counter has overflowed. When written, clears the overflow bit to 0. PMCR_EL0.LC controls whether an overflow is detected from PMCCNTR_EL0[31] or from PMCCNTR_EL0[63].	
		0	When read, means the cycle counter has not overflowed. When written, has no effect.
		1	When read, means the cycle counter has overflowed. When written, clears the overflow bit to 0.
30 (R/W)	P30	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
29 (R/W)	P29	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
27 (R/W)	P27	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
26 (R/W)	P26	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	P25	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
24 (R/W)	P24	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
23 (R/W)	P23	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
21 (R/W)	P21	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
20 (R/W)	P20	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
18 (R/W)	P18	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
17 (R/W)	P17	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
15 (R/W)	P15	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
14 (R/W)	P14	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	P13	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
12 (R/W)	P12	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
11 (R/W)	P11	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
9 (R/W)	P9	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
8 (R/W)	P8	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
6 (R/W)	P6	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
5 (R/W)	P5	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
3 (R/W)	P3	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
2 (R/W)	P2	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Table 48-973: A55_C2_PMOVSCLR_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	P1	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.
0 (R/W)	P0	P<x>.. Event counter overflow clear bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values of each bit are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<n>_EL0 has overflowed. When written, clears the PMEVCNTR<x> overflow bit to 0.

Performance Monitors Overflow Flag Status Set register.

Sets the state of the overflow bit for the Cycle Count Register, PMCCNTR_EL0, and each of the implemented event counters PMEVCNTR0.

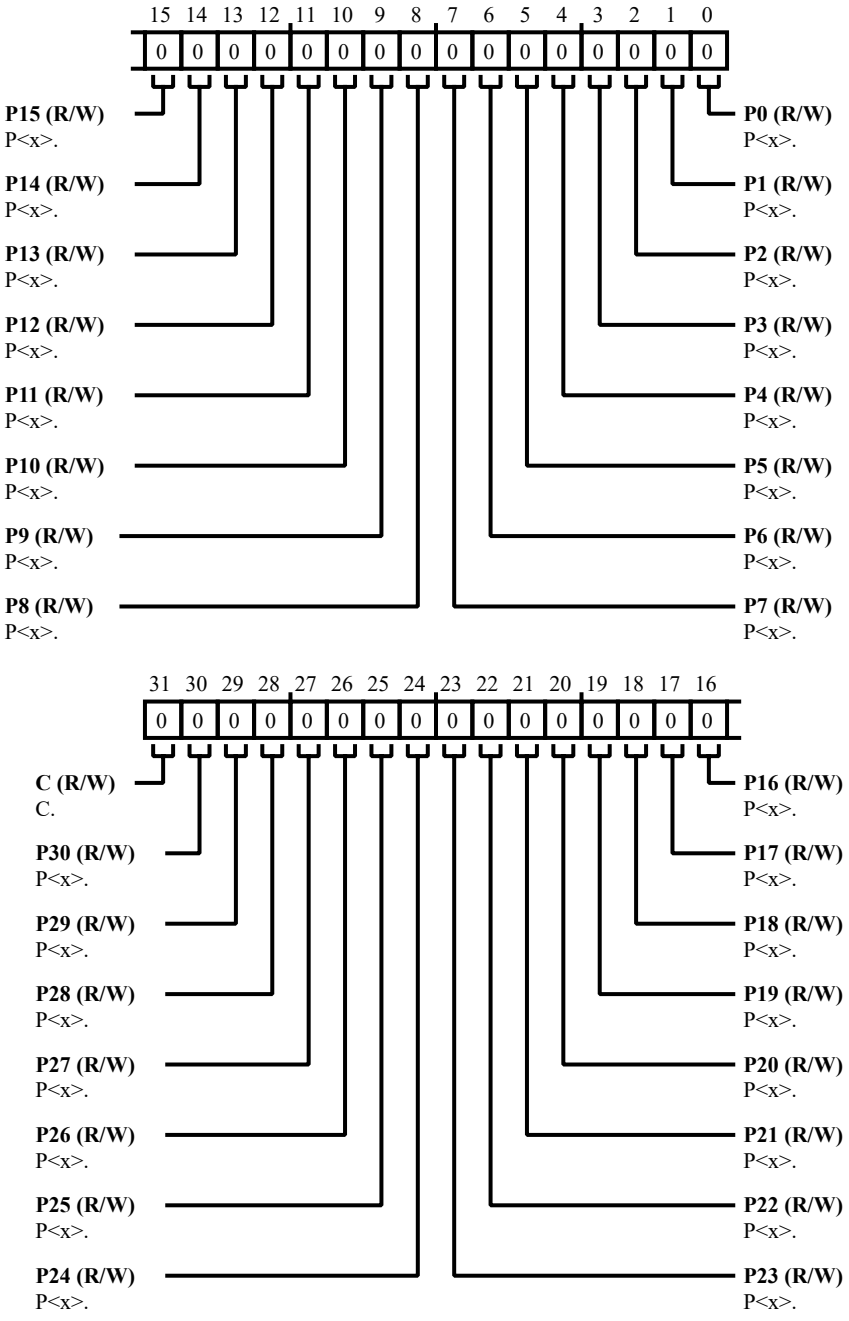


Figure 48-966: A55_C2_PMOVSSET_EL0 Register Diagram

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	C	C.. PMCCNTR_EL0 overflow bit. Possible values are: 0 When read, means the cycle counter has not overflowed. When written, has no effect. 1 When read, means the cycle counter has overflowed. When written, sets the overflow bit to 1.	
		0	When read, means the cycle counter has not overflowed. When written, has no effect.
		1	When read, means the cycle counter has overflowed. When written, sets the overflow bit to 1.
30 (R/W)	P30	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
29 (R/W)	P29	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	P28	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
27 (R/W)	P27	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
26 (R/W)	P26	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	P25	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
24 (R/W)	P24	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
23 (R/W)	P23	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	P22	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
21 (R/W)	P21	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
20 (R/W)	P20	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	P19	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
18 (R/W)	P18	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
17 (R/W)	P17	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	P16	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
15 (R/W)	P15	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
14 (R/W)	P14	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	P13	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
12 (R/W)	P12	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
11 (R/W)	P11	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	P10	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
9 (R/W)	P9	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
8 (R/W)	P8	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	P7	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
6 (R/W)	P6	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
5 (R/W)	P5	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	P4	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
3 (R/W)	P3	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
2 (R/W)	P2	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Table 48-974: A55_C2_PMOVSSET_EL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	P1	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.
0 (R/W)	P0	P<x>.. Event counter overflow set bit for PMEVCNTR<n>_EL0. Bits [30:N] are RAZ/WI. N is the value in PMCFGR.N. Possible values are: 0 When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect. 1 When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.	
		0	When read, means that PMEVCNTR<n>_EL0 has not overflowed. When written, has no effect.
		1	When read, means that PMEVCNTR<x>_EL0 has overflowed. When written, sets the PMEVCNTR<x> overflow bit to 1.

Performance Monitors Software Increment register.

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see .

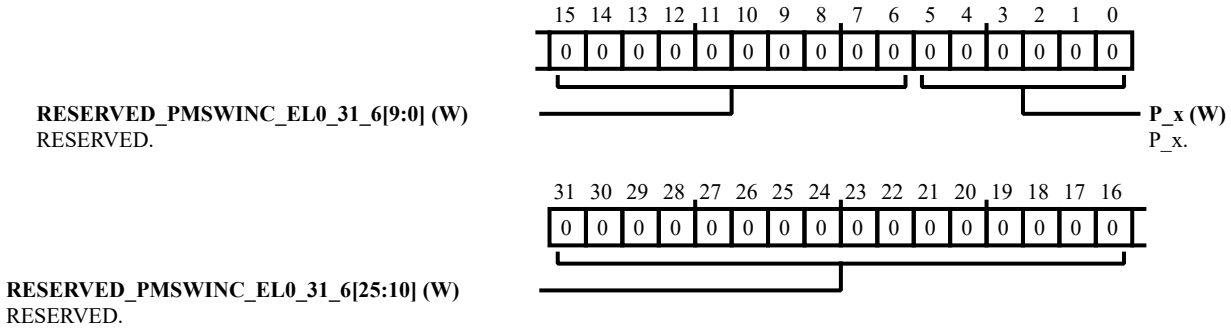


Figure 48-967: A55_C2_PMSWINC_EL0 Register Diagram

Table 48-975: A55_C2_PMSWINC_EL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:6 (RX/W)	RESERVED_PMSWINC_EL0_31_6	RESERVED.. Reserved RES0.	
		0	Reserved.
5:0 (RX/W)	P_X	P_x.. Event counter software increment bit for PMEVCNTR<x>.	
		0	No action. The write to this bit is ignored.
		1	It is CONSTRAINED UNPREDICTABLE whether a SW_INCR event is generated for event counter x.

49 ADSP-2184x MISCREG Register Descriptions

Misc registers for module are for integration purpose (MISCREG) contains the following registers.

Table 49-1: ADSP-2184x MISCREG Register List

Name	Description
MISCREG_CAN_SYSCTL	CANFD Low Power Selection Mode
MISCREG_CLK_MUX_SEL	Miscellaneous Clock Multiplexer Selection Register
MISCREG_CLK_MUX_SEL1	Miscellaneous Clock Multiplexer Selection 1 Register
MISCREG_ECO_REG1	LPDDR Miscellaneous Register
MISCREG_ECO_REG9	EMAC Error Register
MISCREG_EMAC_MISC_REG	EMAC Miscellaneous Register
MISCREG_EMAC_MISC_REG2	EMAC Miscellaneous Register 2
MISCREG_EMMC_MISC_REG	EMMC MISC Register
MISCREG_MINI_CGU0	MINI CGU0 MISC Register
MISCREG_MINI_CGU1	MINI CGU0 MISC Register
MISCREG_SH0_PFB_RANGE_SELECT	SH0 Prefetch Range Selection Register
MISCREG_SH1_PFB_RANGE_SELECT	SH1 Prefetch Range Selection Register
MISCREG_SHARC_BRIDGE_REMAP	SPORT Direct Interrupt Enable for SH0 and SH1
MISCREG_XSPI1_CTL0	XSPI1 Boot/Init Control Register 0
MISCREG_XSPI1_CTL1	XSPI1 Boot/Init Control Register 1
MISCREG_XSPI1_CTL2	XSPI1 Boot/Init Control Register 2
MISCREG_XSPI1_CTL3	XSPI1 Boot/Init Control Register 3
MISCREG_XSPI1_CTL4	XSPI1 Boot/Init Control Register 4
MISCREG_XSPI1_CTL5	XSPI1 Boot/Init Control Register 5
MISCREG_XSPI1_CTL6	XSPI1 Boot/Init Control Register 6

Table 49-1: ADSP-2184x MISCREG Register List (Continued)

Name	Description
MISCREG_XSPI1_CTL7	XSPI1 Boot/Init Control Register 7
MISCREG_XSPI1_CTL8	XSPI1 Boot/Init Control Register 8
MISCREG_XSPI1_RSTCTL	XSPI1 Boot/Init Reset Control Register
MISCREG_XSPI1_STAT0	XSPI1 Boot/Init Status Register
MISCREG_XSPI_CTL0	XSPI Boot/Init Control Register 0
MISCREG_XSPI_CTL1	XSPI Boot/Init Control Register 1
MISCREG_XSPI_CTL2	XSPI Boot/Init Control Register 2
MISCREG_XSPI_CTL3	XSPI Boot/Init Control Register 3
MISCREG_XSPI_CTL4	XSPI Boot/Init Control Register 4
MISCREG_XSPI_CTL5	XSPI Boot/Init Control Register 5
MISCREG_XSPI_CTL6	XSPI Boot/Init Control Register 6
MISCREG_XSPI_CTL7	XSPI Boot/Init Control Register 7
MISCREG_XSPI_CTL8	XSPI Boot/Init Control Register 8
MISCREG_XSPI_RSTCTL	XSPI Boot/Init Reset Control Register
MISCREG_XSPI_STAT0	XSPI Boot/Init Status Register

CANFD Low Power Selection Mode

The MISCREG_CAN_SYSCTL register contains the control bits for the different low power modes supported in the CANFD module.

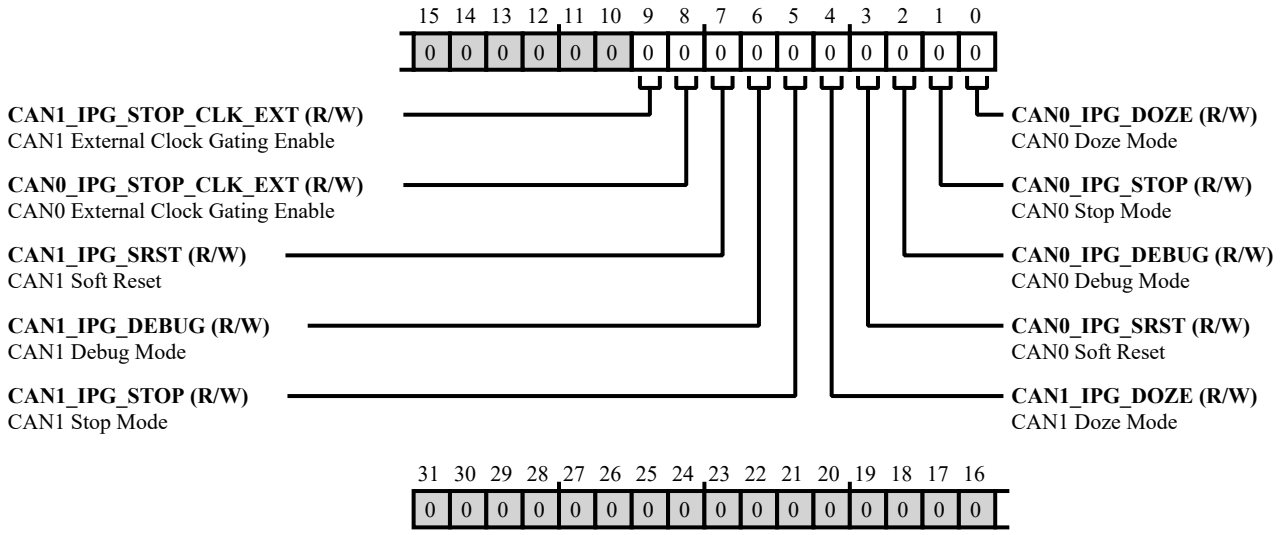


Figure 49-1: MISCREG_CAN_SYSCTL Register Diagram

Table 49-2: MISCREG_CAN_SYSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	CAN1_IPG_STOP_CLK_EXT	CAN1 External Clock Gating Enable. When set (=1), the MISCREG_CAN_SYSCTL.CAN1_IPG_STOP_CLK_EXT bit enables external clock gating for stop mode for CAN1.
8 (R/W)	CAN0_IPG_STOP_CLK_EXT	CAN0 External Clock Gating Enable. When set (=1), the MISCREG_CAN_SYSCTL.CAN0_IPG_STOP_CLK_EXT bit enables external clock gating for stop mode for CAN0.
7 (R/W)	CAN1_IPG_SRST	CAN1 Soft Reset. When set (=1), the MISCREG_CAN_SYSCTL.CAN1_IPG_SRST bit enables a soft reset request to CAN1.
6 (R/W)	CAN1_IPG_DEBUG	CAN1 Debug Mode. When set (=1), the MISCREG_CAN_SYSCTL.CAN1_IPG_DEBUG bit enables a debug mode request to CAN1.
5 (R/W)	CAN1_IPG_STOP	CAN1 Stop Mode. When set (=1), the MISCREG_CAN_SYSCTL.CAN1_IPG_STOP bit enables a stop mode request to CAN1.

Table 49-2: MISCREG_CAN_SYSCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	CAN1_IPG_DOZE	CAN1 Doze Mode. When set (=1), the `MISCREG_CAN_SYSCTL.CAN1_IPG_DOZE` bit enables a doze mode request to CAN1.
3 (R/W)	CAN0_IPG_SRST	CAN0 Soft Reset. When set (=1), the `MISCREG_CAN_SYSCTL.CAN0_IPG_SRST` bit enables a soft reset request to CAN0.
2 (R/W)	CAN0_IPG_DEBUG	CAN0 Debug Mode. When set (=1), the `MISCREG_CAN_SYSCTL.CAN0_IPG_DEBUG` bit enables a debug mode request to CAN0.
1 (R/W)	CAN0_IPG_STOP	CAN0 Stop Mode. When set (=1), the `MISCREG_CAN_SYSCTL.CAN0_IPG_STOP` bit enables a stop mode request to CAN0.
0 (R/W)	CAN0_IPG_DOZE	CAN0 Doze Mode. When set (=1), the `MISCREG_CAN_SYSCTL.CAN0_IPG_DOZE` bit enables a doze mode request to CAN0.

Miscellaneous Clock Multiplexer Selection Register

The MISCREG_CLK_MUX_SEL register contain the clock multiplex selections.

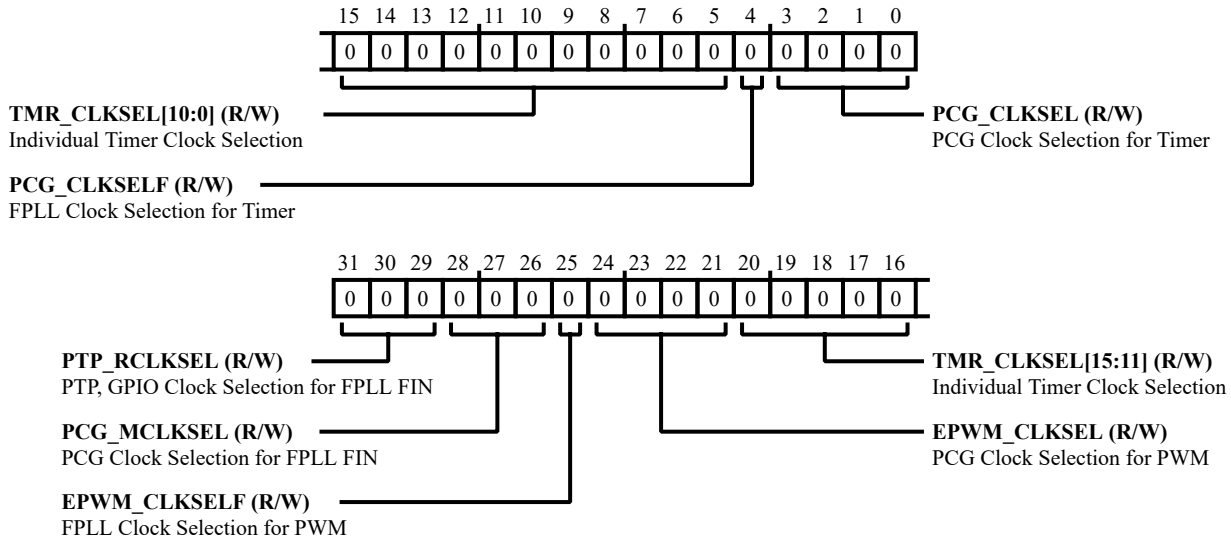


Figure 49-2: MISCREG_CLK_MUX_SEL Register Diagram

Table 49-3: MISCREG_CLK_MUX_SEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:29 (R/W)	PTP_RCLKSEL	PTP, GPIO Clock Selection for FPLL FIN. The MISCREG_CLK_MUX_SEL.PTP_RCLKSEL bit field indicates the PTP GPIO clock selection for FPLL FIN. 3'b000 - Ext Sync over PA[15] 3'b001 - ETH0 PTPPPS[0] ... 3'b101 - ETH0 PTPPPS[3] 3'b110 - N/A ... 3'b111 - N/A

Table 49-3: MISCREG_CLK_MUX_SEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28:26 (R/W)	PCG_MCLKSEL	PCG Clock Selection for FPLL FIN. The `MISCREG_CLK_MUX_SEL.PCG_MCLKSEL` bit field indicates the PCG clock selection for FPLL FIN. 3'b000 - Ext Clk over DAI0- PCGA 3'b001 - Ext Clk over DAI0- PCGB 3'b010 - Ext Clk over DAI0- PCGE 3'b011 - Ext Clk over DAI0- PCGF 3'b100 - Ext Clk over DAI1- PCGC 3'b101 - Ext Clk over DAI1- PCGD 3'b110 - Ext Clk over DAI1- PCGG 3'b111 - Ext Clk over DAI1- PCGH	
25 (R/W)	EPWM_CLKSELF	FPLL Clock Selection for PWM. The `MISCREG_CLK_MUX_SEL.EPWM_CLKSELF` bit field indicates the FPLL clock selection for PWM.	
		0	FPLL Output Clock
		1	Muxed PCG Clock

Table 49-3: MISCREG_CLK_MUX_SEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24:21 (R/W)	EPWM_CLKSEL	PCG Clock Selection for PWM. The `MISCREG_CLK_MUX_SEL.EPWM_CLKSEL` bit field indicates the PCG clock selection for PWM. 4'b0000 - Clock from DAI0- PCGA 4'b0001 - Clock from DAI0- PCGB 4'b0010 - Clock from DAI0- PCGE 4'b0011 - Clock from DAI0- PCGF 4'b0100 - Inverted Clock from DAI0- PCGA 4'b0101 - Inverted Clock from DAI0- PCGB 4'b0110 - Inverted Clock from DAI0- PCGE 4'b0111 - Inverted Clock from DAI0- PCGF 4'b1000 - Clock from DAI1- PCGC 4'b1001 - Clock from DAI1- PCGD 4'b1010 - Clock from DAI1- PCGG 4'b1011 - Clock from DAI1- PCGH 4'b1100 - Inverted Clock from DAI1- PCGC 4'b1101 - Inverted Clock from DAI1- PCGD 4'b1110 - Inverted Clock from DAI1- PCGG 4'b1111 - Inverted Clock from DAI1- PCGH	
20:5 (R/W)	TMR_CLKSEL	Individual Timer Clock Selection. The `MISCREG_CLK_MUX_SEL.TMR_CLKSEL` bit field indicates the input clock for Timer(n) between TM0_ACLK(0) and FPLL output clock or Muxed PCG clock (1), where n = 0 to 15.	
4 (R/W)	PCG_CLKSELF	FPLL Clock Selection for Timer. The `MISCREG_CLK_MUX_SEL.PCG_CLKSELF` bit field indicates the FPLL clock selection for the timer.	
		0	Muxed PCG clock
		1	FPLL Output Clock

Table 49-3: MISCREG_CLK_MUX_SEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	PCG_CLKSEL	PCG Clock Selection for Timer. The `MISCREG_CLK_MUX_SEL.PCG_CLKSEL` bit field indicates the PCG clock selection for the timer. 4'b0000 - Clock from DAI0- PCGA 4'b0001 - Clock from DAI0- PCGB 4'b0010 - Clock from DAI0- PCGE 4'b0011 - Clock from DAI0- PCGF 4'b0100 - Inverted Clock from DAI0- PCGA 4'b0101 - Inverted Clock from DAI0- PCGB 4'b0110 - Inverted Clock from DAI0- PCGE 4'b0111 - Inverted Clock from DAI0- PCGF 4'b1000 - Clock from DAI1- PCGC 4'b1001 - Clock from DAI1- PCGD 4'b1010 - Clock from DAI1- PCGG 4'b1011 - Clock from DAI1- PCGH 4'b1100 - Inverted Clock from DAI1- PCGC 4'b1101 - Inverted Clock from DAI1- PCGD 4'b1110 - Inverted Clock from DAI1- PCGG 4'b1111 - Inverted Clock from DAI1- PCGH

Miscellaneous Clock Multiplexer Selection 1 Register

The MISCREG_CLK_MUX_SEL1 register contain the clock multiplex selections.

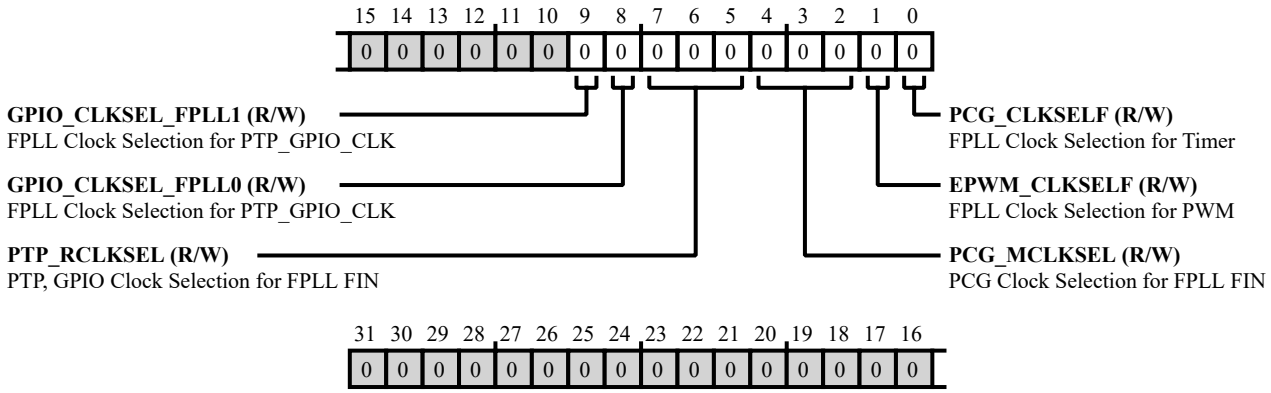


Figure 49-3: MISCREG_CLK_MUX_SEL1 Register Diagram

Table 49-4: MISCREG_CLK_MUX_SEL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	GPIO_CLKSEL_FPLL1	FPLL Clock Selection for PTP_GPIO_CLK. 1'b0 - Muxed PCG clock 1'b1 - FPLL output clock
8 (R/W)	GPIO_CLKSEL_FPLL0	FPLL Clock Selection for PTP_GPIO_CLK. 1'b0 - Muxed PCG clock 1'b1 - FPLL output clock
7:5 (R/W)	PTP_RCLKSEL	PTP, GPIO Clock Selection for FPLL FIN. 3'b000 - Ext Sync over PA[15] 3'b001 - ETH0 PTPPPS[0] ... 3'b101 - ETH0 PTPPPS[3] 3'b110 - N/A ... 3'b111 - N/A
4:2 (R/W)	PCG_MCLKSEL	PCG Clock Selection for FPLL FIN. 3'b000 - Ext Clk over DAI0- PCGA 3'b001 - Ext Clk over DAI0- PCGB 3'b010 - Ext Clk over DAI0- PCGE 3'b011 - Ext Clk over DAI0- PCGF 3'b100 - Ext Clk over DAI1- PCGC 3'b101 - Ext Clk over DAI1- PCGD 3'b110 - Ext Clk over DAI1- PCGG 3'b111 - Ext Clk over DAI1- PCGH
1 (R/W)	EPWM_CLKSELF	FPLL Clock Selection for PWM. 1'b0 - FPLL output clock 1'b1 - Muxed PCG clock
0 (R/W)	PCG_CLKSELF	FPLL Clock Selection for Timer. 1'b0 - Muxed PCG clock 1'b1 - FPLL output clock

LPDDR Miscellaneous Register

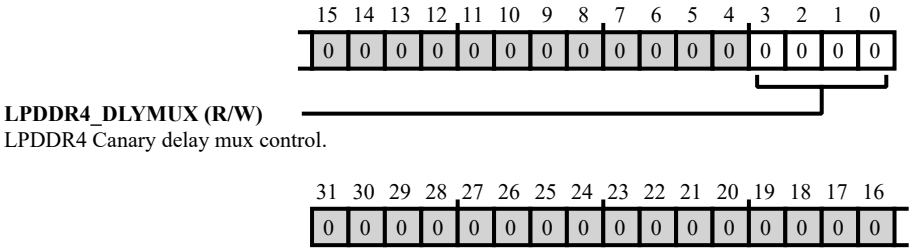


Figure 49-4: MISCREG_ECO_REG1 Register Diagram

Table 49-5: MISCREG_ECO_REG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	LPDDR4_DLYMUX	LPDDR4 Canary delay mux control..

EMAC Error Register

The MISCREG_ECO_REG9 register contains eight Tx channels, eight Rx channels, and DMA errors for the EMAC.

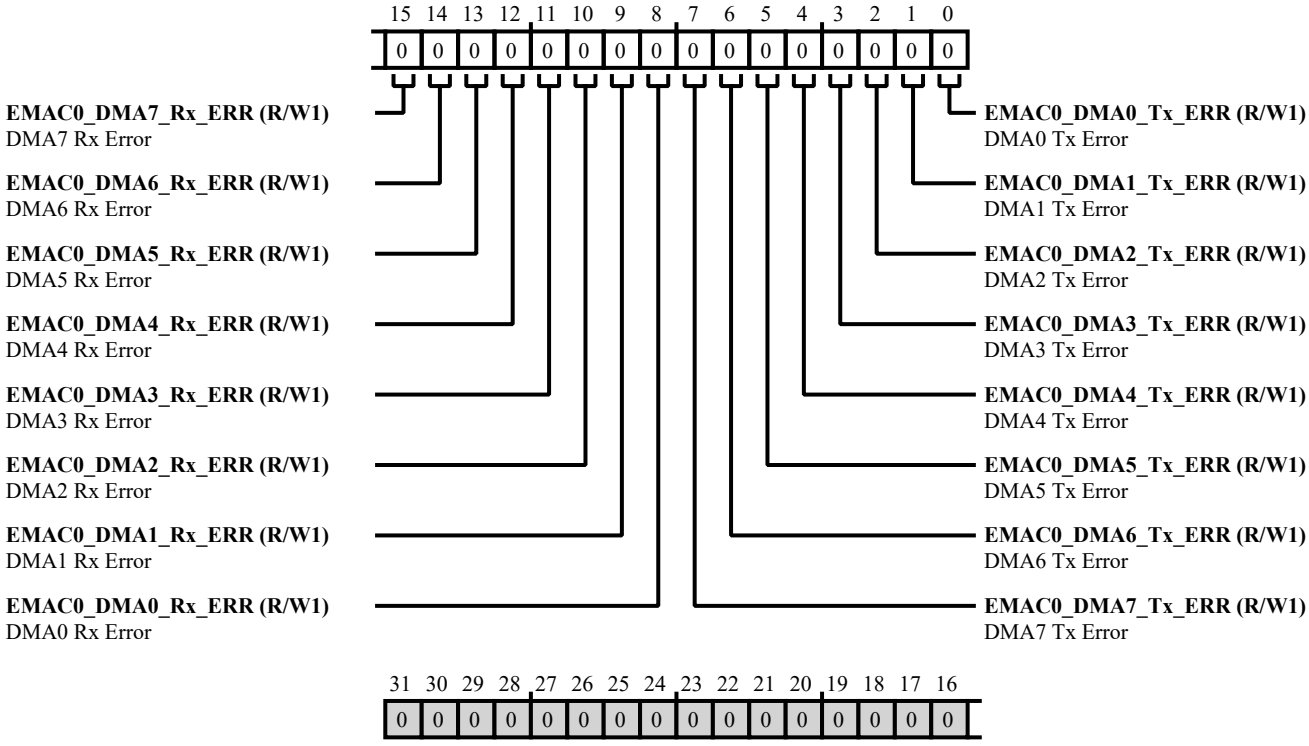


Figure 49-5: MISCREG_ECO_REG9 Register Diagram

Table 49-6: MISCREG_ECO_REG9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1)	EMAC0_DMA7_RX_ERR	DMA7 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
14 (R/W1)	EMAC0_DMA6_RX_ERR	DMA6 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
13 (R/W1)	EMAC0_DMA5_RX_ERR	DMA5 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
12 (R/W1)	EMAC0_DMA4_RX_ERR	DMA4 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
11 (R/W1)	EMAC0_DMA3_RX_ERR	DMA3 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.

Table 49-6: MISCREG_ECO_REG9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W1)	EMAC0_DMA2_RX_ERR	DMA2 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
9 (R/W1)	EMAC0_DMA1_RX_ERR	DMA1 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
8 (R/W1)	EMAC0_DMA0_RX_ERR	DMA0 Rx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
7 (R/W1)	EMAC0_DMA7_TX_ERR	DMA7 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
6 (R/W1)	EMAC0_DMA6_TX_ERR	DMA6 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
5 (R/W1)	EMAC0_DMA5_TX_ERR	DMA5 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
4 (R/W1)	EMAC0_DMA4_TX_ERR	DMA4 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
3 (R/W1)	EMAC0_DMA3_TX_ERR	DMA3 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
2 (R/W1)	EMAC0_DMA2_TX_ERR	DMA2 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
1 (R/W1)	EMAC0_DMA1_TX_ERR	DMA1 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.
0 (R/W1)	EMAC0_DMA0_TX_ERR	DMA0 Tx Error. DMA received a 2nd start (stop) trigger before a stop (start) trigger.

EMAC Miscellaneous Register

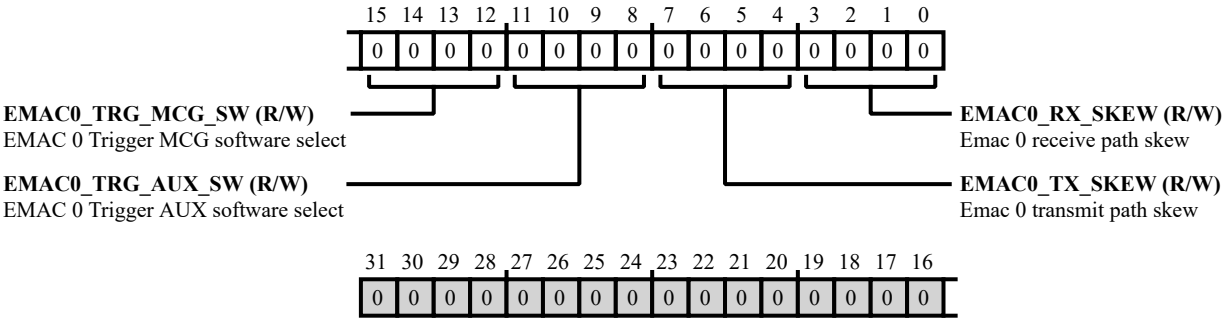


Figure 49-6: MISCREG_EMAC_MISC_REG Register Diagram

Table 49-7: MISCREG_EMAC_MISC_REG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:12 (R/W)	EMAC0_TRG_MCG_SW	EMAC 0 Trigger MCG software select.
11:8 (R/W)	EMAC0_TRG_AUX_SW	EMAC 0 Trigger AUX software select.
7:4 (R/W)	EMAC0_TX_SKEW	Emac 0 transmit path skew.
3:0 (R/W)	EMAC0_RX_SKEW	Emac 0 receive path skew.

EMAC Miscellaneous Register 2

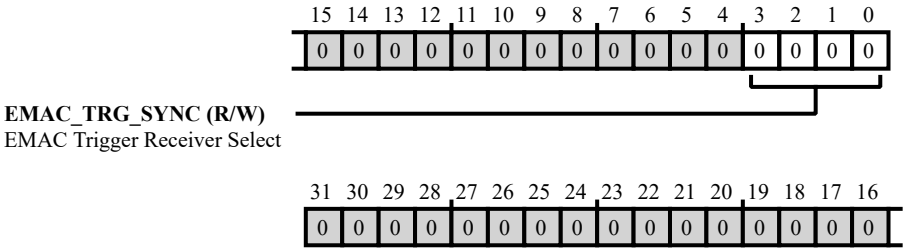


Figure 49-7: MISCREG_EMAC_MISC_REG2 Register Diagram

Table 49-8: MISCREG_EMAC_MISC_REG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	EMAC_TRG_SYNC	EMAC Trigger Receiver Select.

EMMC MISC Register

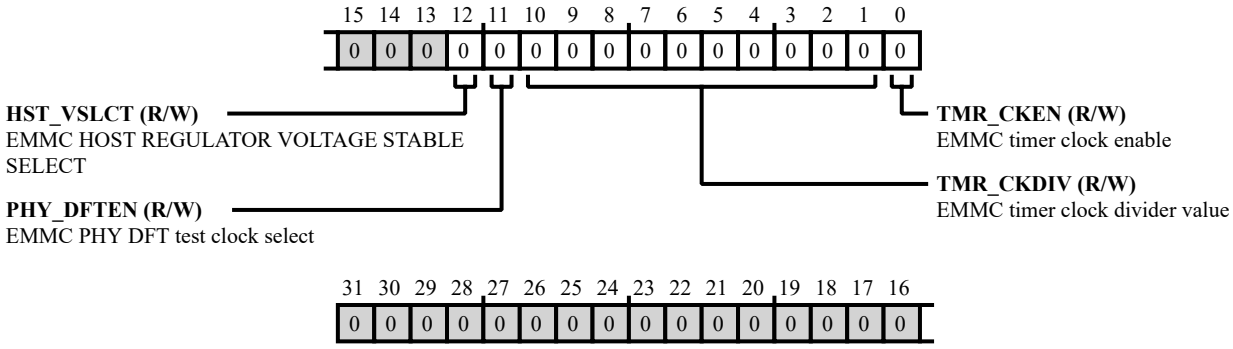


Figure 49-8: MISCREG_EMMC_MISC_REG Register Diagram

Table 49-9: MISCREG_EMMC_MISC_REG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12 (R/W)	HST_VSLCT	EMMC HOST REGULATOR VOLTAGE STABLE SELECT.
11 (R/W)	PHY_DFTEN	EMMC PHY DFT test clock select.
10:1 (R/W)	TMR_CKDIV	EMMC timer clock divider value.
0 (R/W)	TMR_CKEN	EMMC timer clock enable.

MINI CGU0 MISC Register

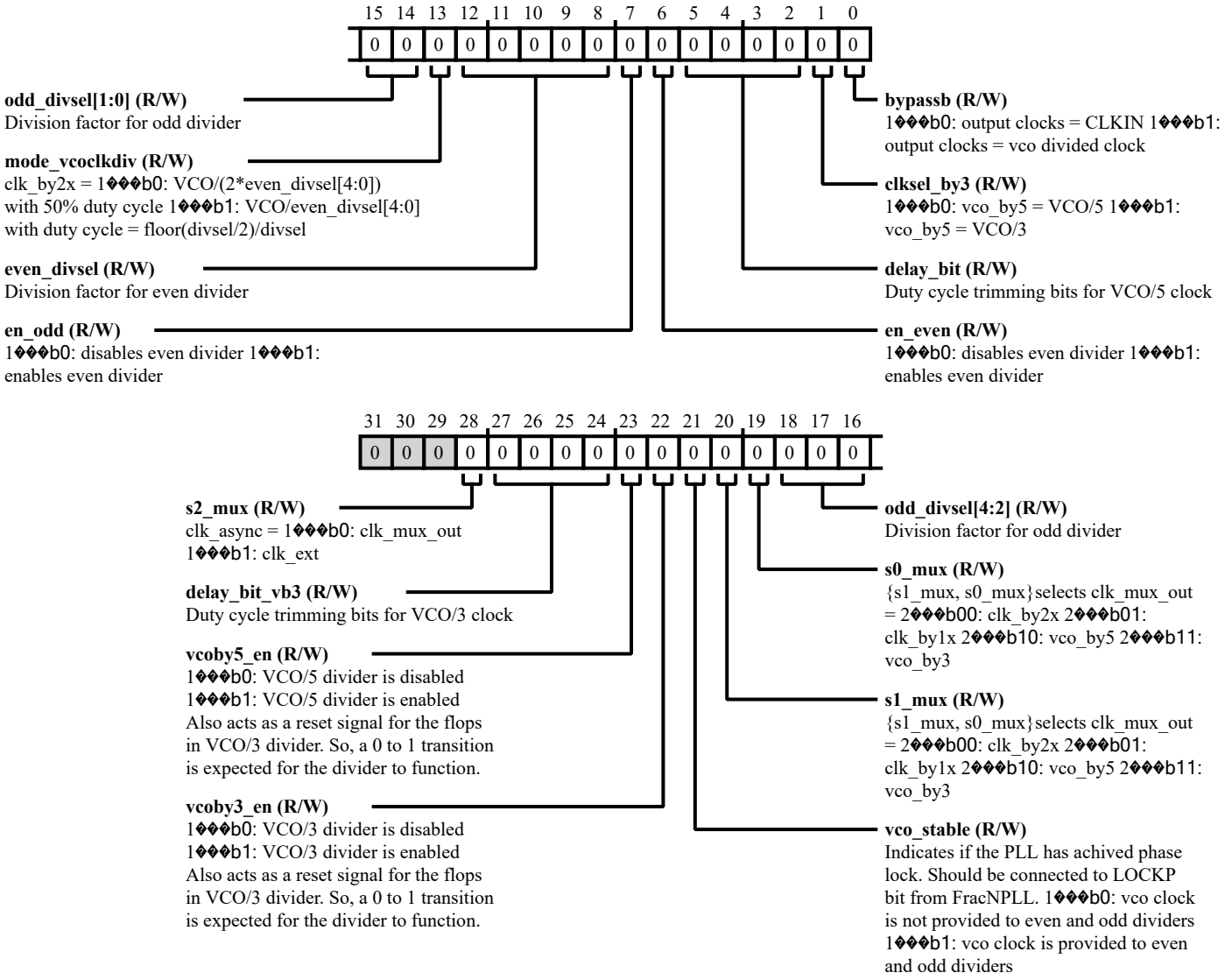


Figure 49-9: MISCREG_MINI_CGU0 Register Diagram

Table 49-10: MISCREG_MINI_CGU0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28 (R/W)	S2_MUX	clk_async = 1b0: clk_mux_out 1b1: clk_ext.
27:24 (R/W)	DELAY_BIT_VB3	Duty cycle trimming bits for VCO/3 clock.

Table 49-10: MISCREG_MINI_CGU0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/W)	VCOBY5_EN	1b0: VCO/5 divider is disabled 1b1: VCO/5 divider is enabled Also acts as a reset signal for the flops in VCO/3 divider. So, a 0 to 1 transition is expected for the divider to function..
22 (R/W)	VCOBY3_EN	1b0: VCO/3 divider is disabled 1b1: VCO/3 divider is enabled Also acts as a reset signal for the flops in VCO/3 divider. So, a 0 to 1 transition is expected for the divider to function..
21 (R/W)	VCO_STABLE	Indicates if the PLL has achived phase lock. Should be connected to LOCKP bit from FracNPLL. 1b0: vco clock is not provided to even and odd dividers 1b1: vco clock is provided to even and odd dividers.
20 (R/W)	S1_MUX	{s1_mux, s0_mux}selects clk_mux_out = 2b00: clk_by2x 2b01: clk_by1x 2b10: vco_by5 2b11: vco_by3.
19 (R/W)	S0_MUX	{s1_mux, s0_mux}selects clk_mux_out = 2b00: clk_by2x 2b01: clk_by1x 2b10: vco_by5 2b11: vco_by3.
18:14 (R/W)	ODD_DIVSEL	Division factor for odd divider.
13 (R/W)	MODE_VCOCLKDIV	clk_by2x = 1b0: VCO/(2*even_divsel[4:0]) with 50% duty cycle 1b1: VCO/even_divsel[4:0] with duty cycle = floor(divsel/2)/divsel.
12:8 (R/W)	EVEN_DIVSEL	Division factor for even divider.
7 (R/W)	EN_ODD	1b0: disables even divider 1b1: enables even divider.
6 (R/W)	EN_EVEN	1b0: disables even divider 1b1: enables even divider.
5:2 (R/W)	DELAY_BIT	Duty cycle trimming bits for VCO/5 clock.
1 (R/W)	CLKSEL_BY3	1b0: vco_by5 = VCO/5 1b1: vco_by5 = VCO/3.
0 (R/W)	BYPASSB	1b0: output clocks = CLKIN 1b1: output clocks = vco divided clock.

MINI CGU0 MISC Register

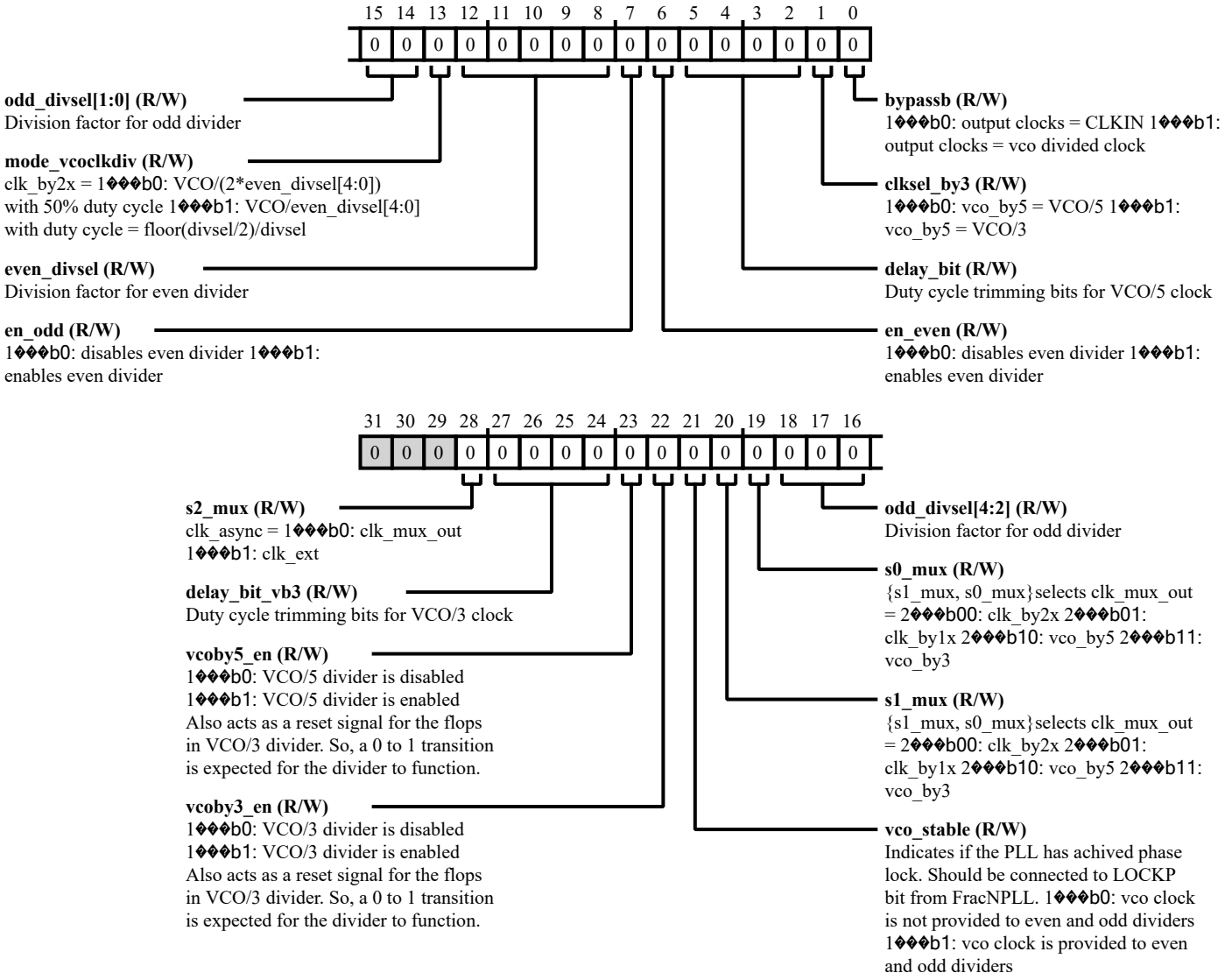


Figure 49-10: MISCREG_MINI_CGU1 Register Diagram

Table 49-11: MISCREG_MINI_CGU1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28 (R/W)	S2_MUX	clk_async = 1b0: clk_mux_out 1b1: clk_ext.
27:24 (R/W)	DELAY_BIT_VB3	Duty cycle trimming bits for VCO/3 clock.

Table 49-11: MISCREG_MINI_CGU1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/W)	VCOBY5_EN	1b0: VCO/5 divider is disabled 1b1: VCO/5 divider is enabled Also acts as a reset signal for the flops in VCO/3 divider. So, a 0 to 1 transition is expected for the divider to function..
22 (R/W)	VCOBY3_EN	1b0: VCO/3 divider is disabled 1b1: VCO/3 divider is enabled Also acts as a reset signal for the flops in VCO/3 divider. So, a 0 to 1 transition is expected for the divider to function..
21 (R/W)	VCO_STABLE	Indicates if the PLL has achived phase lock. Should be connected to LOCKP bit from FracNPLL. 1b0: vco clock is not provided to even and odd dividers 1b1: vco clock is provided to even and odd dividers.
20 (R/W)	S1_MUX	{s1_mux, s0_mux}selects clk_mux_out = 2b00: clk_by2x 2b01: clk_by1x 2b10: vco_by5 2b11: vco_by3.
19 (R/W)	S0_MUX	{s1_mux, s0_mux}selects clk_mux_out = 2b00: clk_by2x 2b01: clk_by1x 2b10: vco_by5 2b11: vco_by3.
18:14 (R/W)	ODD_DIVSEL	Division factor for odd divider.
13 (R/W)	MODE_VCOCLKDIV	clk_by2x = 1b0: VCO/(2*even_divsel[4:0]) with 50% duty cycle 1b1: VCO/even_divsel[4:0] with duty cycle = floor(divsel/2)/divsel.
12:8 (R/W)	EVEN_DIVSEL	Division factor for even divider.
7 (R/W)	EN_ODD	1b0: disables even divider 1b1: enables even divider.
6 (R/W)	EN_EVEN	1b0: disables even divider 1b1: enables even divider.
5:2 (R/W)	DELAY_BIT	Duty cycle trimming bits for VCO/5 clock.
1 (R/W)	CLKSEL_BY3	1b0: vco_by5 = VCO/5 1b1: vco_by5 = VCO/3.
0 (R/W)	BYPASSB	1b0: output clocks = CLKIN 1b1: output clocks = vco divided clock.

SH0 Prefetch Range Selection Register

In the MISCREG_SH0_PFB_RANGE_SELECT register, each bit field controls if the corresponding address range will be prefetched. When an address range is set it is prefetched and when it is zero there is no prefetch.

The MISCREG_SH0_PFB_RANGE_SELECT[31:16] bits control the range selection for the instruction cache, and MISCREG_SH0_PFB_RANGE_SELECT[15:0] bits control the range selection for the data cache.

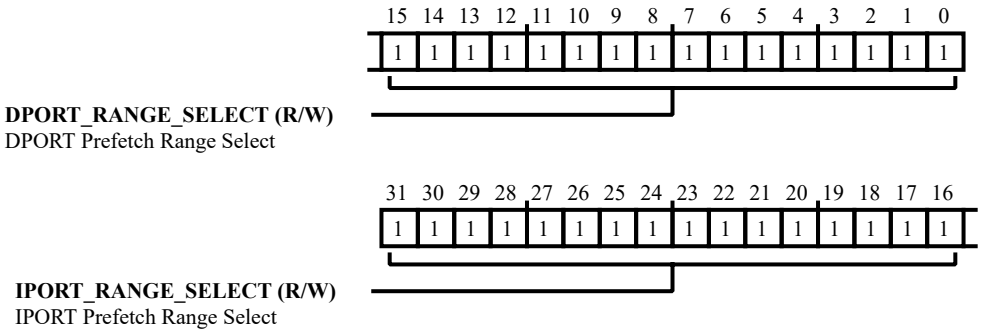


Figure 49-11: MISCREG_SH0_PFB_RANGE_SELECT Register Diagram

Table 49-12: MISCREG_SH0_PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	IPORT_RANGE_SELECT	IPORT Prefetch Range Select. The MISCREG_SH0_PFB_RANGE_SELECT.IPORT_RANGE_SELECT bit field indicates the prefetch range selection for IPORT.
15:0 (R/W)	DPORT_RANGE_SELECT	DPORT Prefetch Range Select. The MISCREG_SH0_PFB_RANGE_SELECT.DPORT_RANGE_SELECT bit field indicates the prefetch range selection for DPORT.

SH1 Prefetch Range Selection Register

Each bit field controls if the corresponding address range will be prefetched. When an address range is set it is prefetched an when it is zero there is no prefetch.

The MISCREG_SH1_PFB_RANGE_SELECT[31:16] bits control the range selection for the instruction cache, and MISCREG_SH1_PFB_RANGE_SELECT[15:0] bits control the range selection for the data cache.

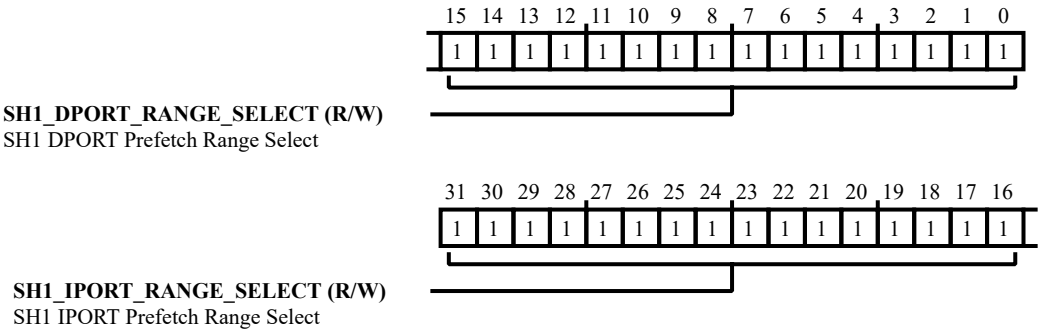


Figure 49-12: MISCREG_SH1_PFB_RANGE_SELECT Register Diagram

Table 49-13: MISCREG_SH1_PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	SH1_IPORT_RANGE_SELECT	SH1 IPORT Prefetch Range Select. The MISCREG_SH1_PFB_RANGE_SELECT.SH1_IPORT_RANGE_SELECT bit field indicates the prefetch range selection for IPORT.
15:0 (R/W)	SH1_DPORT_RANGE_SELECT	SH1 DPORT Prefetch Range Select. The MISCREG_SH1_PFB_RANGE_SELECT.SH1_DPORT_RANGE_SELECT bit field indicates the prefetch range selection for DPORT.

SPORT Direct Interrupt Enable for SH0 and SH1

The bits in the MISCREG_SHARC_BRIDGE_REMAP register control the direct connection of SPORT interrupts to the SHARC core.

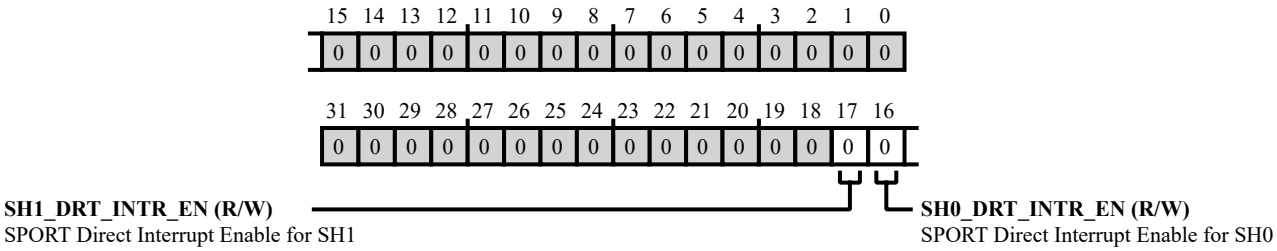


Figure 49-13: MISCREG_SHARC_BRIDGE_REMAP Register Diagram

Table 49-14: MISCREG_SHARC_BRIDGE_REMAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	SH1_DRT_INTR_EN	SPORT Direct Interrupt Enable for SH1. If the MISCREG_SHARC_BRIDGE_REMAP.SH1_DRT_INTR_EN bit is set a direct connection is enabled for SH1. Do not set the MISCREG_SHARC_BRIDGE_REMAP.SH1_DRT_INTR_EN bit when IIR1 of the SH1 is in use. If this bit is zero the direct connection is disabled.
16 (R/W)	SH0_DRT_INTR_EN	SPORT Direct Interrupt Enable for SH0. If the MISCREG_SHARC_BRIDGE_REMAP.SH0_DRT_INTR_EN bit is set a direct connection is enabled for SH0. Do not set the MISCREG_SHARC_BRIDGE_REMAP.SH0_DRT_INTR_EN bit when IIR1 of the SH0 is in use. If this bit is zero the direct connection is disabled.

XSPI1 Boot/Init Control Register 0

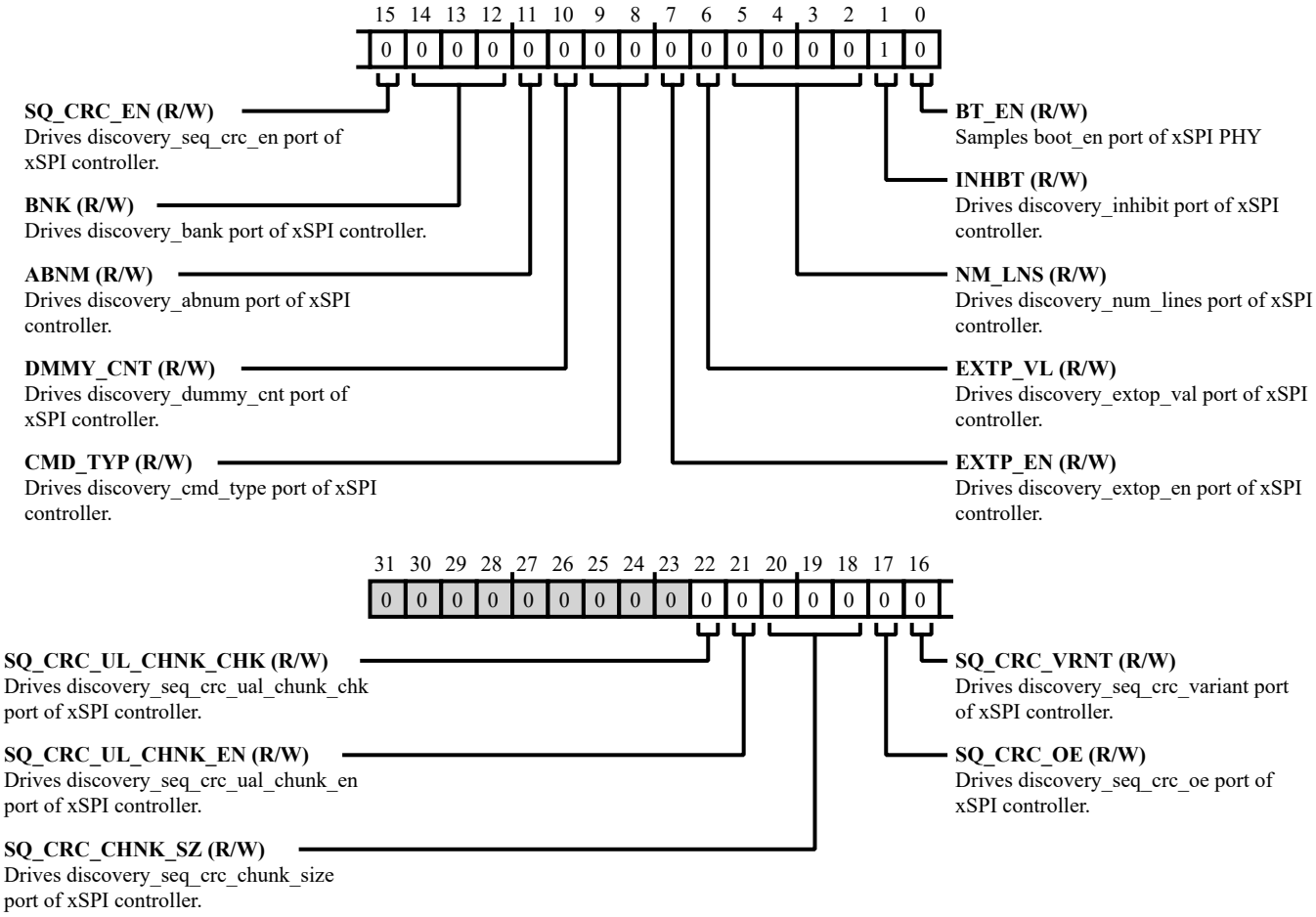


Figure 49-14: MISCREG_XSPI1_CTL0 Register Diagram

Table 49-15: MISCREG_XSPI1_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22 (R/W)	SQ_CRC_UL_CHNK_CHK	Drives discovery_seq_crc_ual_chunk_chk port of xSPI controller..
21 (R/W)	SQ_CRC_UL_CHNK_EN	Drives discovery_seq_crc_ual_chunk_en port of xSPI controller..
20:18 (R/W)	SQ_CRC_CHNK_SZ	Drives discovery_seq_crc_chunk_size port of xSPI controller..
17 (R/W)	SQ_CRC_OE	Drives discovery_seq_crc_oe port of xSPI controller..

Table 49-15: MISCREG_XSPI1_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	SQ_CRC_VRNT	Drives discovery_seq_crc_variant port of xSPI controller..
15 (R/W)	SQ_CRC_EN	Drives discovery_seq_crc_en port of xSPI controller..
14:12 (R/W)	BNK	Drives discovery_bank port of xSPI controller..
11 (R/W)	ABNM	Drives discovery_abnum port of xSPI controller..
10 (R/W)	DMMY_CNT	Drives discovery_dummy_cnt port of xSPI controller..
9:8 (R/W)	CMD_TYP	Drives discovery_cmd_type port of xSPI controller..
7 (R/W)	EXTP_EN	Drives discovery_extop_en port of xSPI controller..
6 (R/W)	EXTP_VL	Drives discovery_extop_val port of xSPI controller..
5:2 (R/W)	NM_LNS	Drives discovery_num_lines port of xSPI controller..
1 (R/W)	INHBT	Drives discovery_inhibit port of xSPI controller..
0 (R/W)	BT_EN	Samples boot_en port of xSPI PHY.

XSPI1 Boot/Init Control Register 1

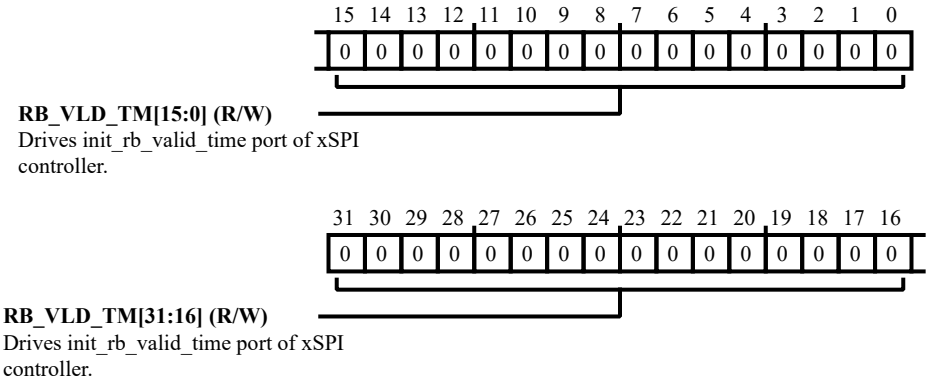


Figure 49-15: MISCREG_XSPI1_CTL1 Register Diagram

Table 49-16: MISCREG_XSPI1_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RB_VLD_TM	Drives init_rb_valid_time port of xSPI controller..

XSPI1 Boot/Init Control Register 2

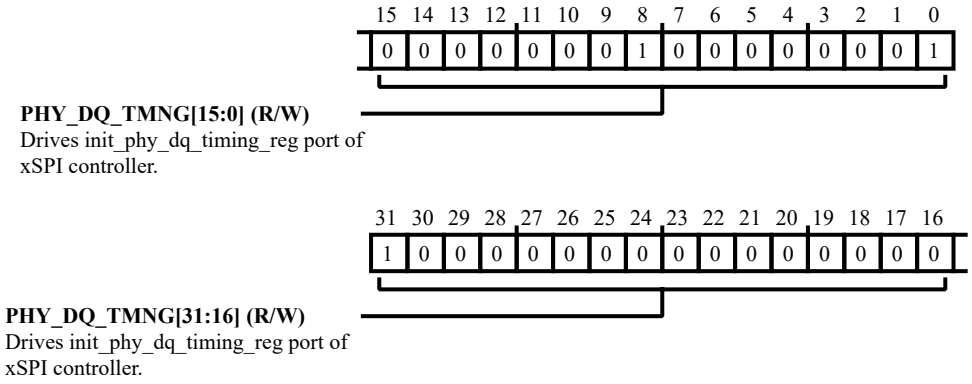


Figure 49-16: MISCREG_XSPI1_CTL2 Register Diagram

Table 49-17: MISCREG_XSPI1_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DQ_TMNG	Drives init_phy_dq_timing_reg port of xSPI controller..

XSPI1 Boot/Init Control Register 3

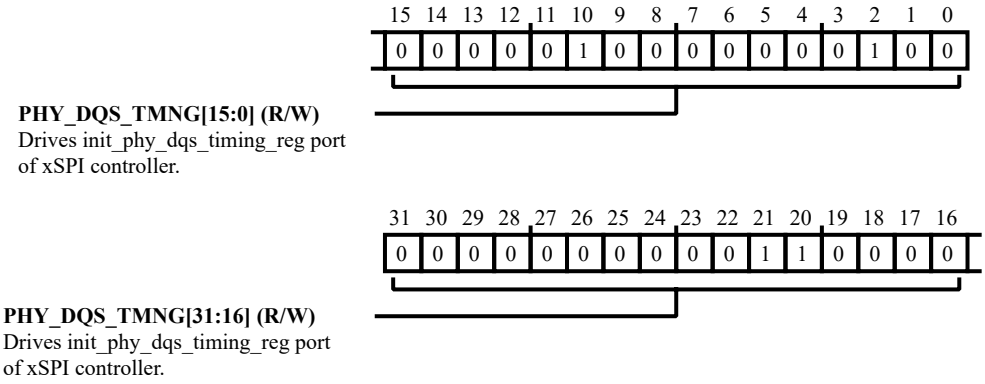


Figure 49-17: MISCREG_XSPI1_CTL3 Register Diagram

Table 49-18: MISCREG_XSPI1_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DQS_TMNG	Drives init_phy_dqs_timing_reg port of xSPI controller..

XSPI1 Boot/Init Control Register 4

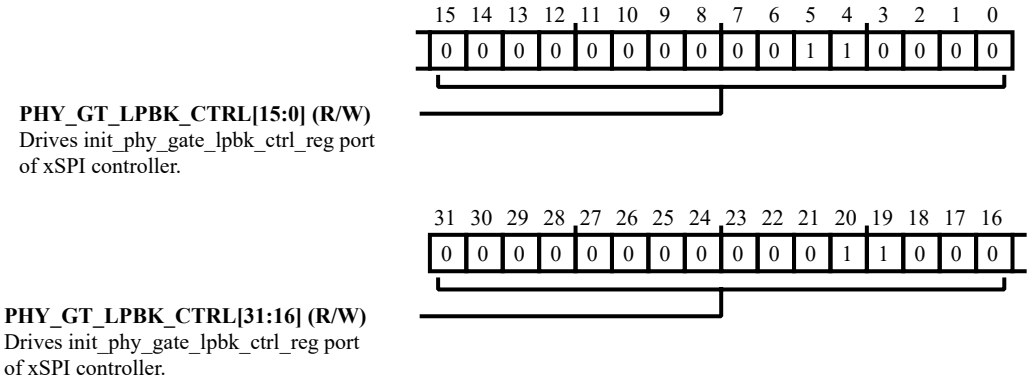


Figure 49-18: MISCREG_XSPI1_CTL4 Register Diagram

Table 49-19: MISCREG_XSPI1_CTL4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_GT_LPBK_CTRL	Drives init_phy_gate_lpbk_ctrl_reg port of xSPI controller..

XSPI1 Boot/Init Control Register 5

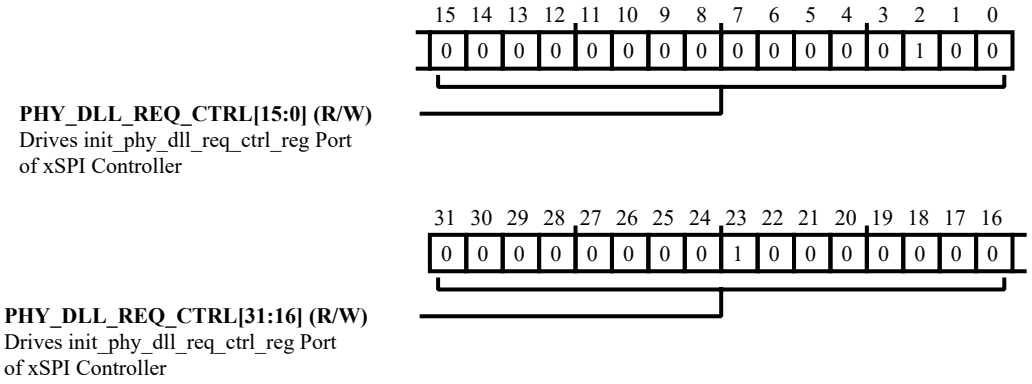


Figure 49-19: MISCREG_XSPI1_CTL5 Register Diagram

Table 49-20: MISCREG_XSPI1_CTL5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DLL_REQ_CTRL	Drives init_phy_dll_req_ctrl_reg Port of xSPI Controller.

XSPI1 Boot/Init Control Register 6

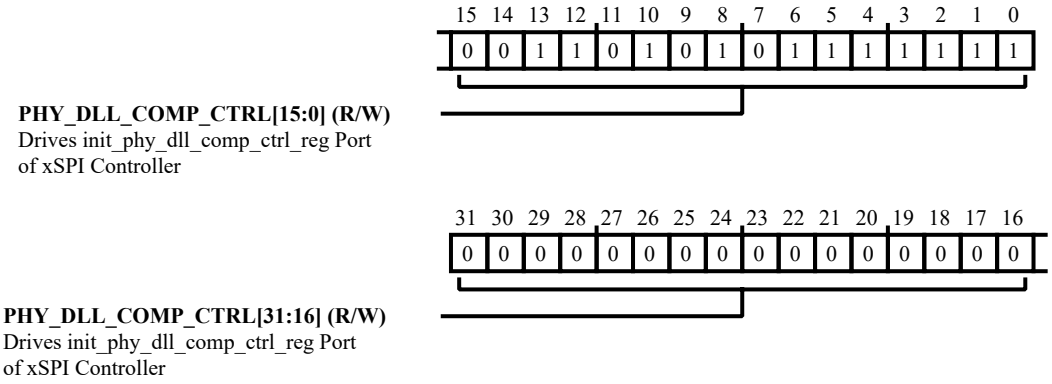


Figure 49-20: MISCREG_XSPI1_CTL6 Register Diagram

Table 49-21: MISCREG_XSPI1_CTL6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DLL_COMP_CTRL	Drives init_phy_dll_comp_ctrl_reg Port of xSPI Controller.

XSPI1 Boot/Init Control Register 7

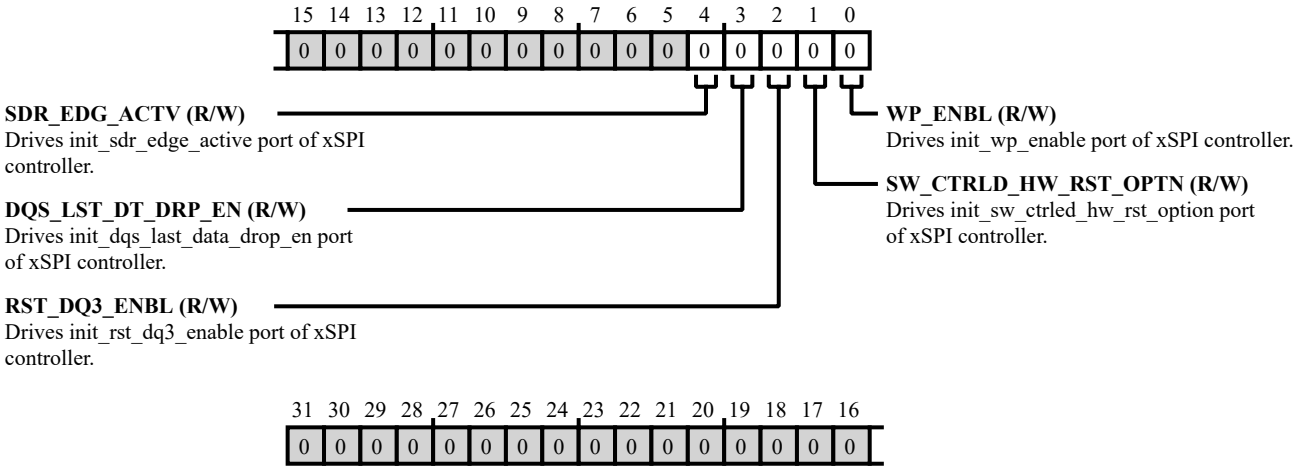


Figure 49-21: MISCREG_XSPI1_CTL7 Register Diagram

Table 49-22: MISCREG_XSPI1_CTL7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	SDR_EDG_ACTV	Drives init_sdr_edge_active port of xSPI controller..
3 (R/W)	DQS_LST_DT_DRP_EN	Drives init_dqs_last_data_drop_en port of xSPI controller..
2 (R/W)	RST_DQ3_ENBL	Drives init_rst_dq3_enable port of xSPI controller..
1 (R/W)	SW_CTRLD_HW_RST_OPTN	Drives init_sw_ctrled_hw_rst_option port of xSPI controller..
0 (R/W)	WP_ENBL	Drives init_wp_enable port of xSPI controller..

XSPI1 Boot/Init Control Register 8

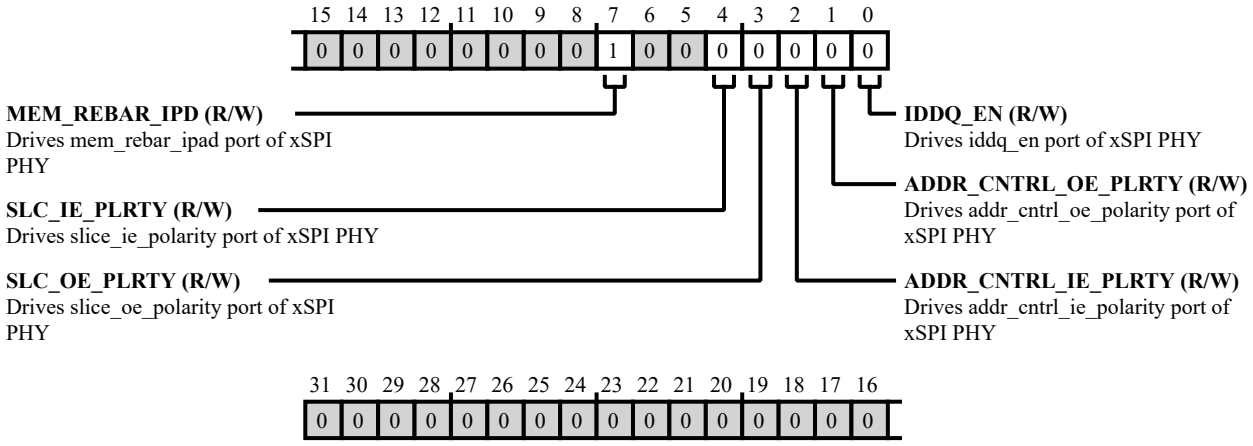


Figure 49-22: MISCREG_XSPI1_CTL8 Register Diagram

Table 49-23: MISCREG_XSPI1_CTL8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	MEM_REBAR_IPD	Drives mem_rebar_ipad port of xSPI PHY.
4 (R/W)	SLC_IE_PLRTY	Drives slice_ie_polarity port of xSPI PHY.
3 (R/W)	SLC_OE_PLRTY	Drives slice_oe_polarity port of xSPI PHY.
2 (R/W)	ADDR_CNTRL_IE_PLRTY	Drives addr_cntrl_ie_polarity port of xSPI PHY.
1 (R/W)	ADDR_CNTRL_OE_PLRTY	Drives addr_cntrl_oe_polarity port of xSPI PHY.
0 (R/W)	IDDQ_EN	Drives iddq_en port of xSPI PHY.

XSPI1 Boot/Init Reset Control Register

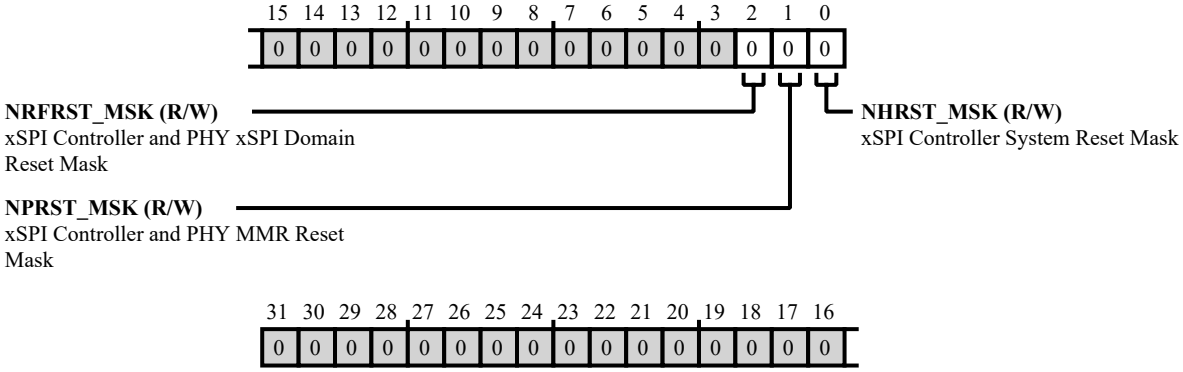


Figure 49-23: MISCREG_XSPI1_RSTCTL Register Diagram

Table 49-24: MISCREG_XSPI1_RSTCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W)	NRFRST_MSK	xSPI Controller and PHY xSPI Domain Reset Mask.
1 (R/W)	NPRST_MSK	xSPI Controller and PHY MMR Reset Mask.
0 (R/W)	NHRST_MSK	xSPI Controller System Reset Mask.

XSPI1 Boot/Init Status Register

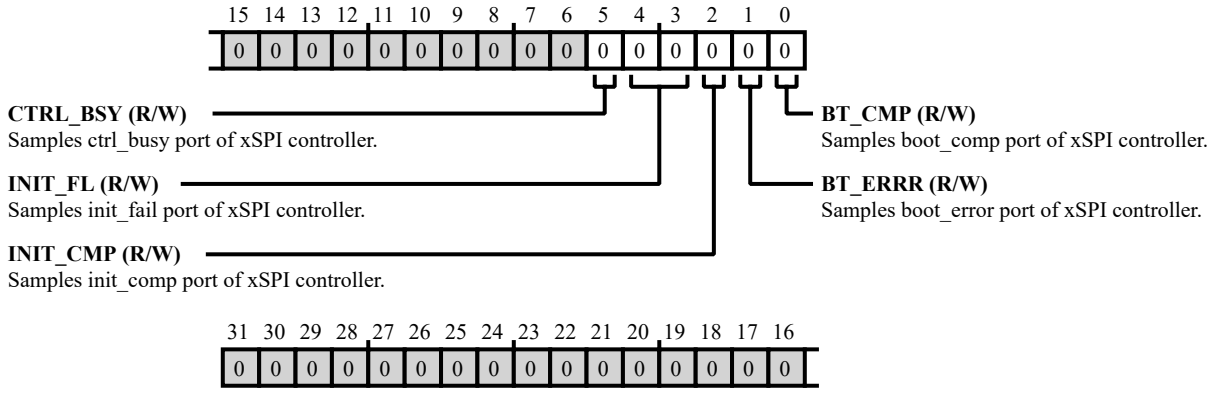


Figure 49-24: MISCREG_XSPI1_STAT0 Register Diagram

Table 49-25: MISCREG_XSPI1_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	CTRL_BSY	Samples ctrl_busy port of xSPI controller..
4:3 (R/W)	INIT_FL	Samples init_fail port of xSPI controller..
2 (R/W)	INIT_CMP	Samples init_comp port of xSPI controller..
1 (R/W)	BT_ERRR	Samples boot_error port of xSPI controller..
0 (R/W)	BT_CMP	Samples boot_comp port of xSPI controller..

XSPI Boot/Init Control Register 0

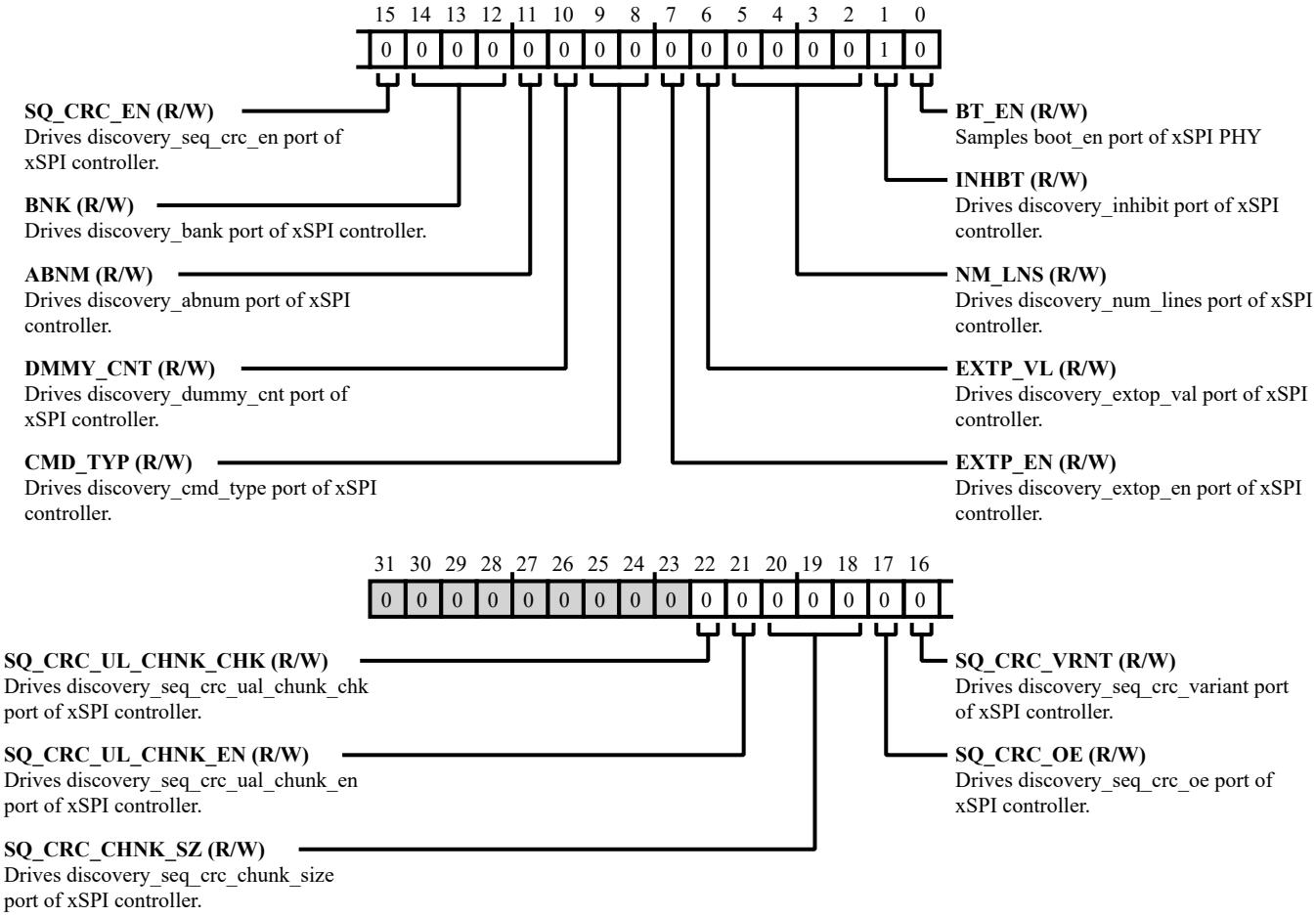


Figure 49-25: MISCREG_XSPI_CTL0 Register Diagram

Table 49-26: MISCREG_XSPI_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22 (R/W)	SQ_CRC_UL_CHNK_CHK	Drives discovery_seq_crc_ual_chunk_chk port of xSPI controller..
21 (R/W)	SQ_CRC_UL_CHNK_EN	Drives discovery_seq_crc_ual_chunk_en port of xSPI controller..
20:18 (R/W)	SQ_CRC_CHNK_SZ	Drives discovery_seq_crc_chunk_size port of xSPI controller..
17 (R/W)	SQ_CRC_OE	Drives discovery_seq_crc_oe port of xSPI controller..

Table 49-26: MISCREG_XSPI_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	SQ_CRC_VRNT	Drives discovery_seq_crc_variant port of xSPI controller..
15 (R/W)	SQ_CRC_EN	Drives discovery_seq_crc_en port of xSPI controller..
14:12 (R/W)	BNK	Drives discovery_bank port of xSPI controller..
11 (R/W)	ABNM	Drives discovery_abnum port of xSPI controller..
10 (R/W)	DMMY_CNT	Drives discovery_dummy_cnt port of xSPI controller..
9:8 (R/W)	CMD_TYP	Drives discovery_cmd_type port of xSPI controller..
7 (R/W)	EXTP_EN	Drives discovery_extop_en port of xSPI controller..
6 (R/W)	EXTP_VL	Drives discovery_extop_val port of xSPI controller..
5:2 (R/W)	NM_LNS	Drives discovery_num_lines port of xSPI controller..
1 (R/W)	INHBT	Drives discovery_inhibit port of xSPI controller..
0 (R/W)	BT_EN	Samples boot_en port of xSPI PHY.

XSPI Boot/Init Control Register 1

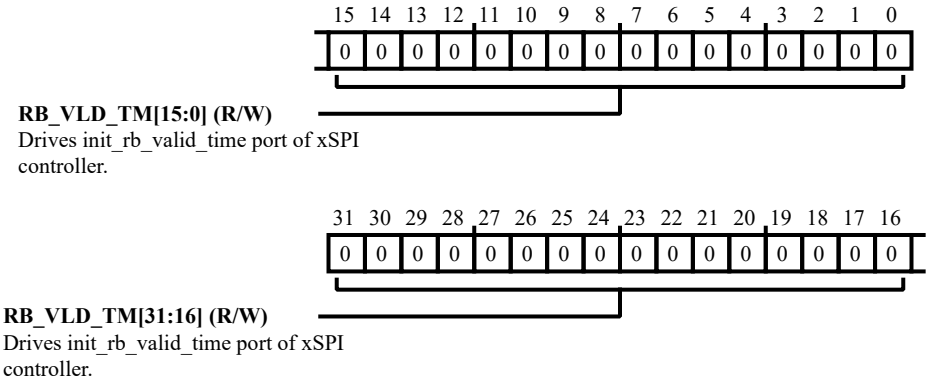


Figure 49-26: MISCREG_XSPI_CTL1 Register Diagram

Table 49-27: MISCREG_XSPI_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	RB_VLD_TM	Drives init_rb_valid_time port of xSPI controller..

XSPI Boot/Init Control Register 2

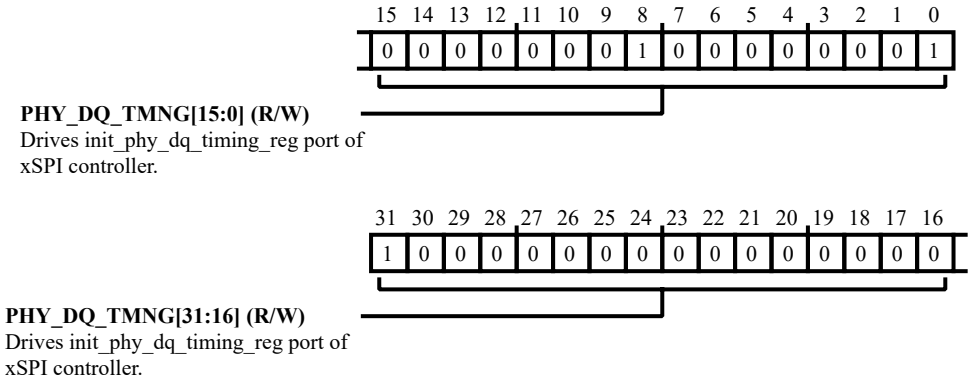


Figure 49-27: MISCREG_XSPI_CTL2 Register Diagram

Table 49-28: MISCREG_XSPI_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DQ_TMNG	Drives init_phy_dq_timing_reg port of xSPI controller..

XSPI Boot/Init Control Register 3

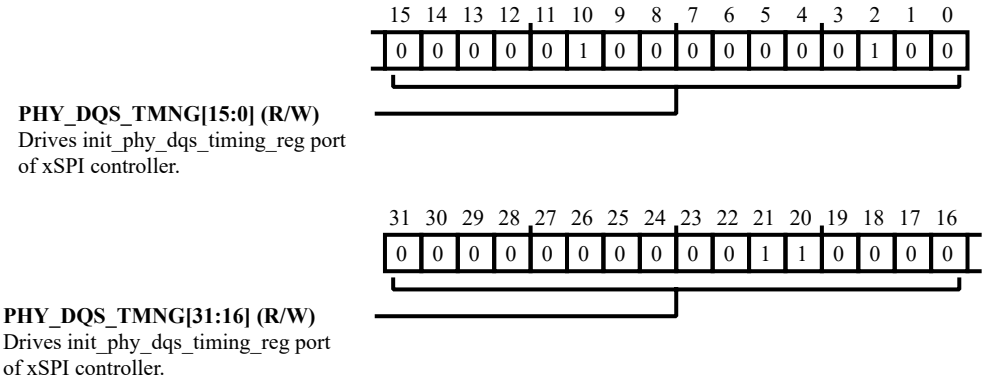


Figure 49-28: MISCREG_XSPI_CTL3 Register Diagram

Table 49-29: MISCREG_XSPI_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DQS_TMNG	Drives init_phy_dqs_timing_reg port of xSPI controller..

XSPI Boot/Init Control Register 4

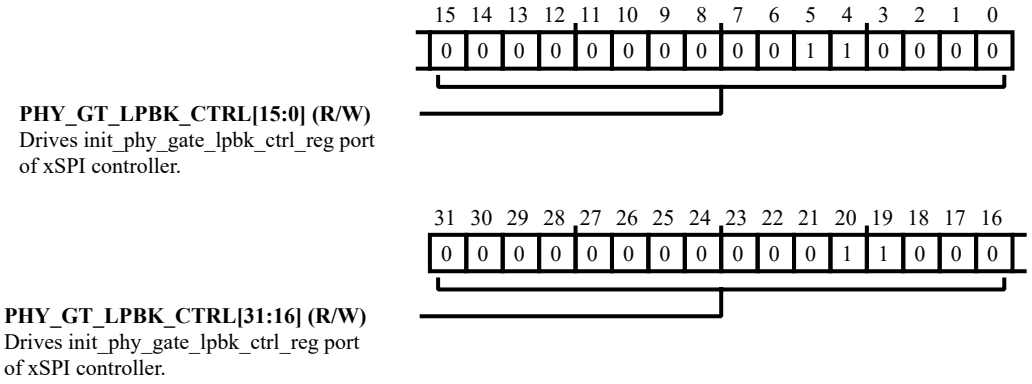


Figure 49-29: MISCREG_XSPI_CTL4 Register Diagram

Table 49-30: MISCREG_XSPI_CTL4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_GT_LPBK_CTRL	Drives init_phy_gate_lpbk_ctrl_reg port of xSPI controller..

XSPI Boot/Init Control Register 5

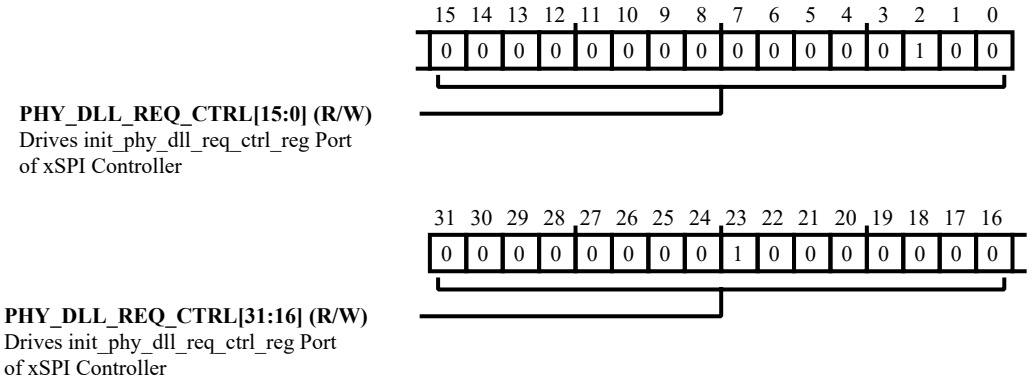


Figure 49-30: MISCREG_XSPI_CTL5 Register Diagram

Table 49-31: MISCREG_XSPI_CTL5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DLL_REQ_CTRL	Drives init_phy_dll_req_ctrl_reg Port of xSPI Controller.

XSPI Boot/Init Control Register 6

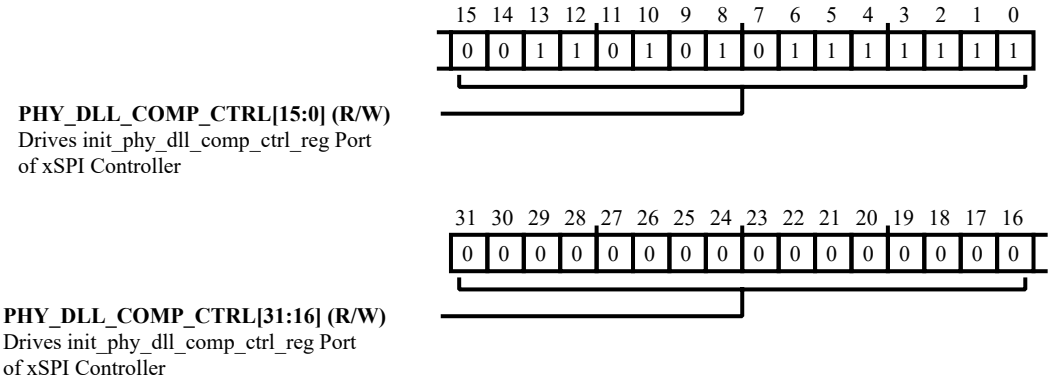


Figure 49-31: MISCREG_XSPI_CTL6 Register Diagram

Table 49-32: MISCREG_XSPI_CTL6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PHY_DLL_COMP_CTRL	Drives init_phy_dll_comp_ctrl_reg Port of xSPI Controller.

XSPI Boot/Init Control Register 7

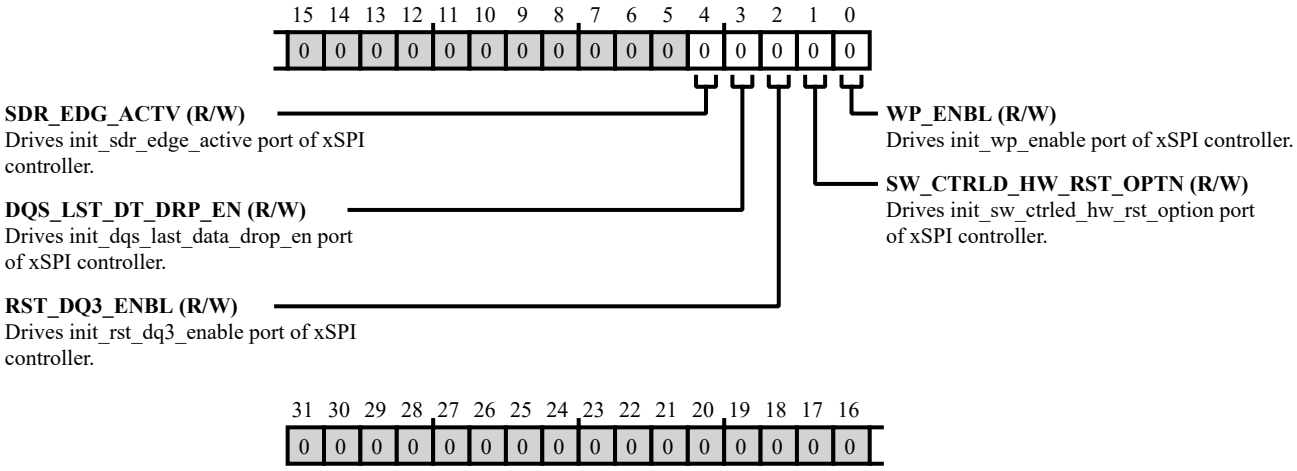


Figure 49-32: MISCREG_XSPI_CTL7 Register Diagram

Table 49-33: MISCREG_XSPI_CTL7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	SDR_EDG_ACTV	Drives init_sdr_edge_active port of xSPI controller..
3 (R/W)	DQS_LST_DT_DRP_EN	Drives init_dqs_last_data_drop_en port of xSPI controller..
2 (R/W)	RST_DQ3_ENBL	Drives init_rst_dq3_enable port of xSPI controller..
1 (R/W)	SW_CTRLD_HW_RST_OPTN	Drives init_sw_ctrled_hw_rst_option port of xSPI controller..
0 (R/W)	WP_ENBL	Drives init_wp_enable port of xSPI controller..

XSPI Boot/Init Control Register 8

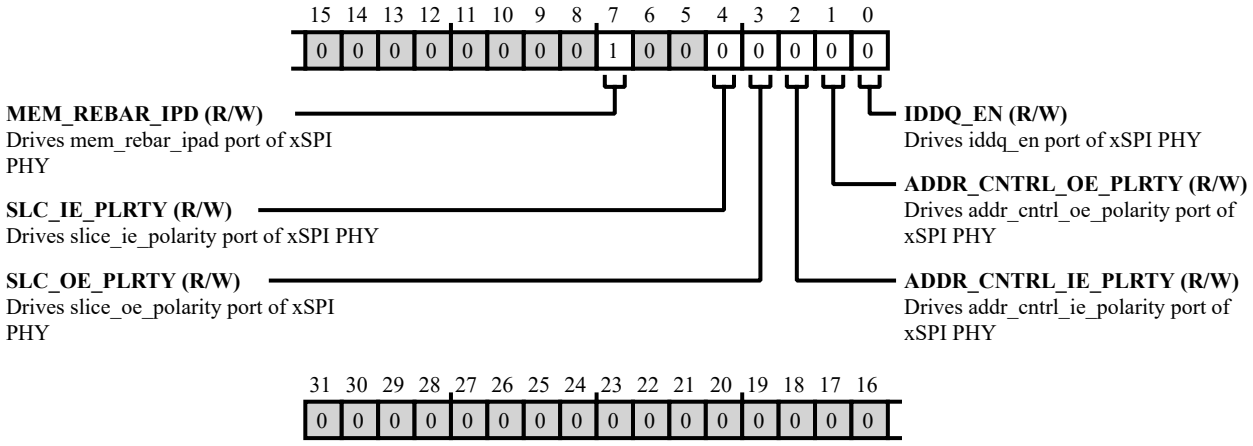


Figure 49-33: MISCREG_XSPI_CTL8 Register Diagram

Table 49-34: MISCREG_XSPI_CTL8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	MEM_REBAR_IPD	Drives mem_rebar_ipad port of xSPI PHY.
4 (R/W)	SLC_IE_PLRTY	Drives slice_ie_polarity port of xSPI PHY.
3 (R/W)	SLC_OE_PLRTY	Drives slice_oe_polarity port of xSPI PHY.
2 (R/W)	ADDR_CNTRL_IE_PLRTY	Drives addr_cntrl_ie_polarity port of xSPI PHY.
1 (R/W)	ADDR_CNTRL_OE_PLRTY	Drives addr_cntrl_oe_polarity port of xSPI PHY.
0 (R/W)	IDDQ_EN	Drives iddq_en port of xSPI PHY.

XSPI Boot/Init Reset Control Register

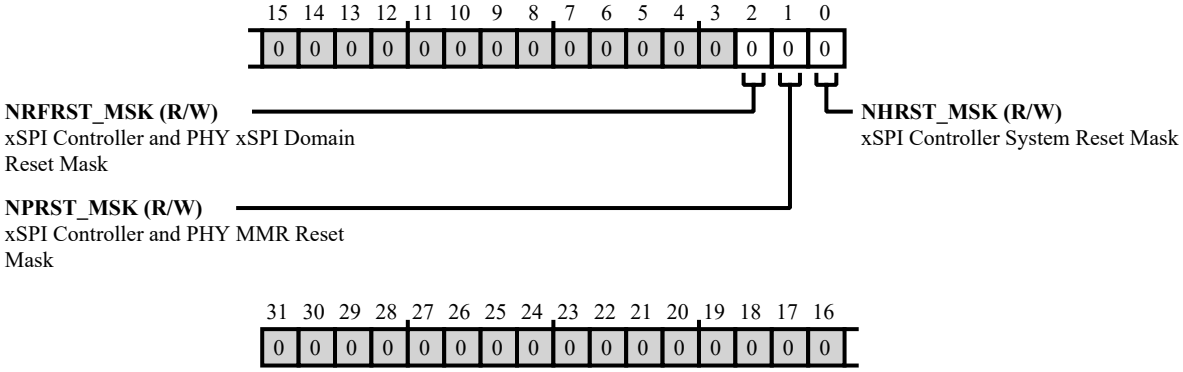


Figure 49-34: MISCREG_XSPI_RSTCTL Register Diagram

Table 49-35: MISCREG_XSPI_RSTCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W)	NRFRST_MSK	xSPI Controller and PHY xSPI Domain Reset Mask.
1 (R/W)	NPRST_MSK	xSPI Controller and PHY MMR Reset Mask.
0 (R/W)	NHRST_MSK	xSPI Controller System Reset Mask.

XSPI Boot/Init Status Register

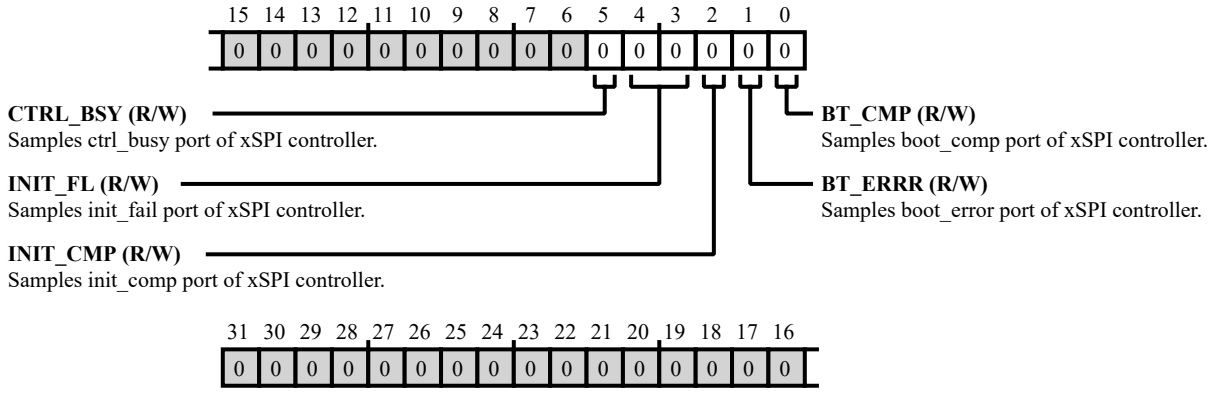


Figure 49-35: MISCREG_XSPI_STAT0 Register Diagram

Table 49-36: MISCREG_XSPI_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	CTRL_BSY	Samples ctrl_busy port of xSPI controller..
4:3 (R/W)	INIT_FL	Samples init_fail port of xSPI controller..
2 (R/W)	INIT_CMP	Samples init_comp port of xSPI controller..
1 (R/W)	BT_ERRR	Samples boot_error port of xSPI controller..
0 (R/W)	BT_CMP	Samples boot_comp port of xSPI controller..

Appendix A ADSP-2184x Register List

This appendix lists Memory-Mapped Register address and register names. The modules are presented in alphabetical order.

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31110000	A55_CLUSTER_ROMENTRY0	A55_CLUSTER ROM Table Entry 0 Register	0x00001002
0x31110004	A55_CLUSTER_ROMENTRY1	A55_CLUSTER ROM Table Entry 1 Register	0x00002002
0x31110008	A55_CLUSTER_ROMENTRY2	A55_CLUSTER ROM Table Entry 2 Register	0x00004003
0x3111000C	A55_CLUSTER_ROMENTRY3	A55_CLUSTER ROM Table Entry 3 Register	0x00005003
0x31110010	A55_CLUSTER_ROMENTRY4	A55_CLUSTER ROM Table Entry 4 Register	0x00000002
0x31110014	A55_CLUSTER_ROMENTRY5	A55_CLUSTER ROM Table Entry 5 Register	0x00000002
0x31110018	A55_CLUSTER_ROMENTRY6	A55_CLUSTER ROM Table Entry 6 Register	0x00008002
0x3111001C	A55_CLUSTER_ROMENTRY7	A55_CLUSTER ROM Table Entry 7 Register	0x00009002
0x31110020	A55_CLUSTER_ROMENTRY8	A55_CLUSTER ROM Table Entry 8 Register	0x00000002
0x31110024	A55_CLUSTER_ROMENTRY9	A55_CLUSTER ROM Table Entry 9 Register	0x00000002
0x31110028	A55_CLUSTER_ROMENTRY10	A55_CLUSTER ROM Table Entry 10 Register	0x0000C003
0x3111002C	A55_CLUSTER_ROMENTRY11	A55_CLUSTER ROM Table Entry 11 Register	0x0000D003
0x31110030	A55_CLUSTER_ROMENTRY12	A55_CLUSTER ROM Table Entry 12 Register	0x00000002
0x31110034	A55_CLUSTER_ROMENTRY13	A55_CLUSTER ROM Table Entry 13 Register	0x00000002
0x31110038	A55_CLUSTER_ROMENTRY14	A55_CLUSTER ROM Table Entry 14 Register	0x00014003
0x3111003C	A55_CLUSTER_ROMENTRY15	A55_CLUSTER ROM Table Entry 15 Register	0x00015003

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110040	A55_CLUSTER_ROMENTRY16	A55_CLUSTER ROM Table Entry 16 Register	0x00000002
0x31110044	A55_CLUSTER_ROMENTRY17	A55_CLUSTER ROM Table Entry 17 Register	0x00000002
0x31110048	A55_CLUSTER_ROMENTRY18	A55_CLUSTER ROM Table Entry 18 Register	0x0001C003
0x3111004C	A55_CLUSTER_ROMENTRY19	A55_CLUSTER ROM Table Entry 19 Register	0x0001D003
0x31110050	A55_CLUSTER_ROMENTRY20	A55_CLUSTER ROM Table Entry 20 Register	0x00000002
0x31110054	A55_CLUSTER_ROMENTRY21	A55_CLUSTER ROM Table Entry 21 Register	0x00000002
0x31110058	A55_CLUSTER_ROMENTRY22	A55_CLUSTER ROM Table Entry 22 Register	0x00000002
0x3111005C	A55_CLUSTER_ROMENTRY23	A55_CLUSTER ROM Table Entry 23 Register	0x00000002
0x31110060	A55_CLUSTER_ROMENTRY24	A55_CLUSTER ROM Table Entry 24 Register	0x00000002
0x31110064	A55_CLUSTER_ROMENTRY25	A55_CLUSTER ROM Table Entry 25 Register	0x00000002
0x31110068	A55_CLUSTER_ROMENTRY26	A55_CLUSTER ROM Table Entry 26 Register	0x00000002
0x3111006C	A55_CLUSTER_ROMENTRY27	A55_CLUSTER ROM Table Entry 27 Register	0x00000002
0x31110070	A55_CLUSTER_ROMENTRY28	A55_CLUSTER ROM Table Entry 28 Register	0x00000002
0x31110074	A55_CLUSTER_ROMENTRY29	A55_CLUSTER ROM Table Entry 29 Register	0x00000002
0x31110078	A55_CLUSTER_ROMENTRY30	A55_CLUSTER ROM Table Entry 30 Register	0x00000002
0x3111007C	A55_CLUSTER_ROMENTRY31	A55_CLUSTER ROM Table Entry 31 Register	0x00000002
0x31110080	A55_CLUSTER_ROMENTRY32	A55_CLUSTER ROM Table Entry 32 Register	0x00000002
0x31110084	A55_CLUSTER_ROMENTRY33	A55_CLUSTER ROM Table Entry 33 Register	0x00000002

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110088	A55_CLUSTER_ROMENTRY34	A55_CLUSTER ROM Table Entry 34 Register	0x00000002
0x3111008C	A55_CLUSTER_ROMENTRY35	A55_CLUSTER ROM Table Entry 35 Register	0x00000002
0x31110090	A55_CLUSTER_ROMENTRY36	A55_CLUSTER ROM Table Entry 36 Register	0x00000002
0x31110094	A55_CLUSTER_ROMENTRY37	A55_CLUSTER ROM Table Entry 37 Register	0x00000002
0x31110098	A55_CLUSTER_ROMENTRY38	A55_CLUSTER ROM Table Entry 38 Register	0x00000002
0x3111009C	A55_CLUSTER_ROMENTRY39	A55_CLUSTER ROM Table Entry 39 Register	0x00000002
0x311100A0	A55_CLUSTER_ROMENTRY40	A55_CLUSTER ROM Table Entry 40 Register	0x00000002
0x311100A4	A55_CLUSTER_ROMENTRY41	A55_CLUSTER ROM Table Entry 41 Register	0x00000002
0x311100A8	A55_CLUSTER_ROMENTRY42	A55_CLUSTER ROM Table Entry 42 Register	0x00000002
0x311100AC	A55_CLUSTER_ROMENTRY43	A55_CLUSTER ROM Table Entry 43 Register	0x00000002
0x311100B0	A55_CLUSTER_ROMENTRY44	A55_CLUSTER ROM Table Entry 44 Register	0x00000002
0x311100B4	A55_CLUSTER_ROMENTRY45	A55_CLUSTER ROM Table Entry 45 Register	0x00000002
0x311100B8	A55_CLUSTER_ROMENTRY46	A55_CLUSTER ROM Table Entry 46 Register	0x00000002
0x311100BC	A55_CLUSTER_ROMENTRY47	A55_CLUSTER ROM Table Entry 47 Register	0x00000002
0x311100C0	A55_CLUSTER_ROMENTRY48	A55_CLUSTER ROM Table Entry 48 Register	0x00000002
0x311100C4	A55_CLUSTER_ROMENTRY49	A55_CLUSTER ROM Table Entry 49 Register	0x00000002
0x311100C8	A55_CLUSTER_ROMENTRY50	A55_CLUSTER ROM Table Entry 50 Register	0x00000002
0x311100CC	A55_CLUSTER_ROMENTRY51	A55_CLUSTER ROM Table Entry 51 Register	0x00000002

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311100D0	A55_CLUSTER_ROMENTRY52	A55_CLUSTER ROM Table Entry 52 Register	0x00000002
0x311100D4	A55_CLUSTER_ROMENTRY53	A55_CLUSTER ROM Table Entry 53 Register	0x00000002
0x311100D8	A55_CLUSTER_ROMENTRY54	A55_CLUSTER ROM Table Entry 54 Register	0x00000002
0x311100DC	A55_CLUSTER_ROMENTRY55	A55_CLUSTER ROM Table Entry 55 Register	0x00000002
0x311100E0	A55_CLUSTER_ROMENTRY56	A55_CLUSTER ROM Table Entry 56 Register	0x00000002
0x311100E4	A55_CLUSTER_ROMENTRY57	A55_CLUSTER ROM Table Entry 57 Register	0x00000002
0x311100E8	A55_CLUSTER_ROMENTRY58	A55_CLUSTER ROM Table Entry 58 Register	0x00000002
0x311100EC	A55_CLUSTER_ROMENTRY59	A55_CLUSTER ROM Table Entry 59 Register	0x00000002
0x311100F0	A55_CLUSTER_ROMENTRY60	A55_CLUSTER ROM Table Entry 60 Register	0x00000002
0x311100F4	A55_CLUSTER_ROMENTRY61	A55_CLUSTER ROM Table Entry 61 Register	0x00000002
0x311100F8	A55_CLUSTER_ROMENTRY62	A55_CLUSTER ROM Table Entry 62 Register	0x00000002
0x311100FC	A55_CLUSTER_ROMENTRY63	A55_CLUSTER ROM Table Entry 63 Register	0x00000002
0x31110100	A55_CLUSTER_ROMENTRY64	A55_CLUSTER ROM Table Entry 64 Register	0x00000002
0x31110104	A55_CLUSTER_ROMENTRY65	A55_CLUSTER ROM Table Entry 65 Register	0x00000002
0x31110108	A55_CLUSTER_ROMENTRY66	A55_CLUSTER ROM Table Entry 66 Register	0x00000002
0x3111010C	A55_CLUSTER_ROMENTRY67	A55_CLUSTER ROM Table Entry 67 Register	0x00000002
0x31110110	A55_CLUSTER_ROMENTRY68	A55_CLUSTER ROM Table Entry 68 Register	0x00000002
0x31110114	A55_CLUSTER_ROMENTRY69	A55_CLUSTER ROM Table Entry 69 Register	0x00000002

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110118	A55_CLUSTER_ROMENTRY70	A55_CLUSTER ROM Table Entry 70 Register	0x00000002
0x3111011C	A55_CLUSTER_ROMENTRY71	A55_CLUSTER ROM Table Entry 71 Register	0x00000002
0x31110120	A55_CLUSTER_ROMENTRY72	A55_CLUSTER ROM Table Entry 72 Register	0x00000002
0x31110124	A55_CLUSTER_ROMENTRY73	A55_CLUSTER ROM Table Entry 73 Register	0x00000002
0x31110128	A55_CLUSTER_ROMENTRY74	A55_CLUSTER ROM Table Entry 74 Register	0x00000002
0x3111012C	A55_CLUSTER_ROMENTRY75	A55_CLUSTER ROM Table Entry 75 Register	0x00000002
0x31110130	A55_CLUSTER_ROMENTRY76	A55_CLUSTER ROM Table Entry 76 Register	0x00000002
0x31110134	A55_CLUSTER_ROMENTRY77	A55_CLUSTER ROM Table Entry 77 Register	0x00000002
0x31110138	A55_CLUSTER_ROMENTRY78	A55_CLUSTER ROM Table Entry 78 Register	0x00000002
0x3111013C	A55_CLUSTER_ROMENTRY79	A55_CLUSTER ROM Table Entry 79 Register	0x00000002
0x31110140	A55_CLUSTER_ROMENTRY80	A55_CLUSTER ROM Table Entry 80 Register	0x00000002
0x31110144	A55_CLUSTER_ROMENTRY81	A55_CLUSTER ROM Table Entry 81 Register	0x00000002
0x31110A00	A55_CLUSTER_DBGPCR0	A55_CLUSTER Debug Power Control 0 Register	0x00000000
0x31110A04	A55_CLUSTER_DBGPCR1	A55_CLUSTER Debug Power Control 1 Register	0x00000000
0x31110A08	A55_CLUSTER_DBGPCR2	A55_CLUSTER Debug Power Control 2 Register	0x00000000
0x31110A0C	A55_CLUSTER_DBGPCR3	A55_CLUSTER Debug Power Control 3 Register	0x00000000
0x31110A10	A55_CLUSTER_DBGPCR4	A55_CLUSTER Debug Power Control 4 Register	0x00000000
0x31110A14	A55_CLUSTER_DBGPCR5	A55_CLUSTER Debug Power Control 5 Register	0x00000000
0x31110A18	A55_CLUSTER_DBGPCR6	A55_CLUSTER Debug Power Control 6 Register	0x00000000
0x31110A1C	A55_CLUSTER_DBGPCR7	A55_CLUSTER Debug Power Control 7 Register	0x00000000
0x31110A20	A55_CLUSTER_DBGPCR8	A55_CLUSTER Debug Power Control 8 Register	0x00000000
0x31110A80	A55_CLUSTER_DBGPSR0	A55_CLUSTER Debug Power Status 0 Register	0x00000000
0x31110A84	A55_CLUSTER_DBGPSR1	A55_CLUSTER Debug Power Status 1 Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110A88	A55_CLUSTER_DBGPSR2	A55_CLUSTER Debug Power Status 2 Register	0x00000000
0x31110A8C	A55_CLUSTER_DBGPSR3	A55_CLUSTER Debug Power Status 3 Register	0x00000000
0x31110A90	A55_CLUSTER_DBGPSR4	A55_CLUSTER Debug Power Status 4 Register	0x00000000
0x31110A94	A55_CLUSTER_DBGPSR5	A55_CLUSTER Debug Power Status 5 Register	0x00000000
0x31110A98	A55_CLUSTER_DBGPSR6	A55_CLUSTER Debug Power Status 6 Register	0x00000000
0x31110A9C	A55_CLUSTER_DBGPSR7	A55_CLUSTER Debug Power Status 7 Register	0x00000000
0x31110AA0	A55_CLUSTER_DBGPSR8	A55_CLUSTER Debug Power Status 8 Register	0x00000000
0x31110C00	A55_CLUSTER_PRIDR0	A55_CLUSTER Power Reset Identification 0 Register	0x00000001
0x31110FB8	A55_CLUSTER_AUTHSTATUS	A55_CLUSTER Authentication Status Register	0x00000008
0x31110FBC	A55_CLUSTER_DEVARCH	A55_CLUSTER Device Architecture Register	0x47700AF7
0x31110FC8	A55_CLUSTER_DEVID	A55_CLUSTER Device ID Register	0x00000020
0x31110FCC	A55_CLUSTER_DEVTYPE	A55_CLUSTER Device Type Register	0x00000000
0x31110FD0	A55_CLUSTER_PIDR4	A55_CLUSTER Peripheral Identification 4 Register	0x00000004
0x31110FD4	A55_CLUSTER_PIDR5	A55_CLUSTER Peripheral Identification 5 Register	0x00000000
0x31110FD8	A55_CLUSTER_PIDR6	A55_CLUSTER Peripheral Identification 6 Register	0x00000000
0x31110FDC	A55_CLUSTER_PIDR7	A55_CLUSTER Peripheral Identification 7 Register	0x00000000
0x31110FE0	A55_CLUSTER_PIDR0	A55_CLUSTER Peripheral Identification 0 Register	0x000000E3
0x31110FE4	A55_CLUSTER_PIDR1	A55_CLUSTER Peripheral Identification 1 Register	0x000000B4
0x31110FE8	A55_CLUSTER_PIDR2	A55_CLUSTER Peripheral Identification 2 Register	0x0000007B
0x31110FEC	A55_CLUSTER_PIDR3	A55_CLUSTER Peripheral Identification 3 Register	0x00000000
0x31110FF0	A55_CLUSTER_CIDR0	A55_CLUSTER Component Identification 0 Register	0x0000000D
0x31110FF4	A55_CLUSTER_CIDR1	A55_CLUSTER Component Identification 1 Register	0x00000090
0x31110FF8	A55_CLUSTER_CIDR2	A55_CLUSTER Component Identification 2 Register	0x00000005
0x31110FFC	A55_CLUSTER_CIDR3	A55_CLUSTER Component Identification 3 Register	0x000000B1
0x31114024	A55_CLUSTER_EDECR	A55_CLUSTER External Debug Execution Control Register	0x00000000
0x31114D00	A55_CLUSTER_MIDR_EL1	A55_CLUSTER Main ID Register	0x412FD050
0x31114D20	A55_CLUSTER_EDPFR_31_0	A55_CLUSTER External Debug Processor Feature Register	0x00112222
0x31114D24	A55_CLUSTER_EDPFR_63_32	A55_CLUSTER External Debug Processor Feature Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31114D28	A55_CLUSTER_EDDFR_31_0	A55_CLUSTER External Debug Feature Register	0x10305408
0x31114D2C	A55_CLUSTER_EDDFR_63_32	A55_CLUSTER External Debug Feature Register	0x00000000
0x31114FA8	A55_CLUSTER_EDDEVAFF0	A55_CLUSTER External Debug Device Affinity 0 Register	0x81000000
0x31114FAC	A55_CLUSTER_EDDEVAFF1	A55_CLUSTER External Debug Device Affinity 1 Register	0x00000000
0x31114FB0	A55_CLUSTER_EDLAR	A55_CLUSTER External Debug Lock Access Register	0x00000000
0x31114FB4	A55_CLUSTER_EDLSR	A55_CLUSTER External Debug Lock Status Register	0x00000000
0x31114FB8	A55_CLUSTER_DBGAUTHSTATUS_EL1	A55_CLUSTER Debug Authentication Status Register	0x000000AA
0x31114FBC	A55_CLUSTER_EDDEVARCH	A55_CLUSTER External Debug Device Architecture Register	0x47708A15
0x31114FC0	A55_CLUSTER_EDDEVID2	A55_CLUSTER External Debug Device ID 2 Register	0x00000000
0x31114FC4	A55_CLUSTER_EDDEVID1	A55_CLUSTER External Debug Device ID 1 Register	0x00000000
0x31114FC8	A55_CLUSTER_EDDEVID	A55_CLUSTER External Debug Device ID 0 Register	0x00000000
0x31114FCC	A55_CLUSTER_EDDEVTYPE	A55_CLUSTER External Debug Device Type Register	0x00000015
0x31114FD0	A55_CLUSTER_EDPIDR4	A55_CLUSTER External Debug Peripheral Identification 4 Register	0x00000004
0x31114FD4	A55_CLUSTER_EDPIDR5	A55_CLUSTER External Debug Peripheral Identification 5 Register	0x00000000
0x31114FD8	A55_CLUSTER_EDPIDR6	A55_CLUSTER External Debug Peripheral Identification 6 Register	0x00000000
0x31114FDC	A55_CLUSTER_EDPIDR7	A55_CLUSTER External Debug Peripheral Identification 7 Register	0x00000000
0x31114FE0	A55_CLUSTER_EDPIDR0	A55_CLUSTER External Debug Peripheral Identification 0 Register	0x00000005
0x31114FE4	A55_CLUSTER_EDPIDR1	A55_CLUSTER External Debug Peripheral Identification 1 Register	0x000000BD
0x31114FE8	A55_CLUSTER_EDPIDR2	A55_CLUSTER External Debug Peripheral Identification 2 Register	0x0000003B

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31114FEC	A55_CLUSTER_EDPIDR3	A55_CLUSTER External Debug Peripheral Identification 3 Register	0x00000000
0x31114FF0	A55_CLUSTER_EDCIDR0	A55_CLUSTER External Debug Component Identification 0 Register	0x0000000D
0x31114FF4	A55_CLUSTER_EDCIDR1	A55_CLUSTER External Debug Component Identification 1 Register	0x00000090
0x31114FF8	A55_CLUSTER_EDCIDR2	A55_CLUSTER External Debug Component Identification 2 Register	0x00000005
0x31114FFC	A55_CLUSTER_EDCIDR3	A55_CLUSTER External Debug Component Identification 3 Register	0x000000B1
0x31115024	A55_CLUSTER_EDECR	A55_CLUSTER External Debug Execution Control Register	0x00000000
0x31115D00	A55_CLUSTER_MIDR_EL1	A55_CLUSTER Main ID Register	0x412FD050
0x31115D20	A55_CLUSTER_EDPFR_31_0	A55_CLUSTER External Debug Processor Feature Register	0x00112222
0x31115D24	A55_CLUSTER_EDPFR_63_32	A55_CLUSTER External Debug Processor Feature Register	0x00000000
0x31115D28	A55_CLUSTER_EDDFR_31_0	A55_CLUSTER External Debug Feature Register	0x10305408
0x31115D2C	A55_CLUSTER_EDDFR_63_32	A55_CLUSTER External Debug Feature Register	0x00000000
0x31115FA8	A55_CLUSTER_EDDEVAFF0	A55_CLUSTER External Debug Device Affinity 0 Register	0x81000100
0x31115FAC	A55_CLUSTER_EDDEVAFF1	A55_CLUSTER External Debug Device Affinity 1 Register	0x00000000
0x31115FB0	A55_CLUSTER_EDLAR	A55_CLUSTER External Debug Lock Access Register	0x00000000
0x31115FB4	A55_CLUSTER_EDLSR	A55_CLUSTER External Debug Lock Status Register	0x00000000
0x31115FB8	A55_CLUSTER_DBGAUTHSTATUS_EL1	A55_CLUSTER Debug Authentication Status Register	0x000000AA
0x31115FBC	A55_CLUSTER_EDDEVARCH	A55_CLUSTER External Debug Device Architecture Register	0x47708A15
0x31115FC0	A55_CLUSTER_EDDEVID2	A55_CLUSTER External Debug Device ID 2 Register	0x00000000
0x31115FC4	A55_CLUSTER_EDDEVID1	A55_CLUSTER External Debug Device ID 1 Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31115FC8	A55_CLUSTER_EDDEVID	A55_CLUSTER External Debug Device ID 0 Register	0x00000000
0x31115FCC	A55_CLUSTER_EDDEVTYPE	A55_CLUSTER External Debug Device Type Register	0x00000015
0x31115FD0	A55_CLUSTER_EDPIDR4	A55_CLUSTER External Debug Peripheral Identification 4 Register	0x00000004
0x31115FD4	A55_CLUSTER_EDPIDR5	A55_CLUSTER External Debug Peripheral Identification 5 Register	0x00000000
0x31115FD8	A55_CLUSTER_EDPIDR6	A55_CLUSTER External Debug Peripheral Identification 6 Register	0x00000000
0x31115FDC	A55_CLUSTER_EDPIDR7	A55_CLUSTER External Debug Peripheral Identification 7 Register	0x00000000
0x31115FE0	A55_CLUSTER_EDPIDR0	A55_CLUSTER External Debug Peripheral Identification 0 Register	0x00000005
0x31115FE4	A55_CLUSTER_EDPIDR1	A55_CLUSTER External Debug Peripheral Identification 1 Register	0x000000BD
0x31115FE8	A55_CLUSTER_EDPIDR2	A55_CLUSTER External Debug Peripheral Identification 2 Register	0x0000003B
0x31115FEC	A55_CLUSTER_EDPIDR3	A55_CLUSTER External Debug Peripheral Identification 3 Register	0x00000000
0x31115FF0	A55_CLUSTER_EDCIDR0	A55_CLUSTER External Debug Component Identification 0 Register	0x0000000D
0x31115FF4	A55_CLUSTER_EDCIDR1	A55_CLUSTER External Debug Component Identification 1 Register	0x00000090
0x31115FF8	A55_CLUSTER_EDCIDR2	A55_CLUSTER External Debug Component Identification 2 Register	0x00000005
0x31115FFC	A55_CLUSTER_EDCIDR3	A55_CLUSTER External Debug Component Identification 3 Register	0x000000B1
0x3111CFA8	A55_CLUSTER_PMDEVAFF0	A55_CLUSTER Performance Monitors Device Affinity 0 Register	0x81000000
0x3111CFAC	A55_CLUSTER_PMDEVAFF1	A55_CLUSTER Performance Monitors Device Affinity 1 Register	0x00000000
0x3111CFB0	A55_CLUSTER_PMLAR	A55_CLUSTER Performance Monitors Lock Access Register	0x00000000
0x3111CFB4	A55_CLUSTER_PMLSR	A55_CLUSTER Performance Monitors Lock Status Register	0x00000000
0x3111CFB8	A55_CLUSTER_PMAUTHSTATUS	A55_CLUSTER Performance Monitors Authentication Status Register	0x00000088

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111CFBC	A55_CLUSTER_PMDEVARCH	A55_CLUSTER Performance Monitors Device Architecture Register	0x47702A16
0x3111CFCC	A55_CLUSTER_PMDEVTYPE	A55_CLUSTER Performance Monitors Device Type Register	0x00000016
0x3111CFD0	A55_CLUSTER_PMPIDR4	A55_CLUSTER Performance Monitors Peripheral Identification 4 Register	0x00000004
0x3111CFD4	A55_CLUSTER_PMPIDR5	A55_CLUSTER Performance Monitors Peripheral Identification 5 Register	0x00000000
0x3111CFD8	A55_CLUSTER_PMPIDR6	A55_CLUSTER Performance Monitors Peripheral Identification 6 Register	0x00000000
0x3111CFDC	A55_CLUSTER_PMPIDR7	A55_CLUSTER Performance Monitors Peripheral Identification 7 Register	0x00000000
0x3111CFE0	A55_CLUSTER_PMPIDR0	A55_CLUSTER Performance Monitors Peripheral Identification 0 Register	0x00000005
0x3111CFE4	A55_CLUSTER_PMPIDR1	A55_CLUSTER Performance Monitors Peripheral Identification 1 Register	0x000000BD
0x3111CFE8	A55_CLUSTER_PMPIDR2	A55_CLUSTER Performance Monitors Peripheral Identification 2 Register	0x0000003B
0x3111CFEC	A55_CLUSTER_PMPIDR3	A55_CLUSTER Performance Monitors Peripheral Identification 3 Register	0x00000000
0x3111CFF0	A55_CLUSTER_PMCIDR0	A55_CLUSTER Performance Monitors Component Identification 0 Register	0x0000000D
0x3111CFF4	A55_CLUSTER_PMCIDR1	A55_CLUSTER Performance Monitors Component Identification 1 Register	0x00000090
0x3111CFF8	A55_CLUSTER_PMCIDR2	A55_CLUSTER Performance Monitors Component Identification 2 Register	0x00000005
0x3111CFFC	A55_CLUSTER_PMCIDR3	A55_CLUSTER Performance Monitors Component Identification 3 Register	0x000000B1
0x3111DFA8	A55_CLUSTER_PMDEVAFF0	A55_CLUSTER Performance Monitors Device Affinity 0 Register	0x81000100
0x3111DFAC	A55_CLUSTER_PMDEVAFF1	A55_CLUSTER Performance Monitors Device Affinity 1 Register	0x00000000
0x3111DFB0	A55_CLUSTER_PMLAR	A55_CLUSTER Performance Monitors Lock Access Register	0x00000000
0x3111DFB4	A55_CLUSTER_PMLSR	A55_CLUSTER Performance Monitors Lock Status Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111DFB8	A55_CLUSTER_PMAUTHSTATUS	A55_CLUSTER Performance Monitors Authentication Status Register	0x00000088
0x3111DFBC	A55_CLUSTER_PMDEVARCH	A55_CLUSTER Performance Monitors Device Architecture Register	0x47702A16
0x3111DFCC	A55_CLUSTER_PMDEVTYPE	A55_CLUSTER Performance Monitors Device Type Register	0x00000016
0x3111DFD0	A55_CLUSTER_PMPIDR4	A55_CLUSTER Performance Monitors Peripheral Identification 4 Register	0x00000004
0x3111DFD4	A55_CLUSTER_PMPIDR5	A55_CLUSTER Performance Monitors Peripheral Identification 5 Register	0x00000000
0x3111DFD8	A55_CLUSTER_PMPIDR6	A55_CLUSTER Performance Monitors Peripheral Identification 6 Register	0x00000000
0x3111DFDC	A55_CLUSTER_PMPIDR7	A55_CLUSTER Performance Monitors Peripheral Identification 7 Register	0x00000000
0x3111DFE0	A55_CLUSTER_PMPIDR0	A55_CLUSTER Performance Monitors Peripheral Identification 0 Register	0x00000005
0x3111DFE4	A55_CLUSTER_PMPIDR1	A55_CLUSTER Performance Monitors Peripheral Identification 1 Register	0x000000BD
0x3111DFE8	A55_CLUSTER_PMPIDR2	A55_CLUSTER Performance Monitors Peripheral Identification 2 Register	0x0000003B
0x3111DFEC	A55_CLUSTER_PMPIDR3	A55_CLUSTER Performance Monitors Peripheral Identification 3 Register	0x00000000
0x3111DFF0	A55_CLUSTER_PMCIDR0	A55_CLUSTER Performance Monitors Component Identification 0 Register	0x0000000D
0x3111DFF4	A55_CLUSTER_PMCIDR1	A55_CLUSTER Performance Monitors Component Identification 1 Register	0x00000090
0x3111DFF8	A55_CLUSTER_PMCIDR2	A55_CLUSTER Performance Monitors Component Identification 2 Register	0x00000005
0x3111DFFC	A55_CLUSTER_PMCIDR3	A55_CLUSTER Performance Monitors Component Identification 3 Register	0x000000B1
0x31124000	A55_CLUSTER_CTICONTROL	A55_CLUSTER CTI Control Register	0x00000000
0x31124010	A55_CLUSTER_CTIINTACK	A55_CLUSTER CTI Output Trigger Acknowledge Register	0x00000000
0x31124014	A55_CLUSTER_CTIAPPSET	A55_CLUSTER CTI Application Trigger Set Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31124018	A55_CLUSTER_CTIAPPCLEAR	A55_CLUSTER CTI Application Trigger Clear Register	0x00000000
0x3112401C	A55_CLUSTER_CTIAPPPULSE	A55_CLUSTER CTI Application Pulse Register	0x00000000
0x31124020	A55_CLUSTER_CTIINEN0	A55_CLUSTER CTI Input Trigger to Output Channel Enable 0 Register	0x00000000
0x31124024	A55_CLUSTER_CTIINEN1	A55_CLUSTER CTI Input Trigger to Output Channel Enable 1 Register	0x00000000
0x31124028	A55_CLUSTER_CTIINEN2	A55_CLUSTER CTI Input Trigger to Output Channel Enable 2 Register	0x00000000
0x3112402C	A55_CLUSTER_CTIINEN3	A55_CLUSTER CTI Input Trigger to Output Channel Enable 3 Register	0x00000000
0x31124030	A55_CLUSTER_CTIINEN4	A55_CLUSTER CTI Input Trigger to Output Channel Enable 4 Register	0x00000000
0x31124034	A55_CLUSTER_CTIINEN5	A55_CLUSTER CTI Input Trigger to Output Channel Enable 5 Register	0x00000000
0x31124038	A55_CLUSTER_CTIINEN6	A55_CLUSTER CTI Input Trigger to Output Channel Enable 6 Register	0x00000000
0x3112403C	A55_CLUSTER_CTIINEN7	A55_CLUSTER CTI Input Trigger to Output Channel Enable 7 Register	0x00000000
0x311240A0	A55_CLUSTER_CTIOUTEN0	A55_CLUSTER CTI Input Channel to Output Trigger Enable 0 Register	0x00000000
0x311240A4	A55_CLUSTER_CTIOUTEN1	A55_CLUSTER CTI Input Channel to Output Trigger Enable 1 Register	0x00000000
0x311240A8	A55_CLUSTER_CTIOUTEN2	A55_CLUSTER CTI Input Channel to Output Trigger Enable 2 Register	0x00000000
0x311240AC	A55_CLUSTER_CTIOUTEN3	A55_CLUSTER CTI Input Channel to Output Trigger Enable 3 Register	0x00000000
0x311240B0	A55_CLUSTER_CTIOUTEN4	A55_CLUSTER CTI Input Channel to Output Trigger Enable 4 Register	0x00000000
0x311240B4	A55_CLUSTER_CTIOUTEN5	A55_CLUSTER CTI Input Channel to Output Trigger Enable 5 Register	0x00000000
0x311240B8	A55_CLUSTER_CTIOUTEN6	A55_CLUSTER CTI Input Channel to Output Trigger Enable 6 Register	0x00000000
0x311240BC	A55_CLUSTER_CTIOUTEN7	A55_CLUSTER CTI Input Channel to Output Trigger Enable 7 Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31124130	A55_CLUSTER_CTITRIGINSTATUS	A55_CLUSTER CTI Trigger In Status Register	0x00000000
0x31124134	A55_CLUSTER_CTITRIGOUTSTATUS	A55_CLUSTER CTI Trigger Out Status Register	0x00000000
0x31124138	A55_CLUSTER_CTICHINSTATUS	A55_CLUSTER CTI Channel In Status Register	0x00000000
0x3112413C	A55_CLUSTER_CTICHOUTSTATUS	A55_CLUSTER CTI Channel Out Status Register	0x00000000
0x31124140	A55_CLUSTER_CTIGATE	A55_CLUSTER CTI Channel Gate Enable Register	0x0000000F
0x31124144	A55_CLUSTER_ASICCTL	A55_CLUSTER CTI External Multiplexer Control Register	0x00000000
0x31124F00	A55_CLUSTER_CTIITCTRL	A55_CLUSTER CTI Integration mode Control Register	0x00000000
0x31124FA0	A55_CLUSTER_CTICLAIMSET	A55_CLUSTER CTI Claim Tag Set Register	0x0000000F
0x31124FA4	A55_CLUSTER_CTICLAIMCLR	A55_CLUSTER CTI Claim Tag Clear Register	0x00000000
0x31124FA8	A55_CLUSTER_CTIDEVAFF0	A55_CLUSTER CTI Device Affinity 0 Register	0x81000000
0x31124FAC	A55_CLUSTER_CTIDEVAFF1	A55_CLUSTER CTI Device Affinity 1 Register	0x00000000
0x31124FB0	A55_CLUSTER_CTILAR	A55_CLUSTER CTI Lock Access Register	0x00000000
0x31124FB4	A55_CLUSTER_CTILSR	A55_CLUSTER CTI Lock Status Register	0x00000000
0x31124FB8	A55_CLUSTER_CTIAUTHSTATUS	A55_CLUSTER CTI Authentication Status Register	0x0000000A
0x31124FBC	A55_CLUSTER_CTIDEVARCH	A55_CLUSTER CTI Device Architecture Register	0x47701A14
0x31124FC0	A55_CLUSTER_CTIDEVID2	A55_CLUSTER CTI Device ID 2 Register	0x00000000
0x31124FC4	A55_CLUSTER_CTIDEVID1	A55_CLUSTER CTI Device ID 1 Register	0x00000000
0x31124FC8	A55_CLUSTER_CTIDEVID	A55_CLUSTER CTI Device ID 0 Register	0x01040A00
0x31124FCC	A55_CLUSTER_CTIDEVTYPE	A55_CLUSTER CTI Device Type Register	0x00000014
0x31124FD0	A55_CLUSTER_CTIPIDR4	A55_CLUSTER CTI Peripheral Identification 4 Register	0x00000004
0x31124FD4	A55_CLUSTER_CTIPIDR5	A55_CLUSTER CTI Peripheral Identification 5 Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31124FD8	A55_CLUSTER_CTIPIDR6	A55_CLUSTER CTI Peripheral Identification 6 Register	0x00000000
0x31124FDC	A55_CLUSTER_CTIPIDR7	A55_CLUSTER CTI Peripheral Identification 7 Register	0x00000000
0x31124FE0	A55_CLUSTER_CTIPIDR0	A55_CLUSTER CTI Peripheral Identification 0 Register	0x00000005
0x31124FE4	A55_CLUSTER_CTIPIDR1	A55_CLUSTER CTI Peripheral Identification 1 Register	0x000000BD
0x31124FE8	A55_CLUSTER_CTIPIDR2	A55_CLUSTER CTI Peripheral Identification 2 Register	0x0000007B
0x31124FEC	A55_CLUSTER_CTIPIDR3	A55_CLUSTER CTI Peripheral Identification 3 Register	0x00000000
0x31124FF0	A55_CLUSTER_CTICIDR0	A55_CLUSTER CTI Component Identification 0 Register	0x0000000D
0x31124FF4	A55_CLUSTER_CTICIDR1	A55_CLUSTER CTI Component Identification 1 Register	0x00000090
0x31124FF8	A55_CLUSTER_CTICIDR2	A55_CLUSTER CTI Component Identification 2 Register	0x00000005
0x31124FFC	A55_CLUSTER_CTICIDR3	A55_CLUSTER CTI Component Identification 3 Register	0x000000B1
0x31125000	A55_CLUSTER_CTICONTROL	A55_CLUSTER CTI Control Register	0x00000000
0x31125010	A55_CLUSTER_CTIINTACK	A55_CLUSTER CTI Output Trigger Acknowledge Register	0x00000000
0x31125014	A55_CLUSTER_CTIAPPSET	A55_CLUSTER CTI Application Trigger Set Register	0x00000000
0x31125018	A55_CLUSTER_CTIAPPCLEAR	A55_CLUSTER CTI Application Trigger Clear Register	0x00000000
0x3112501C	A55_CLUSTER_CTIAPPPULSE	A55_CLUSTER CTI Application Pulse Register	0x00000000
0x31125020	A55_CLUSTER_CTIINEN0	A55_CLUSTER CTI Input Trigger to Output Channel Enable 0 Register	0x00000000
0x31125024	A55_CLUSTER_CTIINEN1	A55_CLUSTER CTI Input Trigger to Output Channel Enable 1 Register	0x00000000
0x31125028	A55_CLUSTER_CTIINEN2	A55_CLUSTER CTI Input Trigger to Output Channel Enable 2 Register	0x00000000
0x3112502C	A55_CLUSTER_CTIINEN3	A55_CLUSTER CTI Input Trigger to Output Channel Enable 3 Register	0x00000000
0x31125030	A55_CLUSTER_CTIINEN4	A55_CLUSTER CTI Input Trigger to Output Channel Enable 4 Register	0x00000000
0x31125034	A55_CLUSTER_CTIINEN5	A55_CLUSTER CTI Input Trigger to Output Channel Enable 5 Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31125038	A55_CLUSTER_CTIINEN6	A55_CLUSTER CTI Input Trigger to Output Channel Enable 6 Register	0x00000000
0x3112503C	A55_CLUSTER_CTIINEN7	A55_CLUSTER CTI Input Trigger to Output Channel Enable 7 Register	0x00000000
0x311250A0	A55_CLUSTER_CTIOUTEN0	A55_CLUSTER CTI Input Channel to Output Trigger Enable 0 Register	0x00000000
0x311250A4	A55_CLUSTER_CTIOUTEN1	A55_CLUSTER CTI Input Channel to Output Trigger Enable 1 Register	0x00000000
0x311250A8	A55_CLUSTER_CTIOUTEN2	A55_CLUSTER CTI Input Channel to Output Trigger Enable 2 Register	0x00000000
0x311250AC	A55_CLUSTER_CTIOUTEN3	A55_CLUSTER CTI Input Channel to Output Trigger Enable 3 Register	0x00000000
0x311250B0	A55_CLUSTER_CTIOUTEN4	A55_CLUSTER CTI Input Channel to Output Trigger Enable 4 Register	0x00000000
0x311250B4	A55_CLUSTER_CTIOUTEN5	A55_CLUSTER CTI Input Channel to Output Trigger Enable 5 Register	0x00000000
0x311250B8	A55_CLUSTER_CTIOUTEN6	A55_CLUSTER CTI Input Channel to Output Trigger Enable 6 Register	0x00000000
0x311250BC	A55_CLUSTER_CTIOUTEN7	A55_CLUSTER CTI Input Channel to Output Trigger Enable 7 Register	0x00000000
0x31125130	A55_CLUSTER_CTITRIGINSTATUS	A55_CLUSTER CTI Trigger In Status Register	0x00000000
0x31125134	A55_CLUSTER_CTITRIGOUTSTATUS	A55_CLUSTER CTI Trigger Out Status Register	0x00000000
0x31125138	A55_CLUSTER_CTICHINSTATUS	A55_CLUSTER CTI Channel In Status Register	0x00000000
0x3112513C	A55_CLUSTER_CTICHOUTSTATUS	A55_CLUSTER CTI Channel Out Status Register	0x00000000
0x31125140	A55_CLUSTER_CTIGATE	A55_CLUSTER CTI Channel Gate Enable Register	0x0000000F
0x31125144	A55_CLUSTER_ASICCTL	A55_CLUSTER CTI External Multiplexer Control Register	0x00000000
0x31125F00	A55_CLUSTER_CTIITCTRL	A55_CLUSTER CTI Integration mode Control Register	0x00000000
0x31125FA0	A55_CLUSTER_CTICLAIMSET	A55_CLUSTER CTI Claim Tag Set Register	0x0000000F
0x31125FA4	A55_CLUSTER_CTICLAIMCLR	A55_CLUSTER CTI Claim Tag Clear Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31125FA8	A55_CLUSTER_CTIDEVAFF0	A55_CLUSTER CTI Device Affinity 0 Register	0x81000100
0x31125FAC	A55_CLUSTER_CTIDEVAFF1	A55_CLUSTER CTI Device Affinity 1 Register	0x00000000
0x31125FB0	A55_CLUSTER_CTILAR	A55_CLUSTER CTI Lock Access Register	0x00000000
0x31125FB4	A55_CLUSTER_CTILSR	A55_CLUSTER CTI Lock Status Register	0x00000000
0x31125FB8	A55_CLUSTER_CTIAUTHSTATUS	A55_CLUSTER CTI Authentication Status Register	0x0000000A
0x31125FBC	A55_CLUSTER_CTIDEVARCH	A55_CLUSTER CTI Device Architecture Register	0x47701A14
0x31125FC0	A55_CLUSTER_CTIDEVID2	A55_CLUSTER CTI Device ID 2 Register	0x00000000
0x31125FC4	A55_CLUSTER_CTIDEVID1	A55_CLUSTER CTI Device ID 1 Register	0x00000000
0x31125FC8	A55_CLUSTER_CTIDEVID	A55_CLUSTER CTI Device ID 0 Register	0x01040A00
0x31125FCC	A55_CLUSTER_CTIDEVTYPE	A55_CLUSTER CTI Device Type Register	0x00000014
0x31125FD0	A55_CLUSTER_CTIPIDR4	A55_CLUSTER CTI Peripheral Identification 4 Register	0x00000004
0x31125FD4	A55_CLUSTER_CTIPIDR5	A55_CLUSTER CTI Peripheral Identification 5 Register	0x00000000
0x31125FD8	A55_CLUSTER_CTIPIDR6	A55_CLUSTER CTI Peripheral Identification 6 Register	0x00000000
0x31125FDC	A55_CLUSTER_CTIPIDR7	A55_CLUSTER CTI Peripheral Identification 7 Register	0x00000000
0x31125FE0	A55_CLUSTER_CTIPIDR0	A55_CLUSTER CTI Peripheral Identification 0 Register	0x00000005
0x31125FE4	A55_CLUSTER_CTIPIDR1	A55_CLUSTER CTI Peripheral Identification 1 Register	0x000000BD
0x31125FE8	A55_CLUSTER_CTIPIDR2	A55_CLUSTER CTI Peripheral Identification 2 Register	0x0000007B
0x31125FEC	A55_CLUSTER_CTIPIDR3	A55_CLUSTER CTI Peripheral Identification 3 Register	0x00000000
0x31125FF0	A55_CLUSTER_CTICIDR0	A55_CLUSTER CTI Component Identification 0 Register	0x0000000D
0x31125FF4	A55_CLUSTER_CTICIDR1	A55_CLUSTER CTI Component Identification 1 Register	0x00000090
0x31125FF8	A55_CLUSTER_CTICIDR2	A55_CLUSTER CTI Component Identification 2 Register	0x00000005
0x31125FFC	A55_CLUSTER_CTICIDR3	A55_CLUSTER CTI Component Identification 3 Register	0x000000B1
0x3112C310	A55_CLUSTER_TRCPDCR	A55_CLUSTER Trace PowerDown Control Register	0x00000000

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112CFA8	A55_CLUSTER_TRCDEVAFF0	A55_CLUSTER Trace Device Affinity 0 Register	0x81000000
0x3112CFAC	A55_CLUSTER_TRCDEVAFF1	A55_CLUSTER Trace Device Affinity 1 Register	0x00000000
0x3112CFB0	A55_CLUSTER_TRCLAR	A55_CLUSTER Trace Software Lock Access Register	0x00000000
0x3112CFB4	A55_CLUSTER_TRCLSR	A55_CLUSTER Trace Software Lock Status Register	0x00000000
0x3112CFB8	A55_CLUSTER_TRCAUTHSTATUS	A55_CLUSTER Trace Authentication Status Register	0x00000088
0x3112CFBC	A55_CLUSTER_TRCDEVARCH	A55_CLUSTER Trace Device Architecture Register	0x47724A13
0x3112CFC8	A55_CLUSTER_TRCDEVID	A55_CLUSTER Trace Device ID Register	0x00000000
0x3112CFCC	A55_CLUSTER_TRCDEVTYPE	A55_CLUSTER Trace Device Type Register	0x00000013
0x3112CFD0	A55_CLUSTER_TRCPIDR4	A55_CLUSTER Trace Peripheral ID 4 Register	0x00000004
0x3112CFE0	A55_CLUSTER_TRCPIDR0	A55_CLUSTER Trace Peripheral ID 0 Register	0x00000005
0x3112CFE4	A55_CLUSTER_TRCPIDR1	A55_CLUSTER Trace Peripheral ID 1 Register	0x000000BD
0x3112CFE8	A55_CLUSTER_TRCPIDR2	A55_CLUSTER Trace Peripheral ID 2 Register	0x0000003B
0x3112CFEC	A55_CLUSTER_TRCPIDR3	A55_CLUSTER Trace Peripheral ID 3 Register	0x00000000
0x3112CFF0	A55_CLUSTER_TRCCIDR0	A55_CLUSTER Trace Component ID 0 Register	0x0000000D
0x3112CFF4	A55_CLUSTER_TRCCIDR1	A55_CLUSTER Trace Component ID 1 Register	0x00000090
0x3112CFF8	A55_CLUSTER_TRCCIDR2	A55_CLUSTER Trace Component ID 2 Register	0x00000005
0x3112CFFC	A55_CLUSTER_TRCCIDR3	A55_CLUSTER Trace Component ID 3 Register	0x000000B1
0x3112D310	A55_CLUSTER_TRCPDCR	A55_CLUSTER Trace PowerDown Control Register	0x00000000
0x3112DFA8	A55_CLUSTER_TRCDEVAFF0	A55_CLUSTER Trace Device Affinity 0 Register	0x81000100
0x3112DFAC	A55_CLUSTER_TRCDEVAFF1	A55_CLUSTER Trace Device Affinity 1 Register	0x00000000
0x3112DFB0	A55_CLUSTER_TRCLAR	A55_CLUSTER Trace Software Lock Access Register	0x00000000
0x3112DFB4	A55_CLUSTER_TRCLSR	A55_CLUSTER Trace Software Lock Status Register	0x00000000
0x3112DFB8	A55_CLUSTER_TRCAUTHSTATUS	A55_CLUSTER Trace Authentication Status Register	0x00000088
0x3112DFBC	A55_CLUSTER_TRCDEVARCH	A55_CLUSTER Trace Device Architecture Register	0x47724A13

Table A-1: ADSP-2184x A55_CLUSTER MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112DFC8	A55_CLUSTER_TRCDEVID	A55_CLUSTER Trace Device ID Register	0x00000000
0x3112DFCC	A55_CLUSTER_TRCDEVTYPE	A55_CLUSTER Trace Device Type Register	0x00000013
0x3112DFD0	A55_CLUSTER_TRCPIDR4	A55_CLUSTER Trace Peripheral ID 4 Register	0x00000004
0x3112DFE0	A55_CLUSTER_TRCPIDR0	A55_CLUSTER Trace Peripheral ID 0 Register	0x00000005
0x3112DFE4	A55_CLUSTER_TRCPIDR1	A55_CLUSTER Trace Peripheral ID 1 Register	0x000000BD
0x3112DFE8	A55_CLUSTER_TRCPIDR2	A55_CLUSTER Trace Peripheral ID 2 Register	0x0000003B
0x3112DFEC	A55_CLUSTER_TRCPIDR3	A55_CLUSTER Trace Peripheral ID 3 Register	0x00000000
0x3112DFF0	A55_CLUSTER_TRCCIDR0	A55_CLUSTER Trace Component ID 0 Register	0x0000000D
0x3112DFF4	A55_CLUSTER_TRCCIDR1	A55_CLUSTER Trace Component ID 1 Register	0x00000090
0x3112DFF8	A55_CLUSTER_TRCCIDR2	A55_CLUSTER Trace Component ID 2 Register	0x00000005
0x3112DFFC	A55_CLUSTER_TRCCIDR3	A55_CLUSTER Trace Component ID 3 Register	0x000000B1

Table A-2: ADSP-2184x ASRC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9240	ASRC0_CTL01	ASRC0 Control Register for ASRC 0 and 1	0x00000000
0x310C9244	ASRC0_CTL23	ASRC0 Control Register for ASRC 2 and 3	0x00000000
0x310C9248	ASRC0_MUTE	ASRC0 Mute Register	0x00000000
0x310C9250	ASRC0_CTL45	ASRC0 Control Register For ASRC 4 And 5	0x00000000
0x310C9254	ASRC0_CTL67	ASRC0 Control Register for ASRC 6 and 7	0x00000000
0x310C9260	ASRC0_RAT01	ASRC0 Ratio Register for ASRC 0 and 1	0x80008000
0x310C9264	ASRC0_RAT23	ASRC0 Ratio Register for ASRC 2 and 3	0x80008000
0x310C9268	ASRC0_RAT45	ASRC0 Ratio Register For ASRC 4 and 5	0x80008000
0x310C926C	ASRC0_RAT67	ASRC0 Ratio Register for ASRC 6 and 7	0x80008000

Table A-3: ADSP-2184x ASRC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA240	ASRC1_CTL01	ASRC1 Control Register for ASRC 0 and 1	0x00000000
0x310CA244	ASRC1_CTL23	ASRC1 Control Register for ASRC 2 and 3	0x00000000
0x310CA248	ASRC1_MUTE	ASRC1 Mute Register	0x00000000

Table A-3: ADSP-2184x ASRC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA250	ASRC1_CTL45	ASRC1 Control Register For ASRC 4 And 5	0x00000000
0x310CA254	ASRC1_CTL67	ASRC1 Control Register for ASRC 6 and 7	0x00000000
0x310CA260	ASRC1_RAT01	ASRC1 Ratio Register for ASRC 0 and 1	0x80008000
0x310CA264	ASRC1_RAT23	ASRC1 Ratio Register for ASRC 2 and 3	0x80008000
0x310CA268	ASRC1_RAT45	ASRC1 Ratio Register For ASRC 4 and 5	0x80008000
0x310CA26C	ASRC1_RAT67	ASRC1 Ratio Register for ASRC 6 and 7	0x80008000

Table A-4: ADSP-2184x CANFD0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31046000	CANFD0_CFG	CANFD0 Module Configuration Register	0x5980000F
0x31046004	CANFD0_CTL1	CANFD0 Control 1 Register	0x00000000
0x31046008	CANFD0_TMR	CANFD0 Free Running Timer Register	0x00000000
0x31046010	CANFD0_RX_MB_GMSK	CANFD0 Receive Mailbox Global Mask Register	0x00000000
0x31046014	CANFD0_RX_14_MSK	CANFD0 Receive Mailbox14 Mask Register	0x00000000
0x31046018	CANFD0_RX_15_MSK	CANFD0 Receive Mailbox15 Mask Register	0x00000000
0x3104601C	CANFD0_ECR	CANFD0 Error Counter Register	0x00000000
0x31046020	CANFD0_ESR1	CANFD0 Error and Status 1 Register	0x00000000
0x31046024	CANFD0_IMSK2	CANFD0 Mailbox Interrupt Mask 2 Register	0x00000000
0x31046028	CANFD0_IMSK1	CANFD0 Mailbox Interrupt Mask 1 Register	0x00000000
0x3104602C	CANFD0_IFLG2	CANFD0 Mailbox Interrupt Flag 2 Register	0x00000000
0x31046030	CANFD0_IFLG1	CANFD0 Mailbox Interrupt Flag 1 Register	0x00000000
0x31046034	CANFD0_CTL2	CANFD0 Control 2 Register	0x00800000
0x31046038	CANFD0_ESR2	CANFD0 Error and Status 2 Register	0x00000000
0x31046044	CANFD0_CRC	CANFD0 CRC Register	0x00000000
0x31046048	CANFD0_RX_FIFO_GMSK	CANFD0 Receive FIFO Global Mask Register	0x00000000
0x3104604C	CANFD0_RX_FIFO	CANFD0 Receive FIFO Information Register	0x00000000
0x31046050	CANFD0_TIMING	CANFD0 Can Bit Timing Register	0x00000000
0x31046880	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046884	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046888	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000

Table A-4: ADSP-2184x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104688C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046890	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046894	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046898	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104689C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468A0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468A4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468A8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468AC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468B0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468B4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468B8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468BC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468C0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468C4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468C8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468CC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468D0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468D4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468D8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468DC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468E0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468E4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468E8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468EC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468F0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468F4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468F8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468FC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046900	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046904	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000

Table A-4: ADSP-2184x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31046908	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104690C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046910	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046914	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046918	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104691C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046920	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046924	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046928	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104692C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046930	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046934	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046938	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104693C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046940	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046944	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046948	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104694C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046950	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046954	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046958	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104695C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046960	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046964	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046968	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104696C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046970	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046974	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046978	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104697C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046AE0	CANFD0_MEC	CANFD0 Memory Error Control Register	0x800C0080

Table A-4: ADSP-2184x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31046AE4	CANFD0_ERR_IADDR	CANFD0 Error Injection Address Register	0x00000000
0x31046AE8	CANFD0_ERR_IDP	CANFD0 Error Injection Data Pattern Register	0x00000000
0x31046AEC	CANFD0_ERR_IPP	CANFD0 Error Injection Parity Pattern Register	0x00000000
0x31046AF0	CANFD0_ERR_RADDR	CANFD0 Error Report Address Register	0x00000000
0x31046AF4	CANFD0_ERR_RDAT	CANFD0 Error Report Data Register	0x00000000
0x31046AF8	CANFD0_ERR_RSYN	CANFD0 Error Report Syndrome Register	0x00000000
0x31046AFC	CANFD0_ERR_STAT	CANFD0 Error Status Register	0x00000000
0x31046B00	CANFD0_PN_CTL1	CANFD0 Pretended Networking Control1 Register	0x00000100
0x31046B04	CANFD0_PN_CTL2	CANFD0 Pretended Networking Control2 Register	0x00000000
0x31046B08	CANFD0_WUM	CANFD0 Pretended Networking Wakeup Match Register	0x00000000
0x31046B0C	CANFD0_FLTR_ID1	CANFD0 Pretended Networking ID Filter1 Register	0x00000000
0x31046B10	CANFD0_FLTR_DLC	CANFD0 Pretended Networking DLC Filter Register	0x00000008
0x31046B14	CANFD0_FLTR_DATA1_LO	CANFD0 Pretended Networking Payload Low Filter1 Register	0x00000000
0x31046B18	CANFD0_FLTR_DATA1_HI	CANFD0 Pretended Networking Payload Low Filter2 Register	0x00000000
0x31046B1C	CANFD0_FLTR_ID2_IDMSK	CANFD0 Pretended Networking ID Filter2 / IDMask Register	0x00000000
0x31046B20	CANFD0_FLTR_DATA2_DMSK_LO	CANFD0 Pretended Networking Payload Low Filter2 / Payload Low Mask Register	0x00000000
0x31046B24	CANFD0_FLTR_DATA2_DMSK_HI	CANFD0 Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register	0x00000000
0x31046B40	CANFD0_WMB0_CS	CANFD0 Pretended Networking Wake Up Message Buffer 0 Control Status Register	0x00000000
0x31046B44	CANFD0_WMB0_ID	CANFD0 Pretended Networking Wake Up Message Buffer ID 0 Register	0x00000000
0x31046B48	CANFD0_WMB0_D03	CANFD0 Pretended Networking Wake Up Message Buffer 0 Data 0-3 Register	0x00000000
0x31046B4C	CANFD0_WMB0_D47	CANFD0 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31046B50	CANFD0_WMB1_CS	CANFD0 Pretended Networking Wake Up Message Buffer 1 Control Status Register	0x00000000
0x31046B54	CANFD0_WMB1_ID	CANFD0 Pretended Networking Wake Up Message Buffer ID 1 Register	0x00000000

Table A-4: ADSP-2184x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31046B58	CANFD0_WMB1_D03	CANFD0 Pretended Networking Wake Up Message Buffer 1 Data 0-3 Register	0x00000000
0x31046B5C	CANFD0_WMB1_D47	CANFD0 Pretended Networking Wake Up Message Buffer 1 Data 4-7 Register	0x00000000
0x31046B60	CANFD0_WMB2_CS	CANFD0 Pretended Networking Wake Up Message Buffer 2 Control Status Register	0x00000000
0x31046B64	CANFD0_WMB2_ID	CANFD0 Pretended Networking Wake Up Message Buffer ID 2 Register	0x00000000
0x31046B68	CANFD0_WMB2_D03	CANFD0 Pretended Networking Wake Up Message Buffer 2 Data 0-3 Register	0x00000000
0x31046B6C	CANFD0_WMB2_D47	CANFD0 Pretended Networking Wake Up Message Buffer 2 Data 4-7 Register	0x00000000
0x31046B70	CANFD0_WMB3_CS	CANFD0 Pretended Networking Wake Up Message Buffer 3 Control Status Register	0x00000000
0x31046B74	CANFD0_WMB3_ID	CANFD0 Pretended Networking Wake Up Message Buffer ID 3 Register	0x00000000
0x31046B78	CANFD0_WMB3_D03	CANFD0 Pretended Networking Wake Up Message Buffer 3 Data 0-3 Register	0x00000000
0x31046B7C	CANFD0_WMB3_D47	CANFD0 Pretended Networking Wake Up Message Buffer 3 Data 4-7 Register	0x00000000
0x31046C00	CANFD0_FD_CTL	CANFD0 CANFD Control Register	0x80000100
0x31046C04	CANFD0_FD_TIMING	CANFD0 CANFD Bit Timing Register	0x00000000
0x31046C08	CANFD0_FD_CRC	CANFD0 CANFD CRC Register	0x00000000

Table A-5: ADSP-2184x CANFD1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31047000	CANFD1_CFG	CANFD1 Module Configuration Register	0x5980000F
0x31047004	CANFD1_CTL1	CANFD1 Control 1 Register	0x00000000
0x31047008	CANFD1_TMR	CANFD1 Free Running Timer Register	0x00000000
0x31047010	CANFD1_RX_MB_GMSK	CANFD1 Receive Mailbox Global Mask Register	0x00000000
0x31047014	CANFD1_RX_14_MSK	CANFD1 Receive Mailbox14 Mask Register	0x00000000
0x31047018	CANFD1_RX_15_MSK	CANFD1 Receive Mailbox15 Mask Register	0x00000000
0x3104701C	CANFD1_ECR	CANFD1 Error Counter Register	0x00000000
0x31047020	CANFD1_ESR1	CANFD1 Error and Status 1 Register	0x00000000

Table A-5: ADSP-2184x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047024	CANFD1_IMSK2	CANFD1 Mailbox Interrupt Mask 2 Register	0x00000000
0x31047028	CANFD1_IMSK1	CANFD1 Mailbox Interrupt Mask 1 Register	0x00000000
0x3104702C	CANFD1_IFLG2	CANFD1 Mailbox Interrupt Flag 2 Register	0x00000000
0x31047030	CANFD1_IFLG1	CANFD1 Mailbox Interrupt Flag 1 Register	0x00000000
0x31047034	CANFD1_CTL2	CANFD1 Control 2 Register	0x00800000
0x31047038	CANFD1_ESR2	CANFD1 Error and Status 2 Register	0x00000000
0x31047044	CANFD1_CRC	CANFD1 CRC Register	0x00000000
0x31047048	CANFD1_RX_FIFO_GMSK	CANFD1 Receive FIFO Global Mask Register	0x00000000
0x3104704C	CANFD1_RX_FIFO	CANFD1 Receive FIFO Information Register	0x00000000
0x31047050	CANFD1_TIMING	CANFD1 Can Bit Timing Register	0x00000000
0x31047880	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047884	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047888	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104788C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047890	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047894	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047898	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104789C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478A0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478A4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478A8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478AC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478B0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478B4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478B8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478BC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478C0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478C4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478C8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478CC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478D0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000

Table A-5: ADSP-2184x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310478D4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478D8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478DC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478E0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478E4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478E8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478EC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478F0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478F4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478F8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478FC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047900	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047904	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047908	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104790C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047910	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047914	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047918	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104791C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047920	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047924	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047928	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104792C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047930	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047934	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047938	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104793C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047940	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047944	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047948	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104794C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000

Table A-5: ADSP-2184x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047950	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047954	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047958	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104795C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047960	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047964	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047968	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104796C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047970	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047974	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047978	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104797C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047AE0	CANFD1_MEC	CANFD1 Memory Error Control Register	0x800C0080
0x31047AE4	CANFD1_ERR_IADDR	CANFD1 Error Injection Address Register	0x00000000
0x31047AE8	CANFD1_ERR_IDP	CANFD1 Error Injection Data Pattern Register	0x00000000
0x31047AEC	CANFD1_ERR_IPP	CANFD1 Error Injection Parity Pattern Register	0x00000000
0x31047AF0	CANFD1_ERR_RADDR	CANFD1 Error Report Address Register	0x00000000
0x31047AF4	CANFD1_ERR_RDAT	CANFD1 Error Report Data Register	0x00000000
0x31047AF8	CANFD1_ERR_RSYN	CANFD1 Error Report Syndrome Register	0x00000000
0x31047AFC	CANFD1_ERR_STAT	CANFD1 Error Status Register	0x00000000
0x31047B00	CANFD1_PN_CTL1	CANFD1 Pretended Networking Control1 Register	0x00000100
0x31047B04	CANFD1_PN_CTL2	CANFD1 Pretended Networking Control2 Register	0x00000000
0x31047B08	CANFD1_WUM	CANFD1 Pretended Networking Wakeup Match Register	0x00000000
0x31047B0C	CANFD1_FLTR_ID1	CANFD1 Pretended Networking ID Filter1 Register	0x00000000
0x31047B10	CANFD1_FLTR_DLC	CANFD1 Pretended Networking DLC Filter Register	0x00000008
0x31047B14	CANFD1_FLTR_DATA1_LO	CANFD1 Pretended Networking Payload Low Filter1 Register	0x00000000
0x31047B18	CANFD1_FLTR_DATA1_HI	CANFD1 Pretended Networking Payload Low Filter2 Register	0x00000000
0x31047B1C	CANFD1_FLTR_ID2_IDMSK	CANFD1 Pretended Networking ID Filter2 / IDMask Register	0x00000000

Table A-5: ADSP-2184x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047B20	CANFD1_FLTR_DATA2_DMSK_LO	CANFD1 Pretended Networking Payload Low Filter2 / Payload Low Mask Register	0x00000000
0x31047B24	CANFD1_FLTR_DATA2_DMSK_HI	CANFD1 Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register	0x00000000
0x31047B40	CANFD1_WMB0_CS	CANFD1 Pretended Networking Wake Up Message Buffer 0 Control Status Register	0x00000000
0x31047B44	CANFD1_WMB0_ID	CANFD1 Pretended Networking Wake Up Message Buffer ID 0 Register	0x00000000
0x31047B48	CANFD1_WMB0_D03	CANFD1 Pretended Networking Wake Up Message Buffer 0 Data 0-3 Register	0x00000000
0x31047B4C	CANFD1_WMB0_D47	CANFD1 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31047B50	CANFD1_WMB1_CS	CANFD1 Pretended Networking Wake Up Message Buffer 1 Control Status Register	0x00000000
0x31047B54	CANFD1_WMB1_ID	CANFD1 Pretended Networking Wake Up Message Buffer ID 1 Register	0x00000000
0x31047B58	CANFD1_WMB1_D03	CANFD1 Pretended Networking Wake Up Message Buffer 1 Data 0-3 Register	0x00000000
0x31047B5C	CANFD1_WMB1_D47	CANFD1 Pretended Networking Wake Up Message Buffer 1 Data 4-7 Register	0x00000000
0x31047B60	CANFD1_WMB2_CS	CANFD1 Pretended Networking Wake Up Message Buffer 2 Control Status Register	0x00000000
0x31047B64	CANFD1_WMB2_ID	CANFD1 Pretended Networking Wake Up Message Buffer ID 2 Register	0x00000000
0x31047B68	CANFD1_WMB2_D03	CANFD1 Pretended Networking Wake Up Message Buffer 2 Data 0-3 Register	0x00000000
0x31047B6C	CANFD1_WMB2_D47	CANFD1 Pretended Networking Wake Up Message Buffer 2 Data 4-7 Register	0x00000000
0x31047B70	CANFD1_WMB3_CS	CANFD1 Pretended Networking Wake Up Message Buffer 3 Control Status Register	0x00000000
0x31047B74	CANFD1_WMB3_ID	CANFD1 Pretended Networking Wake Up Message Buffer ID 3 Register	0x00000000
0x31047B78	CANFD1_WMB3_D03	CANFD1 Pretended Networking Wake Up Message Buffer 3 Data 0-3 Register	0x00000000
0x31047B7C	CANFD1_WMB3_D47	CANFD1 Pretended Networking Wake Up Message Buffer 3 Data 4-7 Register	0x00000000
0x31047C00	CANFD1_FD_CTL	CANFD1 CANFD Control Register	0x80000100

Table A-5: ADSP-2184x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047C04	CANFD1_FD_TIMING	CANFD1 CANFD Bit Timing Register	0x00000000
0x31047C08	CANFD1_FD_CRC	CANFD1 CANFD CRC Register	0x00000000

Table A-6: ADSP-2184x CDU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108F000	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F004	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F008	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F00C	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F010	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F014	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F018	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F01C	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F020	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F024	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F028	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F02C	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F030	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F034	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F038	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F040	CDU0_STAT	CDU0 CDU Status	0x00000000
0x3108F048	CDU0_REVID	CDU0 CDU Revision ID	0x00000011

Table A-7: ADSP-2184x CGU MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108D000	CGU_CTL	CGU Control Register	0x00004200
0x3108D000	CGU0_CTL	CGU0 Control Register	0x00004200
0x3108D004	CGU_PLLCTL	CGU PLL Control Register	0x00000000
0x3108D004	CGU0_PLLCTL	CGU0 PLL Control Register	0x00000000
0x3108D008	CGU_STAT	CGU Status Register	0x0000000F

Table A-7: ADSP-2184x CGU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108D008	CGU0_STAT	CGU0 Status Register	0x0000000F
0x3108D00C	CGU_DIV	CGU Clocks Divisor Register	0x0282C282
0x3108D00C	CGU0_DIV	CGU0 Clocks Divisor Register	0x0282C282
0x3108D010	CGU_CLKOUTSEL	CGU CLKOUT Select Register	0x00000000
0x3108D010	CGU0_CLKOUTSEL	CGU0 CLKOUT Select Register	0x00000000
0x3108D018	CGU_TSCTL	CGU Time Stamp Control Register	0x00000000
0x3108D018	CGU0_TSCTL	CGU0 Time Stamp Control Register	0x00000000
0x3108D01C	CGU_TSVALUE0	CGU Time Stamp Counter Initial 32 LSB Value Register	0x00000000
0x3108D01C	CGU0_TSVALUE0	CGU0 Time Stamp Counter Initial 32 LSB Value Register	0x00000000
0x3108D020	CGU_TSVALUE1	CGU Time Stamp Counter Initial MSB Value Register	0x00000000
0x3108D020	CGU0_TSVALUE1	CGU0 Time Stamp Counter Initial MSB Value Register	0x00000000
0x3108D024	CGU_TSCOUNT0	CGU Time Stamp Counter 32 LSB Register	0x00000000
0x3108D024	CGU0_TSCOUNT0	CGU0 Time Stamp Counter 32 LSB Register	0x00000000
0x3108D028	CGU_TSCOUNT1	CGU Time Stamp Counter 32 MSB Register	0x00000000
0x3108D028	CGU0_TSCOUNT1	CGU0 Time Stamp Counter 32 MSB Register	0x00000000
0x3108D02C	CGU_CCBF_DIS	CGU Core Clock Buffer Disable Register	0x00000000
0x3108D02C	CGU0_CCBF_DIS	CGU0 Core Clock Buffer Disable Register	0x00000000
0x3108D030	CGU_CCBF_STAT	CGU Core Clock Buffer Status Register	0x00000000
0x3108D030	CGU0_CCBF_STAT	CGU0 Core Clock Buffer Status Register	0x00000000
0x3108D038	CGU_SCBF_DIS	CGU System Clock Buffer Disable Register	0x00000000
0x3108D038	CGU0_SCBF_DIS	CGU0 System Clock Buffer Disable Register	0x00000000
0x3108D03C	CGU_SCBF_STAT	CGU System Clock Buffer Status Register	0x00000000
0x3108D03C	CGU0_SCBF_STAT	CGU0 System Clock Buffer Status Register	0x00000000
0x3108D040	CGU_DIVEX	CGU DIV Register Extension	0x00030004
0x3108D040	CGU0_DIVEX	CGU0 DIV Register Extension	0x00030004
0x3108D048	CGU_REVID	CGU Revision ID Register	0x00000030
0x3108D048	CGU0_REVID	CGU0 Revision ID Register	0x00000030

Table A-8: ADSP-2184x CGU1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108E000	CGU1_CTL	CGU1 Control Register	0x00004200
0x3108E004	CGU1_PLLCTL	CGU1 PLL Control Register	0x00000000
0x3108E008	CGU1_STAT	CGU1 Status Register	0x0000000F
0x3108E00C	CGU1_DIV	CGU1 Clocks Divisor Register	0x0282C282
0x3108E02C	CGU1_CCBF_DIS	CGU1 Core Clock Buffer Disable Register	0x00000000
0x3108E030	CGU1_CCBF_STAT	CGU1 Core Clock Buffer Status Register	0x00000000
0x3108E038	CGU1_SCBF_DIS	CGU1 System Clock Buffer Disable Register	0x00000000
0x3108E03C	CGU1_SCBF_STAT	CGU1 System Clock Buffer Status Register	0x00000000
0x3108E040	CGU1_DIVEX	CGU1 DIV Register Extension	0x00030004
0x3108E048	CGU1_REVID	CGU1 Revision ID Register	0x00000030

Table A-9: ADSP-2184x CNT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3100B000	CNT0_CFG	CNT0 Configuration Register	0x00000000
0x3100B004	CNT0_IMSK	CNT0 Interrupt Mask Register	0x00000000
0x3100B008	CNT0_STAT	CNT0 Status Register	0x00000000
0x3100B00C	CNT0_CMD	CNT0 Command Register	0x00000000
0x3100B010	CNT0_DEBNCE	CNT0 Debounce Register	0x00000000
0x3100B014	CNT0_CNTR	CNT0 Counter Register	0x00000000
0x3100B018	CNT0_MAX	CNT0 Maximum Count Register	0x00000000
0x3100B01C	CNT0_MIN	CNT0 Minimum Count Register	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31114020	A55_C0_EDESR	A55_C0 External Debug Event Status Register.	0x00000000
0x31114030	A55_C0_EDWAR_31_0	A55_C0 External Debug Watchpoint Address Register.	0x00000000
0x31114034	A55_C0_EDWAR_63_32	A55_C0 External Debug Watchpoint Address Register.	0x00000000
0x31114080	A55_C0_DBGDTRRX_EL0	A55_C0 Debug Data Transfer Register, Receive.	0x00000000
0x31114084	A55_C0_EDITR	A55_C0 External Debug Instruction Transfer Register.	0x00000000
0x31114088	A55_C0_EDSCR	A55_C0 External Debug Status and Control Register.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111408C	A55_C0_DBGDTRTX_EL0	A55_C0 Debug Data Transfer Register, Transmit.	0x00000000
0x31114090	A55_C0_EDRCR	A55_C0 External Debug Reserve Control Register.	0x00000000
0x31114098	A55_C0_EDECCR	A55_C0 External Debug Exception Catch Control Register.	0x00000000
0x311140A4	A55_C0_EDCIDSR	A55_C0 External Debug Context ID Sample Register.	0x00000000
0x31114300	A55_C0_OSLAR_EL1	A55_C0 OS Lock Access Register.	0x00000000
0x31114310	A55_C0_EDPRCR	A55_C0 External Debug Power/Reset Control Register.	0x00000000
0x31114314	A55_C0_EDPRSR	A55_C0 External Debug Processor Status Register.	0x00000003
0x31114408	A55_C0_DBGBCR0_EL1	A55_C0 Debug Breakpoint Control Registers.	0x00000000
0x31114418	A55_C0_DBGBCR1_EL1	A55_C0 Debug Breakpoint Control Registers.	0x00000000
0x31114428	A55_C0_DBGBCR2_EL1	A55_C0 Debug Breakpoint Control Registers.	0x00000000
0x31114438	A55_C0_DBGBCR3_EL1	A55_C0 Debug Breakpoint Control Registers.	0x00000000
0x31114448	A55_C0_DBGBCR4_EL1	A55_C0 Debug Breakpoint Control Registers.	0x00000000
0x31114458	A55_C0_DBGBCR5_EL1	A55_C0 Debug Breakpoint Control Registers.	0x00000000
0x31114800	A55_C0_DBGWVR0_EL1_31_0	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114804	A55_C0_DBGWVR0_EL1_63_32	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114808	A55_C0_DBGWCR0_EL1	A55_C0 Debug Watchpoint Control Registers.	0x00000000
0x31114810	A55_C0_DBGWVR1_EL1_31_0	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114814	A55_C0_DBGWVR1_EL1_63_32	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114818	A55_C0_DBGWCR1_EL1	A55_C0 Debug Watchpoint Control Registers.	0x00000000
0x31114820	A55_C0_DBGWVR2_EL1_31_0	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114824	A55_C0_DBGWVR2_EL1_63_32	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114828	A55_C0_DBGWCR2_EL1	A55_C0 Debug Watchpoint Control Registers.	0x00000000
0x31114830	A55_C0_DBGWVR3_EL1_31_0	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114834	A55_C0_DBGWVR3_EL1_63_32	A55_C0 Debug Watchpoint Value Registers.	0x00000000
0x31114838	A55_C0_DBGWCR3_EL1	A55_C0 Debug Watchpoint Control Registers.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31114F00	A55_C0_EDITCTRL	A55_C0 External Debug Integration mode Control register.	0x00000000
0x31114FA0	A55_C0_DBGCLAIMSET_EL1	A55_C0 Debug Claim Tag Set register.	0x000000FF
0x31114FA4	A55_C0_DBGCLAIMCLR_EL1	A55_C0 Debug Claim Tag Clear register.	0x00000000
0x3111C000	A55_C0_PMEVCNTR0_EL0	A55_C0 Performance Monitors Event Count Registers.	0x00000000
0x3111C008	A55_C0_PMEVCNTR1_EL0	A55_C0 Performance Monitors Event Count Registers.	0x00000000
0x3111C010	A55_C0_PMEVCNTR2_EL0	A55_C0 Performance Monitors Event Count Registers.	0x00000000
0x3111C018	A55_C0_PMEVCNTR3_EL0	A55_C0 Performance Monitors Event Count Registers.	0x00000000
0x3111C020	A55_C0_PMEVCNTR4_EL0	A55_C0 Performance Monitors Event Count Registers.	0x00000000
0x3111C028	A55_C0_PMEVCNTR5_EL0	A55_C0 Performance Monitors Event Count Registers.	0x00000000
0x3111C0F8	A55_C0_PMCCNTR_EL0_31_0	A55_C0 Performance Monitors Cycle Counter.	0x00000000
0x3111C0FC	A55_C0_PMCCNTR_EL0_63_32	A55_C0 Performance Monitors Cycle Counter.	0x00000000
0x3111C400	A55_C0_PMEVTYPER0_EL0	A55_C0 Performance Monitors Event Type Registers.	0x00000000
0x3111C404	A55_C0_PMEVTYPER1_EL0	A55_C0 Performance Monitors Event Type Registers.	0x00000000
0x3111C408	A55_C0_PMEVTYPER2_EL0	A55_C0 Performance Monitors Event Type Registers.	0x00000000
0x3111C40C	A55_C0_PMEVTYPER3_EL0	A55_C0 Performance Monitors Event Type Registers.	0x00000000
0x3111C410	A55_C0_PMEVTYPER4_EL0	A55_C0 Performance Monitors Event Type Registers.	0x00000000
0x3111C414	A55_C0_PMEVTYPER5_EL0	A55_C0 Performance Monitors Event Type Registers.	0x00000000
0x3111C47C	A55_C0_PMCCFILTR_EL0	A55_C0 Performance Monitors Cycle Counter Filter Register.	0x00000000
0x3111CC00	A55_C0_PMCNTENSET_EL0	A55_C0 Performance Monitors Count Enable Set register.	0x00000000
0x3111CC20	A55_C0_PMCNTENCLR_EL0	A55_C0 Performance Monitors Count Enable Clear register.	0x00000000
0x3111CC40	A55_C0_PMINTENSET_EL1	A55_C0 Performance Monitors Interrupt Enable Set register.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111CC60	A55_C0_PMINTENCLR_EL1	A55_C0 Performance Monitors Interrupt Enable Clear register.	0x00000000
0x3111CC80	A55_C0_PMOVSCLR_EL0	A55_C0 Performance Monitors Overflow Flag Status Clear register.	0x00000000
0x3111CCA0	A55_C0_PMSWINC_EL0	A55_C0 Performance Monitors Software Increment register.	0x00000000
0x3111CCC0	A55_C0_PMOVSSET_EL0	A55_C0 Performance Monitors Overflow Flag Status Set register.	0x00000000
0x3111CE00	A55_C0_PMCFGR	A55_C0 Performance Monitors Configuration Register.	0x0001FF06
0x3111CE04	A55_C0_PMCR_EL0	A55_C0 Performance Monitors Control Register.	0x00000000
0x3111CE20	A55_C0_PMCEID0	A55_C0 Performance Monitors Common Event Identification register 0.	0x6FFFBFFF
0x3111CE24	A55_C0_PMCEID1	A55_C0 Performance Monitors Common Event Identification register 1.	0x00000000
0x3111CE28	A55_C0_PMCEID2	A55_C0 Performance Monitors Common Event Identification register 2.	0x00000000
0x3111CE2C	A55_C0_PMCEID3	A55_C0 Performance Monitors Common Event Identification register 3.	0x00000000
0x3111CF00	A55_C0_PMITCTRL	A55_C0 Performance Monitors Integration mode Control register.	0x00000000
0x3112C004	A55_C0_TRCPRGCTLR	A55_C0 Trace Programming Control Register.	0x00000000
0x3112C00C	A55_C0_TRCSTATR	A55_C0 Trace Status Register.	0x00000000
0x3112C010	A55_C0_TRCCONFIGR	A55_C0 Trace Configuration Register.	0x00000001
0x3112C018	A55_C0_TRCAUXCTLR	A55_C0 Trace Auxiliary Control Register.	0x00000000
0x3112C020	A55_C0_TRCEVENTCTL0R	A55_C0 Trace Event Control 0 Register.	0x00000000
0x3112C024	A55_C0_TRCEVENTCTL1R	A55_C0 Trace Event Control 1 Register.	0x00000000
0x3112C02C	A55_C0_TRCSTALLCTLR	A55_C0 Trace Stall Control Register.	0x00000000
0x3112C030	A55_C0_TRCTSCTLR	A55_C0 Trace Global Timestamp Control Register.	0x00000000
0x3112C034	A55_C0_TRCSYNCPR	A55_C0 Trace Synchronization Period Register.	0x00000000
0x3112C038	A55_C0_TRCCCCTLR	A55_C0 Trace Cycle Count Control Register.	0x00000000
0x3112C03C	A55_C0_TRCBBCTLR	A55_C0 Trace Branch Broadcast Control Register.	0x00000000
0x3112C040	A55_C0_TRCTRACEIDR	A55_C0 Trace Trace ID Register.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C080	A55_C0_TRCVICTLR	A55_C0 Trace ViewInst Main Control Register.	0x00000000
0x3112C084	A55_C0_TRCVIIECTLR	A55_C0 Trace ViewInst Include/Exclude Control Register.	0x00000000
0x3112C088	A55_C0_TRCVISSCTLR	A55_C0 Trace ViewInst Start/Stop Control Register.	0x00000000
0x3112C100	A55_C0_TRCSEQEVR0	A55_C0 Trace Sequencer State Transition Control Registers.	0x00000000
0x3112C104	A55_C0_TRCSEQEVR1	A55_C0 Trace Sequencer State Transition Control Registers.	0x00000000
0x3112C108	A55_C0_TRCSEQEVR2	A55_C0 Trace Sequencer State Transition Control Registers.	0x00000000
0x3112C118	A55_C0_TRCSEQRSTEVR	A55_C0 Trace Sequencer Reset Control Register.	0x00000000
0x3112C11C	A55_C0_TRCSEQSTR	A55_C0 Trace Sequencer State Register.	0x00000000
0x3112C120	A55_C0_TRCEXTINSELR	A55_C0 Trace External Input Select Register.	0x00000000
0x3112C140	A55_C0_TRCCNTRLDVR0	A55_C0 Trace Counter Reload Value Registers.	0x00000000
0x3112C144	A55_C0_TRCCNTRLDVR1	A55_C0 Trace Counter Reload Value Registers.	0x00000000
0x3112C150	A55_C0_TRCCNTCTLR0	A55_C0 Trace Counter Control Registers.	0x00000000
0x3112C154	A55_C0_TRCCNTCTLR1	A55_C0 Trace Counter Control Registers.	0x00000000
0x3112C160	A55_C0_TRCCNTVR0	A55_C0 Trace Counter Value Registers.	0x00000000
0x3112C164	A55_C0_TRCCNTVR1	A55_C0 Trace Counter Value Registers.	0x00000000
0x3112C1C0	A55_C0_TRCIMSPEC0	A55_C0 Trace IMPLEMENTATION DEFINED Register 0.	0x00000000
0x3112C1E0	A55_C0_TRCIDR0	A55_C0 Trace ID Register 0.	0x50000EA1
0x3112C1E4	A55_C0_TRCIDR1	A55_C0 Trace ID Register 1.	0x4100F423
0x3112C1E8	A55_C0_TRCIDR2	A55_C0 Trace ID Register 2.	0x20001088
0x3112C1EC	A55_C0_TRCIDR3	A55_C0 Trace ID Register 3.	0x0D7B0004
0x3112C1F0	A55_C0_TRCIDR4	A55_C0 Trace ID Register 4.	0x11170004
0x3112C1F4	A55_C0_TRCIDR5	A55_C0 Trace ID Register 5.	0x28C7082F
0x3112C208	A55_C0_TRCRSCTLR2	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C20C	A55_C0_TRCRSCTLR3	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C210	A55_C0_TRCRSCTLR4	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C214	A55_C0_TRCRSCTLR5	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C218	A55_C0_TRCRSCTLR6	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C21C	A55_C0_TRCRSCTLR7	A55_C0 Trace Resource Selection Control Registers.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C220	A55_C0_TRCRSCTLR8	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C224	A55_C0_TRCRSCTLR9	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C228	A55_C0_TRCRSCTLR10	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C22C	A55_C0_TRCRSCTLR11	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C230	A55_C0_TRCRSCTLR12	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C234	A55_C0_TRCRSCTLR13	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C238	A55_C0_TRCRSCTLR14	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C23C	A55_C0_TRCRSCTLR15	A55_C0 Trace Resource Selection Control Registers.	0x00000000
0x3112C280	A55_C0_TRCSSCCR0	A55_C0 Trace Single-Shot Comparator Control Registers.	0x00000000
0x3112C2A0	A55_C0_TRCSSCSR0	A55_C0 Trace Single-Shot Comparator Status Registers.	0x00000001
0x3112C300	A55_C0_TRCOSLAR	A55_C0 Trace OS Lock Access Register.	0x00000000
0x3112C304	A55_C0_TRCOSLSR	A55_C0 Trace OS Lock Status Register.	0x0000000A
0x3112C314	A55_C0_TRCPDSR	A55_C0 Trace PowerDown Status Register.	0x00000000
0x3112C400	A55_C0_TRCACVR0_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C404	A55_C0_TRCACVR0_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C408	A55_C0_TRCACVR1_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C40C	A55_C0_TRCACVR1_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C410	A55_C0_TRCACVR2_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C414	A55_C0_TRCACVR2_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C418	A55_C0_TRCACVR3_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C41C	A55_C0_TRCACVR3_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C420	A55_C0_TRCACVR4_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C424	A55_C0_TRCACVR4_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C428	A55_C0_TRCACVR5_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C42C	A55_C0_TRCACVR5_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C430	A55_C0_TRCACVR6_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C434	A55_C0_TRCACVR6_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C438	A55_C0_TRCACVR7_31_0	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C43C	A55_C0_TRCACVR7_63_32	A55_C0 Trace Address Comparator Value Registers.	0x00000000
0x3112C480	A55_C0_TRCACATR0_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C484	A55_C0_TRCACATR0_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C488	A55_C0_TRCACATR1_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C48C	A55_C0_TRCACATR1_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C490	A55_C0_TRCACATR2_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C494	A55_C0_TRCACATR2_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C498	A55_C0_TRCACATR3_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C49C	A55_C0_TRCACATR3_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4A0	A55_C0_TRCACATR4_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4A4	A55_C0_TRCACATR4_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4A8	A55_C0_TRCACATR5_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4AC	A55_C0_TRCACATR5_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4B0	A55_C0_TRCACATR6_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4B4	A55_C0_TRCACATR6_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4B8	A55_C0_TRCACATR7_31_0	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C4BC	A55_C0_TRCACATR7_63_32	A55_C0 Trace Address Comparator Access Type Registers.	0x00000000
0x3112C600	A55_C0_TRCCIDCVR0_31_0	A55_C0 Trace Context ID Comparator Value Registers.	0x00000000
0x3112C604	A55_C0_TRCCIDCVR0_63_32	A55_C0 Trace Context ID Comparator Value Registers.	0x00000000
0x3112C640	A55_C0_TRCVMIDCVR0_31_0	A55_C0 Trace VMID Comparator Value Registers.	0x00000000
0x3112C644	A55_C0_TRCVMIDCVR0_63_32	A55_C0 Trace VMID Comparator Value Registers.	0x00000000
0x3112C680	A55_C0_TRCCIDCCTLR0	A55_C0 Trace Context ID Comparator Control Register 0.	0x00000000
0x3112CEE4	A55_C0_TRCITATBIDR	A55_C0 Integration ATB Identification Register.	0x00000000
0x3112CEEC	A55_C0_TRCITIDATAR	A55_C0 Integration Instruction ATB Data Register.	0x00000000

Table A-10: ADSP-2184x A55_C0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112CEF4	A55_C0_TRCITIATBINR	A55_C0 Integration Instruction ATB In Register.	0x00000000
0x3112CEFC	A55_C0_TRCITIATBOUTR	A55_C0 Integration Instruction ATB Out Register.	0x00000000
0x3112CF00	A55_C0_TRCITCTRL	A55_C0 Trace Integration Mode Control Register.	0x00000000
0x3112CFA0	A55_C0_TRCCLAIMSET	A55_C0 Trace Claim Tag Set Register.	0x0000000F
0x3112CFA4	A55_C0_TRCCLAIMCLR	A55_C0 Trace Claim Tag Clear Register.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31115020	A55_C2_EDESR	A55_C2 External Debug Event Status Register.	0x00000000
0x31115030	A55_C2_EDWAR_31_0	A55_C2 External Debug Watchpoint Address Register.	0x00000000
0x31115034	A55_C2_EDWAR_63_32	A55_C2 External Debug Watchpoint Address Register.	0x00000000
0x31115080	A55_C2_DBGDTRRX_EL0	A55_C2 Debug Data Transfer Register, Receive.	0x00000000
0x31115084	A55_C2_EDITR	A55_C2 External Debug Instruction Transfer Register.	0x00000000
0x31115088	A55_C2_EDSCR	A55_C2 External Debug Status and Control Register.	0x00000000
0x3111508C	A55_C2_DBGDTRTX_EL0	A55_C2 Debug Data Transfer Register, Transmit.	0x00000000
0x31115090	A55_C2_EDRCR	A55_C2 External Debug Reserve Control Register.	0x00000000
0x31115098	A55_C2_EDECCR	A55_C2 External Debug Exception Catch Control Register.	0x00000000
0x311150A4	A55_C2_EDCIDSR	A55_C2 External Debug Context ID Sample Register.	0x00000000
0x31115300	A55_C2_OSLAR_EL1	A55_C2 OS Lock Access Register.	0x00000000
0x31115310	A55_C2_EDPRCR	A55_C2 External Debug Power/Reset Control Register.	0x00000000
0x31115314	A55_C2_EDPRSR	A55_C2 External Debug Processor Status Register.	0x00000003
0x31115408	A55_C2_DBGBCR0_EL1	A55_C2 Debug Breakpoint Control Registers.	0x00000000
0x31115418	A55_C2_DBGBCR1_EL1	A55_C2 Debug Breakpoint Control Registers.	0x00000000
0x31115428	A55_C2_DBGBCR2_EL1	A55_C2 Debug Breakpoint Control Registers.	0x00000000
0x31115438	A55_C2_DBGBCR3_EL1	A55_C2 Debug Breakpoint Control Registers.	0x00000000
0x31115448	A55_C2_DBGBCR4_EL1	A55_C2 Debug Breakpoint Control Registers.	0x00000000
0x31115458	A55_C2_DBGBCR5_EL1	A55_C2 Debug Breakpoint Control Registers.	0x00000000
0x31115800	A55_C2_DBGWVR0_EL1_31_0	A55_C2 Debug Watchpoint Value Registers.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31115804	A55_C2_DBGWVR0_EL1_63_32	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115808	A55_C2_DBGWCR0_EL1	A55_C2 Debug Watchpoint Control Registers.	0x00000000
0x31115810	A55_C2_DBGWVR1_EL1_31_0	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115814	A55_C2_DBGWVR1_EL1_63_32	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115818	A55_C2_DBGWCR1_EL1	A55_C2 Debug Watchpoint Control Registers.	0x00000000
0x31115820	A55_C2_DBGWVR2_EL1_31_0	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115824	A55_C2_DBGWVR2_EL1_63_32	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115828	A55_C2_DBGWCR2_EL1	A55_C2 Debug Watchpoint Control Registers.	0x00000000
0x31115830	A55_C2_DBGWVR3_EL1_31_0	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115834	A55_C2_DBGWVR3_EL1_63_32	A55_C2 Debug Watchpoint Value Registers.	0x00000000
0x31115838	A55_C2_DBGWCR3_EL1	A55_C2 Debug Watchpoint Control Registers.	0x00000000
0x31115F00	A55_C2_EDITCTRL	A55_C2 External Debug Integration mode Control register.	0x00000000
0x31115FA0	A55_C2_DBGCLAIMSET_EL1	A55_C2 Debug Claim Tag Set register.	0x000000FF
0x31115FA4	A55_C2_DBGCLAIMCLR_EL1	A55_C2 Debug Claim Tag Clear register.	0x00000000
0x3111D000	A55_C2_PMEVCNTR0_EL0	A55_C2 Performance Monitors Event Count Registers.	0x00000000
0x3111D008	A55_C2_PMEVCNTR1_EL0	A55_C2 Performance Monitors Event Count Registers.	0x00000000
0x3111D010	A55_C2_PMEVCNTR2_EL0	A55_C2 Performance Monitors Event Count Registers.	0x00000000
0x3111D018	A55_C2_PMEVCNTR3_EL0	A55_C2 Performance Monitors Event Count Registers.	0x00000000
0x3111D020	A55_C2_PMEVCNTR4_EL0	A55_C2 Performance Monitors Event Count Registers.	0x00000000
0x3111D028	A55_C2_PMEVCNTR5_EL0	A55_C2 Performance Monitors Event Count Registers.	0x00000000
0x3111D0F8	A55_C2_PMCCNTR_EL0_31_0	A55_C2 Performance Monitors Cycle Counter.	0x00000000
0x3111D0FC	A55_C2_PMCCNTR_EL0_63_32	A55_C2 Performance Monitors Cycle Counter.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111D400	A55_C2_PMEVTYPER0_EL0	A55_C2 Performance Monitors Event Type Registers.	0x00000000
0x3111D404	A55_C2_PMEVTYPER1_EL0	A55_C2 Performance Monitors Event Type Registers.	0x00000000
0x3111D408	A55_C2_PMEVTYPER2_EL0	A55_C2 Performance Monitors Event Type Registers.	0x00000000
0x3111D40C	A55_C2_PMEVTYPER3_EL0	A55_C2 Performance Monitors Event Type Registers.	0x00000000
0x3111D410	A55_C2_PMEVTYPER4_EL0	A55_C2 Performance Monitors Event Type Registers.	0x00000000
0x3111D414	A55_C2_PMEVTYPER5_EL0	A55_C2 Performance Monitors Event Type Registers.	0x00000000
0x3111D47C	A55_C2_PMCCFILTR_EL0	A55_C2 Performance Monitors Cycle Counter Filter Register.	0x00000000
0x3111DC00	A55_C2_PMCNTENSET_EL0	A55_C2 Performance Monitors Count Enable Set register.	0x00000000
0x3111DC20	A55_C2_PMCNTENCLR_EL0	A55_C2 Performance Monitors Count Enable Clear register.	0x00000000
0x3111DC40	A55_C2_PMINTENSET_EL1	A55_C2 Performance Monitors Interrupt Enable Set register.	0x00000000
0x3111DC60	A55_C2_PMINTENCLR_EL1	A55_C2 Performance Monitors Interrupt Enable Clear register.	0x00000000
0x3111DC80	A55_C2_PMOVSCLR_EL0	A55_C2 Performance Monitors Overflow Flag Status Clear register.	0x00000000
0x3111DCA0	A55_C2_PMSWINC_EL0	A55_C2 Performance Monitors Software Increment register.	0x00000000
0x3111DCC0	A55_C2_PMOVSSET_EL0	A55_C2 Performance Monitors Overflow Flag Status Set register.	0x00000000
0x3111DE00	A55_C2_PMCFGR	A55_C2 Performance Monitors Configuration Register.	0x0001FF06
0x3111DE04	A55_C2_PMCR_EL0	A55_C2 Performance Monitors Control Register.	0x00000000
0x3111DE20	A55_C2_PMCEID0	A55_C2 Performance Monitors Common Event Identification register 0.	0x6FFFBFFF
0x3111DE24	A55_C2_PMCEID1	A55_C2 Performance Monitors Common Event Identification register 1.	0x00000000
0x3111DE28	A55_C2_PMCEID2	A55_C2 Performance Monitors Common Event Identification register 2.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111DE2C	A55_C2_PMCEID3	A55_C2 Performance Monitors Common Event Identification register 3.	0x00000000
0x3111DF00	A55_C2_PMITCTRL	A55_C2 Performance Monitors Integration mode Control register.	0x00000000
0x3112D004	A55_C2_TRCPRGCTLR	A55_C2 Trace Programming Control Register.	0x00000000
0x3112D00C	A55_C2_TRCSTATR	A55_C2 Trace Status Register.	0x00000000
0x3112D010	A55_C2_TRCCONFIGR	A55_C2 Trace Configuration Register.	0x00000001
0x3112D018	A55_C2_TRCAUXCTLR	A55_C2 Trace Auxiliary Control Register.	0x00000000
0x3112D020	A55_C2_TRCEVENTCTL0R	A55_C2 Trace Event Control 0 Register.	0x00000000
0x3112D024	A55_C2_TRCEVENTCTL1R	A55_C2 Trace Event Control 1 Register.	0x00000000
0x3112D02C	A55_C2_TRCSTALLCTLR	A55_C2 Trace Stall Control Register.	0x00000000
0x3112D030	A55_C2_TRCTSCTLR	A55_C2 Trace Global Timestamp Control Register.	0x00000000
0x3112D034	A55_C2_TRCSYNCPR	A55_C2 Trace Synchronization Period Register.	0x00000000
0x3112D038	A55_C2_TRCCCCTLR	A55_C2 Trace Cycle Count Control Register.	0x00000000
0x3112D03C	A55_C2_TRCBBCTLR	A55_C2 Trace Branch Broadcast Control Register.	0x00000000
0x3112D040	A55_C2_TRCTRACEIDR	A55_C2 Trace Trace ID Register.	0x00000000
0x3112D080	A55_C2_TRCVICTLR	A55_C2 Trace ViewInst Main Control Register.	0x00000000
0x3112D084	A55_C2_TRCVIIECTLR	A55_C2 Trace ViewInst Include/Exclude Control Register.	0x00000000
0x3112D088	A55_C2_TRCVISSCTLR	A55_C2 Trace ViewInst Start/Stop Control Register.	0x00000000
0x3112D100	A55_C2_TRCSEQEVR0	A55_C2 Trace Sequencer State Transition Control Registers.	0x00000000
0x3112D104	A55_C2_TRCSEQEVR1	A55_C2 Trace Sequencer State Transition Control Registers.	0x00000000
0x3112D108	A55_C2_TRCSEQEVR2	A55_C2 Trace Sequencer State Transition Control Registers.	0x00000000
0x3112D118	A55_C2_TRCSEQRSTEVR	A55_C2 Trace Sequencer Reset Control Register.	0x00000000
0x3112D11C	A55_C2_TRCSEQSTR	A55_C2 Trace Sequencer State Register.	0x00000000
0x3112D120	A55_C2_TRCEXTINSELR	A55_C2 Trace External Input Select Register.	0x00000000
0x3112D140	A55_C2_TRCCNTRLDVR0	A55_C2 Trace Counter Reload Value Registers.	0x00000000
0x3112D144	A55_C2_TRCCNTRLDVR1	A55_C2 Trace Counter Reload Value Registers.	0x00000000
0x3112D150	A55_C2_TRCCNTCTLR0	A55_C2 Trace Counter Control Registers.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112D154	A55_C2_TRCCNTCTLR1	A55_C2 Trace Counter Control Registers.	0x00000000
0x3112D160	A55_C2_TRCCNTVR0	A55_C2 Trace Counter Value Registers.	0x00000000
0x3112D164	A55_C2_TRCCNTVR1	A55_C2 Trace Counter Value Registers.	0x00000000
0x3112D1C0	A55_C2_TRCIMSPEC0	A55_C2 Trace IMPLEMENTATION DEFINED Register 0.	0x00000000
0x3112D1E0	A55_C2_TRCIDR0	A55_C2 Trace ID Register 0.	0x50000EA1
0x3112D1E4	A55_C2_TRCIDR1	A55_C2 Trace ID Register 1.	0x4100F423
0x3112D1E8	A55_C2_TRCIDR2	A55_C2 Trace ID Register 2.	0x20001088
0x3112D1EC	A55_C2_TRCIDR3	A55_C2 Trace ID Register 3.	0x0D7B0004
0x3112D1F0	A55_C2_TRCIDR4	A55_C2 Trace ID Register 4.	0x11170004
0x3112D1F4	A55_C2_TRCIDR5	A55_C2 Trace ID Register 5.	0x28C7082F
0x3112D208	A55_C2_TRCRSCTLR2	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D20C	A55_C2_TRCRSCTLR3	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D210	A55_C2_TRCRSCTLR4	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D214	A55_C2_TRCRSCTLR5	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D218	A55_C2_TRCRSCTLR6	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D21C	A55_C2_TRCRSCTLR7	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D220	A55_C2_TRCRSCTLR8	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D224	A55_C2_TRCRSCTLR9	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D228	A55_C2_TRCRSCTLR10	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D22C	A55_C2_TRCRSCTLR11	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D230	A55_C2_TRCRSCTLR12	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D234	A55_C2_TRCRSCTLR13	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D238	A55_C2_TRCRSCTLR14	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D23C	A55_C2_TRCRSCTLR15	A55_C2 Trace Resource Selection Control Registers.	0x00000000
0x3112D280	A55_C2_TRCSSCCR0	A55_C2 Trace Single-Shot Comparator Control Registers.	0x00000000
0x3112D2A0	A55_C2_TRCSSCSR0	A55_C2 Trace Single-Shot Comparator Status Registers.	0x00000001
0x3112D300	A55_C2_TRCOSLAR	A55_C2 Trace OS Lock Access Register.	0x00000000
0x3112D304	A55_C2_TRCOSLSR	A55_C2 Trace OS Lock Status Register.	0x0000000A
0x3112D314	A55_C2_TRCPDSR	A55_C2 Trace PowerDown Status Register.	0x00000000
0x3112D400	A55_C2_TRCACVR0_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112D404	A55_C2_TRCACVR0_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D408	A55_C2_TRCACVR1_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D40C	A55_C2_TRCACVR1_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D410	A55_C2_TRCACVR2_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D414	A55_C2_TRCACVR2_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D418	A55_C2_TRCACVR3_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D41C	A55_C2_TRCACVR3_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D420	A55_C2_TRCACVR4_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D424	A55_C2_TRCACVR4_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D428	A55_C2_TRCACVR5_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D42C	A55_C2_TRCACVR5_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D430	A55_C2_TRCACVR6_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D434	A55_C2_TRCACVR6_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D438	A55_C2_TRCACVR7_31_0	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D43C	A55_C2_TRCACVR7_63_32	A55_C2 Trace Address Comparator Value Registers.	0x00000000
0x3112D480	A55_C2_TRCACATR0_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D484	A55_C2_TRCACATR0_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D488	A55_C2_TRCACATR1_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D48C	A55_C2_TRCACATR1_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D490	A55_C2_TRCACATR2_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D494	A55_C2_TRCACATR2_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D498	A55_C2_TRCACATR3_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D49C	A55_C2_TRCACATR3_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4A0	A55_C2_TRCACATR4_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4A4	A55_C2_TRCACATR4_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4A8	A55_C2_TRCACATR5_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4AC	A55_C2_TRCACATR5_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000

Table A-11: ADSP-2184x A55_C2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112D4B0	A55_C2_TRCACATR6_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4B4	A55_C2_TRCACATR6_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4B8	A55_C2_TRCACATR7_31_0	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D4BC	A55_C2_TRCACATR7_63_32	A55_C2 Trace Address Comparator Access Type Registers.	0x00000000
0x3112D600	A55_C2_TRCCIDCVR0_31_0	A55_C2 Trace Context ID Comparator Value Registers.	0x00000000
0x3112D604	A55_C2_TRCCIDCVR0_63_32	A55_C2 Trace Context ID Comparator Value Registers.	0x00000000
0x3112D640	A55_C2_TRCVMIDCVR0_31_0	A55_C2 Trace VMID Comparator Value Registers.	0x00000000
0x3112D644	A55_C2_TRCVMIDCVR0_63_32	A55_C2 Trace VMID Comparator Value Registers.	0x00000000
0x3112D680	A55_C2_TRCCIDCCTLR0	A55_C2 Trace Context ID Comparator Control Register 0.	0x00000000
0x3112DEE4	A55_C2_TRCITATBIDR	A55_C2 Integration ATB Identification Register.	0x00000000
0x3112DEEC	A55_C2_TRCITIDATAR	A55_C2 Integration Instruction ATB Data Register.	0x00000000
0x3112DEF4	A55_C2_TRCITIATBINR	A55_C2 Integration Instruction ATB In Register.	0x00000000
0x3112DEFC	A55_C2_TRCITIATBOUTR	A55_C2 Integration Instruction ATB Out Register.	0x00000000
0x3112DF00	A55_C2_TRCITCTRL	A55_C2 Trace Integration Mode Control Register.	0x00000000
0x3112DFA0	A55_C2_TRCCLAIMSET	A55_C2 Trace Claim Tag Set Register.	0x0000000F
0x3112DFA4	A55_C2_TRCCLAIMCLR	A55_C2 Trace Claim Tag Clear Register.	0x00000000

Table A-12: ADSP-2184x CRC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A5000	CRC0_CTL	CRC0 Control Register	0x00000000
0x310A5004	CRC0_DCNT	CRC0 Data Word Count Register	0x00000000
0x310A5008	CRC0_DCNTRLD	CRC0 Data Word Count Reload Register	0x00000000
0x310A5014	CRC0_COMP	CRC0 Data Compare Register	0x00000000
0x310A5018	CRC0_FILLVAL	CRC0 Fill Value Register	0x00000000
0x310A501C	CRC0_DFIFO	CRC0 Data FIFO Register	0x00000000
0x310A5020	CRC0_INEN	CRC0 Interrupt Enable Register	0x00000000

Table A-12: ADSP-2184x CRC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A5024	CRC0_INEN_SET	CRC0 Interrupt Enable Set Register	0x00000000
0x310A5028	CRC0_INEN_CLR	CRC0 Interrupt Enable Clear Register	0x00000000
0x310A502C	CRC0_POLY	CRC0 Polynomial Register	0x00000000
0x310A5040	CRC0_STAT	CRC0 Status Register	0x00000000
0x310A5044	CRC0_DCNTCAP	CRC0 Data Count Capture Register	0x00000000
0x310A504C	CRC0_RESULT_FIN	CRC0 CRC Final Result Register	0x00000000
0x310A5050	CRC0_RESULT_CUR	CRC0 CRC Current Result Register	0x00000000

Table A-13: ADSP-2184x CRC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A6000	CRC1_CTL	CRC1 Control Register	0x00000000
0x310A6004	CRC1_DCNT	CRC1 Data Word Count Register	0x00000000
0x310A6008	CRC1_DCNTRLD	CRC1 Data Word Count Reload Register	0x00000000
0x310A6014	CRC1_COMP	CRC1 Data Compare Register	0x00000000
0x310A6018	CRC1_FILLVAL	CRC1 Fill Value Register	0x00000000
0x310A601C	CRC1_DFIFO	CRC1 Data FIFO Register	0x00000000
0x310A6020	CRC1_INEN	CRC1 Interrupt Enable Register	0x00000000
0x310A6024	CRC1_INEN_SET	CRC1 Interrupt Enable Set Register	0x00000000
0x310A6028	CRC1_INEN_CLR	CRC1 Interrupt Enable Clear Register	0x00000000
0x310A602C	CRC1_POLY	CRC1 Polynomial Register	0x00000000
0x310A6040	CRC1_STAT	CRC1 Status Register	0x00000000
0x310A6044	CRC1_DCNTCAP	CRC1 Data Count Capture Register	0x00000000
0x310A604C	CRC1_RESULT_FIN	CRC1 CRC Final Result Register	0x00000000
0x310A6050	CRC1_RESULT_CUR	CRC1 CRC Current Result Register	0x00000000

Table A-14: ADSP-2184x CRC2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AA000	CRC2_CTL	CRC2 Control Register	0x00000000
0x310AA004	CRC2_DCNT	CRC2 Data Word Count Register	0x00000000
0x310AA008	CRC2_DCNTRLD	CRC2 Data Word Count Reload Register	0x00000000

Table A-14: ADSP-2184x CRC2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AA014	CRC2_COMP	CRC2 Data Compare Register	0x00000000
0x310AA018	CRC2_FILLVAL	CRC2 Fill Value Register	0x00000000
0x310AA01C	CRC2_DFIFO	CRC2 Data FIFO Register	0x00000000
0x310AA020	CRC2_INEN	CRC2 Interrupt Enable Register	0x00000000
0x310AA024	CRC2_INEN_SET	CRC2 Interrupt Enable Set Register	0x00000000
0x310AA028	CRC2_INEN_CLR	CRC2 Interrupt Enable Clear Register	0x00000000
0x310AA02C	CRC2_POLY	CRC2 Polynomial Register	0x00000000
0x310AA040	CRC2_STAT	CRC2 Status Register	0x00000000
0x310AA044	CRC2_DCNTCAP	CRC2 Data Count Capture Register	0x00000000
0x310AA04C	CRC2_RESULT_FIN	CRC2 CRC Final Result Register	0x00000000
0x310AA050	CRC2_RESULT_CUR	CRC2 CRC Current Result Register	0x00000000

Table A-15: ADSP-2184x CRC3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AB000	CRC3_CTL	CRC3 Control Register	0x00000000
0x310AB004	CRC3_DCNT	CRC3 Data Word Count Register	0x00000000
0x310AB008	CRC3_DCNTRLD	CRC3 Data Word Count Reload Register	0x00000000
0x310AB014	CRC3_COMP	CRC3 Data Compare Register	0x00000000
0x310AB018	CRC3_FILLVAL	CRC3 Fill Value Register	0x00000000
0x310AB01C	CRC3_DFIFO	CRC3 Data FIFO Register	0x00000000
0x310AB020	CRC3_INEN	CRC3 Interrupt Enable Register	0x00000000
0x310AB024	CRC3_INEN_SET	CRC3 Interrupt Enable Set Register	0x00000000
0x310AB028	CRC3_INEN_CLR	CRC3 Interrupt Enable Clear Register	0x00000000
0x310AB02C	CRC3_POLY	CRC3 Polynomial Register	0x00000000
0x310AB040	CRC3_STAT	CRC3 Status Register	0x00000000
0x310AB044	CRC3_DCNTCAP	CRC3 Data Count Capture Register	0x00000000
0x310AB04C	CRC3_RESULT_FIN	CRC3 CRC Final Result Register	0x00000000
0x310AB050	CRC3_RESULT_CUR	CRC3 CRC Current Result Register	0x00000000

Table A-16: ADSP-2184x CTI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31101000	CTI1_CTICONTROL	CTI1 CTI Control Register	0x00000000
0x31101010	CTI1_CTIINTACK	CTI1 CTI Interrupt Acknowledge Register	0x00000000
0x31101014	CTI1_CTIAPPSET	CTI1 CTI Application Trigger Set Register	0x00000000
0x31101018	CTI1_CTIAPPCLEAR	CTI1 CTI Application Trigger Clear Register	0x00000000
0x3110101C	CTI1_CTIAPPPULSE	CTI1 CTI Application Pulse Register	0x00000000
0x31101020	CTI1_CTIINEN0	CTI1 CTI Trigger 0 to Channel Enable Register	0x00000000
0x31101024	CTI1_CTIINEN1	CTI1 CTI Trigger 1 to Channel Enable Register	0x00000000
0x31101028	CTI1_CTIINEN2	CTI1 CTI Trigger 2 to Channel Enable Register	0x00000000
0x3110102C	CTI1_CTIINEN3	CTI1 CTI Trigger 3 to Channel Enable Register	0x00000000
0x31101030	CTI1_CTIINEN4	CTI1 CTI Trigger 4 to Channel Enable Register	0x00000000
0x31101034	CTI1_CTIINEN5	CTI1 CTI Trigger 5 to Channel Enable Register	0x00000000
0x31101038	CTI1_CTIINEN6	CTI1 CTI Trigger 6 to Channel Enable Register	0x00000000
0x3110103C	CTI1_CTIINEN7	CTI1 CTI Trigger 7 to Channel Enable Register	0x00000000
0x311010A0	CTI1_CTIOUTEN0	CTI1 CTI Channel to Trigger 0 Enable Register	0x00000000
0x311010A4	CTI1_CTIOUTEN1	CTI1 CTI Channel to Trigger 1 Enable Register	0x00000000
0x311010A8	CTI1_CTIOUTEN2	CTI1 CTI Channel to Trigger 2 Enable Register	0x00000000
0x311010AC	CTI1_CTIOUTEN3	CTI1 CTI Channel to Trigger 3 Enable Register	0x00000000
0x311010B0	CTI1_CTIOUTEN4	CTI1 CTI Channel to Trigger 4 Enable Register	0x00000000
0x311010B4	CTI1_CTIOUTEN5	CTI1 CTI Channel to Trigger 5 Enable Register	0x00000000
0x311010B8	CTI1_CTIOUTEN6	CTI1 CTI Channel to Trigger 6 Enable Register	0x00000000
0x311010BC	CTI1_CTIOUTEN7	CTI1 CTI Channel to Trigger 7 Enable Register	0x00000000
0x31101130	CTI1_CTITRIGINSTATUS	CTI1 CTI Trigger In Status Register	0x00000000
0x31101134	CTI1_CTITRIGOUTSTATUS	CTI1 CTI Trigger Out Status Register	0x00000000
0x31101138	CTI1_CTICHINSTATUS	CTI1 CTI Channel In Status Register	0x00000000
0x3110113C	CTI1_CTICHOUTSTATUS	CTI1 CTI Channel Out Status Register	0x00000000
0x31101140	CTI1_CTIGATE	CTI1 Enable CTI Channel Gate Register	0x0000000F
0x31101144	CTI1_ASICCTL	CTI1 External Multiplexor Control Register	0x00000000
0x31101EDC	CTI1_ITCHINACK	CTI1 ITCHINACK	0x00000000
0x31101EE0	CTI1_ITTRIGINACK	CTI1 ITTRIGINACK	0x00000000
0x31101EE4	CTI1_ITCHOUT	CTI1 Integration Test Channel Output	0x00000000

Table A-16: ADSP-2184x CTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31101EE8	CTI1_ITTRIGOUT	CTI1 Integration Test Trigger Output	0x00000000
0x31101EEC	CTI1_ITCHOUTACK	CTI1 ITCHOUTACK	0x00000000
0x31101EF0	CTI1_ITTRIGOUTACK	CTI1 ITTRIGOUTACK	0x00000000
0x31101EF4	CTI1_ITCHIN	CTI1 Integration Test Channel Input	0x00000000
0x31101EF8	CTI1_ITTRIGIN	CTI1 Integration Test Trigger Input	0x00000000
0x31101F00	CTI1_ITCTRL	CTI1 Integration Mode Control Register	0x00000000
0x31101FA0	CTI1_CLAIMSET	CTI1 Claim Tag Set Register	0x0000000F
0x31101FA4	CTI1_CLAIMCLR	CTI1 Claim Tag Clear Register	0x00000000
0x31101FB0	CTI1_LAR	CTI1 Lock Access Register	0x00000000
0x31101FB4	CTI1_LSR	CTI1 Lock Status Register	0x00000003
0x31101FB8	CTI1_AUTHSTATUS	CTI1 Authentication Status	0x00000005
0x31101FC8	CTI1_DEVID	CTI1 Device ID	0x00040800
0x31101FCC	CTI1_DEVTYPE	CTI1 Device Type	0x00000014
0x31101FD0	CTI1_PERIPHID4	CTI1 Peripheral ID4	0x00000004
0x31101FD4	CTI1_PERIPHID5	CTI1 Peripheral ID5	0x00000000
0x31101FD8	CTI1_PERIPHID6	CTI1 Peripheral ID6	0x00000000
0x31101FDC	CTI1_PERIPHID7	CTI1 Peripheral ID7	0x00000000
0x31101FE0	CTI1_PERIPHID0	CTI1 Peripheral ID0	0x00000006
0x31101FE4	CTI1_PERIPHID1	CTI1 Peripheral ID1	0x000000B9
0x31101FE8	CTI1_PERIPHID2	CTI1 Peripheral ID2	0x0000003B
0x31101FEC	CTI1_PERIPHID3	CTI1 Peripheral ID3	0x00000000
0x31101FF0	CTI1_COMPID0	CTI1 Component ID0	0x0000000D
0x31101FF4	CTI1_COMPID1	CTI1 Component ID1	0x00000090
0x31101FF8	CTI1_COMPID2	CTI1 Component ID2	0x00000005
0x31101FFC	CTI1_COMPID3	CTI1 Component ID3	0x000000B1

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31100000	DAPROM0_ROMENTRY0	DAPROM0 ROM Entries register 0.	0x00001003
0x31100004	DAPROM0_ROMENTRY1	DAPROM0 ROM Entries register 1.	0x00002003

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100008	DAPROM0_ROMENTRY2	DAPROM0 ROM Entries register 2.	0x00003003
0x3110000C	DAPROM0_ROMENTRY3	DAPROM0 ROM Entries register 3.	0x00004003
0x31100010	DAPROM0_ROMENTRY4	DAPROM0 ROM Entries register 4.	0x00005003
0x31100014	DAPROM0_ROMENTRY5	DAPROM0 ROM Entries register 5.	0x00006002
0x31100018	DAPROM0_ROMENTRY6	DAPROM0 ROM Entries register 6.	0x00007002
0x3110001C	DAPROM0_ROMENTRY7	DAPROM0 ROM Entries register 7.	0x00008003
0x31100020	DAPROM0_ROMENTRY8	DAPROM0 ROM Entries register 8.	0x00009003
0x31100024	DAPROM0_ROMENTRY9	DAPROM0 ROM Entries register 9.	0x0000A003
0x31100028	DAPROM0_ROMENTRY10	DAPROM0 ROM Entries register 10.	0x0000B003
0x3110002C	DAPROM0_ROMENTRY11	DAPROM0 ROM Entries register 11.	0x0000C003
0x31100030	DAPROM0_ROMENTRY12	DAPROM0 ROM Entries register 12.	0x0000D003
0x31100034	DAPROM0_ROMENTRY13	DAPROM0 ROM Entries register 13.	0x0000E003
0x31100038	DAPROM0_ROMENTRY14	DAPROM0 ROM Entries register 14.	0x00010003
0x3110003C	DAPROM0_ROMENTRY15	DAPROM0 ROM Entries register 15.	0x00000000
0x31100040	DAPROM0_ROMENTRY16	DAPROM0 ROM Entries register 16.	0x00000000
0x31100044	DAPROM0_ROMENTRY17	DAPROM0 ROM Entries register 17.	0x00000000
0x31100048	DAPROM0_ROMENTRY18	DAPROM0 ROM Entries register 18.	0x00000000
0x3110004C	DAPROM0_ROMENTRY19	DAPROM0 ROM Entries register 19.	0x00000000
0x31100050	DAPROM0_ROMENTRY20	DAPROM0 ROM Entries register 20.	0x00000000
0x31100054	DAPROM0_ROMENTRY21	DAPROM0 ROM Entries register 21.	0x00000000
0x31100058	DAPROM0_ROMENTRY22	DAPROM0 ROM Entries register 22.	0x00000000
0x3110005C	DAPROM0_ROMENTRY23	DAPROM0 ROM Entries register 23.	0x00000000
0x31100060	DAPROM0_ROMENTRY24	DAPROM0 ROM Entries register 24.	0x00000000
0x31100064	DAPROM0_ROMENTRY25	DAPROM0 ROM Entries register 25.	0x00000000
0x31100068	DAPROM0_ROMENTRY26	DAPROM0 ROM Entries register 26.	0x00000000
0x3110006C	DAPROM0_ROMENTRY27	DAPROM0 ROM Entries register 27.	0x00000000
0x31100070	DAPROM0_ROMENTRY28	DAPROM0 ROM Entries register 28.	0x00000000
0x31100074	DAPROM0_ROMENTRY29	DAPROM0 ROM Entries register 29.	0x00000000
0x31100078	DAPROM0_ROMENTRY30	DAPROM0 ROM Entries register 30.	0x00000000
0x3110007C	DAPROM0_ROMENTRY31	DAPROM0 ROM Entries register 31.	0x00000000
0x31100080	DAPROM0_ROMENTRY32	DAPROM0 ROM Entries register 32.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100084	DAPROM0_ROMENTRY33	DAPROM0 ROM Entries register 33.	0x00000000
0x31100088	DAPROM0_ROMENTRY34	DAPROM0 ROM Entries register 34.	0x00000000
0x3110008C	DAPROM0_ROMENTRY35	DAPROM0 ROM Entries register 35.	0x00000000
0x31100090	DAPROM0_ROMENTRY36	DAPROM0 ROM Entries register 36.	0x00000000
0x31100094	DAPROM0_ROMENTRY37	DAPROM0 ROM Entries register 37.	0x00000000
0x31100098	DAPROM0_ROMENTRY38	DAPROM0 ROM Entries register 38.	0x00000000
0x3110009C	DAPROM0_ROMENTRY39	DAPROM0 ROM Entries register 39.	0x00000000
0x311000A0	DAPROM0_ROMENTRY40	DAPROM0 ROM Entries register 40.	0x00000000
0x311000A4	DAPROM0_ROMENTRY41	DAPROM0 ROM Entries register 41.	0x00000000
0x311000A8	DAPROM0_ROMENTRY42	DAPROM0 ROM Entries register 42.	0x00000000
0x311000AC	DAPROM0_ROMENTRY43	DAPROM0 ROM Entries register 43.	0x00000000
0x311000B0	DAPROM0_ROMENTRY44	DAPROM0 ROM Entries register 44.	0x00000000
0x311000B4	DAPROM0_ROMENTRY45	DAPROM0 ROM Entries register 45.	0x00000000
0x311000B8	DAPROM0_ROMENTRY46	DAPROM0 ROM Entries register 46.	0x00000000
0x311000BC	DAPROM0_ROMENTRY47	DAPROM0 ROM Entries register 47.	0x00000000
0x311000C0	DAPROM0_ROMENTRY48	DAPROM0 ROM Entries register 48.	0x00000000
0x311000C4	DAPROM0_ROMENTRY49	DAPROM0 ROM Entries register 49.	0x00000000
0x311000C8	DAPROM0_ROMENTRY50	DAPROM0 ROM Entries register 50.	0x00000000
0x311000CC	DAPROM0_ROMENTRY51	DAPROM0 ROM Entries register 51.	0x00000000
0x311000D0	DAPROM0_ROMENTRY52	DAPROM0 ROM Entries register 52.	0x00000000
0x311000D4	DAPROM0_ROMENTRY53	DAPROM0 ROM Entries register 53.	0x00000000
0x311000D8	DAPROM0_ROMENTRY54	DAPROM0 ROM Entries register 54.	0x00000000
0x311000DC	DAPROM0_ROMENTRY55	DAPROM0 ROM Entries register 55.	0x00000000
0x311000E0	DAPROM0_ROMENTRY56	DAPROM0 ROM Entries register 56.	0x00000000
0x311000E4	DAPROM0_ROMENTRY57	DAPROM0 ROM Entries register 57.	0x00000000
0x311000E8	DAPROM0_ROMENTRY58	DAPROM0 ROM Entries register 58.	0x00000000
0x311000EC	DAPROM0_ROMENTRY59	DAPROM0 ROM Entries register 59.	0x00000000
0x311000F0	DAPROM0_ROMENTRY60	DAPROM0 ROM Entries register 60.	0x00000000
0x311000F4	DAPROM0_ROMENTRY61	DAPROM0 ROM Entries register 61.	0x00000000
0x311000F8	DAPROM0_ROMENTRY62	DAPROM0 ROM Entries register 62.	0x00000000
0x311000FC	DAPROM0_ROMENTRY63	DAPROM0 ROM Entries register 63.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100100	DAPROM0_ROMENTRY64	DAPROM0 ROM Entries register 64.	0x00000000
0x31100104	DAPROM0_ROMENTRY65	DAPROM0 ROM Entries register 65.	0x00000000
0x31100108	DAPROM0_ROMENTRY66	DAPROM0 ROM Entries register 66.	0x00000000
0x3110010C	DAPROM0_ROMENTRY67	DAPROM0 ROM Entries register 67.	0x00000000
0x31100110	DAPROM0_ROMENTRY68	DAPROM0 ROM Entries register 68.	0x00000000
0x31100114	DAPROM0_ROMENTRY69	DAPROM0 ROM Entries register 69.	0x00000000
0x31100118	DAPROM0_ROMENTRY70	DAPROM0 ROM Entries register 70.	0x00000000
0x3110011C	DAPROM0_ROMENTRY71	DAPROM0 ROM Entries register 71.	0x00000000
0x31100120	DAPROM0_ROMENTRY72	DAPROM0 ROM Entries register 72.	0x00000000
0x31100124	DAPROM0_ROMENTRY73	DAPROM0 ROM Entries register 73.	0x00000000
0x31100128	DAPROM0_ROMENTRY74	DAPROM0 ROM Entries register 74.	0x00000000
0x3110012C	DAPROM0_ROMENTRY75	DAPROM0 ROM Entries register 75.	0x00000000
0x31100130	DAPROM0_ROMENTRY76	DAPROM0 ROM Entries register 76.	0x00000000
0x31100134	DAPROM0_ROMENTRY77	DAPROM0 ROM Entries register 77.	0x00000000
0x31100138	DAPROM0_ROMENTRY78	DAPROM0 ROM Entries register 78.	0x00000000
0x3110013C	DAPROM0_ROMENTRY79	DAPROM0 ROM Entries register 79.	0x00000000
0x31100140	DAPROM0_ROMENTRY80	DAPROM0 ROM Entries register 80.	0x00000000
0x31100144	DAPROM0_ROMENTRY81	DAPROM0 ROM Entries register 81.	0x00000000
0x31100148	DAPROM0_ROMENTRY82	DAPROM0 ROM Entries register 82.	0x00000000
0x3110014C	DAPROM0_ROMENTRY83	DAPROM0 ROM Entries register 83.	0x00000000
0x31100150	DAPROM0_ROMENTRY84	DAPROM0 ROM Entries register 84.	0x00000000
0x31100154	DAPROM0_ROMENTRY85	DAPROM0 ROM Entries register 85.	0x00000000
0x31100158	DAPROM0_ROMENTRY86	DAPROM0 ROM Entries register 86.	0x00000000
0x3110015C	DAPROM0_ROMENTRY87	DAPROM0 ROM Entries register 87.	0x00000000
0x31100160	DAPROM0_ROMENTRY88	DAPROM0 ROM Entries register 88.	0x00000000
0x31100164	DAPROM0_ROMENTRY89	DAPROM0 ROM Entries register 89.	0x00000000
0x31100168	DAPROM0_ROMENTRY90	DAPROM0 ROM Entries register 90.	0x00000000
0x3110016C	DAPROM0_ROMENTRY91	DAPROM0 ROM Entries register 91.	0x00000000
0x31100170	DAPROM0_ROMENTRY92	DAPROM0 ROM Entries register 92.	0x00000000
0x31100174	DAPROM0_ROMENTRY93	DAPROM0 ROM Entries register 93.	0x00000000
0x31100178	DAPROM0_ROMENTRY94	DAPROM0 ROM Entries register 94.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110017C	DAPROM0_ROMENTRY95	DAPROM0 ROM Entries register 95.	0x00000000
0x31100180	DAPROM0_ROMENTRY96	DAPROM0 ROM Entries register 96.	0x00000000
0x31100184	DAPROM0_ROMENTRY97	DAPROM0 ROM Entries register 97.	0x00000000
0x31100188	DAPROM0_ROMENTRY98	DAPROM0 ROM Entries register 98.	0x00000000
0x3110018C	DAPROM0_ROMENTRY99	DAPROM0 ROM Entries register 99.	0x00000000
0x31100190	DAPROM0_ROMENTRY100	DAPROM0 ROM Entries register 100.	0x00000000
0x31100194	DAPROM0_ROMENTRY101	DAPROM0 ROM Entries register 101.	0x00000000
0x31100198	DAPROM0_ROMENTRY102	DAPROM0 ROM Entries register 102.	0x00000000
0x3110019C	DAPROM0_ROMENTRY103	DAPROM0 ROM Entries register 103.	0x00000000
0x311001A0	DAPROM0_ROMENTRY104	DAPROM0 ROM Entries register 104.	0x00000000
0x311001A4	DAPROM0_ROMENTRY105	DAPROM0 ROM Entries register 105.	0x00000000
0x311001A8	DAPROM0_ROMENTRY106	DAPROM0 ROM Entries register 106.	0x00000000
0x311001AC	DAPROM0_ROMENTRY107	DAPROM0 ROM Entries register 107.	0x00000000
0x311001B0	DAPROM0_ROMENTRY108	DAPROM0 ROM Entries register 108.	0x00000000
0x311001B4	DAPROM0_ROMENTRY109	DAPROM0 ROM Entries register 109.	0x00000000
0x311001B8	DAPROM0_ROMENTRY110	DAPROM0 ROM Entries register 110.	0x00000000
0x311001BC	DAPROM0_ROMENTRY111	DAPROM0 ROM Entries register 111.	0x00000000
0x311001C0	DAPROM0_ROMENTRY112	DAPROM0 ROM Entries register 112.	0x00000000
0x311001C4	DAPROM0_ROMENTRY113	DAPROM0 ROM Entries register 113.	0x00000000
0x311001C8	DAPROM0_ROMENTRY114	DAPROM0 ROM Entries register 114.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311001CC	DAPROM0_ROMENTRY115	DAPROM0 ROM Entries register 115.	0x00000000
0x311001D0	DAPROM0_ROMENTRY116	DAPROM0 ROM Entries register 116.	0x00000000
0x311001D4	DAPROM0_ROMENTRY117	DAPROM0 ROM Entries register 117.	0x00000000
0x311001D8	DAPROM0_ROMENTRY118	DAPROM0 ROM Entries register 118.	0x00000000
0x311001DC	DAPROM0_ROMENTRY119	DAPROM0 ROM Entries register 119.	0x00000000
0x311001E0	DAPROM0_ROMENTRY120	DAPROM0 ROM Entries register 120.	0x00000000
0x311001E4	DAPROM0_ROMENTRY121	DAPROM0 ROM Entries register 121.	0x00000000
0x311001E8	DAPROM0_ROMENTRY122	DAPROM0 ROM Entries register 122.	0x00000000
0x311001EC	DAPROM0_ROMENTRY123	DAPROM0 ROM Entries register 123.	0x00000000
0x311001F0	DAPROM0_ROMENTRY124	DAPROM0 ROM Entries register 124.	0x00000000
0x311001F4	DAPROM0_ROMENTRY125	DAPROM0 ROM Entries register 125.	0x00000000
0x311001F8	DAPROM0_ROMENTRY126	DAPROM0 ROM Entries register 126.	0x00000000
0x311001FC	DAPROM0_ROMENTRY127	DAPROM0 ROM Entries register 127.	0x00000000
0x31100200	DAPROM0_ROMENTRY128	DAPROM0 ROM Entries register 128.	0x00000000
0x31100204	DAPROM0_ROMENTRY129	DAPROM0 ROM Entries register 129.	0x00000000
0x31100208	DAPROM0_ROMENTRY130	DAPROM0 ROM Entries register 130.	0x00000000
0x3110020C	DAPROM0_ROMENTRY131	DAPROM0 ROM Entries register 131.	0x00000000
0x31100210	DAPROM0_ROMENTRY132	DAPROM0 ROM Entries register 132.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100214	DAPROM0_ROMENTRY133	DAPROM0 ROM Entries register 133.	0x00000000
0x31100218	DAPROM0_ROMENTRY134	DAPROM0 ROM Entries register 134.	0x00000000
0x3110021C	DAPROM0_ROMENTRY135	DAPROM0 ROM Entries register 135.	0x00000000
0x31100220	DAPROM0_ROMENTRY136	DAPROM0 ROM Entries register 136.	0x00000000
0x31100224	DAPROM0_ROMENTRY137	DAPROM0 ROM Entries register 137.	0x00000000
0x31100228	DAPROM0_ROMENTRY138	DAPROM0 ROM Entries register 138.	0x00000000
0x3110022C	DAPROM0_ROMENTRY139	DAPROM0 ROM Entries register 139.	0x00000000
0x31100230	DAPROM0_ROMENTRY140	DAPROM0 ROM Entries register 140.	0x00000000
0x31100234	DAPROM0_ROMENTRY141	DAPROM0 ROM Entries register 141.	0x00000000
0x31100238	DAPROM0_ROMENTRY142	DAPROM0 ROM Entries register 142.	0x00000000
0x3110023C	DAPROM0_ROMENTRY143	DAPROM0 ROM Entries register 143.	0x00000000
0x31100240	DAPROM0_ROMENTRY144	DAPROM0 ROM Entries register 144.	0x00000000
0x31100244	DAPROM0_ROMENTRY145	DAPROM0 ROM Entries register 145.	0x00000000
0x31100248	DAPROM0_ROMENTRY146	DAPROM0 ROM Entries register 146.	0x00000000
0x3110024C	DAPROM0_ROMENTRY147	DAPROM0 ROM Entries register 147.	0x00000000
0x31100250	DAPROM0_ROMENTRY148	DAPROM0 ROM Entries register 148.	0x00000000
0x31100254	DAPROM0_ROMENTRY149	DAPROM0 ROM Entries register 149.	0x00000000
0x31100258	DAPROM0_ROMENTRY150	DAPROM0 ROM Entries register 150.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110025C	DAPROM0_ROMENTRY151	DAPROM0 ROM Entries register 151.	0x00000000
0x31100260	DAPROM0_ROMENTRY152	DAPROM0 ROM Entries register 152.	0x00000000
0x31100264	DAPROM0_ROMENTRY153	DAPROM0 ROM Entries register 153.	0x00000000
0x31100268	DAPROM0_ROMENTRY154	DAPROM0 ROM Entries register 154.	0x00000000
0x3110026C	DAPROM0_ROMENTRY155	DAPROM0 ROM Entries register 155.	0x00000000
0x31100270	DAPROM0_ROMENTRY156	DAPROM0 ROM Entries register 156.	0x00000000
0x31100274	DAPROM0_ROMENTRY157	DAPROM0 ROM Entries register 157.	0x00000000
0x31100278	DAPROM0_ROMENTRY158	DAPROM0 ROM Entries register 158.	0x00000000
0x3110027C	DAPROM0_ROMENTRY159	DAPROM0 ROM Entries register 159.	0x00000000
0x31100280	DAPROM0_ROMENTRY160	DAPROM0 ROM Entries register 160.	0x00000000
0x31100284	DAPROM0_ROMENTRY161	DAPROM0 ROM Entries register 161.	0x00000000
0x31100288	DAPROM0_ROMENTRY162	DAPROM0 ROM Entries register 162.	0x00000000
0x3110028C	DAPROM0_ROMENTRY163	DAPROM0 ROM Entries register 163.	0x00000000
0x31100290	DAPROM0_ROMENTRY164	DAPROM0 ROM Entries register 164.	0x00000000
0x31100294	DAPROM0_ROMENTRY165	DAPROM0 ROM Entries register 165.	0x00000000
0x31100298	DAPROM0_ROMENTRY166	DAPROM0 ROM Entries register 166.	0x00000000
0x3110029C	DAPROM0_ROMENTRY167	DAPROM0 ROM Entries register 167.	0x00000000
0x311002A0	DAPROM0_ROMENTRY168	DAPROM0 ROM Entries register 168.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311002A4	DAPROM0_ROMENTRY169	DAPROM0 ROM Entries register 169.	0x00000000
0x311002A8	DAPROM0_ROMENTRY170	DAPROM0 ROM Entries register 170.	0x00000000
0x311002AC	DAPROM0_ROMENTRY171	DAPROM0 ROM Entries register 171.	0x00000000
0x311002B0	DAPROM0_ROMENTRY172	DAPROM0 ROM Entries register 172.	0x00000000
0x311002B4	DAPROM0_ROMENTRY173	DAPROM0 ROM Entries register 173.	0x00000000
0x311002B8	DAPROM0_ROMENTRY174	DAPROM0 ROM Entries register 174.	0x00000000
0x311002BC	DAPROM0_ROMENTRY175	DAPROM0 ROM Entries register 175.	0x00000000
0x311002C0	DAPROM0_ROMENTRY176	DAPROM0 ROM Entries register 176.	0x00000000
0x311002C4	DAPROM0_ROMENTRY177	DAPROM0 ROM Entries register 177.	0x00000000
0x311002C8	DAPROM0_ROMENTRY178	DAPROM0 ROM Entries register 178.	0x00000000
0x311002CC	DAPROM0_ROMENTRY179	DAPROM0 ROM Entries register 179.	0x00000000
0x311002D0	DAPROM0_ROMENTRY180	DAPROM0 ROM Entries register 180.	0x00000000
0x311002D4	DAPROM0_ROMENTRY181	DAPROM0 ROM Entries register 181.	0x00000000
0x311002D8	DAPROM0_ROMENTRY182	DAPROM0 ROM Entries register 182.	0x00000000
0x311002DC	DAPROM0_ROMENTRY183	DAPROM0 ROM Entries register 183.	0x00000000
0x311002E0	DAPROM0_ROMENTRY184	DAPROM0 ROM Entries register 184.	0x00000000
0x311002E4	DAPROM0_ROMENTRY185	DAPROM0 ROM Entries register 185.	0x00000000
0x311002E8	DAPROM0_ROMENTRY186	DAPROM0 ROM Entries register 186.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311002EC	DAPROM0_ROMENTRY187	DAPROM0 ROM Entries register 187.	0x00000000
0x311002F0	DAPROM0_ROMENTRY188	DAPROM0 ROM Entries register 188.	0x00000000
0x311002F4	DAPROM0_ROMENTRY189	DAPROM0 ROM Entries register 189.	0x00000000
0x311002F8	DAPROM0_ROMENTRY190	DAPROM0 ROM Entries register 190.	0x00000000
0x311002FC	DAPROM0_ROMENTRY191	DAPROM0 ROM Entries register 191.	0x00000000
0x31100300	DAPROM0_ROMENTRY192	DAPROM0 ROM Entries register 192.	0x00000000
0x31100304	DAPROM0_ROMENTRY193	DAPROM0 ROM Entries register 193.	0x00000000
0x31100308	DAPROM0_ROMENTRY194	DAPROM0 ROM Entries register 194.	0x00000000
0x3110030C	DAPROM0_ROMENTRY195	DAPROM0 ROM Entries register 195.	0x00000000
0x31100310	DAPROM0_ROMENTRY196	DAPROM0 ROM Entries register 196.	0x00000000
0x31100314	DAPROM0_ROMENTRY197	DAPROM0 ROM Entries register 197.	0x00000000
0x31100318	DAPROM0_ROMENTRY198	DAPROM0 ROM Entries register 198.	0x00000000
0x3110031C	DAPROM0_ROMENTRY199	DAPROM0 ROM Entries register 199.	0x00000000
0x31100320	DAPROM0_ROMENTRY200	DAPROM0 ROM Entries register 200.	0x00000000
0x31100324	DAPROM0_ROMENTRY201	DAPROM0 ROM Entries register 201.	0x00000000
0x31100328	DAPROM0_ROMENTRY202	DAPROM0 ROM Entries register 202.	0x00000000
0x3110032C	DAPROM0_ROMENTRY203	DAPROM0 ROM Entries register 203.	0x00000000
0x31100330	DAPROM0_ROMENTRY204	DAPROM0 ROM Entries register 204.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100334	DAPROM0_ROMENTRY205	DAPROM0 ROM Entries register 205.	0x00000000
0x31100338	DAPROM0_ROMENTRY206	DAPROM0 ROM Entries register 206.	0x00000000
0x3110033C	DAPROM0_ROMENTRY207	DAPROM0 ROM Entries register 207.	0x00000000
0x31100340	DAPROM0_ROMENTRY208	DAPROM0 ROM Entries register 208.	0x00000000
0x31100344	DAPROM0_ROMENTRY209	DAPROM0 ROM Entries register 209.	0x00000000
0x31100348	DAPROM0_ROMENTRY210	DAPROM0 ROM Entries register 210.	0x00000000
0x3110034C	DAPROM0_ROMENTRY211	DAPROM0 ROM Entries register 211.	0x00000000
0x31100350	DAPROM0_ROMENTRY212	DAPROM0 ROM Entries register 212.	0x00000000
0x31100354	DAPROM0_ROMENTRY213	DAPROM0 ROM Entries register 213.	0x00000000
0x31100358	DAPROM0_ROMENTRY214	DAPROM0 ROM Entries register 214.	0x00000000
0x3110035C	DAPROM0_ROMENTRY215	DAPROM0 ROM Entries register 215.	0x00000000
0x31100360	DAPROM0_ROMENTRY216	DAPROM0 ROM Entries register 216.	0x00000000
0x31100364	DAPROM0_ROMENTRY217	DAPROM0 ROM Entries register 217.	0x00000000
0x31100368	DAPROM0_ROMENTRY218	DAPROM0 ROM Entries register 218.	0x00000000
0x3110036C	DAPROM0_ROMENTRY219	DAPROM0 ROM Entries register 219.	0x00000000
0x31100370	DAPROM0_ROMENTRY220	DAPROM0 ROM Entries register 220.	0x00000000
0x31100374	DAPROM0_ROMENTRY221	DAPROM0 ROM Entries register 221.	0x00000000
0x31100378	DAPROM0_ROMENTRY222	DAPROM0 ROM Entries register 222.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110037C	DAPROM0_ROMENTRY223	DAPROM0 ROM Entries register 223.	0x00000000
0x31100380	DAPROM0_ROMENTRY224	DAPROM0 ROM Entries register 224.	0x00000000
0x31100384	DAPROM0_ROMENTRY225	DAPROM0 ROM Entries register 225.	0x00000000
0x31100388	DAPROM0_ROMENTRY226	DAPROM0 ROM Entries register 226.	0x00000000
0x3110038C	DAPROM0_ROMENTRY227	DAPROM0 ROM Entries register 227.	0x00000000
0x31100390	DAPROM0_ROMENTRY228	DAPROM0 ROM Entries register 228.	0x00000000
0x31100394	DAPROM0_ROMENTRY229	DAPROM0 ROM Entries register 229.	0x00000000
0x31100398	DAPROM0_ROMENTRY230	DAPROM0 ROM Entries register 230.	0x00000000
0x3110039C	DAPROM0_ROMENTRY231	DAPROM0 ROM Entries register 231.	0x00000000
0x311003A0	DAPROM0_ROMENTRY232	DAPROM0 ROM Entries register 232.	0x00000000
0x311003A4	DAPROM0_ROMENTRY233	DAPROM0 ROM Entries register 233.	0x00000000
0x311003A8	DAPROM0_ROMENTRY234	DAPROM0 ROM Entries register 234.	0x00000000
0x311003AC	DAPROM0_ROMENTRY235	DAPROM0 ROM Entries register 235.	0x00000000
0x311003B0	DAPROM0_ROMENTRY236	DAPROM0 ROM Entries register 236.	0x00000000
0x311003B4	DAPROM0_ROMENTRY237	DAPROM0 ROM Entries register 237.	0x00000000
0x311003B8	DAPROM0_ROMENTRY238	DAPROM0 ROM Entries register 238.	0x00000000
0x311003BC	DAPROM0_ROMENTRY239	DAPROM0 ROM Entries register 239.	0x00000000
0x311003C0	DAPROM0_ROMENTRY240	DAPROM0 ROM Entries register 240.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311003C4	DAPROM0_ROMENTRY241	DAPROM0 ROM Entries register 241.	0x00000000
0x311003C8	DAPROM0_ROMENTRY242	DAPROM0 ROM Entries register 242.	0x00000000
0x311003CC	DAPROM0_ROMENTRY243	DAPROM0 ROM Entries register 243.	0x00000000
0x311003D0	DAPROM0_ROMENTRY244	DAPROM0 ROM Entries register 244.	0x00000000
0x311003D4	DAPROM0_ROMENTRY245	DAPROM0 ROM Entries register 245.	0x00000000
0x311003D8	DAPROM0_ROMENTRY246	DAPROM0 ROM Entries register 246.	0x00000000
0x311003DC	DAPROM0_ROMENTRY247	DAPROM0 ROM Entries register 247.	0x00000000
0x311003E0	DAPROM0_ROMENTRY248	DAPROM0 ROM Entries register 248.	0x00000000
0x311003E4	DAPROM0_ROMENTRY249	DAPROM0 ROM Entries register 249.	0x00000000
0x311003E8	DAPROM0_ROMENTRY250	DAPROM0 ROM Entries register 250.	0x00000000
0x311003EC	DAPROM0_ROMENTRY251	DAPROM0 ROM Entries register 251.	0x00000000
0x311003F0	DAPROM0_ROMENTRY252	DAPROM0 ROM Entries register 252.	0x00000000
0x311003F4	DAPROM0_ROMENTRY253	DAPROM0 ROM Entries register 253.	0x00000000
0x311003F8	DAPROM0_ROMENTRY254	DAPROM0 ROM Entries register 254.	0x00000000
0x311003FC	DAPROM0_ROMENTRY255	DAPROM0 ROM Entries register 255.	0x00000000
0x31100400	DAPROM0_ROMENTRY256	DAPROM0 ROM Entries register 256.	0x00000000
0x31100404	DAPROM0_ROMENTRY257	DAPROM0 ROM Entries register 257.	0x00000000
0x31100408	DAPROM0_ROMENTRY258	DAPROM0 ROM Entries register 258.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110040C	DAPROM0_ROMENTRY259	DAPROM0 ROM Entries register 259.	0x00000000
0x31100410	DAPROM0_ROMENTRY260	DAPROM0 ROM Entries register 260.	0x00000000
0x31100414	DAPROM0_ROMENTRY261	DAPROM0 ROM Entries register 261.	0x00000000
0x31100418	DAPROM0_ROMENTRY262	DAPROM0 ROM Entries register 262.	0x00000000
0x3110041C	DAPROM0_ROMENTRY263	DAPROM0 ROM Entries register 263.	0x00000000
0x31100420	DAPROM0_ROMENTRY264	DAPROM0 ROM Entries register 264.	0x00000000
0x31100424	DAPROM0_ROMENTRY265	DAPROM0 ROM Entries register 265.	0x00000000
0x31100428	DAPROM0_ROMENTRY266	DAPROM0 ROM Entries register 266.	0x00000000
0x3110042C	DAPROM0_ROMENTRY267	DAPROM0 ROM Entries register 267.	0x00000000
0x31100430	DAPROM0_ROMENTRY268	DAPROM0 ROM Entries register 268.	0x00000000
0x31100434	DAPROM0_ROMENTRY269	DAPROM0 ROM Entries register 269.	0x00000000
0x31100438	DAPROM0_ROMENTRY270	DAPROM0 ROM Entries register 270.	0x00000000
0x3110043C	DAPROM0_ROMENTRY271	DAPROM0 ROM Entries register 271.	0x00000000
0x31100440	DAPROM0_ROMENTRY272	DAPROM0 ROM Entries register 272.	0x00000000
0x31100444	DAPROM0_ROMENTRY273	DAPROM0 ROM Entries register 273.	0x00000000
0x31100448	DAPROM0_ROMENTRY274	DAPROM0 ROM Entries register 274.	0x00000000
0x3110044C	DAPROM0_ROMENTRY275	DAPROM0 ROM Entries register 275.	0x00000000
0x31100450	DAPROM0_ROMENTRY276	DAPROM0 ROM Entries register 276.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100454	DAPROM0_ROMENTRY277	DAPROM0 ROM Entries register 277.	0x00000000
0x31100458	DAPROM0_ROMENTRY278	DAPROM0 ROM Entries register 278.	0x00000000
0x3110045C	DAPROM0_ROMENTRY279	DAPROM0 ROM Entries register 279.	0x00000000
0x31100460	DAPROM0_ROMENTRY280	DAPROM0 ROM Entries register 280.	0x00000000
0x31100464	DAPROM0_ROMENTRY281	DAPROM0 ROM Entries register 281.	0x00000000
0x31100468	DAPROM0_ROMENTRY282	DAPROM0 ROM Entries register 282.	0x00000000
0x3110046C	DAPROM0_ROMENTRY283	DAPROM0 ROM Entries register 283.	0x00000000
0x31100470	DAPROM0_ROMENTRY284	DAPROM0 ROM Entries register 284.	0x00000000
0x31100474	DAPROM0_ROMENTRY285	DAPROM0 ROM Entries register 285.	0x00000000
0x31100478	DAPROM0_ROMENTRY286	DAPROM0 ROM Entries register 286.	0x00000000
0x3110047C	DAPROM0_ROMENTRY287	DAPROM0 ROM Entries register 287.	0x00000000
0x31100480	DAPROM0_ROMENTRY288	DAPROM0 ROM Entries register 288.	0x00000000
0x31100484	DAPROM0_ROMENTRY289	DAPROM0 ROM Entries register 289.	0x00000000
0x31100488	DAPROM0_ROMENTRY290	DAPROM0 ROM Entries register 290.	0x00000000
0x3110048C	DAPROM0_ROMENTRY291	DAPROM0 ROM Entries register 291.	0x00000000
0x31100490	DAPROM0_ROMENTRY292	DAPROM0 ROM Entries register 292.	0x00000000
0x31100494	DAPROM0_ROMENTRY293	DAPROM0 ROM Entries register 293.	0x00000000
0x31100498	DAPROM0_ROMENTRY294	DAPROM0 ROM Entries register 294.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110049C	DAPROM0_ROMENTRY295	DAPROM0 ROM Entries register 295.	0x00000000
0x311004A0	DAPROM0_ROMENTRY296	DAPROM0 ROM Entries register 296.	0x00000000
0x311004A4	DAPROM0_ROMENTRY297	DAPROM0 ROM Entries register 297.	0x00000000
0x311004A8	DAPROM0_ROMENTRY298	DAPROM0 ROM Entries register 298.	0x00000000
0x311004AC	DAPROM0_ROMENTRY299	DAPROM0 ROM Entries register 299.	0x00000000
0x311004B0	DAPROM0_ROMENTRY300	DAPROM0 ROM Entries register 300.	0x00000000
0x311004B4	DAPROM0_ROMENTRY301	DAPROM0 ROM Entries register 301.	0x00000000
0x311004B8	DAPROM0_ROMENTRY302	DAPROM0 ROM Entries register 302.	0x00000000
0x311004BC	DAPROM0_ROMENTRY303	DAPROM0 ROM Entries register 303.	0x00000000
0x311004C0	DAPROM0_ROMENTRY304	DAPROM0 ROM Entries register 304.	0x00000000
0x311004C4	DAPROM0_ROMENTRY305	DAPROM0 ROM Entries register 305.	0x00000000
0x311004C8	DAPROM0_ROMENTRY306	DAPROM0 ROM Entries register 306.	0x00000000
0x311004CC	DAPROM0_ROMENTRY307	DAPROM0 ROM Entries register 307.	0x00000000
0x311004D0	DAPROM0_ROMENTRY308	DAPROM0 ROM Entries register 308.	0x00000000
0x311004D4	DAPROM0_ROMENTRY309	DAPROM0 ROM Entries register 309.	0x00000000
0x311004D8	DAPROM0_ROMENTRY310	DAPROM0 ROM Entries register 310.	0x00000000
0x311004DC	DAPROM0_ROMENTRY311	DAPROM0 ROM Entries register 311.	0x00000000
0x311004E0	DAPROM0_ROMENTRY312	DAPROM0 ROM Entries register 312.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311004E4	DAPROM0_ROMENTRY313	DAPROM0 ROM Entries register 313.	0x00000000
0x311004E8	DAPROM0_ROMENTRY314	DAPROM0 ROM Entries register 314.	0x00000000
0x311004EC	DAPROM0_ROMENTRY315	DAPROM0 ROM Entries register 315.	0x00000000
0x311004F0	DAPROM0_ROMENTRY316	DAPROM0 ROM Entries register 316.	0x00000000
0x311004F4	DAPROM0_ROMENTRY317	DAPROM0 ROM Entries register 317.	0x00000000
0x311004F8	DAPROM0_ROMENTRY318	DAPROM0 ROM Entries register 318.	0x00000000
0x311004FC	DAPROM0_ROMENTRY319	DAPROM0 ROM Entries register 319.	0x00000000
0x31100500	DAPROM0_ROMENTRY320	DAPROM0 ROM Entries register 320.	0x00000000
0x31100504	DAPROM0_ROMENTRY321	DAPROM0 ROM Entries register 321.	0x00000000
0x31100508	DAPROM0_ROMENTRY322	DAPROM0 ROM Entries register 322.	0x00000000
0x3110050C	DAPROM0_ROMENTRY323	DAPROM0 ROM Entries register 323.	0x00000000
0x31100510	DAPROM0_ROMENTRY324	DAPROM0 ROM Entries register 324.	0x00000000
0x31100514	DAPROM0_ROMENTRY325	DAPROM0 ROM Entries register 325.	0x00000000
0x31100518	DAPROM0_ROMENTRY326	DAPROM0 ROM Entries register 326.	0x00000000
0x3110051C	DAPROM0_ROMENTRY327	DAPROM0 ROM Entries register 327.	0x00000000
0x31100520	DAPROM0_ROMENTRY328	DAPROM0 ROM Entries register 328.	0x00000000
0x31100524	DAPROM0_ROMENTRY329	DAPROM0 ROM Entries register 329.	0x00000000
0x31100528	DAPROM0_ROMENTRY330	DAPROM0 ROM Entries register 330.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110052C	DAPROM0_ROMENTRY331	DAPROM0 ROM Entries register 331.	0x00000000
0x31100530	DAPROM0_ROMENTRY332	DAPROM0 ROM Entries register 332.	0x00000000
0x31100534	DAPROM0_ROMENTRY333	DAPROM0 ROM Entries register 333.	0x00000000
0x31100538	DAPROM0_ROMENTRY334	DAPROM0 ROM Entries register 334.	0x00000000
0x3110053C	DAPROM0_ROMENTRY335	DAPROM0 ROM Entries register 335.	0x00000000
0x31100540	DAPROM0_ROMENTRY336	DAPROM0 ROM Entries register 336.	0x00000000
0x31100544	DAPROM0_ROMENTRY337	DAPROM0 ROM Entries register 337.	0x00000000
0x31100548	DAPROM0_ROMENTRY338	DAPROM0 ROM Entries register 338.	0x00000000
0x3110054C	DAPROM0_ROMENTRY339	DAPROM0 ROM Entries register 339.	0x00000000
0x31100550	DAPROM0_ROMENTRY340	DAPROM0 ROM Entries register 340.	0x00000000
0x31100554	DAPROM0_ROMENTRY341	DAPROM0 ROM Entries register 341.	0x00000000
0x31100558	DAPROM0_ROMENTRY342	DAPROM0 ROM Entries register 342.	0x00000000
0x3110055C	DAPROM0_ROMENTRY343	DAPROM0 ROM Entries register 343.	0x00000000
0x31100560	DAPROM0_ROMENTRY344	DAPROM0 ROM Entries register 344.	0x00000000
0x31100564	DAPROM0_ROMENTRY345	DAPROM0 ROM Entries register 345.	0x00000000
0x31100568	DAPROM0_ROMENTRY346	DAPROM0 ROM Entries register 346.	0x00000000
0x3110056C	DAPROM0_ROMENTRY347	DAPROM0 ROM Entries register 347.	0x00000000
0x31100570	DAPROM0_ROMENTRY348	DAPROM0 ROM Entries register 348.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100574	DAPROM0_ROMENTRY349	DAPROM0 ROM Entries register 349.	0x00000000
0x31100578	DAPROM0_ROMENTRY350	DAPROM0 ROM Entries register 350.	0x00000000
0x3110057C	DAPROM0_ROMENTRY351	DAPROM0 ROM Entries register 351.	0x00000000
0x31100580	DAPROM0_ROMENTRY352	DAPROM0 ROM Entries register 352.	0x00000000
0x31100584	DAPROM0_ROMENTRY353	DAPROM0 ROM Entries register 353.	0x00000000
0x31100588	DAPROM0_ROMENTRY354	DAPROM0 ROM Entries register 354.	0x00000000
0x3110058C	DAPROM0_ROMENTRY355	DAPROM0 ROM Entries register 355.	0x00000000
0x31100590	DAPROM0_ROMENTRY356	DAPROM0 ROM Entries register 356.	0x00000000
0x31100594	DAPROM0_ROMENTRY357	DAPROM0 ROM Entries register 357.	0x00000000
0x31100598	DAPROM0_ROMENTRY358	DAPROM0 ROM Entries register 358.	0x00000000
0x3110059C	DAPROM0_ROMENTRY359	DAPROM0 ROM Entries register 359.	0x00000000
0x311005A0	DAPROM0_ROMENTRY360	DAPROM0 ROM Entries register 360.	0x00000000
0x311005A4	DAPROM0_ROMENTRY361	DAPROM0 ROM Entries register 361.	0x00000000
0x311005A8	DAPROM0_ROMENTRY362	DAPROM0 ROM Entries register 362.	0x00000000
0x311005AC	DAPROM0_ROMENTRY363	DAPROM0 ROM Entries register 363.	0x00000000
0x311005B0	DAPROM0_ROMENTRY364	DAPROM0 ROM Entries register 364.	0x00000000
0x311005B4	DAPROM0_ROMENTRY365	DAPROM0 ROM Entries register 365.	0x00000000
0x311005B8	DAPROM0_ROMENTRY366	DAPROM0 ROM Entries register 366.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311005BC	DAPROM0_ROMENTRY367	DAPROM0 ROM Entries register 367.	0x00000000
0x311005C0	DAPROM0_ROMENTRY368	DAPROM0 ROM Entries register 368.	0x00000000
0x311005C4	DAPROM0_ROMENTRY369	DAPROM0 ROM Entries register 369.	0x00000000
0x311005C8	DAPROM0_ROMENTRY370	DAPROM0 ROM Entries register 370.	0x00000000
0x311005CC	DAPROM0_ROMENTRY371	DAPROM0 ROM Entries register 371.	0x00000000
0x311005D0	DAPROM0_ROMENTRY372	DAPROM0 ROM Entries register 372.	0x00000000
0x311005D4	DAPROM0_ROMENTRY373	DAPROM0 ROM Entries register 373.	0x00000000
0x311005D8	DAPROM0_ROMENTRY374	DAPROM0 ROM Entries register 374.	0x00000000
0x311005DC	DAPROM0_ROMENTRY375	DAPROM0 ROM Entries register 375.	0x00000000
0x311005E0	DAPROM0_ROMENTRY376	DAPROM0 ROM Entries register 376.	0x00000000
0x311005E4	DAPROM0_ROMENTRY377	DAPROM0 ROM Entries register 377.	0x00000000
0x311005E8	DAPROM0_ROMENTRY378	DAPROM0 ROM Entries register 378.	0x00000000
0x311005EC	DAPROM0_ROMENTRY379	DAPROM0 ROM Entries register 379.	0x00000000
0x311005F0	DAPROM0_ROMENTRY380	DAPROM0 ROM Entries register 380.	0x00000000
0x311005F4	DAPROM0_ROMENTRY381	DAPROM0 ROM Entries register 381.	0x00000000
0x311005F8	DAPROM0_ROMENTRY382	DAPROM0 ROM Entries register 382.	0x00000000
0x311005FC	DAPROM0_ROMENTRY383	DAPROM0 ROM Entries register 383.	0x00000000
0x31100600	DAPROM0_ROMENTRY384	DAPROM0 ROM Entries register 384.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100604	DAPROM0_ROMENTRY385	DAPROM0 ROM Entries register 385.	0x00000000
0x31100608	DAPROM0_ROMENTRY386	DAPROM0 ROM Entries register 386.	0x00000000
0x3110060C	DAPROM0_ROMENTRY387	DAPROM0 ROM Entries register 387.	0x00000000
0x31100610	DAPROM0_ROMENTRY388	DAPROM0 ROM Entries register 388.	0x00000000
0x31100614	DAPROM0_ROMENTRY389	DAPROM0 ROM Entries register 389.	0x00000000
0x31100618	DAPROM0_ROMENTRY390	DAPROM0 ROM Entries register 390.	0x00000000
0x3110061C	DAPROM0_ROMENTRY391	DAPROM0 ROM Entries register 391.	0x00000000
0x31100620	DAPROM0_ROMENTRY392	DAPROM0 ROM Entries register 392.	0x00000000
0x31100624	DAPROM0_ROMENTRY393	DAPROM0 ROM Entries register 393.	0x00000000
0x31100628	DAPROM0_ROMENTRY394	DAPROM0 ROM Entries register 394.	0x00000000
0x3110062C	DAPROM0_ROMENTRY395	DAPROM0 ROM Entries register 395.	0x00000000
0x31100630	DAPROM0_ROMENTRY396	DAPROM0 ROM Entries register 396.	0x00000000
0x31100634	DAPROM0_ROMENTRY397	DAPROM0 ROM Entries register 397.	0x00000000
0x31100638	DAPROM0_ROMENTRY398	DAPROM0 ROM Entries register 398.	0x00000000
0x3110063C	DAPROM0_ROMENTRY399	DAPROM0 ROM Entries register 399.	0x00000000
0x31100640	DAPROM0_ROMENTRY400	DAPROM0 ROM Entries register 400.	0x00000000
0x31100644	DAPROM0_ROMENTRY401	DAPROM0 ROM Entries register 401.	0x00000000
0x31100648	DAPROM0_ROMENTRY402	DAPROM0 ROM Entries register 402.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110064C	DAPROM0_ROMENTRY403	DAPROM0 ROM Entries register 403.	0x00000000
0x31100650	DAPROM0_ROMENTRY404	DAPROM0 ROM Entries register 404.	0x00000000
0x31100654	DAPROM0_ROMENTRY405	DAPROM0 ROM Entries register 405.	0x00000000
0x31100658	DAPROM0_ROMENTRY406	DAPROM0 ROM Entries register 406.	0x00000000
0x3110065C	DAPROM0_ROMENTRY407	DAPROM0 ROM Entries register 407.	0x00000000
0x31100660	DAPROM0_ROMENTRY408	DAPROM0 ROM Entries register 408.	0x00000000
0x31100664	DAPROM0_ROMENTRY409	DAPROM0 ROM Entries register 409.	0x00000000
0x31100668	DAPROM0_ROMENTRY410	DAPROM0 ROM Entries register 410.	0x00000000
0x3110066C	DAPROM0_ROMENTRY411	DAPROM0 ROM Entries register 411.	0x00000000
0x31100670	DAPROM0_ROMENTRY412	DAPROM0 ROM Entries register 412.	0x00000000
0x31100674	DAPROM0_ROMENTRY413	DAPROM0 ROM Entries register 413.	0x00000000
0x31100678	DAPROM0_ROMENTRY414	DAPROM0 ROM Entries register 414.	0x00000000
0x3110067C	DAPROM0_ROMENTRY415	DAPROM0 ROM Entries register 415.	0x00000000
0x31100680	DAPROM0_ROMENTRY416	DAPROM0 ROM Entries register 416.	0x00000000
0x31100684	DAPROM0_ROMENTRY417	DAPROM0 ROM Entries register 417.	0x00000000
0x31100688	DAPROM0_ROMENTRY418	DAPROM0 ROM Entries register 418.	0x00000000
0x3110068C	DAPROM0_ROMENTRY419	DAPROM0 ROM Entries register 419.	0x00000000
0x31100690	DAPROM0_ROMENTRY420	DAPROM0 ROM Entries register 420.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100694	DAPROM0_ROMENTRY421	DAPROM0 ROM Entries register 421.	0x00000000
0x31100698	DAPROM0_ROMENTRY422	DAPROM0 ROM Entries register 422.	0x00000000
0x3110069C	DAPROM0_ROMENTRY423	DAPROM0 ROM Entries register 423.	0x00000000
0x311006A0	DAPROM0_ROMENTRY424	DAPROM0 ROM Entries register 424.	0x00000000
0x311006A4	DAPROM0_ROMENTRY425	DAPROM0 ROM Entries register 425.	0x00000000
0x311006A8	DAPROM0_ROMENTRY426	DAPROM0 ROM Entries register 426.	0x00000000
0x311006AC	DAPROM0_ROMENTRY427	DAPROM0 ROM Entries register 427.	0x00000000
0x311006B0	DAPROM0_ROMENTRY428	DAPROM0 ROM Entries register 428.	0x00000000
0x311006B4	DAPROM0_ROMENTRY429	DAPROM0 ROM Entries register 429.	0x00000000
0x311006B8	DAPROM0_ROMENTRY430	DAPROM0 ROM Entries register 430.	0x00000000
0x311006BC	DAPROM0_ROMENTRY431	DAPROM0 ROM Entries register 431.	0x00000000
0x311006C0	DAPROM0_ROMENTRY432	DAPROM0 ROM Entries register 432.	0x00000000
0x311006C4	DAPROM0_ROMENTRY433	DAPROM0 ROM Entries register 433.	0x00000000
0x311006C8	DAPROM0_ROMENTRY434	DAPROM0 ROM Entries register 434.	0x00000000
0x311006CC	DAPROM0_ROMENTRY435	DAPROM0 ROM Entries register 435.	0x00000000
0x311006D0	DAPROM0_ROMENTRY436	DAPROM0 ROM Entries register 436.	0x00000000
0x311006D4	DAPROM0_ROMENTRY437	DAPROM0 ROM Entries register 437.	0x00000000
0x311006D8	DAPROM0_ROMENTRY438	DAPROM0 ROM Entries register 438.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311006DC	DAPROM0_ROMENTRY439	DAPROM0 ROM Entries register 439.	0x00000000
0x311006E0	DAPROM0_ROMENTRY440	DAPROM0 ROM Entries register 440.	0x00000000
0x311006E4	DAPROM0_ROMENTRY441	DAPROM0 ROM Entries register 441.	0x00000000
0x311006E8	DAPROM0_ROMENTRY442	DAPROM0 ROM Entries register 442.	0x00000000
0x311006EC	DAPROM0_ROMENTRY443	DAPROM0 ROM Entries register 443.	0x00000000
0x311006F0	DAPROM0_ROMENTRY444	DAPROM0 ROM Entries register 444.	0x00000000
0x311006F4	DAPROM0_ROMENTRY445	DAPROM0 ROM Entries register 445.	0x00000000
0x311006F8	DAPROM0_ROMENTRY446	DAPROM0 ROM Entries register 446.	0x00000000
0x311006FC	DAPROM0_ROMENTRY447	DAPROM0 ROM Entries register 447.	0x00000000
0x31100700	DAPROM0_ROMENTRY448	DAPROM0 ROM Entries register 448.	0x00000000
0x31100704	DAPROM0_ROMENTRY449	DAPROM0 ROM Entries register 449.	0x00000000
0x31100708	DAPROM0_ROMENTRY450	DAPROM0 ROM Entries register 450.	0x00000000
0x3110070C	DAPROM0_ROMENTRY451	DAPROM0 ROM Entries register 451.	0x00000000
0x31100710	DAPROM0_ROMENTRY452	DAPROM0 ROM Entries register 452.	0x00000000
0x31100714	DAPROM0_ROMENTRY453	DAPROM0 ROM Entries register 453.	0x00000000
0x31100718	DAPROM0_ROMENTRY454	DAPROM0 ROM Entries register 454.	0x00000000
0x3110071C	DAPROM0_ROMENTRY455	DAPROM0 ROM Entries register 455.	0x00000000
0x31100720	DAPROM0_ROMENTRY456	DAPROM0 ROM Entries register 456.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100724	DAPROM0_ROMENTRY457	DAPROM0 ROM Entries register 457.	0x00000000
0x31100728	DAPROM0_ROMENTRY458	DAPROM0 ROM Entries register 458.	0x00000000
0x3110072C	DAPROM0_ROMENTRY459	DAPROM0 ROM Entries register 459.	0x00000000
0x31100730	DAPROM0_ROMENTRY460	DAPROM0 ROM Entries register 460.	0x00000000
0x31100734	DAPROM0_ROMENTRY461	DAPROM0 ROM Entries register 461.	0x00000000
0x31100738	DAPROM0_ROMENTRY462	DAPROM0 ROM Entries register 462.	0x00000000
0x3110073C	DAPROM0_ROMENTRY463	DAPROM0 ROM Entries register 463.	0x00000000
0x31100740	DAPROM0_ROMENTRY464	DAPROM0 ROM Entries register 464.	0x00000000
0x31100744	DAPROM0_ROMENTRY465	DAPROM0 ROM Entries register 465.	0x00000000
0x31100748	DAPROM0_ROMENTRY466	DAPROM0 ROM Entries register 466.	0x00000000
0x3110074C	DAPROM0_ROMENTRY467	DAPROM0 ROM Entries register 467.	0x00000000
0x31100750	DAPROM0_ROMENTRY468	DAPROM0 ROM Entries register 468.	0x00000000
0x31100754	DAPROM0_ROMENTRY469	DAPROM0 ROM Entries register 469.	0x00000000
0x31100758	DAPROM0_ROMENTRY470	DAPROM0 ROM Entries register 470.	0x00000000
0x3110075C	DAPROM0_ROMENTRY471	DAPROM0 ROM Entries register 471.	0x00000000
0x31100760	DAPROM0_ROMENTRY472	DAPROM0 ROM Entries register 472.	0x00000000
0x31100764	DAPROM0_ROMENTRY473	DAPROM0 ROM Entries register 473.	0x00000000
0x31100768	DAPROM0_ROMENTRY474	DAPROM0 ROM Entries register 474.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110076C	DAPROM0_ROMENTRY475	DAPROM0 ROM Entries register 475.	0x00000000
0x31100770	DAPROM0_ROMENTRY476	DAPROM0 ROM Entries register 476.	0x00000000
0x31100774	DAPROM0_ROMENTRY477	DAPROM0 ROM Entries register 477.	0x00000000
0x31100778	DAPROM0_ROMENTRY478	DAPROM0 ROM Entries register 478.	0x00000000
0x3110077C	DAPROM0_ROMENTRY479	DAPROM0 ROM Entries register 479.	0x00000000
0x31100780	DAPROM0_ROMENTRY480	DAPROM0 ROM Entries register 480.	0x00000000
0x31100784	DAPROM0_ROMENTRY481	DAPROM0 ROM Entries register 481.	0x00000000
0x31100788	DAPROM0_ROMENTRY482	DAPROM0 ROM Entries register 482.	0x00000000
0x3110078C	DAPROM0_ROMENTRY483	DAPROM0 ROM Entries register 483.	0x00000000
0x31100790	DAPROM0_ROMENTRY484	DAPROM0 ROM Entries register 484.	0x00000000
0x31100794	DAPROM0_ROMENTRY485	DAPROM0 ROM Entries register 485.	0x00000000
0x31100798	DAPROM0_ROMENTRY486	DAPROM0 ROM Entries register 486.	0x00000000
0x3110079C	DAPROM0_ROMENTRY487	DAPROM0 ROM Entries register 487.	0x00000000
0x311007A0	DAPROM0_ROMENTRY488	DAPROM0 ROM Entries register 488.	0x00000000
0x311007A4	DAPROM0_ROMENTRY489	DAPROM0 ROM Entries register 489.	0x00000000
0x311007A8	DAPROM0_ROMENTRY490	DAPROM0 ROM Entries register 490.	0x00000000
0x311007AC	DAPROM0_ROMENTRY491	DAPROM0 ROM Entries register 491.	0x00000000
0x311007B0	DAPROM0_ROMENTRY492	DAPROM0 ROM Entries register 492.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311007B4	DAPROM0_ROMENTRY493	DAPROM0 ROM Entries register 493.	0x00000000
0x311007B8	DAPROM0_ROMENTRY494	DAPROM0 ROM Entries register 494.	0x00000000
0x311007BC	DAPROM0_ROMENTRY495	DAPROM0 ROM Entries register 495.	0x00000000
0x311007C0	DAPROM0_ROMENTRY496	DAPROM0 ROM Entries register 496.	0x00000000
0x311007C4	DAPROM0_ROMENTRY497	DAPROM0 ROM Entries register 497.	0x00000000
0x311007C8	DAPROM0_ROMENTRY498	DAPROM0 ROM Entries register 498.	0x00000000
0x311007CC	DAPROM0_ROMENTRY499	DAPROM0 ROM Entries register 499.	0x00000000
0x311007D0	DAPROM0_ROMENTRY500	DAPROM0 ROM Entries register 500.	0x00000000
0x311007D4	DAPROM0_ROMENTRY501	DAPROM0 ROM Entries register 501.	0x00000000
0x311007D8	DAPROM0_ROMENTRY502	DAPROM0 ROM Entries register 502.	0x00000000
0x311007DC	DAPROM0_ROMENTRY503	DAPROM0 ROM Entries register 503.	0x00000000
0x311007E0	DAPROM0_ROMENTRY504	DAPROM0 ROM Entries register 504.	0x00000000
0x311007E4	DAPROM0_ROMENTRY505	DAPROM0 ROM Entries register 505.	0x00000000
0x311007E8	DAPROM0_ROMENTRY506	DAPROM0 ROM Entries register 506.	0x00000000
0x311007EC	DAPROM0_ROMENTRY507	DAPROM0 ROM Entries register 507.	0x00000000
0x311007F0	DAPROM0_ROMENTRY508	DAPROM0 ROM Entries register 508.	0x00000000
0x311007F4	DAPROM0_ROMENTRY509	DAPROM0 ROM Entries register 509.	0x00000000
0x311007F8	DAPROM0_ROMENTRY510	DAPROM0 ROM Entries register 510.	0x00000000

Table A-17: ADSP-2184x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311007FC	DAPROM0_ROMENTRY511	DAPROM0 ROM Entries register 511.	0x00000000
0x31100FBC	DAPROM0_DEVARCH	DAPROM0 Device Architecture Register.	0x47700AF7
0x31100FC8	DAPROM0_DEVID	DAPROM0 Device Configuration Register.	0x00000000
0x31100FD0	DAPROM0_PIDR4	DAPROM0 Peripheral Identification Register 4.	0x00000000
0x31100FD4	DAPROM0_PIDR5	DAPROM0 Peripheral Identification Register 5.	0x00000000
0x31100FD8	DAPROM0_PIDR6	DAPROM0 Peripheral Identification Register 6.	0x00000000
0x31100FDC	DAPROM0_PIDR7	DAPROM0 Peripheral Identification Register 7.	0x00000000
0x31100FE0	DAPROM0_PIDR0	DAPROM0 Peripheral Identification Register 0.	0x00000000
0x31100FE4	DAPROM0_PIDR1	DAPROM0 Peripheral Identification Register 1.	0x00000000
0x31100FE8	DAPROM0_PIDR2	DAPROM0 Peripheral Identification Register 2.	0x00000008
0x31100FEC	DAPROM0_PIDR3	DAPROM0 Peripheral Identification Register 3.	0x00000000
0x31100FF0	DAPROM0_CIDR0	DAPROM0 Component Identification Register 0.	0x0000000D
0x31100FF4	DAPROM0_CIDR1	DAPROM0 Component Identification Register 1.	0x00000090
0x31100FF8	DAPROM0_CIDR2	DAPROM0 Component Identification Register 2.	0x00000005
0x31100FFC	DAPROM0_CIDR3	DAPROM0 Component Identification Register 3.	0x000000B1

Table A-18: ADSP-2184x CSTF0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31108000	CSTF0_FUNNELCONTROL	CSTF0 Funnel Control register.	0x00000300
0x31108004	CSTF0_PRIORITYCONTROL	CSTF0 Priority Control register.	0x00000000
0x31108EEC	CSTF0_ITATBDATA0	CSTF0 Integration test data register.	0x00000000
0x31108EF0	CSTF0_ITATBCTR3	CSTF0 Integration test control register 3.	0x00000000
0x31108EF4	CSTF0_ITATBCTR2	CSTF0 Integration test control register 2.	0x00000000
0x31108EF8	CSTF0_ITATBCTR1	CSTF0 Integration test control register 1.	0x00000000
0x31108EFC	CSTF0_ITATBCTR0	CSTF0 Integration test control register 0.	0x00000000
0x31108F00	CSTF0_ITCTRL	CSTF0 Integration Mode Control Register.	0x00000000
0x31108FA0	CSTF0_CLAIMSET	CSTF0 Claim Tag Set Register.	0x0000000F
0x31108FA4	CSTF0_CLAIMCLR	CSTF0 Claim Tag Clear Register.	0x00000000

Table A-18: ADSP-2184x CSTF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31108FA8	CSTF0_DEVAFF0	CSTF0 Device Affinity register 0.	0x00000000
0x31108FAC	CSTF0_DEVAFF1	CSTF0 Device Affinity register 1.	0x00000000
0x31108FB8	CSTF0_AUTHSTATUS	CSTF0 Authentication Status Register.	0x00000000
0x31108FBC	CSTF0_DEVARCH	CSTF0 Device Architecture Register.	0x00000000
0x31108FC0	CSTF0_DEVID2	CSTF0 Device Configuration Register 2.	0x00000000
0x31108FC4	CSTF0_DEVID1	CSTF0 Device Configuration Register 1.	0x00000000
0x31108FC8	CSTF0_DEVID	CSTF0 Device Configuration Register.	0x00000033
0x31108FCC	CSTF0_DEVTYPE	CSTF0 Device Type Identifier Register.	0x00000012
0x31108FD0	CSTF0_PIDR4	CSTF0 Peripheral Identification Register 4.	0x00000004
0x31108FD4	CSTF0_PIDR5	CSTF0 Peripheral Identification Register 5.	0x00000000
0x31108FD8	CSTF0_PIDR6	CSTF0 Peripheral Identification Register 6.	0x00000000
0x31108FDC	CSTF0_PIDR7	CSTF0 Peripheral Identification Register 7.	0x00000000
0x31108FE0	CSTF0_PIDR0	CSTF0 Peripheral Identification Register 0.	0x000000EB
0x31108FE4	CSTF0_PIDR1	CSTF0 Peripheral Identification Register 1.	0x000000B9
0x31108FE8	CSTF0_PIDR2	CSTF0 Peripheral Identification Register 2.	0x0000004B
0x31108FEC	CSTF0_PIDR3	CSTF0 Peripheral Identification Register 3.	0x00000000
0x31108FF0	CSTF0_CIDR0	CSTF0 Component Identification Register 0.	0x0000000D
0x31108FF4	CSTF0_CIDR1	CSTF0 Component Identification Register 1.	0x00000090
0x31108FF8	CSTF0_CIDR2	CSTF0 Component Identification Register 2.	0x00000005
0x31108FFC	CSTF0_CIDR3	CSTF0 Component Identification Register 3.	0x000000B1

Table A-19: ADSP-2184x SYSCTI MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110D000	SYSCTI_CTICONTROL	SYSCTI CTI Control register.	0x00000000
0x3110D010	SYSCTI_CTIINTACK	SYSCTI CTI Interrupt Acknowledge register.	0x00000000
0x3110D014	SYSCTI_CTIAPPSET	SYSCTI CTI Application Channel Set register.	0x00000000
0x3110D018	SYSCTI_CTIAPPCLEAR	SYSCTI CTI Application Channel Clear register.	0x00000000
0x3110D01C	SYSCTI_CTIAPPPULSE	SYSCTI CTI Application Channel Pulse register.	0x00000000
0x3110D020	SYSCTI_CTIINEN0	SYSCTI CTI Trigger 0 to Channel Enable register.	0x00000000
0x3110D024	SYSCTI_CTIINEN1	SYSCTI CTI Trigger 1 to Channel Enable register.	0x00000000

Table A-19: ADSP-2184x SYSCTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110D028	SYSCTI_CTIINEN2	SYSCTI CTI Trigger 2 to Channel Enable register.	0x00000000
0x3110D02C	SYSCTI_CTIINEN3	SYSCTI CTI Trigger 3 to Channel Enable register.	0x00000000
0x3110D030	SYSCTI_CTIINEN4	SYSCTI CTI Trigger 4 to Channel Enable register.	0x00000000
0x3110D034	SYSCTI_CTIINEN5	SYSCTI CTI Trigger 5 to Channel Enable register.	0x00000000
0x3110D038	SYSCTI_CTIINEN6	SYSCTI CTI Trigger 6 to Channel Enable register.	0x00000000
0x3110D03C	SYSCTI_CTIINEN7	SYSCTI CTI Trigger 7 to Channel Enable register.	0x00000000
0x3110D0A0	SYSCTI_CTIOUTEN0	SYSCTI CTI Channel to Trigger 0 Enable register.	0x00000000
0x3110D0A4	SYSCTI_CTIOUTEN1	SYSCTI CTI Channel to Trigger 1 Enable register.	0x00000000
0x3110D0A8	SYSCTI_CTIOUTEN2	SYSCTI CTI Channel to Trigger 2 Enable register.	0x00000000
0x3110D0AC	SYSCTI_CTIOUTEN3	SYSCTI CTI Channel to Trigger 3 Enable register.	0x00000000
0x3110D0B0	SYSCTI_CTIOUTEN4	SYSCTI CTI Channel to Trigger 4 Enable register.	0x00000000
0x3110D0B4	SYSCTI_CTIOUTEN5	SYSCTI CTI Channel to Trigger 5 Enable register.	0x00000000
0x3110D0B8	SYSCTI_CTIOUTEN6	SYSCTI CTI Channel to Trigger 6 Enable register.	0x00000000
0x3110D0BC	SYSCTI_CTIOUTEN7	SYSCTI CTI Channel to Trigger 7 Enable register.	0x00000000
0x3110D0C0	SYSCTI_CTIOUTEN8	SYSCTI CTI Channel to Trigger 8 Enable register.	0x00000000
0x3110D0C4	SYSCTI_CTIOUTEN9	SYSCTI CTI Channel to Trigger 9 Enable register.	0x00000000
0x3110D0C8	SYSCTI_CTIOUTEN10	SYSCTI CTI Channel to Trigger 10 Enable register.	0x00000000
0x3110D0CC	SYSCTI_CTIOUTEN11	SYSCTI CTI Channel to Trigger 11 Enable register.	0x00000000
0x3110D0D0	SYSCTI_CTIOUTEN12	SYSCTI CTI Channel to Trigger 12 Enable register.	0x00000000
0x3110D0D4	SYSCTI_CTIOUTEN13	SYSCTI CTI Channel to Trigger 13 Enable register.	0x00000000
0x3110D130	SYSCTI_CTITRIGINSTATUS	SYSCTI CTI Trigger Input Status register.	0x00000000
0x3110D134	SYSCTI_CTITRIGOUTSTATUS	SYSCTI CTI Trigger Output Status register.	0x00000000
0x3110D138	SYSCTI_CTICHINSTATUS	SYSCTI CTI Channel Input Status register.	0x00000000
0x3110D13C	SYSCTI_CTICHOUTSTATUS	SYSCTI CTI Channel Output Status register.	0x00000000
0x3110D140	SYSCTI_CTIGATE	SYSCTI Enable CTI Channel Gate register.	0x0000000F
0x3110D144	SYSCTI_ASICCTRL	SYSCTI External Multiplexer Control register.	0x00000000
0x3110DEE4	SYSCTI_ITCHOUT	SYSCTI Integration Test Channel Output register.	0x00000000
0x3110DEE8	SYSCTI_ITTRIGOUT	SYSCTI Integration Test Trigger Output register.	0x00000000
0x3110DEF4	SYSCTI_ITCHIN	SYSCTI Integration Test Channel Input register.	0x00000000

Table A-19: ADSP-2184x SYSCTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110DEF8	SYSCTI_ITTRIGIN	SYSCTI Integration Test Trigger Input register.	0x00000000
0x3110DF00	SYSCTI_ITCTRL	SYSCTI Integration Mode Control Register.	0x00000000
0x3110DFA0	SYSCTI_CLAIMSET	SYSCTI Claim Tag Set Register.	0x0000000F
0x3110DFA4	SYSCTI_CLAIMCLR	SYSCTI Claim Tag Clear Register.	0x00000000
0x3110DFA8	SYSCTI_DEVAFF0	SYSCTI Device Affinity register 0.	0x00000000
0x3110DFAC	SYSCTI_DEVAFF1	SYSCTI Device Affinity register 1.	0x00000000
0x3110DFB8	SYSCTI_AUTHSTATUS	SYSCTI Authentication Status Register.	0x00000000
0x3110DFBC	SYSCTI_DEVARCH	SYSCTI Device Architecture Register.	0x47701A14
0x3110DFC8	SYSCTI_DEVID	SYSCTI Device Configuration Register.	0x01040E00
0x3110DFCC	SYSCTI_DEVTYPE	SYSCTI Device Type Identifier Register.	0x00000014
0x3110DFD0	SYSCTI_PIDR4	SYSCTI Peripheral Identification Register 4.	0x00000004
0x3110DFD4	SYSCTI_PIDR5	SYSCTI Peripheral Identification Register 5.	0x00000000
0x3110DFD8	SYSCTI_PIDR6	SYSCTI Peripheral Identification Register 6.	0x00000000
0x3110DFDC	SYSCTI_PIDR7	SYSCTI Peripheral Identification Register 7.	0x00000000
0x3110DFE0	SYSCTI_PIDR0	SYSCTI Peripheral Identification Register 0.	0x000000ED
0x3110DFE4	SYSCTI_PIDR1	SYSCTI Peripheral Identification Register 1.	0x000000B9
0x3110DFE8	SYSCTI_PIDR2	SYSCTI Peripheral Identification Register 2.	0x0000004B
0x3110DFEC	SYSCTI_PIDR3	SYSCTI Peripheral Identification Register 3.	0x00000000
0x3110DFF0	SYSCTI_CIDR0	SYSCTI Component Identification Register 0.	0x0000000D
0x3110DFF4	SYSCTI_CIDR1	SYSCTI Component Identification Register 1.	0x00000090
0x3110DFF8	SYSCTI_CIDR2	SYSCTI Component Identification Register 2.	0x00000005
0x3110DFFC	SYSCTI_CIDR3	SYSCTI Component Identification Register 3.	0x000000B1

Table A-20: ADSP-2184x SOC600CTI_TRACE4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C000	SOC600CTI_TRACE4_CTICONTROL	SOC600CTI_TRACE4 CTI Control register.	0x00000000
0x3110C010	SOC600CTI_TRACE4_CTIINTACK	SOC600CTI_TRACE4 CTI Interrupt Acknowledge register.	0x00000000
0x3110C014	SOC600CTI_TRACE4_CTIAPPSET	SOC600CTI_TRACE4 CTI Application Channel Set register.	0x00000000

Table A-20: ADSP-2184x SOC600CTI_TRACE4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C018	SOC600CTI_TRACE4_CTIAPPCLEAR	SOC600CTI_TRACE4 CTI Application Channel Clear register.	0x00000000
0x3110C01C	SOC600CTI_TRACE4_CTIAPPPULSE	SOC600CTI_TRACE4 CTI Application Channel Pulse register.	0x00000000
0x3110C020	SOC600CTI_TRACE4_CTIINEN0	SOC600CTI_TRACE4 CTI Trigger 0 to Channel Enable register.	0x00000000
0x3110C024	SOC600CTI_TRACE4_CTIINEN1	SOC600CTI_TRACE4 CTI Trigger 1 to Channel Enable register.	0x00000000
0x3110C028	SOC600CTI_TRACE4_CTIINEN2	SOC600CTI_TRACE4 CTI Trigger 2 to Channel Enable register.	0x00000000
0x3110C02C	SOC600CTI_TRACE4_CTIINEN3	SOC600CTI_TRACE4 CTI Trigger 3 to Channel Enable register.	0x00000000
0x3110C030	SOC600CTI_TRACE4_CTIINEN4	SOC600CTI_TRACE4 CTI Trigger 4 to Channel Enable register.	0x00000000
0x3110C034	SOC600CTI_TRACE4_CTIINEN5	SOC600CTI_TRACE4 CTI Trigger 5 to Channel Enable register.	0x00000000
0x3110C038	SOC600CTI_TRACE4_CTIINEN6	SOC600CTI_TRACE4 CTI Trigger 6 to Channel Enable register.	0x00000000
0x3110C0A0	SOC600CTI_TRACE4_CTIOUTEN0	SOC600CTI_TRACE4 CTI Channel to Trigger 0 Enable register.	0x00000000
0x3110C0A4	SOC600CTI_TRACE4_CTIOUTEN1	SOC600CTI_TRACE4 CTI Channel to Trigger 1 Enable register.	0x00000000
0x3110C0A8	SOC600CTI_TRACE4_CTIOUTEN2	SOC600CTI_TRACE4 CTI Channel to Trigger 2 Enable register.	0x00000000
0x3110C0AC	SOC600CTI_TRACE4_CTIOUTEN3	SOC600CTI_TRACE4 CTI Channel to Trigger 3 Enable register.	0x00000000
0x3110C0B0	SOC600CTI_TRACE4_CTIOUTEN4	SOC600CTI_TRACE4 CTI Channel to Trigger 4 Enable register.	0x00000000
0x3110C0B4	SOC600CTI_TRACE4_CTIOUTEN5	SOC600CTI_TRACE4 CTI Channel to Trigger 5 Enable register.	0x00000000
0x3110C130	SOC600CTI_TRACE4_CTI-TRIGINSTATUS	SOC600CTI_TRACE4 CTI Trigger Input Status register.	0x00000000
0x3110C134	SOC600CTI_TRACE4_CTI-TRIGOUTSTATUS	SOC600CTI_TRACE4 CTI Trigger Output Status register.	0x00000000
0x3110C138	SOC600CTI_TRACE4_CTI-CHINSTATUS	SOC600CTI_TRACE4 CTI Channel Input Status register.	0x00000000

Table A-20: ADSP-2184x SOC600CTI_TRACE4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C13C	SOC600CTI_TRACE4_CTICHOUTSTATUS	SOC600CTI_TRACE4 CTI Channel Output Status register.	0x00000000
0x3110C140	SOC600CTI_TRACE4_CTIGATE	SOC600CTI_TRACE4 Enable CTI Channel Gate register.	0x0000000F
0x3110C144	SOC600CTI_TRACE4_ASICCTRL	SOC600CTI_TRACE4 External Multiplexer Control register.	0x00000000
0x3110CEE4	SOC600CTI_TRACE4_ITCHOUT	SOC600CTI_TRACE4 Integration Test Channel Output register.	0x00000000
0x3110CEE8	SOC600CTI_TRACE4_ITTRIGOUT	SOC600CTI_TRACE4 Integration Test Trigger Output register.	0x00000000
0x3110CEF4	SOC600CTI_TRACE4_ITCHIN	SOC600CTI_TRACE4 Integration Test Channel Input register.	0x00000000
0x3110CEF8	SOC600CTI_TRACE4_ITTRIGIN	SOC600CTI_TRACE4 Integration Test Trigger Input register.	0x00000000
0x3110CF00	SOC600CTI_TRACE4_ITCTRL	SOC600CTI_TRACE4 Integration Mode Control Register.	0x00000000
0x3110CFA0	SOC600CTI_TRACE4_CLAIMSET	SOC600CTI_TRACE4 Claim Tag Set Register.	0x0000000F
0x3110CFA4	SOC600CTI_TRACE4_CLAIMCLR	SOC600CTI_TRACE4 Claim Tag Clear Register.	0x00000000
0x3110CFA8	SOC600CTI_TRACE4_DEVAFF0	SOC600CTI_TRACE4 Device Affinity register 0.	0x00000000
0x3110CFAC	SOC600CTI_TRACE4_DEVAFF1	SOC600CTI_TRACE4 Device Affinity register 1.	0x00000000
0x3110CFB8	SOC600CTI_TRACE4_AUTHSTATUS	SOC600CTI_TRACE4 Authentication Status Register.	0x00000000
0x3110CFBC	SOC600CTI_TRACE4_DEVARCH	SOC600CTI_TRACE4 Device Architecture Register.	0x47701A14
0x3110CFC8	SOC600CTI_TRACE4_DEVID	SOC600CTI_TRACE4 Device Configuration Register.	0x01040700
0x3110CFCC	SOC600CTI_TRACE4_DEVTYPE	SOC600CTI_TRACE4 Device Type Identifier Register.	0x00000014
0x3110CFD0	SOC600CTI_TRACE4_PIDR4	SOC600CTI_TRACE4 Peripheral Identification Register 4.	0x00000004
0x3110CFD4	SOC600CTI_TRACE4_PIDR5	SOC600CTI_TRACE4 Peripheral Identification Register 5.	0x00000000

Table A-20: ADSP-2184x SOC600CTI_TRACE4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110CFD8	SOC600CTI_TRACE4_PIDR6	SOC600CTI_TRACE4 Peripheral Identification Register 6.	0x00000000
0x3110CFDC	SOC600CTI_TRACE4_PIDR7	SOC600CTI_TRACE4 Peripheral Identification Register 7.	0x00000000
0x3110CFE0	SOC600CTI_TRACE4_PIDR0	SOC600CTI_TRACE4 Peripheral Identification Register 0.	0x000000ED
0x3110CFE4	SOC600CTI_TRACE4_PIDR1	SOC600CTI_TRACE4 Peripheral Identification Register 1.	0x000000B9
0x3110CFE8	SOC600CTI_TRACE4_PIDR2	SOC600CTI_TRACE4 Peripheral Identification Register 2.	0x0000004B
0x3110CFEC	SOC600CTI_TRACE4_PIDR3	SOC600CTI_TRACE4 Peripheral Identification Register 3.	0x00000000
0x3110CFF0	SOC600CTI_TRACE4_CIDR0	SOC600CTI_TRACE4 Component Identification Register 0.	0x0000000D
0x3110CFF4	SOC600CTI_TRACE4_CIDR1	SOC600CTI_TRACE4 Component Identification Register 1.	0x00000090
0x3110CFF8	SOC600CTI_TRACE4_CIDR2	SOC600CTI_TRACE4 Component Identification Register 2.	0x00000005
0x3110CFFC	SOC600CTI_TRACE4_CIDR3	SOC600CTI_TRACE4 Component Identification Register 3.	0x000000B1

Table A-21: ADSP-2184x CSETF0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31109004	CSETF0_RSZ	CSETF0 RAM Size register.	0x00001000
0x3110900C	CSETF0_STS	CSETF0 Status register.	0x00000018
0x31109010	CSETF0_RRD	CSETF0 RAM Read Data register.	0x00000000
0x31109014	CSETF0_RRP	CSETF0 RAM Read Pointer register.	0x00000000
0x31109018	CSETF0_RWP	CSETF0 RAM Write Pointer register.	0x00000000
0x3110901C	CSETF0_TRG	CSETF0 Trigger Counter register.	0x00000000
0x31109020	CSETF0_CTL	CSETF0 Control Register.	0x00000000
0x31109024	CSETF0_RWD	CSETF0 RAM Write Data register.	0x00000000
0x31109028	CSETF0_MODE	CSETF0 Mode register.	0x00000000
0x3110902C	CSETF0_LBUFLEVEL	CSETF0 Latched Buffer Fill Level.	0x00000000
0x31109030	CSETF0_CBUFLEVEL	CSETF0 Current Buffer Fill Level.	0x00000000

Table A-21: ADSP-2184x CSETF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31109034	CSETF0_BUFWM	CSETF0 Buffer Level Water Mark.	0x00000000
0x31109300	CSETF0_FFSR	CSETF0 Formatter and Flush Status Register.	0x00000000
0x31109304	CSETF0_FFCR	CSETF0 Formatter and Flush Control Register.	0x00000000
0x31109308	CSETF0_PSCR	CSETF0 Periodic Synchronization Counter Register.	0x0000000A
0x31109ED0	CSETF0_ITATBMDATA0	CSETF0 Integration Test ATB Transmitter Data 0 register.	0x00000000
0x31109ED4	CSETF0_ITATBMCTR2	CSETF0 Integration Test ATB Transmitter Control 2 register.	0x00000000
0x31109ED8	CSETF0_ITATBMCTR1	CSETF0 Integration Test ATB Transmitter Control 1 register.	0x00000000
0x31109EDC	CSETF0_ITATBMCTR0	CSETF0 Integration Test ATB Transmitter Control 0 register.	0x00000000
0x31109EE0	CSETF0_ITEVTINTR	CSETF0 Integration Test Event and Interrupt Control Register.	0x00000000
0x31109EE8	CSETF0_ITTRFLIN	CSETF0 Integration Test Trigger In and Flush In register.	0x00000000
0x31109EEC	CSETF0_ITATBDATA0	CSETF0 Integration Test ATB Data 0 Register.	0x00000000
0x31109EF0	CSETF0_ITATBCTR2	CSETF0 Integration Test ATB Control 2 Register.	0x00000000
0x31109EF4	CSETF0_ITATBCTR1	CSETF0 Integration Test ATB Control 1 Register.	0x00000000
0x31109EF8	CSETF0_ITATBCTR0	CSETF0 Integration Test ATB Control 0 Register.	0x00000000
0x31109F00	CSETF0_ITCTRL	CSETF0 Integration Mode Control Register.	0x00000000
0x31109FA0	CSETF0_CLAIMSET	CSETF0 Claim Tag Set Register.	0x0000000F
0x31109FA4	CSETF0_CLAIMCLR	CSETF0 Claim Tag Clear Register.	0x00000000
0x31109FB8	CSETF0_AUTHSTATUS	CSETF0 Authentication Status Register.	0x00000000
0x31109FC4	CSETF0_DEVID1	CSETF0 Device Configuration Register 1.	0x00000001
0x31109FC8	CSETF0_DEVID	CSETF0 Device Configuration Register.	0x00000380
0x31109FCC	CSETF0_DEVTYPE	CSETF0 Device Type Identifier Register.	0x00000032
0x31109FD0	CSETF0_PIDR4	CSETF0 Peripheral Identification Register 4.	0x00000004
0x31109FD4	CSETF0_PIDR5	CSETF0 Peripheral Identification Register 5.	0x00000000
0x31109FD8	CSETF0_PIDR6	CSETF0 Peripheral Identification Register 6.	0x00000000
0x31109FDC	CSETF0_PIDR7	CSETF0 Peripheral Identification Register 7.	0x00000000
0x31109FE0	CSETF0_PIDR0	CSETF0 Peripheral Identification Register 0.	0x000000EA
0x31109FE4	CSETF0_PIDR1	CSETF0 Peripheral Identification Register 1.	0x000000B9
0x31109FE8	CSETF0_PIDR2	CSETF0 Peripheral Identification Register 2.	0x0000008B

Table A-21: ADSP-2184x CSETF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31109FEC	CSETF0_PIDR3	CSETF0 Peripheral Identification Register 3.	0x00000000
0x31109FF0	CSETF0_CIDR0	CSETF0 Component Identification Register 0.	0x0000000D
0x31109FF4	CSETF0_CIDR1	CSETF0 Component Identification Register 1.	0x00000090
0x31109FF8	CSETF0_CIDR2	CSETF0 Component Identification Register 2.	0x00000005
0x31109FFC	CSETF0_CIDR3	CSETF0 Component Identification Register 3.	0x000000B1

Table A-22: ADSP-2184x CSETR1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110A004	CSETR1_RSZ	CSETR1 RAM Size register.	0x00000000
0x3110A00C	CSETR1_STS	CSETR1 Status register.	0x00000018
0x3110A010	CSETR1_RRD	CSETR1 RAM Read Data register.	0x00000000
0x3110A014	CSETR1_RRP	CSETR1 RAM Read Pointer register.	0x00000000
0x3110A018	CSETR1_RWP	CSETR1 RAM Write Pointer register.	0x00000000
0x3110A01C	CSETR1_TRG	CSETR1 Trigger Counter register.	0x00000000
0x3110A020	CSETR1_CTL	CSETR1 Control Register.	0x00000000
0x3110A024	CSETR1_RWD	CSETR1 RAM Write Data register.	0x00000000
0x3110A028	CSETR1_MODE	CSETR1 Mode register.	0x00000000
0x3110A02C	CSETR1_LBUFLEVEL	CSETR1 Latched Buffer Fill Level.	0x00000000
0x3110A030	CSETR1_CBUFLEVEL	CSETR1 Current Buffer Fill Level.	0x00000000
0x3110A034	CSETR1_BUFWM	CSETR1 Buffer Level Water Mark.	0x00000000
0x3110A038	CSETR1_RRPHI	CSETR1 RAM Read Pointer High register.	0x00000000
0x3110A03C	CSETR1_RWPHI	CSETR1 RAM Write Pointer High register.	0x00000000
0x3110A110	CSETR1_AXICTL	CSETR1 AXI Control Register.	0x00000000
0x3110A118	CSETR1_DBALO	CSETR1 Data Buffer Address Low register.	0x00000000
0x3110A11C	CSETR1_DBAHI	CSETR1 Data Buffer Address HIGH register.	0x00000000
0x3110A120	CSETR1_RURP	CSETR1 RAM Update Read Pointer register.	0x00000000
0x3110A300	CSETR1_FFSR	CSETR1 Formatter and Flush Status Register.	0x00000000
0x3110A304	CSETR1_FFCR	CSETR1 Formatter and Flush Control Register.	0x00000000
0x3110A308	CSETR1_PSCR	CSETR1 Periodic Synchronization Counter Register.	0x0000000A
0x3110AED0	CSETR1_AXICTL1	CSETR1 AXI Control Register 1.	0x00000000

Table A-22: ADSP-2184x CSETR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110AEE0	CSETR1_ITEVTINTR	CSETR1 Integration Test Event and Interrupt Control Register.	0x00000000
0x3110AEE8	CSETR1_ITTRFLIN	CSETR1 Integration Test Trigger In and Flush In register.	0x00000000
0x3110AEEC	CSETR1_ITATBDATA0	CSETR1 Integration Test ATB Data 0 Register.	0x00000000
0x3110AEF0	CSETR1_ITATBCTR2	CSETR1 Integration Test ATB Control 2 Register.	0x00000000
0x3110AEF4	CSETR1_ITATBCTR1	CSETR1 Integration Test ATB Control 1 Register.	0x00000000
0x3110AEF8	CSETR1_ITATBCTR0	CSETR1 Integration Test ATB Control 0 Register.	0x00000000
0x3110AF00	CSETR1_ITCTRL	CSETR1 Integration Mode Control Register.	0x00000000
0x3110AFA0	CSETR1_CLAIMSET	CSETR1 Claim Tag Set Register.	0x0000000F
0x3110AFA4	CSETR1_CLAIMCLR	CSETR1 Claim Tag Clear Register.	0x00000000
0x3110AFB8	CSETR1_AUTHSTATUS	CSETR1 Authentication Status Register.	0x00000000
0x3110AFC4	CSETR1_DEVID1	CSETR1 Device Configuration Register 1.	0x00000001
0x3110AFC8	CSETR1_DEVID	CSETR1 Device Configuration Register.	0x03412A40
0x3110AFCC	CSETR1_DEVTYPE	CSETR1 Device Type Identifier Register.	0x00000021
0x3110AFD0	CSETR1_PIDR4	CSETR1 Peripheral Identification Register 4.	0x00000004
0x3110AFD4	CSETR1_PIDR5	CSETR1 Peripheral Identification Register 5.	0x00000000
0x3110AFD8	CSETR1_PIDR6	CSETR1 Peripheral Identification Register 6.	0x00000000
0x3110AFDC	CSETR1_PIDR7	CSETR1 Peripheral Identification Register 7.	0x00000000
0x3110AFE0	CSETR1_PIDR0	CSETR1 Peripheral Identification Register 0.	0x000000E8
0x3110AFE4	CSETR1_PIDR1	CSETR1 Peripheral Identification Register 1.	0x000000B9
0x3110AFE8	CSETR1_PIDR2	CSETR1 Peripheral Identification Register 2.	0x0000008B
0x3110AFEC	CSETR1_PIDR3	CSETR1 Peripheral Identification Register 3.	0x00000000
0x3110AFF0	CSETR1_CIDR0	CSETR1 Component Identification Register 0.	0x0000000D
0x3110AFF4	CSETR1_CIDR1	CSETR1 Component Identification Register 1.	0x00000090
0x3110AFF8	CSETR1_CIDR2	CSETR1 Component Identification Register 2.	0x00000005
0x3110AFFC	CSETR1_CIDR3	CSETR1 Component Identification Register 3.	0x000000B1

Table A-23: ADSP-2184x TRACE0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110B000	TRACE0_SSPSR	TRACE0 Supported Port Size Register.	0x00000000
0x3110B004	TRACE0_CSPSR	TRACE0 Current Port Size Register.	0x00000001

Table A-23: ADSP-2184x TRACE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110B100	TRACE0_STMR	TRACE0 Supported Trigger Modes Register.	0x0000011F
0x3110B104	TRACE0_TCVR	TRACE0 Trigger Counter Value Register.	0x00000000
0x3110B108	TRACE0_TCMR	TRACE0 Trigger Counter Multiplier Register.	0x00000000
0x3110B200	TRACE0_STPMR	TRACE0 Supported Test Patterns/Modes Register.	0x0003000F
0x3110B204	TRACE0_CTPMR	TRACE0 Current Test Patterns/Modes Register.	0x00000000
0x3110B208	TRACE0_TPRCR	TRACE0 Test Pattern Repeat Counter Register.	0x00000000
0x3110B300	TRACE0_FFSR	TRACE0 Formatter and Flush Status Register.	0x00000002
0x3110B304	TRACE0_FFCR	TRACE0 Formatter and Flush Control Register.	0x00001000
0x3110B308	TRACE0_FSCR	TRACE0 Formatter Synchronization Count Register.	0x00000040
0x3110B400	TRACE0_EXTCTLIN	TRACE0 External Control Port In Register.	0x00000000
0x3110B404	TRACE0_EXTCTLOUT	TRACE0 External Control Port Out Register.	0x00000000
0x3110BEE8	TRACE0_ITTRFLIN	TRACE0 Integration Test Trigger In and Flush In Register.	0x00000000
0x3110BEEC	TRACE0_ITATBDATA0	TRACE0 Integration Test ATB Data Register 0.	0x00000000
0x3110BEF0	TRACE0_ITATBCTR2	TRACE0 Integration Test ATB Control Register 2.	0x00000000
0x3110BEF4	TRACE0_ITATBCTR1	TRACE0 Integration Test ATB Control Register 1.	0x00000000
0x3110BEF8	TRACE0_ITATBCTR0	TRACE0 Integration Test ATB Control Register 0.	0x00000000
0x3110BEFC	TRACE0_ITOUTCTR	TRACE0 Integration Test Output Control Register.	0x00000000
0x3110BF00	TRACE0_ITCTRL	TRACE0 Integration Mode Control Register.	0x00000000
0x3110BFA0	TRACE0_CLAIMSET	TRACE0 Claim Tag Set Register.	0x0000000F
0x3110BFA4	TRACE0_CLAIMCLR	TRACE0 Claim Tag Clear Register.	0x00000000
0x3110BFB8	TRACE0_AUTHSTATUS	TRACE0 Authentication Status Register.	0x00000000
0x3110BFBC	TRACE0_DEVARCH	TRACE0 Device Architecture Register.	0x00000000
0x3110BFC8	TRACE0_DEVID	TRACE0 Device Configuration Register.	0x00010020
0x3110BFCC	TRACE0_DEVTYPE	TRACE0 Device Type Identifier Register.	0x00000011
0x3110BFD0	TRACE0_PIDR4	TRACE0 Peripheral Identification Register 4.	0x00000004
0x3110BFD4	TRACE0_PIDR5	TRACE0 Peripheral Identification Register 5.	0x00000000
0x3110BFD8	TRACE0_PIDR6	TRACE0 Peripheral Identification Register 6.	0x00000000
0x3110BFDC	TRACE0_PIDR7	TRACE0 Peripheral Identification Register 7.	0x00000000
0x3110BFE0	TRACE0_PIDR0	TRACE0 Peripheral Identification Register 0.	0x000000E7
0x3110BFE4	TRACE0_PIDR1	TRACE0 Peripheral Identification Register 1.	0x000000B9

Table A-23: ADSP-2184x TRACE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110BFE8	TRACE0_PIDR2	TRACE0 Peripheral Identification Register 2.	0x0000003B
0x3110BFEC	TRACE0_PIDR3	TRACE0 Peripheral Identification Register 3.	0x00000000
0x3110BFF0	TRACE0_CIDR0	TRACE0 Component Identification Register 0.	0x0000000D
0x3110BFF4	TRACE0_CIDR1	TRACE0 Component Identification Register 1.	0x00000090
0x3110BFF8	TRACE0_CIDR2	TRACE0 Component Identification Register 2.	0x00000005
0x3110BFFC	TRACE0_CIDR3	TRACE0 Component Identification Register 3.	0x000000B1

Table A-24: ADSP-2184x CSTSGENWR0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110E000	CSTSGENWR0_CNTCR	CSTSGENWR0 Counter Control Register.	0x00000000
0x3110E000	CSS600_TSGEN_EHP2_CNTCR	CSS600_TSGEN_EHP2 Counter Control Register.	0x00000000
0x3110E004	CSTSGENWR0_CNTSR	CSTSGENWR0 Counter Status Register.	0x00000000
0x3110E004	CSS600_TSGEN_EHP2_CNTSR	CSS600_TSGEN_EHP2 Counter Status Register.	0x00000000
0x3110E008	CSTSGENWR0_CNTCVL	CSTSGENWR0 Current value of Counter[31:0].	0x00000000
0x3110E008	CSS600_TSGEN_EHP2_CNTCVL	CSS600_TSGEN_EHP2 Current value of Counter[31:0].	0x00000000
0x3110E00C	CSTSGENWR0_CNTCVU	CSTSGENWR0 Current value of Counter[63:32].	0x00000000
0x3110E00C	CSS600_TSGEN_EHP2_CNTCVU	CSS600_TSGEN_EHP2 Current value of Counter[63:32].	0x00000000
0x3110E020	CSTSGENWR0_CNTFID0	CSTSGENWR0 Base Frequency ID register.	0x00000000
0x3110E020	CSS600_TSGEN_EHP2_CNTFID0	CSS600_TSGEN_EHP2 Base Frequency ID register.	0x00000000
0x3110EEF8	CSTSGENWR0_ITSTAT	CSTSGENWR0 Integration Test Status Register.	0x00000000
0x3110EEF8	CSS600_TSGEN_EHP2_ITSTAT	CSS600_TSGEN_EHP2 Integration Test Status Register.	0x00000000
0x3110EF00	CSTSGENWR0_ITCTRL	CSTSGENWR0 Integration Mode Control Register.	0x00000000
0x3110EF00	CSS600_TSGEN_EHP2_ITCTRL	CSS600_TSGEN_EHP2 Integration Mode Control Register.	0x00000000
0x3110EFD0	CSTSGENWR0_PIDR4	CSTSGENWR0 Peripheral Identification Register 4.	0x00000004
0x3110EFD0	CSS600_TSGEN_EHP2_PIDR4	CSS600_TSGEN_EHP2 Peripheral Identification Register 4.	0x00000004

Table A-24: ADSP-2184x CSTSGENWR0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110EFD4	CSTSGENWR0_PIDR5	CSTSGENWR0 Peripheral Identification Register 5.	0x00000000
0x3110EFD4	CSS600_TSGEN_EHP2_PIDR5	CSS600_TSGEN_EHP2 Peripheral Identification Register 5.	0x00000000
0x3110EFD8	CSTSGENWR0_PIDR6	CSTSGENWR0 Peripheral Identification Register 6.	0x00000000
0x3110EFD8	CSS600_TSGEN_EHP2_PIDR6	CSS600_TSGEN_EHP2 Peripheral Identification Register 6.	0x00000000
0x3110EFDC	CSTSGENWR0_PIDR7	CSTSGENWR0 Peripheral Identification Register 7.	0x00000000
0x3110EFDC	CSS600_TSGEN_EHP2_PIDR7	CSS600_TSGEN_EHP2 Peripheral Identification Register 7.	0x00000000
0x3110EFE0	CSTSGENWR0_PIDR0	CSTSGENWR0 Peripheral Identification Register 0.	0x00000093
0x3110EFE0	CSS600_TSGEN_EHP2_PIDR0	CSS600_TSGEN_EHP2 Peripheral Identification Register 0.	0x00000093
0x3110EFE4	CSTSGENWR0_PIDR1	CSTSGENWR0 Peripheral Identification Register 1.	0x000000B1
0x3110EFE4	CSS600_TSGEN_EHP2_PIDR1	CSS600_TSGEN_EHP2 Peripheral Identification Register 1.	0x000000B1
0x3110EFE8	CSTSGENWR0_PIDR2	CSTSGENWR0 Peripheral Identification Register 2.	0x0000000B
0x3110EFE8	CSS600_TSGEN_EHP2_PIDR2	CSS600_TSGEN_EHP2 Peripheral Identification Register 2.	0x0000000B
0x3110EFEC	CSTSGENWR0_PIDR3	CSTSGENWR0 Peripheral Identification Register 3.	0x00000000
0x3110EFEC	CSS600_TSGEN_EHP2_PIDR3	CSS600_TSGEN_EHP2 Peripheral Identification Register 3.	0x00000000
0x3110EFF0	CSTSGENWR0_CIDR0	CSTSGENWR0 Component Identification Register 0.	0x0000000D
0x3110EFF0	CSS600_TSGEN_EHP2_CIDR0	CSS600_TSGEN_EHP2 Component Identification Register 0.	0x0000000D
0x3110EFF4	CSTSGENWR0_CIDR1	CSTSGENWR0 Component Identification Register 1.	0x000000F0
0x3110EFF4	CSS600_TSGEN_EHP2_CIDR1	CSS600_TSGEN_EHP2 Component Identification Register 1.	0x000000F0
0x3110EFF8	CSTSGENWR0_CIDR2	CSTSGENWR0 Component Identification Register 2.	0x00000005
0x3110EFF8	CSS600_TSGEN_EHP2_CIDR2	CSS600_TSGEN_EHP2 Component Identification Register 2.	0x00000005
0x3110EFFC	CSTSGENWR0_CIDR3	CSTSGENWR0 Component Identification Register 3.	0x000000B1
0x3110EFFC	CSS600_TSGEN_EHP2_CIDR3	CSS600_TSGEN_EHP2 Component Identification Register 3.	0x000000B1
0x31149000	A55_SYSTMRWR_CNTCR	A55_SYSTMRWR Counter Control Register.	0x00000000

Table A-24: ADSP-2184x CSTSGENWR0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31149004	A55_SYSTMRWR_CNTSR	A55_SYSTMRWR Counter Status Register.	0x00000000
0x31149008	A55_SYSTMRWR_CNTCVL	A55_SYSTMRWR Current value of Counter[31:0].	0x00000000
0x3114900C	A55_SYSTMRWR_CNTCVU	A55_SYSTMRWR Current value of Counter[63:32].	0x00000000
0x31149020	A55_SYSTMRWR_CNTFID0	A55_SYSTMRWR Base Frequency ID register.	0x00000000
0x31149EF8	A55_SYSTMRWR_ITSTAT	A55_SYSTMRWR Integration Test Status Register.	0x00000000
0x31149F00	A55_SYSTMRWR_ITCTRL	A55_SYSTMRWR Integration Mode Control Register.	0x00000000
0x31149FD0	A55_SYSTMRWR_PIDR4	A55_SYSTMRWR Peripheral Identification Register 4.	0x00000004
0x31149FD4	A55_SYSTMRWR_PIDR5	A55_SYSTMRWR Peripheral Identification Register 5.	0x00000000
0x31149FD8	A55_SYSTMRWR_PIDR6	A55_SYSTMRWR Peripheral Identification Register 6.	0x00000000
0x31149FDC	A55_SYSTMRWR_PIDR7	A55_SYSTMRWR Peripheral Identification Register 7.	0x00000000
0x31149FE0	A55_SYSTMRWR_PIDR0	A55_SYSTMRWR Peripheral Identification Register 0.	0x00000093
0x31149FE4	A55_SYSTMRWR_PIDR1	A55_SYSTMRWR Peripheral Identification Register 1.	0x000000B1
0x31149FE8	A55_SYSTMRWR_PIDR2	A55_SYSTMRWR Peripheral Identification Register 2.	0x0000000B
0x31149FEC	A55_SYSTMRWR_PIDR3	A55_SYSTMRWR Peripheral Identification Register 3.	0x00000000
0x31149FF0	A55_SYSTMRWR_CIDR0	A55_SYSTMRWR Component Identification Register 0.	0x0000000D
0x31149FF4	A55_SYSTMRWR_CIDR1	A55_SYSTMRWR Component Identification Register 1.	0x000000F0
0x31149FF8	A55_SYSTMRWR_CIDR2	A55_SYSTMRWR Component Identification Register 2.	0x00000005
0x31149FFC	A55_SYSTMRWR_CIDR3	A55_SYSTMRWR Component Identification Register 3.	0x000000B1
0x3114A000	A55_SYSTMRRD_CNTCVLREAD	A55_SYSTMRRD Current value of Counter[31:0].	0x00000000
0x3114A004	A55_SYSTMRRD_CNTCVUREAD	A55_SYSTMRRD Current value of Counter[63:32].	0x00000000
0x3114AFD0	A55_SYSTMRRD_PIDR4	A55_SYSTMRRD Peripheral Identification Register 4.	0x00000004
0x3114AFD4	A55_SYSTMRRD_PIDR5	A55_SYSTMRRD Peripheral Identification Register 5.	0x00000000
0x3114AFD8	A55_SYSTMRRD_PIDR6	A55_SYSTMRRD Peripheral Identification Register 6.	0x00000000
0x3114AFDC	A55_SYSTMRRD_PIDR7	A55_SYSTMRRD Peripheral Identification Register 7.	0x00000000
0x3114AFE0	A55_SYSTMRRD_PIDR0	A55_SYSTMRRD Peripheral Identification Register 0.	0x00000093
0x3114AFE4	A55_SYSTMRRD_PIDR1	A55_SYSTMRRD Peripheral Identification Register 1.	0x000000B1
0x3114AFE8	A55_SYSTMRRD_PIDR2	A55_SYSTMRRD Peripheral Identification Register 2.	0x0000000B
0x3114AFEC	A55_SYSTMRRD_PIDR3	A55_SYSTMRRD Peripheral Identification Register 3.	0x00000000

Table A-24: ADSP-2184x CSTSGENWR0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114AFF0	A55_SYSTMRRD_CIDR0	A55_SYSTMRRD Component Identification Register 0.	0x0000000D
0x3114AFF4	A55_SYSTMRRD_CIDR1	A55_SYSTMRRD Component Identification Register 1.	0x000000F0
0x3114AFF8	A55_SYSTMRRD_CIDR2	A55_SYSTMRRD Component Identification Register 2.	0x00000005
0x3114AFFC	A55_SYSTMRRD_CIDR3	A55_SYSTMRRD Component Identification Register 3.	0x000000B1

Table A-25: ADSP-2184x DAI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9000	DAI0_EXTD_CLK0	DAI0 Extended Clock Routing Control Register 0	0x00000000
0x310C9000	DAI_EXTD_CLK0	DAI Extended Clock Routing Control Register 0	0x00000000
0x310C9004	DAI0_EXTD_CLK1	DAI0 Extended Clock Routing Control Register 1	0x00000000
0x310C9004	DAI_EXTD_CLK1	DAI Extended Clock Routing Control Register 1	0x00000000
0x310C9008	DAI0_EXTD_CLK2	DAI0 Extended Clock Routing Control Register 2	0x00000000
0x310C9008	DAI_EXTD_CLK2	DAI Extended Clock Routing Control Register 2	0x00000000
0x310C900C	DAI0_EXTD_CLK3	DAI0 Extended Clock Routing Control Register 3	0x00000000
0x310C900C	DAI_EXTD_CLK3	DAI Extended Clock Routing Control Register 3	0x00000000
0x310C9010	DAI0_EXTD_CLK4	DAI0 Extended Clock Routing Control Register 4	0x00000000
0x310C9010	DAI_EXTD_CLK4	DAI Extended Clock Routing Control Register 4	0x00000000
0x310C9014	DAI0_EXTD_CLK5	DAI0 Extended Clock Routing Control Register 5	0x00000000
0x310C9014	DAI_EXTD_CLK5	DAI Extended Clock Routing Control Register 5	0x00000000
0x310C9018	DAI0_EXTD_DAT0	DAI0 Extended Serial Data Routing Control Register 0	0x00000000
0x310C9018	DAI_EXTD_DAT0	DAI Extended Serial Data Routing Control Register 0	0x00000000
0x310C901C	DAI0_EXTD_DAT1	DAI0 Extended Serial Data Routing Control Register 1	0x00000000
0x310C901C	DAI_EXTD_DAT1	DAI Extended Serial Data Routing Control Register 1	0x00000000
0x310C9020	DAI0_EXTD_DAT2	DAI0 Extended Serial Data Routing Control Register 2	0x00000000
0x310C9020	DAI_EXTD_DAT2	DAI Extended Serial Data Routing Control Register 2	0x00000000
0x310C9024	DAI0_EXTD_DAT3	DAI0 Extended Serial Data Routing Control Register 3	0x00000000
0x310C9024	DAI_EXTD_DAT3	DAI Extended Serial Data Routing Control Register 3	0x00000000
0x310C9028	DAI0_EXTD_DAT4	DAI0 Extended Serial Data Routing Control Register 4	0x00000000
0x310C9028	DAI_EXTD_DAT4	DAI Extended Serial Data Routing Control Register 4	0x00000000
0x310C902C	DAI0_EXTD_DAT5	DAI0 Extended Serial Data Routing Control Register 5	0x00000000

Table A-25: ADSP-2184x DAI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C902C	DAI_EXTD_DAT5	DAI Extended Serial Data Routing Control Register 5	0x00000000
0x310C9030	DAI0_EXTD_DAT6	DAI0 Extended Serial Data Routing Control Register 6	0x00000000
0x310C9030	DAI_EXTD_DAT6	DAI Extended Serial Data Routing Control Register 6	0x00000000
0x310C9034	DAI0_EXTD_FS0	DAI0 Extended Frame Sync Routing Control Register 0	0x00000000
0x310C9034	DAI_EXTD_FS0	DAI Extended Frame Sync Routing Control Register 0	0x00000000
0x310C9038	DAI0_EXTD_FS1	DAI0 Extended Frame Sync Routing Control Register 1	0x00000000
0x310C9038	DAI_EXTD_FS1	DAI Extended Frame Sync Routing Control Register 1	0x00000000
0x310C903C	DAI0_EXTD_FS2	DAI0 Extended Frame Sync Routing Control Register 2	0x00000000
0x310C903C	DAI_EXTD_FS2	DAI Extended Frame Sync Routing Control Register 2	0x00000000
0x310C9044	DAI0_EXTD_FS4	DAI0 Extended Frame Sync Routing Control Register 4	0x00000000
0x310C9044	DAI_EXTD_FS4	DAI Extended Frame Sync Routing Control Register 4	0x00000000
0x310C9048	DAI0_EXTD_PIN0	DAI0 Extended Pin Buffer Assignment Register 0	0x00000000
0x310C9048	DAI_EXTD_PIN0	DAI Extended Pin Buffer Assignment Register 0	0x00000000
0x310C904C	DAI0_EXTD_PIN1	DAI0 Extended Pin Buffer Assignment Register 1	0x00000000
0x310C904C	DAI_EXTD_PIN1	DAI Extended Pin Buffer Assignment Register 1	0x00000000
0x310C9050	DAI0_EXTD_PIN2	DAI0 Extended Pin Buffer Assignment Register 2	0x00000000
0x310C9050	DAI_EXTD_PIN2	DAI Extended Pin Buffer Assignment Register 2	0x00000000
0x310C9054	DAI0_EXTD_PIN3	DAI0 Extended Pin Buffer Assignment Register 3	0x00000000
0x310C9054	DAI_EXTD_PIN3	DAI Extended Pin Buffer Assignment Register 3	0x00000000
0x310C9058	DAI0_EXTD_PIN4	DAI0 Extended Pin Buffer Assignment Register 4	0x00000000
0x310C9058	DAI_EXTD_PIN4	DAI Extended Pin Buffer Assignment Register 4	0x00000000
0x310C905C	DAI0_EXTD_MISC0	DAI0 Extended Miscellaneous Control Register 0	0x00000000
0x310C905C	DAI_EXTD_MISC0	DAI Extended Miscellaneous Control Register 0	0x00000000
0x310C9060	DAI0_EXTD_MISC1	DAI0 Extended Miscellaneous Control Register 1	0x00000000
0x310C9060	DAI_EXTD_MISC1	DAI Extended Miscellaneous Control Register 1	0x00000000
0x310C9064	DAI0_EXTD_MISC2	DAI0 Extended Miscellaneous Control Register 2	0x00000000
0x310C9064	DAI_EXTD_MISC2	DAI Extended Miscellaneous Control Register 2	0x00000000
0x310C9068	DAI0_EXTD_PBEN0	DAI0 Extended Pin Buffer Enable Register 0	0x00000000
0x310C9068	DAI_EXTD_PBEN0	DAI Extended Pin Buffer Enable Register 0	0x00000000
0x310C906C	DAI0_EXTD_PBEN1	DAI0 Extended Pin Buffer Enable Register 1	0x00000000
0x310C906C	DAI_EXTD_PBEN1	DAI Extended Pin Buffer Enable Register 1	0x00000000

Table A-25: ADSP-2184x DAI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9070	DAI0_EXTD_PBEN2	DAI0 Extended Pin Buffer Enable Register 2	0x00000000
0x310C9070	DAI_EXTD_PBEN2	DAI Extended Pin Buffer Enable Register 2	0x00000000
0x310C9074	DAI0_EXTD_PBEN3	DAI0 Extended Pin Buffer Enable Register 3	0x00000000
0x310C9074	DAI_EXTD_PBEN3	DAI Extended Pin Buffer Enable Register 3	0x00000000
0x310C9078	DAI0_EXTD_CLK6	DAI0 Extended Clock Routing Control Register 6	0x00000000
0x310C9078	DAI_EXTD_CLK6	DAI Extended Clock Routing Control Register 6	0x00000000
0x310C907C	DAI0_EXTD_CLK7	DAI0 Extended Clock Routing Control Register 7	0x00000000
0x310C907C	DAI_EXTD_CLK7	DAI Extended Clock Routing Control Register 7	0x00000000
0x310C9080	DAI0_EXTD_DAT7	DAI0 Extended Serial Data Routing Control Register 7	0x00000000
0x310C9080	DAI_EXTD_DAT7	DAI Extended Serial Data Routing Control Register 7	0x00000000
0x310C9084	DAI0_EXTD_DAT8	DAI0 Extended Serial Data Routing Control Register 8	0x00000000
0x310C9084	DAI_EXTD_DAT8	DAI Extended Serial Data Routing Control Register 8	0x00000000
0x310C9088	DAI0_EXTD_FS5	DAI0 Extended Frame Sync Routing Control Register 5	0x00000000
0x310C9088	DAI_EXTD_FS5	DAI Extended Frame Sync Routing Control Register 5	0x00000000
0x310C908C	DAI0_EXTD_FS6	DAI0 Extended Frame Sync Routing Control Register 6	0x00000000
0x310C908C	DAI_EXTD_FS6	DAI Extended Frame Sync Routing Control Register 6	0x00000000
0x310C909C	DAI0_DAI_MISC_CTL	DAI0 DAI MISC Control Register	0x00000000
0x310C909C	DAI_DAI_MISC_CTL	DAI DAI MISC Control Register	0x00000000
0x310C90C0	DAI0_CLK0	DAI0 Clock Routing Control Register 0	0x252630C2
0x310C90C0	DAI_CLK0	DAI Clock Routing Control Register 0	0x252630C2
0x310C90C4	DAI0_CLK1	DAI0 Clock Routing Control Register 1	0x3DEF7BDE
0x310C90C4	DAI_CLK1	DAI Clock Routing Control Register 1	0x3DEF7BDE
0x310C90C8	DAI0_CLK2	DAI0 Clock Routing Control Register 2	0x01EF7BDE
0x310C90C8	DAI_CLK2	DAI Clock Routing Control Register 2	0x01EF7BDE
0x310C90CC	DAI0_CLK3	DAI0 Clock Routing Control Register 3	0x3C000000
0x310C90CC	DAI_CLK3	DAI Clock Routing Control Register 3	0x3C000000
0x310C90D0	DAI0_CLK4	DAI0 Clock Routing Control Register 4	0x3DEF03DE
0x310C90D0	DAI_CLK4	DAI Clock Routing Control Register 4	0x3DEF03DE
0x310C90D4	DAI0_CLK5	DAI0 Clock Routing Control Register 5	0x3DEF7BDE
0x310C90D4	DAI_CLK5	DAI Clock Routing Control Register 5	0x3DEF7BDE
0x310C90D8	DAI0_CLK6	DAI0 Clock Routing Control Register 6	0x3DEF7BDE

Table A-25: ADSP-2184x DAI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C90D8	DAI_CLK6	DAI Clock Routing Control Register 6	0x3DEF7BDE
0x310C90DC	DAI0_CLK7	DAI0 Clock Routing Control Register 7	0x000003DE
0x310C90DC	DAI_CLK7	DAI Clock Routing Control Register 7	0x000003DE
0x310C9100	DAI0_DAT0	DAI0 Serial Data Routing Control Register 0	0x08144040
0x310C9100	DAI_DAT0	DAI Serial Data Routing Control Register 0	0x08144040
0x310C9104	DAI0_DAT1	DAI0 Serial Data Routing Control Register 1	0x0F38B289
0x310C9104	DAI_DAT1	DAI Serial Data Routing Control Register 1	0x0F38B289
0x310C9108	DAI0_DAT2	DAI0 Serial Data Routing Control Register 2	0x00000450
0x310C9108	DAI_DAT2	DAI Serial Data Routing Control Register 2	0x00000450
0x310C910C	DAI0_DAT3	DAI0 Serial Data Routing Control Register 3	0x00000000
0x310C910C	DAI_DAT3	DAI Serial Data Routing Control Register 3	0x00000000
0x310C9110	DAI0_DAT4	DAI0 Serial Data Routing Control Register 4	0x00000000
0x310C9110	DAI_DAT4	DAI Serial Data Routing Control Register 4	0x00000000
0x310C9114	DAI0_DAT5	DAI0 Serial Data Routing Control Register 5	0x00000000
0x310C9114	DAI_DAT5	DAI Serial Data Routing Control Register 5	0x00000000
0x310C9118	DAI0_DAT6	DAI0 Serial Data Routing Control Register 6	0x00FBEFBE
0x310C9118	DAI_DAT6	DAI Serial Data Routing Control Register 6	0x00FBEFBE
0x310C911C	DAI0_DAT7	DAI0 Serial Data Routing Control Register 7	0x00000000
0x310C911C	DAI_DAT7	DAI Serial Data Routing Control Register 7	0x00000000
0x310C9120	DAI0_DAT8	DAI0 Serial Data Routing Control Register 8	0x00000000
0x310C9120	DAI_DAT8	DAI Serial Data Routing Control Register 8	0x00000000
0x310C9140	DAI0_FS0	DAI0 Frame Sync Routing Control Register 0	0x2736B4E3
0x310C9140	DAI_FS0	DAI Frame Sync Routing Control Register 0	0x2736B4E3
0x310C9144	DAI0_FS1	DAI0 Frame Sync Routing Control Register 1	0x3DEF7BDE
0x310C9144	DAI_FS1	DAI Frame Sync Routing Control Register 1	0x3DEF7BDE
0x310C9148	DAI0_FS2	DAI0 Frame Sync Routing Control Register 2	0x00007BDE
0x310C9148	DAI_FS2	DAI Frame Sync Routing Control Register 2	0x00007BDE
0x310C9150	DAI0_FS4	DAI0 Frame Sync Routing Control Register 4	0x000F7BDE
0x310C9150	DAI_FS4	DAI Frame Sync Routing Control Register 4	0x000F7BDE
0x310C9154	DAI0_FS5	DAI0 Frame Sync Routing Control Register 5	0x3DEF7BDE
0x310C9154	DAI_FS5	DAI Frame Sync Routing Control Register 5	0x3DEF7BDE

Table A-25: ADSP-2184x DAI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9158	DAI0_FS6	DAI0 Frame Sync Routing Control Register 6	0x000003DE
0x310C9158	DAI_FS6	DAI Frame Sync Routing Control Register 6	0x000003DE
0x310C9180	DAI0_PIN0	DAI0 Pin Buffer Assignment Register 0	0x04C80A94
0x310C9180	DAI_PIN0	DAI Pin Buffer Assignment Register 0	0x04C80A94
0x310C9184	DAI0_PIN1	DAI0 Pin Buffer Assignment Register 1	0x04E84B96
0x310C9184	DAI_PIN1	DAI Pin Buffer Assignment Register 1	0x04E84B96
0x310C9188	DAI0_PIN2	DAI0 Pin Buffer Assignment Register 2	0x03668C98
0x310C9188	DAI_PIN2	DAI Pin Buffer Assignment Register 2	0x03668C98
0x310C918C	DAI0_PIN3	DAI0 Pin Buffer Assignment Register 3	0x03A714A3
0x310C918C	DAI_PIN3	DAI Pin Buffer Assignment Register 3	0x03A714A3
0x310C9190	DAI0_PIN4	DAI0 Pin Buffer Assignment Register 4	0x05694F9E
0x310C9190	DAI_PIN4	DAI Pin Buffer Assignment Register 4	0x05694F9E
0x310C9194	DAI0_PIN5	DAI0 Dummy Pin Buffer Assignment Register	0x06891120
0x310C9194	DAI_PIN5	DAI Dummy Pin Buffer Assignment Register	0x06891120
0x310C91C0	DAI0_MISC0	DAI0 Miscellaneous Control Register 0	0x3DEF7BDE
0x310C91C0	DAI_MISC0	DAI Miscellaneous Control Register 0	0x3DEF7BDE
0x310C91C4	DAI0_MISC1	DAI0 Miscellaneous Control Register 1	0x3DEF7BDE
0x310C91C4	DAI_MISC1	DAI Miscellaneous Control Register 1	0x3DEF7BDE
0x310C91C8	DAI0_MISC2	DAI0 Miscellaneous Control Register 1	0x000F7BDE
0x310C91C8	DAI_MISC2	DAI Miscellaneous Control Register 1	0x000F7BDE
0x310C91E0	DAI0_PBEN0	DAI0 Pin Buffer Enable Register 0	0x0E2482CA
0x310C91E0	DAI_PBEN0	DAI Pin Buffer Enable Register 0	0x0E2482CA
0x310C91E4	DAI0_PBEN1	DAI0 Pin Buffer Enable Register 1	0x1348D30F
0x310C91E4	DAI_PBEN1	DAI Pin Buffer Enable Register 1	0x1348D30F
0x310C91E8	DAI0_PBEN2	DAI0 Pin Buffer Enable Register 2	0x1A5545D6
0x310C91E8	DAI_PBEN2	DAI Pin Buffer Enable Register 2	0x1A5545D6
0x310C91EC	DAI0_PBEN3	DAI0 Pin Buffer Enable Register 3	0x1D71F79B
0x310C91EC	DAI_PBEN3	DAI Pin Buffer Enable Register 3	0x1D71F79B
0x310C9200	DAI0_IMSK_FE	DAI0 Falling-Edge Interrupt Mask Register	0x00000000
0x310C9200	DAI_IMSK_FE	DAI Falling-Edge Interrupt Mask Register	0x00000000
0x310C9204	DAI0_IMSK_RE	DAI0 Rising-Edge Interrupt Mask Register	0x00000000

Table A-25: ADSP-2184x DAI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9204	DAI_IMSK_RE	DAI Rising-Edge Interrupt Mask Register	0x00000000
0x310C9210	DAI0_IMSK_PRI	DAI0 Core Interrupt Priority Assignment Register	0x00000000
0x310C9210	DAI_IMSK_PRI	DAI Core Interrupt Priority Assignment Register	0x00000000
0x310C9220	DAI0_IRPTL_H	DAI0 High Priority Interrupt Latch Register	0x00000000
0x310C9220	DAI_IRPTL_H	DAI High Priority Interrupt Latch Register	0x00000000
0x310C9224	DAI0_IRPTL_L	DAI0 Low Priority Interrupt Latch Register	0x00000000
0x310C9224	DAI_IRPTL_L	DAI Low Priority Interrupt Latch Register	0x00000000
0x310C9230	DAI0_IRPTL_HS	DAI0 Shadow High Priority Interrupt Latch Register	0x00000000
0x310C9230	DAI_IRPTL_HS	DAI Shadow High Priority Interrupt Latch Register	0x00000000
0x310C9234	DAI0_IRPTL_LS	DAI0 Shadow Low Priority Interrupt Latch Register	0x00000000
0x310C9234	DAI_IRPTL_LS	DAI Shadow Low Priority Interrupt Latch Register	0x00000000
0x310C92E4	DAI0_PIN_STAT	DAI0 Pin Status Register	0x00000000
0x310C92E4	DAI_PIN_STAT	DAI Pin Status Register	0x00000000
0x310C92E8	DAI0_GBL_SP_EN	DAI0 Global SPORT Enable Register	0x00000000
0x310C92E8	DAI_GBL_SP_EN	DAI Global SPORT Enable Register	0x00000000
0x310C92EC	DAI0_GBL_INT_EN	DAI0 Global SPORT Interrupt Grouping Register	0x00000000
0x310C92EC	DAI_GBL_INT_EN	DAI Global SPORT Interrupt Grouping Register	0x00000000
0x310C92F0	DAI0_GBL_PCG_EN	DAI0 Global PCG Enable Control Register	0x00000000
0x310C92F0	DAI_GBL_PCG_EN	DAI Global PCG Enable Control Register	0x00000000

Table A-26: ADSP-2184x DAI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA000	DAI1_EXTD_CLK0	DAI1 Extended Clock Routing Control Register 0	0x00000000
0x310CA004	DAI1_EXTD_CLK1	DAI1 Extended Clock Routing Control Register 1	0x00000000
0x310CA008	DAI1_EXTD_CLK2	DAI1 Extended Clock Routing Control Register 2	0x00000000
0x310CA00C	DAI1_EXTD_CLK3	DAI1 Extended Clock Routing Control Register 3	0x00000000
0x310CA010	DAI1_EXTD_CLK4	DAI1 Extended Clock Routing Control Register 4	0x00000000
0x310CA014	DAI1_EXTD_CLK5	DAI1 Extended Clock Routing Control Register 5	0x00000000
0x310CA018	DAI1_EXTD_DAT0	DAI1 Extended Serial Data Routing Control Register 0	0x00000000
0x310CA01C	DAI1_EXTD_DAT1	DAI1 Extended Serial Data Routing Control Register 1	0x00000000

Table A-26: ADSP-2184x DAI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA020	DAI1_EXTD_DAT2	DAI1 Extended Serial Data Routing Control Register 2	0x00000000
0x310CA024	DAI1_EXTD_DAT3	DAI1 Extended Serial Data Routing Control Register 3	0x00000000
0x310CA028	DAI1_EXTD_DAT4	DAI1 Extended Serial Data Routing Control Register 4	0x00000000
0x310CA02C	DAI1_EXTD_DAT5	DAI1 Extended Serial Data Routing Control Register 5	0x00000000
0x310CA030	DAI1_EXTD_DAT6	DAI1 Extended Serial Data Routing Control Register 6	0x00000000
0x310CA034	DAI1_EXTD_FS0	DAI1 Extended Frame Sync Routing Control Register 0	0x00000000
0x310CA038	DAI1_EXTD_FS1	DAI1 Extended Frame Sync Routing Control Register 1	0x00000000
0x310CA03C	DAI1_EXTD_FS2	DAI1 Extended Frame Sync Routing Control Register 2	0x00000000
0x310CA044	DAI1_EXTD_FS4	DAI1 Extended Frame Sync Routing Control Register 4	0x00000000
0x310CA048	DAI1_EXTD_PIN0	DAI1 Extended Pin Buffer Assignment Register 0	0x00000000
0x310CA04C	DAI1_EXTD_PIN1	DAI1 Extended Pin Buffer Assignment Register 1	0x00000000
0x310CA050	DAI1_EXTD_PIN2	DAI1 Extended Pin Buffer Assignment Register 2	0x00000000
0x310CA054	DAI1_EXTD_PIN3	DAI1 Extended Pin Buffer Assignment Register 3	0x00000000
0x310CA058	DAI1_EXTD_PIN4	DAI1 Extended Pin Buffer Assignment Register 4	0x00000000
0x310CA05C	DAI1_EXTD_MISC0	DAI1 Extended Miscellaneous Control Register 0	0x00000000
0x310CA060	DAI1_EXTD_MISC1	DAI1 Extended Miscellaneous Control Register 1	0x00000000
0x310CA064	DAI1_EXTD_MISC2	DAI1 Extended Miscellaneous Control Register 2	0x00000000
0x310CA068	DAI1_EXTD_PBEN0	DAI1 Extended Pin Buffer Enable Register 0	0x00000000
0x310CA06C	DAI1_EXTD_PBEN1	DAI1 Extended Pin Buffer Enable Register 1	0x00000000
0x310CA070	DAI1_EXTD_PBEN2	DAI1 Extended Pin Buffer Enable Register 2	0x00000000
0x310CA074	DAI1_EXTD_PBEN3	DAI1 Extended Pin Buffer Enable Register 3	0x00000000
0x310CA078	DAI1_EXTD_CLK6	DAI1 Extended Clock Routing Control Register 6	0x00000000
0x310CA07C	DAI1_EXTD_CLK7	DAI1 Extended Clock Routing Control Register 7	0x00000000
0x310CA080	DAI1_EXTD_DAT7	DAI1 Extended Serial Data Routing Control Register 7	0x00000000
0x310CA084	DAI1_EXTD_DAT8	DAI1 Extended Serial Data Routing Control Register 8	0x00000000
0x310CA088	DAI1_EXTD_FS5	DAI1 Extended Frame Sync Routing Control Register 5	0x00000000
0x310CA08C	DAI1_EXTD_FS6	DAI1 Extended Frame Sync Routing Control Register 6	0x00000000
0x310CA09C	DAI1_DAI_MISC_CTL	DAI1 DAI MISC Control Register	0x00000000
0x310CA0C0	DAI1_CLK0	DAI1 Clock Routing Control Register 0	0x252630C2
0x310CA0C4	DAI1_CLK1	DAI1 Clock Routing Control Register 1	0x3DEF7BDE
0x310CA0C8	DAI1_CLK2	DAI1 Clock Routing Control Register 2	0x01EF7BDE

Table A-26: ADSP-2184x DAI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA0CC	DAI1_CLK3	DAI1 Clock Routing Control Register 3	0x3C000000
0x310CA0D0	DAI1_CLK4	DAI1 Clock Routing Control Register 4	0x3DEF03DE
0x310CA0D4	DAI1_CLK5	DAI1 Clock Routing Control Register 5	0x3DEF7BDE
0x310CA0D8	DAI1_CLK6	DAI1 Clock Routing Control Register 6	0x3DEF7BDE
0x310CA0DC	DAI1_CLK7	DAI1 Clock Routing Control Register 7	0x000003DE
0x310CA100	DAI1_DAT0	DAI1 Serial Data Routing Control Register 0	0x08144040
0x310CA104	DAI1_DAT1	DAI1 Serial Data Routing Control Register 1	0x0F38B289
0x310CA108	DAI1_DAT2	DAI1 Serial Data Routing Control Register 2	0x00000450
0x310CA10C	DAI1_DAT3	DAI1 Serial Data Routing Control Register 3	0x00000000
0x310CA110	DAI1_DAT4	DAI1 Serial Data Routing Control Register 4	0x00000000
0x310CA114	DAI1_DAT5	DAI1 Serial Data Routing Control Register 5	0x00000000
0x310CA118	DAI1_DAT6	DAI1 Serial Data Routing Control Register 6	0x00FBEFBE
0x310CA11C	DAI1_DAT7	DAI1 Serial Data Routing Control Register 7	0x00000000
0x310CA120	DAI1_DAT8	DAI1 Serial Data Routing Control Register 8	0x00000000
0x310CA140	DAI1_FS0	DAI1 Frame Sync Routing Control Register 0	0x2736B4E3
0x310CA144	DAI1_FS1	DAI1 Frame Sync Routing Control Register 1	0x3DEF7BDE
0x310CA148	DAI1_FS2	DAI1 Frame Sync Routing Control Register 2	0x00007BDE
0x310CA150	DAI1_FS4	DAI1 Frame Sync Routing Control Register 4	0x000F7BDE
0x310CA154	DAI1_FS5	DAI1 Frame Sync Routing Control Register 5	0x3DEF7BDE
0x310CA158	DAI1_FS6	DAI1 Frame Sync Routing Control Register 6	0x000003DE
0x310CA180	DAI1_PIN0	DAI1 Pin Buffer Assignment Register 0	0x04C80A94
0x310CA184	DAI1_PIN1	DAI1 Pin Buffer Assignment Register 1	0x04E84B96
0x310CA188	DAI1_PIN2	DAI1 Pin Buffer Assignment Register 2	0x03668C98
0x310CA18C	DAI1_PIN3	DAI1 Pin Buffer Assignment Register 3	0x03A714A3
0x310CA190	DAI1_PIN4	DAI1 Pin Buffer Assignment Register 4	0x05694F9E
0x310CA194	DAI1_PIN5	DAI1 Dummy Pin Buffer Assignment Register	0x06891120
0x310CA1C0	DAI1_MISC0	DAI1 Miscellaneous Control Register 0	0x3DEF7BDE
0x310CA1C4	DAI1_MISC1	DAI1 Miscellaneous Control Register 1	0x3DEF7BDE
0x310CA1C8	DAI1_MISC2	DAI1 Miscellaneous Control Register 1	0x000F7BDE
0x310CA1E0	DAI1_PBEN0	DAI1 Pin Buffer Enable Register 0	0x0E2482CA
0x310CA1E4	DAI1_PBEN1	DAI1 Pin Buffer Enable Register 1	0x1348D30F

Table A-26: ADSP-2184x DAI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA1E8	DAI1_PBEN2	DAI1 Pin Buffer Enable Register 2	0x1A5545D6
0x310CA1EC	DAI1_PBEN3	DAI1 Pin Buffer Enable Register 3	0x1D71F79B
0x310CA200	DAI1_IMSK_FE	DAI1 Falling-Edge Interrupt Mask Register	0x00000000
0x310CA204	DAI1_IMSK_RE	DAI1 Rising-Edge Interrupt Mask Register	0x00000000
0x310CA210	DAI1_IMSK_PRI	DAI1 Core Interrupt Priority Assignment Register	0x00000000
0x310CA220	DAI1_IRPTL_H	DAI1 High Priority Interrupt Latch Register	0x00000000
0x310CA224	DAI1_IRPTL_L	DAI1 Low Priority Interrupt Latch Register	0x00000000
0x310CA230	DAI1_IRPTL_HS	DAI1 Shadow High Priority Interrupt Latch Register	0x00000000
0x310CA234	DAI1_IRPTL_LS	DAI1 Shadow Low Priority Interrupt Latch Register	0x00000000
0x310CA2E4	DAI1_PIN_STAT	DAI1 Pin Status Register	0x00000000
0x310CA2E8	DAI1_GBL_SP_EN	DAI1 Global SPORT Enable Register	0x00000000
0x310CA2EC	DAI1_GBL_INT_EN	DAI1 Global SPORT Interrupt Grouping Register	0x00000000

Table A-27: ADSP-2184x DDE MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022000	DMA0_DSCPTR_NXT	DMA0 Pointer to Next Initial Descriptor Register	0x00000000
0x31022004	DMA0_ADDRSTART	DMA0 Start Address of Current Buffer Register	0x00000000
0x31022008	DMA0_CFG	DMA0 Configuration Register	0x00000000
0x3102200C	DMA0_XCNT	DMA0 Inner Loop Count Start Value Register	0x00000000
0x31022010	DMA0_XMOD	DMA0 Inner Loop Address Increment Register	0x00000000
0x31022014	DMA0_YCNT	DMA0 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022018	DMA0_YMOD	DMA0 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022024	DMA0_DSCPTR_CUR	DMA0 Current Descriptor Pointer Register	0x00000000
0x31022028	DMA0_DSCPTR_PRV	DMA0 Previous Initial Descriptor Pointer Register	0x00000000
0x3102202C	DMA0_ADDR_CUR	DMA0 Current Address Register	0x00000000
0x31022030	DMA0_STAT	DMA0 Status Register	0x00006000
0x31022034	DMA0_XCNT_CUR	DMA0 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022038	DMA0_YCNT_CUR	DMA0 Current Row Count (2D only) Register	0x00000000
0x31022040	DMA0_BWLCNT	DMA0 Bandwidth Limit Count Register	0x00000000
0x31022044	DMA0_BWLCNT_CUR	DMA0 Bandwidth Limit Count Current Register	0x00000000

Table A-27: ADSP-2184x DDE MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31022048	DMA0_BWMCNT	DMA0 Bandwidth Monitor Count Register	0x00000000
0x3102204C	DMA0_BWMCNT_CUR	DMA0 Bandwidth Monitor Count Current Register	0x00000000
0x3102D200	DDE_DSCPTR_NXT	DDE Pointer to Next Initial Descriptor Register	0x00000000
0x3102D204	DDE_ADDRSTART	DDE Start Address of Current Buffer Register	0x00000000
0x3102D208	DDE_CFG	DDE Configuration Register	0x00000220
0x3102D20C	DDE_XCNT	DDE Inner Loop Count Start Value Register	0x00000000
0x3102D210	DDE_XMOD	DDE Inner Loop Address Increment Register	0x00000000
0x3102D214	DDE_YCNT	DDE Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D218	DDE_YMOD	DDE Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D224	DDE_DSCPTR_CUR	DDE Current Descriptor Pointer Register	0x00000000
0x3102D228	DDE_DSCPTR_PRV	DDE Previous Initial Descriptor Pointer Register	0x00000000
0x3102D22C	DDE_ADDR_CUR	DDE Current Address Register	0x00000000
0x3102D230	DDE_STAT	DDE Status Register	0x0000B000
0x3102D234	DDE_XCNT_CUR	DDE Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D238	DDE_YCNT_CUR	DDE Current Row Count (2D only) Register	0x00000000
0x3102D240	DDE_BWLCNT	DDE Bandwidth Limit Count Register	0x00000000
0x3102D244	DDE_BWLCNT_CUR	DDE Bandwidth Limit Count Current Register	0x00000000
0x3102D248	DDE_BWMCNT	DDE Bandwidth Monitor Count Register	0x00000000
0x3102D24C	DDE_BWMCNT_CUR	DDE Bandwidth Monitor Count Current Register	0x00000000

Table A-28: ADSP-2184x DMA1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022080	DMA1_DSCPTR_NXT	DMA1 Pointer to Next Initial Descriptor Register	0x00000000
0x31022084	DMA1_ADDRSTART	DMA1 Start Address of Current Buffer Register	0x00000000
0x31022088	DMA1_CFG	DMA1 Configuration Register	0x00000000
0x3102208C	DMA1_XCNT	DMA1 Inner Loop Count Start Value Register	0x00000000
0x31022090	DMA1_XMOD	DMA1 Inner Loop Address Increment Register	0x00000000
0x31022094	DMA1_YCNT	DMA1 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022098	DMA1_YMOD	DMA1 Outer Loop Address Increment (2D only) Register	0x00000000
0x310220A4	DMA1_DSCPTR_CUR	DMA1 Current Descriptor Pointer Register	0x00000000

Table A-28: ADSP-2184x DMA1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310220A8	DMA1_DSCPTR_PRV	DMA1 Previous Initial Descriptor Pointer Register	0x00000000
0x310220AC	DMA1_ADDR_CUR	DMA1 Current Address Register	0x00000000
0x310220B0	DMA1_STAT	DMA1 Status Register	0x00006000
0x310220B4	DMA1_XCNT_CUR	DMA1 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310220B8	DMA1_YCNT_CUR	DMA1 Current Row Count (2D only) Register	0x00000000
0x310220C0	DMA1_BWLCNT	DMA1 Bandwidth Limit Count Register	0x00000000
0x310220C4	DMA1_BWLCNT_CUR	DMA1 Bandwidth Limit Count Current Register	0x00000000
0x310220C8	DMA1_BWMCNT	DMA1 Bandwidth Monitor Count Register	0x00000000
0x310220CC	DMA1_BWMCNT_CUR	DMA1 Bandwidth Monitor Count Current Register	0x00000000

Table A-29: ADSP-2184x DMA10 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023000	DMA10_DSCPTR_NXT	DMA10 Pointer to Next Initial Descriptor Register	0x00000000
0x31023004	DMA10_ADDRSTART	DMA10 Start Address of Current Buffer Register	0x00000000
0x31023008	DMA10_CFG	DMA10 Configuration Register	0x00000000
0x3102300C	DMA10_XCNT	DMA10 Inner Loop Count Start Value Register	0x00000000
0x31023010	DMA10_XMOD	DMA10 Inner Loop Address Increment Register	0x00000000
0x31023014	DMA10_YCNT	DMA10 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023018	DMA10_YMOD	DMA10 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023024	DMA10_DSCPTR_CUR	DMA10 Current Descriptor Pointer Register	0x00000000
0x31023028	DMA10_DSCPTR_PRV	DMA10 Previous Initial Descriptor Pointer Register	0x00000000
0x3102302C	DMA10_ADDR_CUR	DMA10 Current Address Register	0x00000000
0x31023030	DMA10_STAT	DMA10 Status Register	0x00006000
0x31023034	DMA10_XCNT_CUR	DMA10 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023038	DMA10_YCNT_CUR	DMA10 Current Row Count (2D only) Register	0x00000000
0x31023040	DMA10_BWLCNT	DMA10 Bandwidth Limit Count Register	0x00000000
0x31023044	DMA10_BWLCNT_CUR	DMA10 Bandwidth Limit Count Current Register	0x00000000
0x31023048	DMA10_BWMCNT	DMA10 Bandwidth Monitor Count Register	0x00000000

Table A-29: ADSP-2184x DMA10 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102304C	DMA10_BWMCNT_CUR	DMA10 Bandwidth Monitor Count Current Register	0x00000000

Table A-30: ADSP-2184x DMA11 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023080	DMA11_DSCPTR_NXT	DMA11 Pointer to Next Initial Descriptor Register	0x00000000
0x31023084	DMA11_ADDRSTART	DMA11 Start Address of Current Buffer Register	0x00000000
0x31023088	DMA11_CFG	DMA11 Configuration Register	0x00000000
0x3102308C	DMA11_XCNT	DMA11 Inner Loop Count Start Value Register	0x00000000
0x31023090	DMA11_XMOD	DMA11 Inner Loop Address Increment Register	0x00000000
0x31023094	DMA11_YCNT	DMA11 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023098	DMA11_YMOD	DMA11 Outer Loop Address Increment (2D only) Register	0x00000000
0x310230A4	DMA11_DSCPTR_CUR	DMA11 Current Descriptor Pointer Register	0x00000000
0x310230A8	DMA11_DSCPTR_PRV	DMA11 Previous Initial Descriptor Pointer Register	0x00000000
0x310230AC	DMA11_ADDR_CUR	DMA11 Current Address Register	0x00000000
0x310230B0	DMA11_STAT	DMA11 Status Register	0x00006000
0x310230B4	DMA11_XCNT_CUR	DMA11 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310230B8	DMA11_YCNT_CUR	DMA11 Current Row Count (2D only) Register	0x00000000
0x310230C0	DMA11_BWLCNT	DMA11 Bandwidth Limit Count Register	0x00000000
0x310230C4	DMA11_BWLCNT_CUR	DMA11 Bandwidth Limit Count Current Register	0x00000000
0x310230C8	DMA11_BWMCNT	DMA11 Bandwidth Monitor Count Register	0x00000000
0x310230CC	DMA11_BWMCNT_CUR	DMA11 Bandwidth Monitor Count Current Register	0x00000000

Table A-31: ADSP-2184x DMA12 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023100	DMA12_DSCPTR_NXT	DMA12 Pointer to Next Initial Descriptor Register	0x00000000
0x31023104	DMA12_ADDRSTART	DMA12 Start Address of Current Buffer Register	0x00000000
0x31023108	DMA12_CFG	DMA12 Configuration Register	0x00000000
0x3102310C	DMA12_XCNT	DMA12 Inner Loop Count Start Value Register	0x00000000

Table A-31: ADSP-2184x DMA12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31023110	DMA12_XMOD	DMA12 Inner Loop Address Increment Register	0x00000000
0x31023114	DMA12_YCNT	DMA12 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023118	DMA12_YMOD	DMA12 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023124	DMA12_DSCPTR_CUR	DMA12 Current Descriptor Pointer Register	0x00000000
0x31023128	DMA12_DSCPTR_PRV	DMA12 Previous Initial Descriptor Pointer Register	0x00000000
0x3102312C	DMA12_ADDR_CUR	DMA12 Current Address Register	0x00000000
0x31023130	DMA12_STAT	DMA12 Status Register	0x00006000
0x31023134	DMA12_XCNT_CUR	DMA12 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023138	DMA12_YCNT_CUR	DMA12 Current Row Count (2D only) Register	0x00000000
0x31023140	DMA12_BWLCNT	DMA12 Bandwidth Limit Count Register	0x00000000
0x31023144	DMA12_BWLCNT_CUR	DMA12 Bandwidth Limit Count Current Register	0x00000000
0x31023148	DMA12_BWMCNT	DMA12 Bandwidth Monitor Count Register	0x00000000
0x3102314C	DMA12_BWMCNT_CUR	DMA12 Bandwidth Monitor Count Current Register	0x00000000

Table A-32: ADSP-2184x DMA13 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023180	DMA13_DSCPTR_NXT	DMA13 Pointer to Next Initial Descriptor Register	0x00000000
0x31023184	DMA13_ADDRSTART	DMA13 Start Address of Current Buffer Register	0x00000000
0x31023188	DMA13_CFG	DMA13 Configuration Register	0x00000000
0x3102318C	DMA13_XCNT	DMA13 Inner Loop Count Start Value Register	0x00000000
0x31023190	DMA13_XMOD	DMA13 Inner Loop Address Increment Register	0x00000000
0x31023194	DMA13_YCNT	DMA13 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023198	DMA13_YMOD	DMA13 Outer Loop Address Increment (2D only) Register	0x00000000
0x310231A4	DMA13_DSCPTR_CUR	DMA13 Current Descriptor Pointer Register	0x00000000
0x310231A8	DMA13_DSCPTR_PRV	DMA13 Previous Initial Descriptor Pointer Register	0x00000000
0x310231AC	DMA13_ADDR_CUR	DMA13 Current Address Register	0x00000000
0x310231B0	DMA13_STAT	DMA13 Status Register	0x00006000

Table A-32: ADSP-2184x DMA13 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310231B4	DMA13_XCNT_CUR	DMA13 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310231B8	DMA13_YCNT_CUR	DMA13 Current Row Count (2D only) Register	0x00000000
0x310231C0	DMA13_BWLCNT	DMA13 Bandwidth Limit Count Register	0x00000000
0x310231C4	DMA13_BWLCNT_CUR	DMA13 Bandwidth Limit Count Current Register	0x00000000
0x310231C8	DMA13_BWMCNT	DMA13 Bandwidth Monitor Count Register	0x00000000
0x310231CC	DMA13_BWMCNT_CUR	DMA13 Bandwidth Monitor Count Current Register	0x00000000

Table A-33: ADSP-2184x DMA14 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023200	DMA14_DSCPTR_NXT	DMA14 Pointer to Next Initial Descriptor Register	0x00000000
0x31023204	DMA14_ADDRSTART	DMA14 Start Address of Current Buffer Register	0x00000000
0x31023208	DMA14_CFG	DMA14 Configuration Register	0x00000000
0x3102320C	DMA14_XCNT	DMA14 Inner Loop Count Start Value Register	0x00000000
0x31023210	DMA14_XMOD	DMA14 Inner Loop Address Increment Register	0x00000000
0x31023214	DMA14_YCNT	DMA14 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023218	DMA14_YMOD	DMA14 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023224	DMA14_DSCPTR_CUR	DMA14 Current Descriptor Pointer Register	0x00000000
0x31023228	DMA14_DSCPTR_PRV	DMA14 Previous Initial Descriptor Pointer Register	0x00000000
0x3102322C	DMA14_ADDR_CUR	DMA14 Current Address Register	0x00000000
0x31023230	DMA14_STAT	DMA14 Status Register	0x00006000
0x31023234	DMA14_XCNT_CUR	DMA14 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023238	DMA14_YCNT_CUR	DMA14 Current Row Count (2D only) Register	0x00000000
0x31023240	DMA14_BWLCNT	DMA14 Bandwidth Limit Count Register	0x00000000
0x31023244	DMA14_BWLCNT_CUR	DMA14 Bandwidth Limit Count Current Register	0x00000000
0x31023248	DMA14_BWMCNT	DMA14 Bandwidth Monitor Count Register	0x00000000
0x3102324C	DMA14_BWMCNT_CUR	DMA14 Bandwidth Monitor Count Current Register	0x00000000

Table A-34: ADSP-2184x DMA15 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023280	DMA15_DSCPTR_NXT	DMA15 Pointer to Next Initial Descriptor Register	0x00000000
0x31023284	DMA15_ADDRSTART	DMA15 Start Address of Current Buffer Register	0x00000000
0x31023288	DMA15_CFG	DMA15 Configuration Register	0x00000000
0x3102328C	DMA15_XCNT	DMA15 Inner Loop Count Start Value Register	0x00000000
0x31023290	DMA15_XMOD	DMA15 Inner Loop Address Increment Register	0x00000000
0x31023294	DMA15_YCNT	DMA15 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023298	DMA15_YMOD	DMA15 Outer Loop Address Increment (2D only) Register	0x00000000
0x310232A4	DMA15_DSCPTR_CUR	DMA15 Current Descriptor Pointer Register	0x00000000
0x310232A8	DMA15_DSCPTR_PRV	DMA15 Previous Initial Descriptor Pointer Register	0x00000000
0x310232AC	DMA15_ADDR_CUR	DMA15 Current Address Register	0x00000000
0x310232B0	DMA15_STAT	DMA15 Status Register	0x00006000
0x310232B4	DMA15_XCNT_CUR	DMA15 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310232B8	DMA15_YCNT_CUR	DMA15 Current Row Count (2D only) Register	0x00000000
0x310232C0	DMA15_BWLCNT	DMA15 Bandwidth Limit Count Register	0x00000000
0x310232C4	DMA15_BWLCNT_CUR	DMA15 Bandwidth Limit Count Current Register	0x00000000
0x310232C8	DMA15_BWMCNT	DMA15 Bandwidth Monitor Count Register	0x00000000
0x310232CC	DMA15_BWMCNT_CUR	DMA15 Bandwidth Monitor Count Current Register	0x00000000

Table A-35: ADSP-2184x DMA16 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023300	DMA16_DSCPTR_NXT	DMA16 Pointer to Next Initial Descriptor Register	0x00000000
0x31023304	DMA16_ADDRSTART	DMA16 Start Address of Current Buffer Register	0x00000000
0x31023308	DMA16_CFG	DMA16 Configuration Register	0x00000000
0x3102330C	DMA16_XCNT	DMA16 Inner Loop Count Start Value Register	0x00000000
0x31023310	DMA16_XMOD	DMA16 Inner Loop Address Increment Register	0x00000000
0x31023314	DMA16_YCNT	DMA16 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023318	DMA16_YMOD	DMA16 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-35: ADSP-2184x DMA16 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31023324	DMA16_DSCPTR_CUR	DMA16 Current Descriptor Pointer Register	0x00000000
0x31023328	DMA16_DSCPTR_PRV	DMA16 Previous Initial Descriptor Pointer Register	0x00000000
0x3102332C	DMA16_ADDR_CUR	DMA16 Current Address Register	0x00000000
0x31023330	DMA16_STAT	DMA16 Status Register	0x00006000
0x31023334	DMA16_XCNT_CUR	DMA16 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023338	DMA16_YCNT_CUR	DMA16 Current Row Count (2D only) Register	0x00000000
0x31023340	DMA16_BWLCNT	DMA16 Bandwidth Limit Count Register	0x00000000
0x31023344	DMA16_BWLCNT_CUR	DMA16 Bandwidth Limit Count Current Register	0x00000000
0x31023348	DMA16_BWMCNT	DMA16 Bandwidth Monitor Count Register	0x00000000
0x3102334C	DMA16_BWMCNT_CUR	DMA16 Bandwidth Monitor Count Current Register	0x00000000

Table A-36: ADSP-2184x DMA17 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023380	DMA17_DSCPTR_NXT	DMA17 Pointer to Next Initial Descriptor Register	0x00000000
0x31023384	DMA17_ADDRSTART	DMA17 Start Address of Current Buffer Register	0x00000000
0x31023388	DMA17_CFG	DMA17 Configuration Register	0x00000000
0x3102338C	DMA17_XCNT	DMA17 Inner Loop Count Start Value Register	0x00000000
0x31023390	DMA17_XMOD	DMA17 Inner Loop Address Increment Register	0x00000000
0x31023394	DMA17_YCNT	DMA17 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023398	DMA17_YMOD	DMA17 Outer Loop Address Increment (2D only) Register	0x00000000
0x310233A4	DMA17_DSCPTR_CUR	DMA17 Current Descriptor Pointer Register	0x00000000
0x310233A8	DMA17_DSCPTR_PRV	DMA17 Previous Initial Descriptor Pointer Register	0x00000000
0x310233AC	DMA17_ADDR_CUR	DMA17 Current Address Register	0x00000000
0x310233B0	DMA17_STAT	DMA17 Status Register	0x00006000
0x310233B4	DMA17_XCNT_CUR	DMA17 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310233B8	DMA17_YCNT_CUR	DMA17 Current Row Count (2D only) Register	0x00000000
0x310233C0	DMA17_BWLCNT	DMA17 Bandwidth Limit Count Register	0x00000000
0x310233C4	DMA17_BWLCNT_CUR	DMA17 Bandwidth Limit Count Current Register	0x00000000

Table A-36: ADSP-2184x DMA17 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310233C8	DMA17_BWMCNT	DMA17 Bandwidth Monitor Count Register	0x00000000
0x310233CC	DMA17_BWMCNT_CUR	DMA17 Bandwidth Monitor Count Current Register	0x00000000

Table A-37: ADSP-2184x DMA18 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7100	DMA18_DSCPTR_NXT	DMA18 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7104	DMA18_ADDRSTART	DMA18 Start Address of Current Buffer Register	0x00000000
0x310A7108	DMA18_CFG	DMA18 Configuration Register	0x00000000
0x310A710C	DMA18_XCNT	DMA18 Inner Loop Count Start Value Register	0x00000000
0x310A7110	DMA18_XMOD	DMA18 Inner Loop Address Increment Register	0x00000000
0x310A7114	DMA18_YCNT	DMA18 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7118	DMA18_YMOD	DMA18 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7124	DMA18_DSCPTR_CUR	DMA18 Current Descriptor Pointer Register	0x00000000
0x310A7128	DMA18_DSCPTR_PRV	DMA18 Previous Initial Descriptor Pointer Register	0x00000000
0x310A712C	DMA18_ADDR_CUR	DMA18 Current Address Register	0x00000000
0x310A7130	DMA18_STAT	DMA18 Status Register	0x00006000
0x310A7134	DMA18_XCNT_CUR	DMA18 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7138	DMA18_YCNT_CUR	DMA18 Current Row Count (2D only) Register	0x00000000
0x310A7140	DMA18_BWLCNT	DMA18 Bandwidth Limit Count Register	0x00000000
0x310A7144	DMA18_BWLCNT_CUR	DMA18 Bandwidth Limit Count Current Register	0x00000000
0x310A7148	DMA18_BWMCNT	DMA18 Bandwidth Monitor Count Register	0x00000000
0x310A714C	DMA18_BWMCNT_CUR	DMA18 Bandwidth Monitor Count Current Register	0x00000000

Table A-38: ADSP-2184x DMA19 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7180	DMA19_DSCPTR_NXT	DMA19 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7184	DMA19_ADDRSTART	DMA19 Start Address of Current Buffer Register	0x00000000
0x310A7188	DMA19_CFG	DMA19 Configuration Register	0x00000000

Table A-38: ADSP-2184x DMA19 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A718C	DMA19_XCNT	DMA19 Inner Loop Count Start Value Register	0x00000000
0x310A7190	DMA19_XMOD	DMA19 Inner Loop Address Increment Register	0x00000000
0x310A7194	DMA19_YCNT	DMA19 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7198	DMA19_YMOD	DMA19 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A71A4	DMA19_DSCPTR_CUR	DMA19 Current Descriptor Pointer Register	0x00000000
0x310A71A8	DMA19_DSCPTR_PRV	DMA19 Previous Initial Descriptor Pointer Register	0x00000000
0x310A71AC	DMA19_ADDR_CUR	DMA19 Current Address Register	0x00000000
0x310A71B0	DMA19_STAT	DMA19 Status Register	0x00006000
0x310A71B4	DMA19_XCNT_CUR	DMA19 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A71B8	DMA19_YCNT_CUR	DMA19 Current Row Count (2D only) Register	0x00000000
0x310A71C0	DMA19_BWLCNT	DMA19 Bandwidth Limit Count Register	0x00000000
0x310A71C4	DMA19_BWLCNT_CUR	DMA19 Bandwidth Limit Count Current Register	0x00000000
0x310A71C8	DMA19_BWMCNT	DMA19 Bandwidth Monitor Count Register	0x00000000
0x310A71CC	DMA19_BWMCNT_CUR	DMA19 Bandwidth Monitor Count Current Register	0x00000000

Table A-39: ADSP-2184x DMA2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022100	DMA2_DSCPTR_NXT	DMA2 Pointer to Next Initial Descriptor Register	0x00000000
0x31022104	DMA2_ADDRSTART	DMA2 Start Address of Current Buffer Register	0x00000000
0x31022108	DMA2_CFG	DMA2 Configuration Register	0x00000000
0x3102210C	DMA2_XCNT	DMA2 Inner Loop Count Start Value Register	0x00000000
0x31022110	DMA2_XMOD	DMA2 Inner Loop Address Increment Register	0x00000000
0x31022114	DMA2_YCNT	DMA2 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022118	DMA2_YMOD	DMA2 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022124	DMA2_DSCPTR_CUR	DMA2 Current Descriptor Pointer Register	0x00000000
0x31022128	DMA2_DSCPTR_PRV	DMA2 Previous Initial Descriptor Pointer Register	0x00000000
0x3102212C	DMA2_ADDR_CUR	DMA2 Current Address Register	0x00000000
0x31022130	DMA2_STAT	DMA2 Status Register	0x00006000

Table A-39: ADSP-2184x DMA2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31022134	DMA2_XCNT_CUR	DMA2 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022138	DMA2_YCNT_CUR	DMA2 Current Row Count (2D only) Register	0x00000000
0x31022140	DMA2_BWLCNT	DMA2 Bandwidth Limit Count Register	0x00000000
0x31022144	DMA2_BWLCNT_CUR	DMA2 Bandwidth Limit Count Current Register	0x00000000
0x31022148	DMA2_BWMCNT	DMA2 Bandwidth Monitor Count Register	0x00000000
0x3102214C	DMA2_BWMCNT_CUR	DMA2 Bandwidth Monitor Count Current Register	0x00000000

Table A-40: ADSP-2184x DMA20 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026080	DMA20_DSCPTR_NXT	DMA20 Pointer to Next Initial Descriptor Register	0x00000000
0x31026084	DMA20_ADDRSTART	DMA20 Start Address of Current Buffer Register	0x00000000
0x31026088	DMA20_CFG	DMA20 Configuration Register	0x00000000
0x3102608C	DMA20_XCNT	DMA20 Inner Loop Count Start Value Register	0x00000000
0x31026090	DMA20_XMOD	DMA20 Inner Loop Address Increment Register	0x00000000
0x31026094	DMA20_YCNT	DMA20 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026098	DMA20_YMOD	DMA20 Outer Loop Address Increment (2D only) Register	0x00000000
0x310260A4	DMA20_DSCPTR_CUR	DMA20 Current Descriptor Pointer Register	0x00000000
0x310260A8	DMA20_DSCPTR_PRV	DMA20 Previous Initial Descriptor Pointer Register	0x00000000
0x310260AC	DMA20_ADDR_CUR	DMA20 Current Address Register	0x00000000
0x310260B0	DMA20_STAT	DMA20 Status Register	0x00006000
0x310260B4	DMA20_XCNT_CUR	DMA20 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310260B8	DMA20_YCNT_CUR	DMA20 Current Row Count (2D only) Register	0x00000000
0x310260C0	DMA20_BWLCNT	DMA20 Bandwidth Limit Count Register	0x00000000
0x310260C4	DMA20_BWLCNT_CUR	DMA20 Bandwidth Limit Count Current Register	0x00000000
0x310260C8	DMA20_BWMCNT	DMA20 Bandwidth Monitor Count Register	0x00000000
0x310260CC	DMA20_BWMCNT_CUR	DMA20 Bandwidth Monitor Count Current Register	0x00000000

Table A-41: ADSP-2184x DMA21 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026000	DMA21_DSCPTR_NXT	DMA21 Pointer to Next Initial Descriptor Register	0x00000000
0x31026004	DMA21_ADDRSTART	DMA21 Start Address of Current Buffer Register	0x00000000
0x31026008	DMA21_CFG	DMA21 Configuration Register	0x00000000
0x3102600C	DMA21_XCNT	DMA21 Inner Loop Count Start Value Register	0x00000000
0x31026010	DMA21_XMOD	DMA21 Inner Loop Address Increment Register	0x00000000
0x31026014	DMA21_YCNT	DMA21 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026018	DMA21_YMOD	DMA21 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026024	DMA21_DSCPTR_CUR	DMA21 Current Descriptor Pointer Register	0x00000000
0x31026028	DMA21_DSCPTR_PRV	DMA21 Previous Initial Descriptor Pointer Register	0x00000000
0x3102602C	DMA21_ADDR_CUR	DMA21 Current Address Register	0x00000000
0x31026030	DMA21_STAT	DMA21 Status Register	0x00006000
0x31026034	DMA21_XCNT_CUR	DMA21 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026038	DMA21_YCNT_CUR	DMA21 Current Row Count (2D only) Register	0x00000000
0x31026040	DMA21_BWLCNT	DMA21 Bandwidth Limit Count Register	0x00000000
0x31026044	DMA21_BWLCNT_CUR	DMA21 Bandwidth Limit Count Current Register	0x00000000
0x31026048	DMA21_BWMCNT	DMA21 Bandwidth Monitor Count Register	0x00000000
0x3102604C	DMA21_BWMCNT_CUR	DMA21 Bandwidth Monitor Count Current Register	0x00000000

Table A-42: ADSP-2184x DMA22 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D000	DMA22_DSCPTR_NXT	DMA22 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D004	DMA22_ADDRSTART	DMA22 Start Address of Current Buffer Register	0x00000000
0x3102D008	DMA22_CFG	DMA22 Configuration Register	0x00000000
0x3102D00C	DMA22_XCNT	DMA22 Inner Loop Count Start Value Register	0x00000000
0x3102D010	DMA22_XMOD	DMA22 Inner Loop Address Increment Register	0x00000000
0x3102D014	DMA22_YCNT	DMA22 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D018	DMA22_YMOD	DMA22 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-42: ADSP-2184x DMA22 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D024	DMA22_DSCPTR_CUR	DMA22 Current Descriptor Pointer Register	0x00000000
0x3102D028	DMA22_DSCPTR_PRV	DMA22 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D02C	DMA22_ADDR_CUR	DMA22 Current Address Register	0x00000000
0x3102D030	DMA22_STAT	DMA22 Status Register	0x00006000
0x3102D034	DMA22_XCNT_CUR	DMA22 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D038	DMA22_YCNT_CUR	DMA22 Current Row Count (2D only) Register	0x00000000
0x3102D040	DMA22_BWLCNT	DMA22 Bandwidth Limit Count Register	0x00000000
0x3102D044	DMA22_BWLCNT_CUR	DMA22 Bandwidth Limit Count Current Register	0x00000000
0x3102D048	DMA22_BWMCNT	DMA22 Bandwidth Monitor Count Register	0x00000000
0x3102D04C	DMA22_BWMCNT_CUR	DMA22 Bandwidth Monitor Count Current Register	0x00000000

Table A-43: ADSP-2184x DMA23 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D080	DMA23_DSCPTR_NXT	DMA23 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D084	DMA23_ADDRSTART	DMA23 Start Address of Current Buffer Register	0x00000000
0x3102D088	DMA23_CFG	DMA23 Configuration Register	0x00000000
0x3102D08C	DMA23_XCNT	DMA23 Inner Loop Count Start Value Register	0x00000000
0x3102D090	DMA23_XMOD	DMA23 Inner Loop Address Increment Register	0x00000000
0x3102D094	DMA23_YCNT	DMA23 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D098	DMA23_YMOD	DMA23 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D0A4	DMA23_DSCPTR_CUR	DMA23 Current Descriptor Pointer Register	0x00000000
0x3102D0A8	DMA23_DSCPTR_PRV	DMA23 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D0AC	DMA23_ADDR_CUR	DMA23 Current Address Register	0x00000000
0x3102D0B0	DMA23_STAT	DMA23 Status Register	0x00006000
0x3102D0B4	DMA23_XCNT_CUR	DMA23 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D0B8	DMA23_YCNT_CUR	DMA23 Current Row Count (2D only) Register	0x00000000
0x3102D0C0	DMA23_BWLCNT	DMA23 Bandwidth Limit Count Register	0x00000000
0x3102D0C4	DMA23_BWLCNT_CUR	DMA23 Bandwidth Limit Count Current Register	0x00000000

Table A-43: ADSP-2184x DMA23 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D0C8	DMA23_BWMCNT	DMA23 Bandwidth Monitor Count Register	0x00000000
0x3102D0CC	DMA23_BWMCNT_CUR	DMA23 Bandwidth Monitor Count Current Register	0x00000000

Table A-44: ADSP-2184x DMA24 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D100	DMA24_DSCPTR_NXT	DMA24 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D104	DMA24_ADDRSTART	DMA24 Start Address of Current Buffer Register	0x00000000
0x3102D108	DMA24_CFG	DMA24 Configuration Register	0x00000000
0x3102D10C	DMA24_XCNT	DMA24 Inner Loop Count Start Value Register	0x00000000
0x3102D110	DMA24_XMOD	DMA24 Inner Loop Address Increment Register	0x00000000
0x3102D114	DMA24_YCNT	DMA24 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D118	DMA24_YMOD	DMA24 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D124	DMA24_DSCPTR_CUR	DMA24 Current Descriptor Pointer Register	0x00000000
0x3102D128	DMA24_DSCPTR_PRV	DMA24 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D12C	DMA24_ADDR_CUR	DMA24 Current Address Register	0x00000000
0x3102D130	DMA24_STAT	DMA24 Status Register	0x00006000
0x3102D134	DMA24_XCNT_CUR	DMA24 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D138	DMA24_YCNT_CUR	DMA24 Current Row Count (2D only) Register	0x00000000
0x3102D140	DMA24_BWLCNT	DMA24 Bandwidth Limit Count Register	0x00000000
0x3102D144	DMA24_BWLCNT_CUR	DMA24 Bandwidth Limit Count Current Register	0x00000000
0x3102D148	DMA24_BWMCNT	DMA24 Bandwidth Monitor Count Register	0x00000000
0x3102D14C	DMA24_BWMCNT_CUR	DMA24 Bandwidth Monitor Count Current Register	0x00000000

Table A-45: ADSP-2184x DMA25 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D180	DMA25_DSCPTR_NXT	DMA25 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D184	DMA25_ADDRSTART	DMA25 Start Address of Current Buffer Register	0x00000000
0x3102D188	DMA25_CFG	DMA25 Configuration Register	0x00000000

Table A-45: ADSP-2184x DMA25 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D18C	DMA25_XCNT	DMA25 Inner Loop Count Start Value Register	0x00000000
0x3102D190	DMA25_XMOD	DMA25 Inner Loop Address Increment Register	0x00000000
0x3102D194	DMA25_YCNT	DMA25 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D198	DMA25_YMOD	DMA25 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D1A4	DMA25_DSCPTR_CUR	DMA25 Current Descriptor Pointer Register	0x00000000
0x3102D1A8	DMA25_DSCPTR_PRV	DMA25 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D1AC	DMA25_ADDR_CUR	DMA25 Current Address Register	0x00000000
0x3102D1B0	DMA25_STAT	DMA25 Status Register	0x00006000
0x3102D1B4	DMA25_XCNT_CUR	DMA25 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D1B8	DMA25_YCNT_CUR	DMA25 Current Row Count (2D only) Register	0x00000000
0x3102D1C0	DMA25_BWLCNT	DMA25 Bandwidth Limit Count Register	0x00000000
0x3102D1C4	DMA25_BWLCNT_CUR	DMA25 Bandwidth Limit Count Current Register	0x00000000
0x3102D1C8	DMA25_BWMCNT	DMA25 Bandwidth Monitor Count Register	0x00000000
0x3102D1CC	DMA25_BWMCNT_CUR	DMA25 Bandwidth Monitor Count Current Register	0x00000000

Table A-46: ADSP-2184x DMA26 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D200	DMA26_DSCPTR_NXT	DMA26 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D204	DMA26_ADDRSTART	DMA26 Start Address of Current Buffer Register	0x00000000
0x3102D208	DMA26_CFG	DMA26 Configuration Register	0x00000000
0x3102D20C	DMA26_XCNT	DMA26 Inner Loop Count Start Value Register	0x00000000
0x3102D210	DMA26_XMOD	DMA26 Inner Loop Address Increment Register	0x00000000
0x3102D214	DMA26_YCNT	DMA26 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D218	DMA26_YMOD	DMA26 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D224	DMA26_DSCPTR_CUR	DMA26 Current Descriptor Pointer Register	0x00000000
0x3102D228	DMA26_DSCPTR_PRV	DMA26 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D22C	DMA26_ADDR_CUR	DMA26 Current Address Register	0x00000000

Table A-46: ADSP-2184x DMA26 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D230	DMA26_STAT	DMA26 Status Register	0x00006000
0x3102D234	DMA26_XCNT_CUR	DMA26 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D238	DMA26_YCNT_CUR	DMA26 Current Row Count (2D only) Register	0x00000000
0x3102D240	DMA26_BWLCNT	DMA26 Bandwidth Limit Count Register	0x00000000
0x3102D244	DMA26_BWLCNT_CUR	DMA26 Bandwidth Limit Count Current Register	0x00000000
0x3102D248	DMA26_BWMCNT	DMA26 Bandwidth Monitor Count Register	0x00000000
0x3102D24C	DMA26_BWMCNT_CUR	DMA26 Bandwidth Monitor Count Current Register	0x00000000

Table A-47: ADSP-2184x DMA27 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D280	DMA27_DSCPTR_NXT	DMA27 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D284	DMA27_ADDRSTART	DMA27 Start Address of Current Buffer Register	0x00000000
0x3102D288	DMA27_CFG	DMA27 Configuration Register	0x00000000
0x3102D28C	DMA27_XCNT	DMA27 Inner Loop Count Start Value Register	0x00000000
0x3102D290	DMA27_XMOD	DMA27 Inner Loop Address Increment Register	0x00000000
0x3102D294	DMA27_YCNT	DMA27 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D298	DMA27_YMOD	DMA27 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D2A4	DMA27_DSCPTR_CUR	DMA27 Current Descriptor Pointer Register	0x00000000
0x3102D2A8	DMA27_DSCPTR_PRV	DMA27 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D2AC	DMA27_ADDR_CUR	DMA27 Current Address Register	0x00000000
0x3102D2B0	DMA27_STAT	DMA27 Status Register	0x00006000
0x3102D2B4	DMA27_XCNT_CUR	DMA27 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D2B8	DMA27_YCNT_CUR	DMA27 Current Row Count (2D only) Register	0x00000000
0x3102D2C0	DMA27_BWLCNT	DMA27 Bandwidth Limit Count Register	0x00000000
0x3102D2C4	DMA27_BWLCNT_CUR	DMA27 Bandwidth Limit Count Current Register	0x00000000
0x3102D2C8	DMA27_BWMCNT	DMA27 Bandwidth Monitor Count Register	0x00000000
0x3102D2CC	DMA27_BWMCNT_CUR	DMA27 Bandwidth Monitor Count Current Register	0x00000000

Table A-48: ADSP-2184x DMA3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022180	DMA3_DSCPTR_NXT	DMA3 Pointer to Next Initial Descriptor Register	0x00000000
0x31022184	DMA3_ADDRSTART	DMA3 Start Address of Current Buffer Register	0x00000000
0x31022188	DMA3_CFG	DMA3 Configuration Register	0x00000000
0x3102218C	DMA3_XCNT	DMA3 Inner Loop Count Start Value Register	0x00000000
0x31022190	DMA3_XMOD	DMA3 Inner Loop Address Increment Register	0x00000000
0x31022194	DMA3_YCNT	DMA3 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022198	DMA3_YMOD	DMA3 Outer Loop Address Increment (2D only) Register	0x00000000
0x310221A4	DMA3_DSCPTR_CUR	DMA3 Current Descriptor Pointer Register	0x00000000
0x310221A8	DMA3_DSCPTR_PRV	DMA3 Previous Initial Descriptor Pointer Register	0x00000000
0x310221AC	DMA3_ADDR_CUR	DMA3 Current Address Register	0x00000000
0x310221B0	DMA3_STAT	DMA3 Status Register	0x00006000
0x310221B4	DMA3_XCNT_CUR	DMA3 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310221B8	DMA3_YCNT_CUR	DMA3 Current Row Count (2D only) Register	0x00000000
0x310221C0	DMA3_BWLCNT	DMA3 Bandwidth Limit Count Register	0x00000000
0x310221C4	DMA3_BWLCNT_CUR	DMA3 Bandwidth Limit Count Current Register	0x00000000
0x310221C8	DMA3_BWMCNT	DMA3 Bandwidth Monitor Count Register	0x00000000
0x310221CC	DMA3_BWMCNT_CUR	DMA3 Bandwidth Monitor Count Current Register	0x00000000

Table A-49: ADSP-2184x DMA30 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF000	DMA30_DSCPTR_NXT	DMA30 Pointer to Next Initial Descriptor Register	0x00000000
0x30FFF004	DMA30_ADDRSTART	DMA30 Start Address of Current Buffer Register	0x00000000
0x30FFF008	DMA30_CFG	DMA30 Configuration Register	0x00000000
0x30FFF00C	DMA30_XCNT	DMA30 Inner Loop Count Start Value Register	0x00000000
0x30FFF010	DMA30_XMOD	DMA30 Inner Loop Address Increment Register	0x00000000
0x30FFF014	DMA30_YCNT	DMA30 Outer Loop Count Start Value (2D only) Register	0x00000000
0x30FFF018	DMA30_YMOD	DMA30 Outer Loop Address Increment (2D only) Register	0x00000000
0x30FFF024	DMA30_DSCPTR_CUR	DMA30 Current Descriptor Pointer Register	0x00000000

Table A-49: ADSP-2184x DMA30 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF028	DMA30_DSCPTR_PRV	DMA30 Previous Initial Descriptor Pointer Register	0x00000000
0x30FFF02C	DMA30_ADDR_CUR	DMA30 Current Address Register	0x00000000
0x30FFF030	DMA30_STAT	DMA30 Status Register	0x00006000
0x30FFF034	DMA30_XCNT_CUR	DMA30 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x30FFF038	DMA30_YCNT_CUR	DMA30 Current Row Count (2D only) Register	0x00000000
0x30FFF040	DMA30_BWLCNT	DMA30 Bandwidth Limit Count Register	0x00000000
0x30FFF044	DMA30_BWLCNT_CUR	DMA30 Bandwidth Limit Count Current Register	0x00000000
0x30FFF048	DMA30_BWMCNT	DMA30 Bandwidth Monitor Count Register	0x00000000
0x30FFF04C	DMA30_BWMCNT_CUR	DMA30 Bandwidth Monitor Count Current Register	0x00000000

Table A-50: ADSP-2184x DMA34 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026180	DMA34_DSCPTR_NXT	DMA34 Pointer to Next Initial Descriptor Register	0x00000000
0x31026184	DMA34_ADDRSTART	DMA34 Start Address of Current Buffer Register	0x00000000
0x31026188	DMA34_CFG	DMA34 Configuration Register	0x00000000
0x3102618C	DMA34_XCNT	DMA34 Inner Loop Count Start Value Register	0x00000000
0x31026190	DMA34_XMOD	DMA34 Inner Loop Address Increment Register	0x00000000
0x31026194	DMA34_YCNT	DMA34 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026198	DMA34_YMOD	DMA34 Outer Loop Address Increment (2D only) Register	0x00000000
0x310261A4	DMA34_DSCPTR_CUR	DMA34 Current Descriptor Pointer Register	0x00000000
0x310261A8	DMA34_DSCPTR_PRV	DMA34 Previous Initial Descriptor Pointer Register	0x00000000
0x310261AC	DMA34_ADDR_CUR	DMA34 Current Address Register	0x00000000
0x310261B0	DMA34_STAT	DMA34 Status Register	0x00006000
0x310261B4	DMA34_XCNT_CUR	DMA34 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310261B8	DMA34_YCNT_CUR	DMA34 Current Row Count (2D only) Register	0x00000000
0x310261C0	DMA34_BWLCNT	DMA34 Bandwidth Limit Count Register	0x00000000
0x310261C4	DMA34_BWLCNT_CUR	DMA34 Bandwidth Limit Count Current Register	0x00000000
0x310261C8	DMA34_BWMCNT	DMA34 Bandwidth Monitor Count Register	0x00000000

Table A-50: ADSP-2184x DMA34 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310261CC	DMA34_BWMCNT_CUR	DMA34 Bandwidth Monitor Count Current Register	0x00000000

Table A-51: ADSP-2184x DMA35 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026100	DMA35_DSCPTR_NXT	DMA35 Pointer to Next Initial Descriptor Register	0x00000000
0x31026104	DMA35_ADDRSTART	DMA35 Start Address of Current Buffer Register	0x00000000
0x31026108	DMA35_CFG	DMA35 Configuration Register	0x00000000
0x3102610C	DMA35_XCNT	DMA35 Inner Loop Count Start Value Register	0x00000000
0x31026110	DMA35_XMOD	DMA35 Inner Loop Address Increment Register	0x00000000
0x31026114	DMA35_YCNT	DMA35 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026118	DMA35_YMOD	DMA35 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026124	DMA35_DSCPTR_CUR	DMA35 Current Descriptor Pointer Register	0x00000000
0x31026128	DMA35_DSCPTR_PRV	DMA35 Previous Initial Descriptor Pointer Register	0x00000000
0x3102612C	DMA35_ADDR_CUR	DMA35 Current Address Register	0x00000000
0x31026130	DMA35_STAT	DMA35 Status Register	0x00006000
0x31026134	DMA35_XCNT_CUR	DMA35 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026138	DMA35_YCNT_CUR	DMA35 Current Row Count (2D only) Register	0x00000000
0x31026140	DMA35_BWLCNT	DMA35 Bandwidth Limit Count Register	0x00000000
0x31026144	DMA35_BWLCNT_CUR	DMA35 Bandwidth Limit Count Current Register	0x00000000
0x31026148	DMA35_BWMCNT	DMA35 Bandwidth Monitor Count Register	0x00000000
0x3102614C	DMA35_BWMCNT_CUR	DMA35 Bandwidth Monitor Count Current Register	0x00000000

Table A-52: ADSP-2184x DMA36 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF080	DMA36_DSCPTR_NXT	DMA36 Pointer to Next Initial Descriptor Register	0x00000000
0x30FFF084	DMA36_ADDRSTART	DMA36 Start Address of Current Buffer Register	0x00000000
0x30FFF088	DMA36_CFG	DMA36 Configuration Register	0x00000000
0x30FFF08C	DMA36_XCNT	DMA36 Inner Loop Count Start Value Register	0x00000000

Table A-52: ADSP-2184x DMA36 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF090	DMA36_XMOD	DMA36 Inner Loop Address Increment Register	0x00000000
0x30FFF094	DMA36_YCNT	DMA36 Outer Loop Count Start Value (2D only) Register	0x00000000
0x30FFF098	DMA36_YMOD	DMA36 Outer Loop Address Increment (2D only) Register	0x00000000
0x30FFF0A4	DMA36_DSCPTR_CUR	DMA36 Current Descriptor Pointer Register	0x00000000
0x30FFF0A8	DMA36_DSCPTR_PRV	DMA36 Previous Initial Descriptor Pointer Register	0x00000000
0x30FFF0AC	DMA36_ADDR_CUR	DMA36 Current Address Register	0x00000000
0x30FFF0B0	DMA36_STAT	DMA36 Status Register	0x00006000
0x30FFF0B4	DMA36_XCNT_CUR	DMA36 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x30FFF0B8	DMA36_YCNT_CUR	DMA36 Current Row Count (2D only) Register	0x00000000
0x30FFF0C0	DMA36_BWLCNT	DMA36 Bandwidth Limit Count Register	0x00000000
0x30FFF0C4	DMA36_BWLCNT_CUR	DMA36 Bandwidth Limit Count Current Register	0x00000000
0x30FFF0C8	DMA36_BWMCNT	DMA36 Bandwidth Monitor Count Register	0x00000000
0x30FFF0CC	DMA36_BWMCNT_CUR	DMA36 Bandwidth Monitor Count Current Register	0x00000000

Table A-53: ADSP-2184x DMA37 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026280	DMA37_DSCPTR_NXT	DMA37 Pointer to Next Initial Descriptor Register	0x00000000
0x31026284	DMA37_ADDRSTART	DMA37 Start Address of Current Buffer Register	0x00000000
0x31026288	DMA37_CFG	DMA37 Configuration Register	0x00000000
0x3102628C	DMA37_XCNT	DMA37 Inner Loop Count Start Value Register	0x00000000
0x31026290	DMA37_XMOD	DMA37 Inner Loop Address Increment Register	0x00000000
0x31026294	DMA37_YCNT	DMA37 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026298	DMA37_YMOD	DMA37 Outer Loop Address Increment (2D only) Register	0x00000000
0x310262A4	DMA37_DSCPTR_CUR	DMA37 Current Descriptor Pointer Register	0x00000000
0x310262A8	DMA37_DSCPTR_PRV	DMA37 Previous Initial Descriptor Pointer Register	0x00000000
0x310262AC	DMA37_ADDR_CUR	DMA37 Current Address Register	0x00000000
0x310262B0	DMA37_STAT	DMA37 Status Register	0x00006000

Table A-53: ADSP-2184x DMA37 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310262B4	DMA37_XCNT_CUR	DMA37 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310262B8	DMA37_YCNT_CUR	DMA37 Current Row Count (2D only) Register	0x00000000
0x310262C0	DMA37_BWLCNT	DMA37 Bandwidth Limit Count Register	0x00000000
0x310262C4	DMA37_BWLCNT_CUR	DMA37 Bandwidth Limit Count Current Register	0x00000000
0x310262C8	DMA37_BWMCNT	DMA37 Bandwidth Monitor Count Register	0x00000000
0x310262CC	DMA37_BWMCNT_CUR	DMA37 Bandwidth Monitor Count Current Register	0x00000000

Table A-54: ADSP-2184x DMA38 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026200	DMA38_DSCPTR_NXT	DMA38 Pointer to Next Initial Descriptor Register	0x00000000
0x31026204	DMA38_ADDRSTART	DMA38 Start Address of Current Buffer Register	0x00000000
0x31026208	DMA38_CFG	DMA38 Configuration Register	0x00000000
0x3102620C	DMA38_XCNT	DMA38 Inner Loop Count Start Value Register	0x00000000
0x31026210	DMA38_XMOD	DMA38 Inner Loop Address Increment Register	0x00000000
0x31026214	DMA38_YCNT	DMA38 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026218	DMA38_YMOD	DMA38 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026224	DMA38_DSCPTR_CUR	DMA38 Current Descriptor Pointer Register	0x00000000
0x31026228	DMA38_DSCPTR_PRV	DMA38 Previous Initial Descriptor Pointer Register	0x00000000
0x3102622C	DMA38_ADDR_CUR	DMA38 Current Address Register	0x00000000
0x31026230	DMA38_STAT	DMA38 Status Register	0x00006000
0x31026234	DMA38_XCNT_CUR	DMA38 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026238	DMA38_YCNT_CUR	DMA38 Current Row Count (2D only) Register	0x00000000
0x31026240	DMA38_BWLCNT	DMA38 Bandwidth Limit Count Register	0x00000000
0x31026244	DMA38_BWLCNT_CUR	DMA38 Bandwidth Limit Count Current Register	0x00000000
0x31026248	DMA38_BWMCNT	DMA38 Bandwidth Monitor Count Register	0x00000000
0x3102624C	DMA38_BWMCNT_CUR	DMA38 Bandwidth Monitor Count Current Register	0x00000000

Table A-55: ADSP-2184x DMA39 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109A000	DMA39_DSCPTR_NXT	DMA39 Pointer to Next Initial Descriptor Register	0x00000000
0x3109A004	DMA39_ADDRSTART	DMA39 Start Address of Current Buffer Register	0x00000000
0x3109A008	DMA39_CFG	DMA39 Configuration Register	0x00000220
0x3109A00C	DMA39_XCNT	DMA39 Inner Loop Count Start Value Register	0x00000000
0x3109A010	DMA39_XMOD	DMA39 Inner Loop Address Increment Register	0x00000000
0x3109A014	DMA39_YCNT	DMA39 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109A018	DMA39_YMOD	DMA39 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109A024	DMA39_DSCPTR_CUR	DMA39 Current Descriptor Pointer Register	0x00000000
0x3109A028	DMA39_DSCPTR_PRV	DMA39 Previous Initial Descriptor Pointer Register	0x00000000
0x3109A02C	DMA39_ADDR_CUR	DMA39 Current Address Register	0x00000000
0x3109A030	DMA39_STAT	DMA39 Status Register	0x00006000
0x3109A034	DMA39_XCNT_CUR	DMA39 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109A038	DMA39_YCNT_CUR	DMA39 Current Row Count (2D only) Register	0x00000000
0x3109A040	DMA39_BWLCNT	DMA39 Bandwidth Limit Count Register	0x00000000
0x3109A044	DMA39_BWLCNT_CUR	DMA39 Bandwidth Limit Count Current Register	0x00000000
0x3109A048	DMA39_BWMCNT	DMA39 Bandwidth Monitor Count Register	0x00000000
0x3109A04C	DMA39_BWMCNT_CUR	DMA39 Bandwidth Monitor Count Current Register	0x00000000

Table A-56: ADSP-2184x DMA4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022200	DMA4_DSCPTR_NXT	DMA4 Pointer to Next Initial Descriptor Register	0x00000000
0x31022204	DMA4_ADDRSTART	DMA4 Start Address of Current Buffer Register	0x00000000
0x31022208	DMA4_CFG	DMA4 Configuration Register	0x00000000
0x3102220C	DMA4_XCNT	DMA4 Inner Loop Count Start Value Register	0x00000000
0x31022210	DMA4_XMOD	DMA4 Inner Loop Address Increment Register	0x00000000
0x31022214	DMA4_YCNT	DMA4 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022218	DMA4_YMOD	DMA4 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022224	DMA4_DSCPTR_CUR	DMA4 Current Descriptor Pointer Register	0x00000000

Table A-56: ADSP-2184x DMA4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31022228	DMA4_DSCPTR_PRV	DMA4 Previous Initial Descriptor Pointer Register	0x00000000
0x3102222C	DMA4_ADDR_CUR	DMA4 Current Address Register	0x00000000
0x31022230	DMA4_STAT	DMA4 Status Register	0x00006000
0x31022234	DMA4_XCNT_CUR	DMA4 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022238	DMA4_YCNT_CUR	DMA4 Current Row Count (2D only) Register	0x00000000
0x31022240	DMA4_BWLCNT	DMA4 Bandwidth Limit Count Register	0x00000000
0x31022244	DMA4_BWLCNT_CUR	DMA4 Bandwidth Limit Count Current Register	0x00000000
0x31022248	DMA4_BWMCNT	DMA4 Bandwidth Monitor Count Register	0x00000000
0x3102224C	DMA4_BWMCNT_CUR	DMA4 Bandwidth Monitor Count Current Register	0x00000000

Table A-57: ADSP-2184x DMA40 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109A080	DMA40_DSCPTR_NXT	DMA40 Pointer to Next Initial Descriptor Register	0x00000000
0x3109A084	DMA40_ADDRSTART	DMA40 Start Address of Current Buffer Register	0x00000000
0x3109A088	DMA40_CFG	DMA40 Configuration Register	0x00000222
0x3109A08C	DMA40_XCNT	DMA40 Inner Loop Count Start Value Register	0x00000000
0x3109A090	DMA40_XMOD	DMA40 Inner Loop Address Increment Register	0x00000000
0x3109A094	DMA40_YCNT	DMA40 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109A098	DMA40_YMOD	DMA40 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109A0A4	DMA40_DSCPTR_CUR	DMA40 Current Descriptor Pointer Register	0x00000000
0x3109A0A8	DMA40_DSCPTR_PRV	DMA40 Previous Initial Descriptor Pointer Register	0x00000000
0x3109A0AC	DMA40_ADDR_CUR	DMA40 Current Address Register	0x00000000
0x3109A0B0	DMA40_STAT	DMA40 Status Register	0x00006000
0x3109A0B4	DMA40_XCNT_CUR	DMA40 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109A0B8	DMA40_YCNT_CUR	DMA40 Current Row Count (2D only) Register	0x00000000
0x3109A0C0	DMA40_BWLCNT	DMA40 Bandwidth Limit Count Register	0x00000000
0x3109A0C4	DMA40_BWLCNT_CUR	DMA40 Bandwidth Limit Count Current Register	0x00000000
0x3109A0C8	DMA40_BWMCNT	DMA40 Bandwidth Monitor Count Register	0x00000000

Table A-57: ADSP-2184x DMA40 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109A0CC	DMA40_BWMCNT_CUR	DMA40 Bandwidth Monitor Count Current Register	0x00000000

Table A-58: ADSP-2184x DMA43 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109B000	DMA43_DSCPTR_NXT	DMA43 Pointer to Next Initial Descriptor Register	0x00000000
0x3109B004	DMA43_ADDRSTART	DMA43 Start Address of Current Buffer Register	0x00000000
0x3109B008	DMA43_CFG	DMA43 Configuration Register	0x00000220
0x3109B00C	DMA43_XCNT	DMA43 Inner Loop Count Start Value Register	0x00000000
0x3109B010	DMA43_XMOD	DMA43 Inner Loop Address Increment Register	0x00000000
0x3109B014	DMA43_YCNT	DMA43 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109B018	DMA43_YMOD	DMA43 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109B024	DMA43_DSCPTR_CUR	DMA43 Current Descriptor Pointer Register	0x00000000
0x3109B028	DMA43_DSCPTR_PRV	DMA43 Previous Initial Descriptor Pointer Register	0x00000000
0x3109B02C	DMA43_ADDR_CUR	DMA43 Current Address Register	0x00000000
0x3109B030	DMA43_STAT	DMA43 Status Register	0x0000B000
0x3109B034	DMA43_XCNT_CUR	DMA43 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109B038	DMA43_YCNT_CUR	DMA43 Current Row Count (2D only) Register	0x00000000
0x3109B040	DMA43_BWLCNT	DMA43 Bandwidth Limit Count Register	0x00000000
0x3109B044	DMA43_BWLCNT_CUR	DMA43 Bandwidth Limit Count Current Register	0x00000000
0x3109B048	DMA43_BWMCNT	DMA43 Bandwidth Monitor Count Register	0x00000000
0x3109B04C	DMA43_BWMCNT_CUR	DMA43 Bandwidth Monitor Count Current Register	0x00000000

Table A-59: ADSP-2184x DMA44 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109B080	DMA44_DSCPTR_NXT	DMA44 Pointer to Next Initial Descriptor Register	0x00000000
0x3109B084	DMA44_ADDRSTART	DMA44 Start Address of Current Buffer Register	0x00000000
0x3109B088	DMA44_CFG	DMA44 Configuration Register	0x00000222
0x3109B08C	DMA44_XCNT	DMA44 Inner Loop Count Start Value Register	0x00000000

Table A-59: ADSP-2184x DMA44 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109B090	DMA44_XMOD	DMA44 Inner Loop Address Increment Register	0x00000000
0x3109B094	DMA44_YCNT	DMA44 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109B098	DMA44_YMOD	DMA44 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109B0A4	DMA44_DSCPTR_CUR	DMA44 Current Descriptor Pointer Register	0x00000000
0x3109B0A8	DMA44_DSCPTR_PRV	DMA44 Previous Initial Descriptor Pointer Register	0x00000000
0x3109B0AC	DMA44_ADDR_CUR	DMA44 Current Address Register	0x00000000
0x3109B0B0	DMA44_STAT	DMA44 Status Register	0x0000B000
0x3109B0B4	DMA44_XCNT_CUR	DMA44 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109B0B8	DMA44_YCNT_CUR	DMA44 Current Row Count (2D only) Register	0x00000000
0x3109B0C0	DMA44_BWLCNT	DMA44 Bandwidth Limit Count Register	0x00000000
0x3109B0C4	DMA44_BWLCNT_CUR	DMA44 Bandwidth Limit Count Current Register	0x00000000
0x3109B0C8	DMA44_BWMCNT	DMA44 Bandwidth Monitor Count Register	0x00000000
0x3109B0CC	DMA44_BWMCNT_CUR	DMA44 Bandwidth Monitor Count Current Register	0x00000000

Table A-60: ADSP-2184x DMA45 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7200	DMA45_DSCPTR_NXT	DMA45 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7204	DMA45_ADDRSTART	DMA45 Start Address of Current Buffer Register	0x00000000
0x310A7208	DMA45_CFG	DMA45 Configuration Register	0x00000000
0x310A720C	DMA45_XCNT	DMA45 Inner Loop Count Start Value Register	0x00000000
0x310A7210	DMA45_XMOD	DMA45 Inner Loop Address Increment Register	0x00000000
0x310A7214	DMA45_YCNT	DMA45 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7218	DMA45_YMOD	DMA45 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7224	DMA45_DSCPTR_CUR	DMA45 Current Descriptor Pointer Register	0x00000000
0x310A7228	DMA45_DSCPTR_PRV	DMA45 Previous Initial Descriptor Pointer Register	0x00000000
0x310A722C	DMA45_ADDR_CUR	DMA45 Current Address Register	0x00000000
0x310A7230	DMA45_STAT	DMA45 Status Register	0x00006000

Table A-60: ADSP-2184x DMA45 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7234	DMA45_XCNT_CUR	DMA45 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7238	DMA45_YCNT_CUR	DMA45 Current Row Count (2D only) Register	0x00000000
0x310A7240	DMA45_BWLCNT	DMA45 Bandwidth Limit Count Register	0x00000000
0x310A7244	DMA45_BWLCNT_CUR	DMA45 Bandwidth Limit Count Current Register	0x00000000
0x310A7248	DMA45_BWMCNT	DMA45 Bandwidth Monitor Count Register	0x00000000
0x310A724C	DMA45_BWMCNT_CUR	DMA45 Bandwidth Monitor Count Current Register	0x00000000

Table A-61: ADSP-2184x DMA46 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7280	DMA46_DSCPTR_NXT	DMA46 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7284	DMA46_ADDRSTART	DMA46 Start Address of Current Buffer Register	0x00000000
0x310A7288	DMA46_CFG	DMA46 Configuration Register	0x00000000
0x310A728C	DMA46_XCNT	DMA46 Inner Loop Count Start Value Register	0x00000000
0x310A7290	DMA46_XMOD	DMA46 Inner Loop Address Increment Register	0x00000000
0x310A7294	DMA46_YCNT	DMA46 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7298	DMA46_YMOD	DMA46 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A72A4	DMA46_DSCPTR_CUR	DMA46 Current Descriptor Pointer Register	0x00000000
0x310A72A8	DMA46_DSCPTR_PRV	DMA46 Previous Initial Descriptor Pointer Register	0x00000000
0x310A72AC	DMA46_ADDR_CUR	DMA46 Current Address Register	0x00000000
0x310A72B0	DMA46_STAT	DMA46 Status Register	0x00006000
0x310A72B4	DMA46_XCNT_CUR	DMA46 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A72B8	DMA46_YCNT_CUR	DMA46 Current Row Count (2D only) Register	0x00000000
0x310A72C0	DMA46_BWLCNT	DMA46 Bandwidth Limit Count Register	0x00000000
0x310A72C4	DMA46_BWLCNT_CUR	DMA46 Bandwidth Limit Count Current Register	0x00000000
0x310A72C8	DMA46_BWMCNT	DMA46 Bandwidth Monitor Count Register	0x00000000
0x310A72CC	DMA46_BWMCNT_CUR	DMA46 Bandwidth Monitor Count Current Register	0x00000000

Table A-62: ADSP-2184x DMA47 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7300	DMA47_DSCPTR_NXT	DMA47 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7304	DMA47_ADDRSTART	DMA47 Start Address of Current Buffer Register	0x00000000
0x310A7308	DMA47_CFG	DMA47 Configuration Register	0x00000000
0x310A730C	DMA47_XCNT	DMA47 Inner Loop Count Start Value Register	0x00000000
0x310A7310	DMA47_XMOD	DMA47 Inner Loop Address Increment Register	0x00000000
0x310A7314	DMA47_YCNT	DMA47 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7318	DMA47_YMOD	DMA47 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7324	DMA47_DSCPTR_CUR	DMA47 Current Descriptor Pointer Register	0x00000000
0x310A7328	DMA47_DSCPTR_PRV	DMA47 Previous Initial Descriptor Pointer Register	0x00000000
0x310A732C	DMA47_ADDR_CUR	DMA47 Current Address Register	0x00000000
0x310A7330	DMA47_STAT	DMA47 Status Register	0x00006000
0x310A7334	DMA47_XCNT_CUR	DMA47 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7338	DMA47_YCNT_CUR	DMA47 Current Row Count (2D only) Register	0x00000000
0x310A7340	DMA47_BWLCNT	DMA47 Bandwidth Limit Count Register	0x00000000
0x310A7344	DMA47_BWLCNT_CUR	DMA47 Bandwidth Limit Count Current Register	0x00000000
0x310A7348	DMA47_BWMCNT	DMA47 Bandwidth Monitor Count Register	0x00000000
0x310A734C	DMA47_BWMCNT_CUR	DMA47 Bandwidth Monitor Count Current Register	0x00000000

Table A-63: ADSP-2184x DMA48 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7380	DMA48_DSCPTR_NXT	DMA48 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7384	DMA48_ADDRSTART	DMA48 Start Address of Current Buffer Register	0x00000000
0x310A7388	DMA48_CFG	DMA48 Configuration Register	0x00000000
0x310A738C	DMA48_XCNT	DMA48 Inner Loop Count Start Value Register	0x00000000
0x310A7390	DMA48_XMOD	DMA48 Inner Loop Address Increment Register	0x00000000
0x310A7394	DMA48_YCNT	DMA48 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7398	DMA48_YMOD	DMA48 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-63: ADSP-2184x DMA48 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A73A4	DMA48_DSCPTR_CUR	DMA48 Current Descriptor Pointer Register	0x00000000
0x310A73A8	DMA48_DSCPTR_PRV	DMA48 Previous Initial Descriptor Pointer Register	0x00000000
0x310A73AC	DMA48_ADDR_CUR	DMA48 Current Address Register	0x00000000
0x310A73B0	DMA48_STAT	DMA48 Status Register	0x00006000
0x310A73B4	DMA48_XCNT_CUR	DMA48 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A73B8	DMA48_YCNT_CUR	DMA48 Current Row Count (2D only) Register	0x00000000
0x310A73C0	DMA48_BWLCNT	DMA48 Bandwidth Limit Count Register	0x00000000
0x310A73C4	DMA48_BWLCNT_CUR	DMA48 Bandwidth Limit Count Current Register	0x00000000
0x310A73C8	DMA48_BWMCNT	DMA48 Bandwidth Monitor Count Register	0x00000000
0x310A73CC	DMA48_BWMCNT_CUR	DMA48 Bandwidth Monitor Count Current Register	0x00000000

Table A-64: ADSP-2184x DMA49 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC000	DMA49_DSCPTR_NXT	DMA49 Pointer to Next Initial Descriptor Register	0x00000000
0x310AC004	DMA49_ADDRSTART	DMA49 Start Address of Current Buffer Register	0x00000000
0x310AC008	DMA49_CFG	DMA49 Configuration Register	0x00000220
0x310AC00C	DMA49_XCNT	DMA49 Inner Loop Count Start Value Register	0x00000000
0x310AC010	DMA49_XMOD	DMA49 Inner Loop Address Increment Register	0x00000000
0x310AC014	DMA49_YCNT	DMA49 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310AC018	DMA49_YMOD	DMA49 Outer Loop Address Increment (2D only) Register	0x00000000
0x310AC024	DMA49_DSCPTR_CUR	DMA49 Current Descriptor Pointer Register	0x00000000
0x310AC028	DMA49_DSCPTR_PRV	DMA49 Previous Initial Descriptor Pointer Register	0x00000000
0x310AC02C	DMA49_ADDR_CUR	DMA49 Current Address Register	0x00000000
0x310AC030	DMA49_STAT	DMA49 Status Register	0x00006000
0x310AC034	DMA49_XCNT_CUR	DMA49 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310AC038	DMA49_YCNT_CUR	DMA49 Current Row Count (2D only) Register	0x00000000
0x310AC040	DMA49_BWLCNT	DMA49 Bandwidth Limit Count Register	0x00000000
0x310AC044	DMA49_BWLCNT_CUR	DMA49 Bandwidth Limit Count Current Register	0x00000000

Table A-64: ADSP-2184x DMA49 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC048	DMA49_BWMCNT	DMA49 Bandwidth Monitor Count Register	0x00000000
0x310AC04C	DMA49_BWMCNT_CUR	DMA49 Bandwidth Monitor Count Current Register	0x00000000

Table A-65: ADSP-2184x DMA5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022280	DMA5_DSCPTR_NXT	DMA5 Pointer to Next Initial Descriptor Register	0x00000000
0x31022284	DMA5_ADDRSTART	DMA5 Start Address of Current Buffer Register	0x00000000
0x31022288	DMA5_CFG	DMA5 Configuration Register	0x00000000
0x3102228C	DMA5_XCNT	DMA5 Inner Loop Count Start Value Register	0x00000000
0x31022290	DMA5_XMOD	DMA5 Inner Loop Address Increment Register	0x00000000
0x31022294	DMA5_YCNT	DMA5 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022298	DMA5_YMOD	DMA5 Outer Loop Address Increment (2D only) Register	0x00000000
0x310222A4	DMA5_DSCPTR_CUR	DMA5 Current Descriptor Pointer Register	0x00000000
0x310222A8	DMA5_DSCPTR_PRV	DMA5 Previous Initial Descriptor Pointer Register	0x00000000
0x310222AC	DMA5_ADDR_CUR	DMA5 Current Address Register	0x00000000
0x310222B0	DMA5_STAT	DMA5 Status Register	0x00006000
0x310222B4	DMA5_XCNT_CUR	DMA5 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310222B8	DMA5_YCNT_CUR	DMA5 Current Row Count (2D only) Register	0x00000000
0x310222C0	DMA5_BWLCNT	DMA5 Bandwidth Limit Count Register	0x00000000
0x310222C4	DMA5_BWLCNT_CUR	DMA5 Bandwidth Limit Count Current Register	0x00000000
0x310222C8	DMA5_BWMCNT	DMA5 Bandwidth Monitor Count Register	0x00000000
0x310222CC	DMA5_BWMCNT_CUR	DMA5 Bandwidth Monitor Count Current Register	0x00000000

Table A-66: ADSP-2184x DMA50 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC080	DMA50_DSCPTR_NXT	DMA50 Pointer to Next Initial Descriptor Register	0x00000000
0x310AC084	DMA50_ADDRSTART	DMA50 Start Address of Current Buffer Register	0x00000000
0x310AC088	DMA50_CFG	DMA50 Configuration Register	0x00000222
0x310AC08C	DMA50_XCNT	DMA50 Inner Loop Count Start Value Register	0x00000000

Table A-66: ADSP-2184x DMA50 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC090	DMA50_XMOD	DMA50 Inner Loop Address Increment Register	0x00000000
0x310AC094	DMA50_YCNT	DMA50 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310AC098	DMA50_YMOD	DMA50 Outer Loop Address Increment (2D only) Register	0x00000000
0x310AC0A4	DMA50_DSCPTR_CUR	DMA50 Current Descriptor Pointer Register	0x00000000
0x310AC0A8	DMA50_DSCPTR_PRV	DMA50 Previous Initial Descriptor Pointer Register	0x00000000
0x310AC0AC	DMA50_ADDR_CUR	DMA50 Current Address Register	0x00000000
0x310AC0B0	DMA50_STAT	DMA50 Status Register	0x00006000
0x310AC0B4	DMA50_XCNT_CUR	DMA50 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310AC0B8	DMA50_YCNT_CUR	DMA50 Current Row Count (2D only) Register	0x00000000
0x310AC0C0	DMA50_BWLCNT	DMA50 Bandwidth Limit Count Register	0x00000000
0x310AC0C4	DMA50_BWLCNT_CUR	DMA50 Bandwidth Limit Count Current Register	0x00000000
0x310AC0C8	DMA50_BWMCNT	DMA50 Bandwidth Monitor Count Register	0x00000000
0x310AC0CC	DMA50_BWMCNT_CUR	DMA50 Bandwidth Monitor Count Current Register	0x00000000

Table A-67: ADSP-2184x DMA51 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109C000	DMA51_DSCPTR_NXT	DMA51 Pointer to Next Initial Descriptor Register	0x00000000
0x3109C004	DMA51_ADDRSTART	DMA51 Start Address of Current Buffer Register	0x00000000
0x3109C008	DMA51_CFG	DMA51 Configuration Register	0x00000220
0x3109C00C	DMA51_XCNT	DMA51 Inner Loop Count Start Value Register	0x00000000
0x3109C010	DMA51_XMOD	DMA51 Inner Loop Address Increment Register	0x00000000
0x3109C014	DMA51_YCNT	DMA51 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109C018	DMA51_YMOD	DMA51 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109C024	DMA51_DSCPTR_CUR	DMA51 Current Descriptor Pointer Register	0x00000000
0x3109C028	DMA51_DSCPTR_PRV	DMA51 Previous Initial Descriptor Pointer Register	0x00000000
0x3109C02C	DMA51_ADDR_CUR	DMA51 Current Address Register	0x00000000
0x3109C030	DMA51_STAT	DMA51 Status Register	0x0000B000

Table A-67: ADSP-2184x DMA51 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109C034	DMA51_XCNT_CUR	DMA51 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109C038	DMA51_YCNT_CUR	DMA51 Current Row Count (2D only) Register	0x00000000
0x3109C040	DMA51_BWLCNT	DMA51 Bandwidth Limit Count Register	0x00000000
0x3109C044	DMA51_BWLCNT_CUR	DMA51 Bandwidth Limit Count Current Register	0x00000000
0x3109C048	DMA51_BWMCNT	DMA51 Bandwidth Monitor Count Register	0x00000000
0x3109C04C	DMA51_BWMCNT_CUR	DMA51 Bandwidth Monitor Count Current Register	0x00000000

Table A-68: ADSP-2184x DMA52 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109C080	DMA52_DSCPTR_NXT	DMA52 Pointer to Next Initial Descriptor Register	0x00000000
0x3109C084	DMA52_ADDRSTART	DMA52 Start Address of Current Buffer Register	0x00000000
0x3109C088	DMA52_CFG	DMA52 Configuration Register	0x00000222
0x3109C08C	DMA52_XCNT	DMA52 Inner Loop Count Start Value Register	0x00000000
0x3109C090	DMA52_XMOD	DMA52 Inner Loop Address Increment Register	0x00000000
0x3109C094	DMA52_YCNT	DMA52 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109C098	DMA52_YMOD	DMA52 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109C0A4	DMA52_DSCPTR_CUR	DMA52 Current Descriptor Pointer Register	0x00000000
0x3109C0A8	DMA52_DSCPTR_PRV	DMA52 Previous Initial Descriptor Pointer Register	0x00000000
0x3109C0AC	DMA52_ADDR_CUR	DMA52 Current Address Register	0x00000000
0x3109C0B0	DMA52_STAT	DMA52 Status Register	0x0000B000
0x3109C0B4	DMA52_XCNT_CUR	DMA52 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109C0B8	DMA52_YCNT_CUR	DMA52 Current Row Count (2D only) Register	0x00000000
0x3109C0C0	DMA52_BWLCNT	DMA52 Bandwidth Limit Count Register	0x00000000
0x3109C0C4	DMA52_BWLCNT_CUR	DMA52 Bandwidth Limit Count Current Register	0x00000000
0x3109C0C8	DMA52_BWMCNT	DMA52 Bandwidth Monitor Count Register	0x00000000
0x3109C0CC	DMA52_BWMCNT_CUR	DMA52 Bandwidth Monitor Count Current Register	0x00000000

Table A-69: ADSP-2184x DMA59 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D500	DMA59_DSCPTR_NXT	DMA59 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D504	DMA59_ADDRSTART	DMA59 Start Address of Current Buffer Register	0x00000000
0x3102D508	DMA59_CFG	DMA59 Configuration Register	0x00000000
0x3102D50C	DMA59_XCNT	DMA59 Inner Loop Count Start Value Register	0x00000000
0x3102D510	DMA59_XMOD	DMA59 Inner Loop Address Increment Register	0x00000000
0x3102D514	DMA59_YCNT	DMA59 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D518	DMA59_YMOD	DMA59 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D524	DMA59_DSCPTR_CUR	DMA59 Current Descriptor Pointer Register	0x00000000
0x3102D528	DMA59_DSCPTR_PRV	DMA59 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D52C	DMA59_ADDR_CUR	DMA59 Current Address Register	0x00000000
0x3102D530	DMA59_STAT	DMA59 Status Register	0x00006000
0x3102D534	DMA59_XCNT_CUR	DMA59 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D538	DMA59_YCNT_CUR	DMA59 Current Row Count (2D only) Register	0x00000000
0x3102D540	DMA59_BWLCNT	DMA59 Bandwidth Limit Count Register	0x00000000
0x3102D544	DMA59_BWLCNT_CUR	DMA59 Bandwidth Limit Count Current Register	0x00000000
0x3102D548	DMA59_BWMCNT	DMA59 Bandwidth Monitor Count Register	0x00000000
0x3102D54C	DMA59_BWMCNT_CUR	DMA59 Bandwidth Monitor Count Current Register	0x00000000

Table A-70: ADSP-2184x DMA6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022300	DMA6_DSCPTR_NXT	DMA6 Pointer to Next Initial Descriptor Register	0x00000000
0x31022304	DMA6_ADDRSTART	DMA6 Start Address of Current Buffer Register	0x00000000
0x31022308	DMA6_CFG	DMA6 Configuration Register	0x00000000
0x3102230C	DMA6_XCNT	DMA6 Inner Loop Count Start Value Register	0x00000000
0x31022310	DMA6_XMOD	DMA6 Inner Loop Address Increment Register	0x00000000
0x31022314	DMA6_YCNT	DMA6 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022318	DMA6_YMOD	DMA6 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022324	DMA6_DSCPTR_CUR	DMA6 Current Descriptor Pointer Register	0x00000000

Table A-70: ADSP-2184x DMA6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31022328	DMA6_DSCPTR_PRV	DMA6 Previous Initial Descriptor Pointer Register	0x00000000
0x3102232C	DMA6_ADDR_CUR	DMA6 Current Address Register	0x00000000
0x31022330	DMA6_STAT	DMA6 Status Register	0x00006000
0x31022334	DMA6_XCNT_CUR	DMA6 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022338	DMA6_YCNT_CUR	DMA6 Current Row Count (2D only) Register	0x00000000
0x31022340	DMA6_BWLCNT	DMA6 Bandwidth Limit Count Register	0x00000000
0x31022344	DMA6_BWLCNT_CUR	DMA6 Bandwidth Limit Count Current Register	0x00000000
0x31022348	DMA6_BWMCNT	DMA6 Bandwidth Monitor Count Register	0x00000000
0x3102234C	DMA6_BWMCNT_CUR	DMA6 Bandwidth Monitor Count Current Register	0x00000000

Table A-71: ADSP-2184x DMA60 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D580	DMA60_DSCPTR_NXT	DMA60 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D584	DMA60_ADDRSTART	DMA60 Start Address of Current Buffer Register	0x00000000
0x3102D588	DMA60_CFG	DMA60 Configuration Register	0x00000000
0x3102D58C	DMA60_XCNT	DMA60 Inner Loop Count Start Value Register	0x00000000
0x3102D590	DMA60_XMOD	DMA60 Inner Loop Address Increment Register	0x00000000
0x3102D594	DMA60_YCNT	DMA60 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D598	DMA60_YMOD	DMA60 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D5A4	DMA60_DSCPTR_CUR	DMA60 Current Descriptor Pointer Register	0x00000000
0x3102D5A8	DMA60_DSCPTR_PRV	DMA60 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D5AC	DMA60_ADDR_CUR	DMA60 Current Address Register	0x00000000
0x3102D5B0	DMA60_STAT	DMA60 Status Register	0x00006000
0x3102D5B4	DMA60_XCNT_CUR	DMA60 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D5B8	DMA60_YCNT_CUR	DMA60 Current Row Count (2D only) Register	0x00000000
0x3102D5C0	DMA60_BWLCNT	DMA60 Bandwidth Limit Count Register	0x00000000
0x3102D5C4	DMA60_BWLCNT_CUR	DMA60 Bandwidth Limit Count Current Register	0x00000000
0x3102D5C8	DMA60_BWMCNT	DMA60 Bandwidth Monitor Count Register	0x00000000

Table A-71: ADSP-2184x DMA60 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D5CC	DMA60_BWMCNT_CUR	DMA60 Bandwidth Monitor Count Current Register	0x00000000

Table A-72: ADSP-2184x DMA7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022380	DMA7_DSCPTR_NXT	DMA7 Pointer to Next Initial Descriptor Register	0x00000000
0x31022384	DMA7_ADDRSTART	DMA7 Start Address of Current Buffer Register	0x00000000
0x31022388	DMA7_CFG	DMA7 Configuration Register	0x00000000
0x3102238C	DMA7_XCNT	DMA7 Inner Loop Count Start Value Register	0x00000000
0x31022390	DMA7_XMOD	DMA7 Inner Loop Address Increment Register	0x00000000
0x31022394	DMA7_YCNT	DMA7 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022398	DMA7_YMOD	DMA7 Outer Loop Address Increment (2D only) Register	0x00000000
0x310223A4	DMA7_DSCPTR_CUR	DMA7 Current Descriptor Pointer Register	0x00000000
0x310223A8	DMA7_DSCPTR_PRV	DMA7 Previous Initial Descriptor Pointer Register	0x00000000
0x310223AC	DMA7_ADDR_CUR	DMA7 Current Address Register	0x00000000
0x310223B0	DMA7_STAT	DMA7 Status Register	0x00006000
0x310223B4	DMA7_XCNT_CUR	DMA7 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310223B8	DMA7_YCNT_CUR	DMA7 Current Row Count (2D only) Register	0x00000000
0x310223C0	DMA7_BWLCNT	DMA7 Bandwidth Limit Count Register	0x00000000
0x310223C4	DMA7_BWLCNT_CUR	DMA7 Bandwidth Limit Count Current Register	0x00000000
0x310223C8	DMA7_BWMCNT	DMA7 Bandwidth Monitor Count Register	0x00000000
0x310223CC	DMA7_BWMCNT_CUR	DMA7 Bandwidth Monitor Count Current Register	0x00000000

Table A-73: ADSP-2184x DMA8 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7000	DMA8_DSCPTR_NXT	DMA8 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7004	DMA8_ADDRSTART	DMA8 Start Address of Current Buffer Register	0x00000000
0x310A7008	DMA8_CFG	DMA8 Configuration Register	0x00000000
0x310A700C	DMA8_XCNT	DMA8 Inner Loop Count Start Value Register	0x00000000
0x310A7010	DMA8_XMOD	DMA8 Inner Loop Address Increment Register	0x00000000

Table A-73: ADSP-2184x DMA8 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7014	DMA8_YCNT	DMA8 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7018	DMA8_YMOD	DMA8 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7024	DMA8_DSCPTR_CUR	DMA8 Current Descriptor Pointer Register	0x00000000
0x310A7028	DMA8_DSCPTR_PRV	DMA8 Previous Initial Descriptor Pointer Register	0x00000000
0x310A702C	DMA8_ADDR_CUR	DMA8 Current Address Register	0x00000000
0x310A7030	DMA8_STAT	DMA8 Status Register	0x00006000
0x310A7034	DMA8_XCNT_CUR	DMA8 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7038	DMA8_YCNT_CUR	DMA8 Current Row Count (2D only) Register	0x00000000
0x310A7040	DMA8_BWLCNT	DMA8 Bandwidth Limit Count Register	0x00000000
0x310A7044	DMA8_BWLCNT_CUR	DMA8 Bandwidth Limit Count Current Register	0x00000000
0x310A7048	DMA8_BWMCNT	DMA8 Bandwidth Monitor Count Register	0x00000000
0x310A704C	DMA8_BWMCNT_CUR	DMA8 Bandwidth Monitor Count Current Register	0x00000000

Table A-74: ADSP-2184x DMA9 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7080	DMA9_DSCPTR_NXT	DMA9 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7084	DMA9_ADDRSTART	DMA9 Start Address of Current Buffer Register	0x00000000
0x310A7088	DMA9_CFG	DMA9 Configuration Register	0x00000000
0x310A708C	DMA9_XCNT	DMA9 Inner Loop Count Start Value Register	0x00000000
0x310A7090	DMA9_XMOD	DMA9 Inner Loop Address Increment Register	0x00000000
0x310A7094	DMA9_YCNT	DMA9 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7098	DMA9_YMOD	DMA9 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A70A4	DMA9_DSCPTR_CUR	DMA9 Current Descriptor Pointer Register	0x00000000
0x310A70A8	DMA9_DSCPTR_PRV	DMA9 Previous Initial Descriptor Pointer Register	0x00000000
0x310A70AC	DMA9_ADDR_CUR	DMA9 Current Address Register	0x00000000
0x310A70B0	DMA9_STAT	DMA9 Status Register	0x00006000
0x310A70B4	DMA9_XCNT_CUR	DMA9 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A70B8	DMA9_YCNT_CUR	DMA9 Current Row Count (2D only) Register	0x00000000
0x310A70C0	DMA9_BWLCNT	DMA9 Bandwidth Limit Count Register	0x00000000

Table A-74: ADSP-2184x DMA9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A70C4	DMA9_BWLCNT_CUR	DMA9 Bandwidth Limit Count Current Register	0x00000000
0x310A70C8	DMA9_BWMCNT	DMA9 Bandwidth Monitor Count Register	0x00000000
0x310A70CC	DMA9_BWMCNT_CUR	DMA9 Bandwidth Monitor Count Current Register	0x00000000

Table A-75: ADSP-2184x DPM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31090000	DPM0_CTL	DPM0 Control Register	0x00000000
0x31090004	DPM0_STAT	DPM0 Status Register	0x00000001
0x31090070	DPM0_PER_DIS0	DPM0 Peripherals Disable Register 0	0x00000000
0x31090074	DPM0_PER_DIS1	DPM0 Peripherals Disable Register 1	0x00000000
0x31090084	DPM0_REVID	DPM0 Revision ID	0x00000020

Table A-76: ADSP-2184x LPDDR4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31158000	LPDDR4_CTLR	LPDDR4 Controller Register 0	0x03040000
0x31158004	LPDDR4_STAT	LPDDR4 Operating Mode Status Register	0x00000000
0x31158010	LPDDR4_MRCTRL0	LPDDR4 Mode Register Read/Write Control Register 0	0x00000030
0x31158014	LPDDR4_MRCTRL1	LPDDR4 Mode Register Read/Write Control Register 1	0x00000000
0x31158018	LPDDR4_MRSTAT	LPDDR4 Mode Register Read/Write Status Register	0x00000000
0x31158020	LPDDR4_DERATEEN	LPDDR4 Temperature Derate Enable Register	0x00000000
0x31158024	LPDDR4_DERATEINT	LPDDR4 Temperature Derate Interval Register	0x00800000
0x3115802C	LPDDR4_DERATECTL	LPDDR4 Temperature Derate Control Register	0x00000001
0x31158030	LPDDR4_PWRCTL	LPDDR4 Low Power Control Register	0x00000000
0x31158034	LPDDR4_PWRTMG	LPDDR4 Low Power Timing Register	0x00402010
0x31158038	LPDDR4_HWLPCTL	LPDDR4 Hardware Low Power Control Register	0x00000003
0x31158050	LPDDR4_RFSHCTL0	LPDDR4 Refresh Control Register 0	0x80210000
0x31158054	LPDDR4_RFSHCTL1	LPDDR4 Refresh Control Register 1	0x00000000
0x31158060	LPDDR4_RFSHCTL3	LPDDR4 Refresh Control Register 3	0x00000000
0x31158064	LPDDR4_RFSHTMG	LPDDR4 Refresh Timing Register	0x0062008C
0x31158068	LPDDR4_RFSHTMG1	LPDDR4 Refresh Timing Register 1	0x008C0000

Table A-76: ADSP-2184x LPDDR4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31158070	LPDDR4_ECCCFG0	LPDDR4 ECC Configuration Register 0	0x003F7F00
0x31158074	LPDDR4_ECCCFG1	LPDDR4 ECC Configuration Register 1	0x00001FB0
0x31158078	LPDDR4_ECCSTAT	LPDDR4 SECDED ECC Status Register	0x00000000
0x3115807C	LPDDR4_ECCCTL	LPDDR4 ECC Clear Register	0x00000300
0x31158080	LPDDR4_ECCERRCNT	LPDDR4 ECC Error Counter Register	0x00000000
0x31158084	LPDDR4_ECCCADDR0	LPDDR4 ECC Corrected Error Address Register 0	0x00000000
0x31158088	LPDDR4_ECCCADDR1	LPDDR4 ECC Corrected Error Address Register 1	0x00000000
0x3115808C	LPDDR4_ECCCSYN0	LPDDR4 ECC Corrected Syndrome Register 0	0x00000000
0x31158090	LPDDR4_ECCCSYN1	LPDDR4 ECC Corrected Syndrome Register 1	0x00000000
0x31158094	LPDDR4_ECCCSYN2	LPDDR4 ECC Corrected Syndrome Register 2	0x00000000
0x31158098	LPDDR4_ECCBITMASK0	LPDDR4 ECC Corrected Data Bit Mask Register 0	0x00000000
0x3115809C	LPDDR4_ECCBITMASK1	LPDDR4 ECC Corrected Data Bit Mask Register 1	0x00000000
0x311580A0	LPDDR4_ECCBITMASK2	LPDDR4 ECC Corrected Data Bit Mask Register 2	0x00000000
0x311580A4	LPDDR4_ECCUADDR0	LPDDR4 ECC Uncorrected Error Address Register 0	0x00000000
0x311580A8	LPDDR4_ECCUADDR1	LPDDR4 ECC Uncorrected Error Address Register 1	0x00000000
0x311580AC	LPDDR4_ECCUSYN0	LPDDR4 ECC Uncorrected Syndrome Register 0	0x00000000
0x311580B0	LPDDR4_ECCUSYN1	LPDDR4 ECC Uncorrected Syndrome Register 1	0x00000000
0x311580B4	LPDDR4_ECCUSYN2	LPDDR4 ECC Uncorrected Syndrome Register 2	0x00000000
0x311580B8	LPDDR4_ECCPOISONADDR0	LPDDR4 ECC Data Poisoning Address Register 0	0x00000000
0x311580BC	LPDDR4_ECCPOISONADDR1	LPDDR4 ECC Data Poisoning Address Register 1	0x00000000
0x311580C0	LPDDR4_CRCPARCTL0	LPDDR4 CRC Parity Control Register 0	0x00000000
0x311580CC	LPDDR4_CRCPARSTAT	LPDDR4 CRC Parity Status Register	0x00000000
0x311580D0	LPDDR4_INIT0	LPDDR4 SDRAM Initialization Register 0	0x0002004E
0x311580D4	LPDDR4_INIT1	LPDDR4 SDRAM Initialization Register 1	0x00000000
0x311580D8	LPDDR4_INIT2	LPDDR4 SDRAM Initialization Register 2	0x00000D00
0x311580DC	LPDDR4_INIT3	LPDDR4 SDRAM Initialization Register 3	0x00000510
0x311580E0	LPDDR4_INIT4	LPDDR4 SDRAM Initialization Register 4	0x00000000
0x311580E4	LPDDR4_INIT5	LPDDR4 SDRAM Initialization Register 5	0x00100004
0x311580E8	LPDDR4_INIT6	LPDDR4 SDRAM Initialization Register 6	0x00000000

Table A-76: ADSP-2184x LPDDR4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311580EC	LPDDR4_INIT7	LPDDR4 SDRAM Initialization Register 7	0x00000000
0x311580F0	LPDDR4_DIMMCTL	LPDDR4 DIMM Control Register	0x00000000
0x311580F4	LPDDR4_RANKCTL	LPDDR4 Rank Control Register	0x0000066F
0x311580F8	LPDDR4_RANKCTL1	LPDDR4 Rank Control Register 1	0x00000008
0x31158100	LPDDR4_DRAMTMG0	LPDDR4 SDRAM Timing Register 0	0x0F101B0F
0x31158104	LPDDR4_DRAMTMG1	LPDDR4 SDRAM Timing Register 1	0x00080414
0x31158108	LPDDR4_DRAMTMG2	LPDDR4 SDRAM Timing Register 2	0x0305060D
0x3115810C	LPDDR4_DRAMTMG3	LPDDR4 SDRAM Timing Register 3	0x00504000
0x31158110	LPDDR4_DRAMTMG4	LPDDR4 SDRAM Timing Register 4	0x05040405
0x31158114	LPDDR4_DRAMTMG5	LPDDR4 SDRAM Timing Register 5	0x05050403
0x31158118	LPDDR4_DRAMTMG6	LPDDR4 SDRAM Timing Register 6	0x02020005
0x3115811C	LPDDR4_DRAMTMG7	LPDDR4 SDRAM Timing Register 7	0x00000202
0x31158120	LPDDR4_DRAMTMG8	LPDDR4 SDRAM Timing Register 8	0x00004405
0x31158130	LPDDR4_DRAMTMG12	LPDDR4 SDRAM Timing Register 12	0x00020000
0x31158134	LPDDR4_DRAMTMG13	LPDDR4 SDRAM Timing Register 13	0x1C200004
0x31158138	LPDDR4_DRAMTMG14	LPDDR4 SDRAM Timing Register 14	0x000000A0
0x31158180	LPDDR4_ZQCTL0	LPDDR4 ZQ Control Register 0	0x02000040
0x31158184	LPDDR4_ZQCTL1	LPDDR4 ZQ Control Register 1	0x02000100
0x31158188	LPDDR4_ZQCTL2	LPDDR4 ZQ Control Register 2	0x00000000
0x3115818C	LPDDR4_ZQSTAT	LPDDR4 ZQ Status Register	0x00000000
0x31158190	LPDDR4_DFITMG0	LPDDR4 DFI Timing Register 0	0x07020002
0x31158194	LPDDR4_DFITMG1	LPDDR4 DFI Timing Register 1	0x00000404
0x31158198	LPDDR4_DFILPCFG0	LPDDR4 DFI Low Power Configuration Register 0	0x07000000
0x311581A0	LPDDR4_DFIUPD0	LPDDR4 DFI Update Register 0	0x00400003
0x311581A4	LPDDR4_DFIUPD1	LPDDR4 DFI Update Register 1	0x00010001
0x311581A8	LPDDR4_DFIUPD2	LPDDR4 DFI Update Register 2	0x80000000
0x311581B0	LPDDR4_DFIMISC	LPDDR4 DFI Miscellaneous Control Register	0x00000001
0x311581B4	LPDDR4_DFITMG2	LPDDR4 DFI Timing Register 2	0x00000202
0x311581BC	LPDDR4_DFISTAT	LPDDR4 DFI Status Register	0x00000000
0x311581C0	LPDDR4_DBICTL	LPDDR4 DM/DBI Control Register	0x00000001
0x311581C4	LPDDR4_DFIPHYCTLR	LPDDR4 DFI PHY Controller	0x80000001

Table A-76: ADSP-2184x LPDDR4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31158200	LPDDR4_ADDRMAP0	LPDDR4 Address Map Register 0	0x00000000
0x31158204	LPDDR4_ADDRMAP1	LPDDR4 Address Map Register 1	0x00000000
0x31158208	LPDDR4_ADDRMAP2	LPDDR4 Address Map Register 2	0x00000000
0x3115820C	LPDDR4_ADDRMAP3	LPDDR4 Address Map Register 3	0x00000000
0x31158210	LPDDR4_ADDRMAP4	LPDDR4 Address Map Register 4	0x00000000
0x31158214	LPDDR4_ADDRMAP5	LPDDR4 Address Map Register 5	0x00000000
0x31158218	LPDDR4_ADDRMAP6	LPDDR4 Address Map Register 6	0x00000000
0x3115821C	LPDDR4_ADDRMAP7	LPDDR4 Address Map Register 7	0x00000000
0x31158224	LPDDR4_ADDRMAP9	LPDDR4 Address Map Register 9	0x00000000
0x31158228	LPDDR4_ADDRMAP10	LPDDR4 Address Map Register 10	0x00000000
0x3115822C	LPDDR4_ADDRMAP11	LPDDR4 Address Map Register 11	0x00000000
0x31158240	LPDDR4_ODTCFG	LPDDR4 ODT Configuration Register	0x04000400
0x31158244	LPDDR4_ODTMAP	LPDDR4 ODT/Rank Map Register	0x00002211
0x31158250	LPDDR4_SCHED	LPDDR4 Scheduler Control Register	0x8000101D
0x31158254	LPDDR4_SCHED1	LPDDR4 Scheduler Control Register 1	0x00002000
0x3115825C	LPDDR4_PERFHPR1	LPDDR4 High Priority Read CAM Register 1	0x0F000001
0x31158264	LPDDR4_PERFLPR1	LPDDR4 Low Priority Read CAM Register 1	0x0F00007F
0x3115826C	LPDDR4_PERFWR1	LPDDR4 Write CAM Register 1	0x0F00007F
0x31158270	LPDDR4_SCHED3	LPDDR4 Scheduler Control Register 3	0x04040208
0x31158274	LPDDR4_SCHED4	LPDDR4 Scheduler Control Register 4	0x08400810
0x31158278	LPDDR4_SCHED5	LPDDR4 Scheduler Control Register 5	0x10000204
0x31158300	LPDDR4_DBG0	LPDDR4 Debug Register 0	0x00000000
0x31158304	LPDDR4_DBG1	LPDDR4 Debug Register 1	0x00000000
0x31158308	LPDDR4_DBGCAM	LPDDR4 CAM Debug Register	0x00000000
0x3115830C	LPDDR4_DBGCMD	LPDDR4 Command Debug Register	0x00000000
0x31158310	LPDDR4_DBGSTAT	LPDDR4 Status Debug Register	0x00000000
0x31158318	LPDDR4_DBGCAM1	LPDDR4 CAM Debug Register 1	0x00000000
0x31158320	LPDDR4_SWCTL	LPDDR4 Software Register Programming Control Enable	0x00000001
0x31158324	LPDDR4_SWSTAT	LPDDR4 Software Register Programming Control Status	0x00000001
0x31158328	LPDDR4_SWCTLSTATIC	LPDDR4 Static Registers Write Enable	0x00000000
0x3115836C	LPDDR4_POISONCFG	LPDDR4 AXI Poison Configuration Register	0x00110011

Table A-76: ADSP-2184x LPDDR4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31158370	LPDDR4_POISONSTAT	LPDDR4 AXI Poison Status Register	0x00000000
0x31158374	LPDDR4_ADVECCINDEX	LPDDR4 Advanced ECC Index Register	0x00000000
0x3115837C	LPDDR4_ECCPOISONPAT0	LPDDR4 ECC Poison Pattern 0 Register	0x00000000
0x31158384	LPDDR4_ECCPOISONPAT2	LPDDR4 ECC Poison Pattern 2 Register	0x00000000
0x311583F0	LPDDR4_DERATESTAT	LPDDR4 Temperature Derate Status Register	0x00000000
0x311583FC	LPDDR4_PSTAT	LPDDR4 Port Status Register.	0x00000000
0x31158400	LPDDR4_PCCFG	LPDDR4 Port Common Configuration Register.	0x00000000
0x31158404	LPDDR4_PCFGR_0	LPDDR4 Port n Configuration Read Register.	0x00004000
0x31158408	LPDDR4_PCFGW_0	LPDDR4 Port n Configuration Write Register.	0x00004000
0x31158490	LPDDR4_PCTRL_0	LPDDR4 Port n Control Register.	0x00000000
0x31158494	LPDDR4_PCFGQOS0_0	LPDDR4 Port n Read QoS Configuration Register 0.	0x00000000
0x31158498	LPDDR4_PCFGQOS1_0	LPDDR4 Port n Read QoS Configuration Register 1.	0x00000000
0x3115849C	LPDDR4_PCFGWQOS0_0	LPDDR4 Port n Write QoS Configuration Register 0.	0x00000E00
0x311584A0	LPDDR4_PCFGWQOS1_0	LPDDR4 Port n Write QoS Configuration Register 1.	0x00000000
0x31158F24	LPDDR4_SBRCTL	LPDDR4 Scrubber Control Register.	0x0000FF10
0x31158F28	LPDDR4_SBRSTAT	LPDDR4 Scrubber Status Register.	0x00000000
0x31158F2C	LPDDR4_SBRWDATA0	LPDDR4 Scrubber Write Data Pattern 0.	0x00000000
0x31158F38	LPDDR4_SBRSTART0	LPDDR4 Scrubber Start Address Mask Register 0.	0x00000000
0x31158F3C	LPDDR4_SBRSTART1	LPDDR4 Scrubber Start Address Mask Register 1.	0x00000000
0x31158F40	LPDDR4_SBRRANGE0	LPDDR4 Scrubber Address Range Mask Register 0.	0x00000000
0x31158F44	LPDDR4_SBRRANGE1	LPDDR4 Scrubber Address Range Mask Register 1.	0x00000000
0x31158FF0	LPDDR4_UMCTL2_VER_NUMBER	LPDDR4 UMCTL2 Version Number Register.	0x3339312A
0x31158FF4	LPDDR4_UMCTL2_VER_TYPE	LPDDR4 UMCTL2 Version Type Register.	0x67612A2A
0x3115A020	LPDDR4_DERATEEN	LPDDR4 Temperature Derate Enable Register.	0x00000000
0x3115A024	LPDDR4_DERATEINT	LPDDR4 Temperature Derate Interval Register.	0x00800000
0x3115A034	LPDDR4_PWRTMG	LPDDR4 Low Power Timing Register.	0x00402010
0x3115A050	LPDDR4_RFSHCTL0	LPDDR4 Refresh Control Register 0.	0x80210000
0x3115A064	LPDDR4_RFSHTMG	LPDDR4 Refresh Timing Register.	0x0062008C

Table A-76: ADSP-2184x LPDDR4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3115A068	LPDDR4_RFSHTMG1	LPDDR4 Refresh Timing Register 1.	0x008C0000
0x3115A0DC	LPDDR4_INIT3	LPDDR4 SDRAM Initialization Register 3.	0x00000510
0x3115A0E0	LPDDR4_INIT4	LPDDR4 SDRAM Initialization Register 4.	0x00000000
0x3115A0E8	LPDDR4_INIT6	LPDDR4 SDRAM Initialization Register 6.	0x00000000
0x3115A0EC	LPDDR4_INIT7	LPDDR4 SDRAM Initialization Register 7.	0x00000000
0x3115A0F4	LPDDR4_RANKCTL	LPDDR4 Rank Control Register.	0x0000066F
0x3115A0F8	LPDDR4_RANKCTL1	LPDDR4 Rank Control Register 1.	0x00000008
0x3115A100	LPDDR4_DRAMTMG0	LPDDR4 SDRAM Timing Register 0.	0x0F101B0F
0x3115A104	LPDDR4_DRAMTMG1	LPDDR4 SDRAM Timing Register 1.	0x00080414
0x3115A108	LPDDR4_DRAMTMG2	LPDDR4 SDRAM Timing Register 2.	0x0305060D
0x3115A10C	LPDDR4_DRAMTMG3	LPDDR4 SDRAM Timing Register 3.	0x00504000
0x3115A110	LPDDR4_DRAMTMG4	LPDDR4 SDRAM Timing Register 4.	0x05040405
0x3115A114	LPDDR4_DRAMTMG5	LPDDR4 SDRAM Timing Register 5.	0x05050403
0x3115A118	LPDDR4_DRAMTMG6	LPDDR4 SDRAM Timing Register 6.	0x02020005
0x3115A11C	LPDDR4_DRAMTMG7	LPDDR4 SDRAM Timing Register 7.	0x00000202
0x3115A120	LPDDR4_DRAMTMG8	LPDDR4 SDRAM Timing Register 8.	0x00004405
0x3115A130	LPDDR4_DRAMTMG12	LPDDR4 SDRAM Timing Register 12.	0x00020000
0x3115A134	LPDDR4_DRAMTMG13	LPDDR4 SDRAM Timing Register 13.	0x1C200004
0x3115A138	LPDDR4_DRAMTMG14	LPDDR4 SDRAM Timing Register 14.	0x000000A0
0x3115A180	LPDDR4_ZQCTL0	LPDDR4 ZQ Control Register 0.	0x02000040
0x3115A190	LPDDR4_DFITMG0	LPDDR4 DFI Timing Register 0.	0x07020002
0x3115A194	LPDDR4_DFITMG1	LPDDR4 DFI Timing Register 1.	0x00000404
0x3115A1B4	LPDDR4_DFITMG2	LPDDR4 DFI Timing Register 2.	0x00000202
0x3115A240	LPDDR4_ODTCFG	LPDDR4 ODT Configuration Register.	0x04000400

Table A-77: ADSP-2184x EMSI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31152000	EMSI0_SDMA_ADDR	EMSI0 SDMA System Address register.	0x00000000
0x31152004	EMSI0_BLKSZ	EMSI0 Block Size register.	0x00000000
0x31152006	EMSI0_BLKCNT	EMSI0 16-bit Block Count register.	0x00000000

Table A-77: ADSP-2184x EMSI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31152008	EMSI0_ARG	EMSI0 Argument register.	0x00000000
0x3115200C	EMSI0_TRNSFRMODE	EMSI0 Transfer Mode register.	0x00000000
0x3115200E	EMSI0_CMD	EMSI0 Command register.	0x00000000
0x31152010	EMSI0_RESP0	EMSI0 Response Register 0/1.	0x00000000
0x31152014	EMSI0_RESP1	EMSI0 Response Register 2/3.	0x00000000
0x31152018	EMSI0_RESP2	EMSI0 Response Register 4/5.	0x00000000
0x3115201C	EMSI0_RESP3	EMSI0 Response Register 6/7.	0x00000000
0x31152020	EMSI0_BUF_DATA	EMSI0 Buffer Data Port Register.	0x00000000
0x31152024	EMSI0_PSTATE	EMSI0 Present State Register.	0x00000000
0x31152028	EMSI0_CTL1	EMSI0 Host Control 1 Register.	0x00000000
0x31152029	EMSI0_PWR_CTL	EMSI0 Power Control Register.	0x00000000
0x3115202A	EMSI0_BGAP_CTL	EMSI0 Block Gap Control Register.	0x00000000
0x3115202B	EMSI0_WU_CTL	EMSI0 Wakeup Control Register.	0x00000000
0x3115202C	EMSI0_CLK_CTL	EMSI0 Clock Control Register.	0x00000000
0x3115202E	EMSI0_TO_CTL	EMSI0 Timeout Control Register.	0x00000000
0x3115202F	EMSI0_SWRST	EMSI0 Software Reset Register.	0x00000000
0x31152030	EMSI0_ISTAT	EMSI0 Normal Interrupt Status Register.	0x00000000
0x31152032	EMSI0_ERR_STAT	EMSI0 Error Interrupt Status Register.	0x00000000
0x31152034	EMSI0_ISTAT_EN	EMSI0 Normal Interrupt Status Enable Register.	0x00000000
0x31152036	EMSI0_ERR_STAT_EN	EMSI0 Error Interrupt Status Enable Register.	0x00000000
0x31152038	EMSI0_ISTAT_INTEN	EMSI0 Normal Interrupt Signal Enable Register.	0x00000000
0x3115203A	EMSI0_ERR_STAT_INTEN	EMSI0 Error Interrupt Signal Enable Register.	0x00000000
0x3115203C	EMSI0_AUTOCMD_STAT	EMSI0 Auto CMD Status Register.	0x00000000
0x3115203E	EMSI0_CTL2	EMSI0 Host Control 2 Register.	0x00000000
0x31152040	EMSI0_CAP1	EMSI0 Capabilities 1 Register - 0 to 31.	0x256EC88A
0x31152044	EMSI0_CAP2	EMSI0 Capabilities Register - 32 to 63.	0x0800AB77
0x31152050	EMSI0_FRC_AUTOCMD-STAT	EMSI0 Force Event Register for Auto CMD Error Status register.	0x00000000
0x31152052	EMSI0_FRC_ERRSTAT	EMSI0 Force Event Register for Error Interrupt Status.	0x00000000
0x31152054	EMSI0_ADMA_ERR_STAT	EMSI0 ADMA Error Status Register.	0x00000000
0x31152058	EMSI0_ADMA_ADDR_LO	EMSI0 ADMA System Address Register - Low.	0x00000000

Table A-77: ADSP-2184x EMSI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31152060	EMSI0_PRESET_INIT	EMSI0 Preset Value for Initialization.	0x000000FA
0x31152062	EMSI0_PRESET_DS	EMSI0 Preset Value for Default Speed.	0x00000004
0x31152064	EMSI0_PRESET_LEGACY	EMSI0 Preset Value for High Speed.	0x00000002
0x31152066	EMSI0_PRESET_HSSDR	EMSI0 Preset Value for SDR12.	0x00000004
0x31152068	EMSI0_PRESET_HSDDR	EMSI0 Preset Value for SDR25.	0x00000002
0x3115206A	EMSI0_PRESET_SDR50	EMSI0 Preset Value for SDR50.	0x00000001
0x3115206C	EMSI0_PRESET_SDR104	EMSI0 Preset Value for SDR104.	0x00000000
0x3115206E	EMSI0_PRESET_DDR50	EMSI0 Preset Value for DDR50.	0x00000002
0x31152074	EMSI0_PRESET_HS400	EMSI0 Preset Value for UHS-II.	0x00000000
0x31152078	EMSI0_ADMA_DESADDR_LO	EMSI0 ADMA3 Integrated Descriptor Address Register - Low.	0x00000000
0x311520E8	EMSI0_BASEADDR0	EMSI0 Pointer for Vendor Specific Area 1.	0x00000500
0x311520EA	EMSI0_BASEADDR1	EMSI0 Pointer for Vendor Specific Area 2.	0x00000180
0x311520FE	EMSI0_HOST_CNTRL_VERS	EMSI0 Host Controller Version.	0x00000005
0x31152180	EMSI0_CQVER	EMSI0 Command Queuing Version register.	0x00000510
0x31152184	EMSI0_CQ_CAP	EMSI0 Command Queuing Capabilities register.	0x00004000
0x31152188	EMSI0_CQ_CFG	EMSI0 Command Queuing Configuration register.	0x00000000
0x3115218C	EMSI0_CQ_CTL	EMSI0 Command Queuing Control register.	0x00000000
0x31152190	EMSI0_CQ_ISTAT	EMSI0 Command Queuing Interrupt Status register.	0x00000000
0x31152194	EMSI0_CQ_ISTAT_EN	EMSI0 Command Queuing Interrupt Status Enable register.	0x00000000
0x31152198	EMSI0_CQ_ISTAT_INTEN	EMSI0 Command Queuing Interrupt signal enable register.	0x00000000
0x3115219C	EMSI0_CQ_IC	EMSI0 Command Queuing Interrupt Coalescing register.	0x00000000
0x311521A0	EMSI0_CQ_TDL_BADDR	EMSI0 Command Queuing Task Descriptor List Base Address register.	0x00000000
0x311521A8	EMSI0_CQ_TDBR	EMSI0 Command Queuing DoorBell register.	0x00000000
0x311521AC	EMSI0_CQ_TCN	EMSI0 Command Queuing TaskClear Notification register.	0x00000000
0x311521B0	EMSI0_CQ_DQSTAT	EMSI0 Device queue status register.	0x00000000
0x311521B4	EMSI0_CQ_DPT	EMSI0 Device pending tasks register.	0x00000000
0x311521B8	EMSI0_CQ_TCLR	EMSI0 Command Queuing DoorBell register.	0x00000000

Table A-77: ADSP-2184x EMSI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311521C0	EMSI0_CQ_SSCFG1	EMSI0 CQ Send Status Configuration 1 register.	0x00011000
0x311521C4	EMSI0_CQ_SSCFG2	EMSI0 CQ Send Status Configuration 2 register.	0x00000000
0x311521C8	EMSI0_CQ_CRDCT	EMSI0 Command response for direct command register.	0x00000000
0x311521D0	EMSI0_CQ_RMEM	EMSI0 Command response mode error mask register.	0xFDF9A080
0x311521D4	EMSI0_CQ_TERRINFO	EMSI0 CQ Task Error Information register.	0x00000000
0x311521D8	EMSI0_CQ_CRI	EMSI0 CQ Command response index.	0x00000000
0x311521DC	EMSI0_CQ_CRARG	EMSI0 CQ Command response argument register.	0x00000000
0x31152300	EMSI0_PHY_CFG	EMSI0 SD/eMMC PHY General Configuration.	0x00000000
0x31152304	EMSI0_PHY_CMDPAD_CFG	EMSI0 SD/eMMC PHY CMD/RESP PAD Setting.	0x00000440
0x31152306	EMSI0_PHY_DATPAD_CFG	EMSI0 SD/eMMC PHY Data PAD Setting.	0x00000440
0x31152308	EMSI0_PHY_CLKPAD_CFG	EMSI0 SD/eMMC PHY Clock PAD Setting.	0x00000440
0x3115230A	EMSI0_PHY_STBPAD_CFG	EMSI0 SD/eMMC PHY Strobe PAD Setting.	0x00000440
0x3115230C	EMSI0_PHY_RSTPAD_CFG	EMSI0 SD/eMMC PHY RSTN PAD Setting.	0x00000440
0x3115231C	EMSI0_PHY_COMMDL_CFG	EMSI0 Common DelayLine config settings register.	0x00000000
0x3115231D	EMSI0_PHY_SDCLKDL_CFG	EMSI0 SD/eMMC DelayLine settings.	0x00000000
0x3115231E	EMSI0_PHY_SDCLKDL_DC	EMSI0 SD/eMMC DelayLine Delay code setting.	0x00000000
0x31152320	EMSI0_PHY_SMPLDL_CFG	EMSI0 SD/eMMC cclk_rx DelayLine settings.	0x0000000C
0x31152321	EMSI0_PHY_ATDL_CFG	EMSI0 SD/eMMC drift_cclk_rx DelayLine configuration settings.	0x00000000
0x31152324	EMSI0_PHY_DLL_CTL	EMSI0 SD/eMMC PHY DLL control setting.	0x00000000
0x31152325	EMSI0_PHY_DLL_CFG1	EMSI0 DLL Config register 1.	0x00000000
0x31152326	EMSI0_PHY_DLL_CFG2	EMSI0 DLL Config register 2.	0x00000000
0x31152328	EMSI0_PHY_DLLDL_CFG	EMSI0 DLL Config register 2.	0x00000000
0x31152329	EMSI0_PHY_DLL_OFFST	EMSI0 DLL Offset setting register.	0x00000000
0x3115232C	EMSI0_PHY_DLLLBT_CFG	EMSI0 DLL LBT setting register.	0x00000000
0x3115232E	EMSI0_PHY_DLL_STAT	EMSI0 DLL Status register.	0x00000000

Table A-77: ADSP-2184x EMSI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31152330	EMSI0_PHY_DLLDBG_MLKDC	EMSI0 DLL Manager lock code debug register.	0x00000000
0x31152332	EMSI0_PHY_DLLDBG_SLKDC	EMSI0 DLL Manager Subordinate code debug register.	0x00000000
0x31152334	EMSI0_PHY_TST_PHY1_DL0	EMSI0 Testmode Phy1 Delay line0 Settings register.	0x00000000
0x31152335	EMSI0_PHY_TST_PHY2_DL0	EMSI0 Testmode Phy2 Delay line0 Settings register.	0x00000000
0x31152336	EMSI0_PHY_TST_PHY2_DL1	EMSI0 Testmode Phy2 Delay line1 Settings register.	0x00000000
0x31152500	EMSI0_VER_ID	EMSI0 MSHC version.	0x30323030
0x31152504	EMSI0_VER_TYPE	EMSI0 MSHC version type.	0x67612A2A
0x31152508	EMSI0_MISC_CTL	EMSI0 MSHC Control register.	0x00000001
0x31152510	EMSI0_MBIU_CTL	EMSI0 MBIU Control register.	0x0000000F
0x3115252C	EMSI0_EMMC_CTL	EMSI0 eMMC Control register.	0x0000000C
0x3115252E	EMSI0_BOOT_CTL	EMSI0 eMMC Boot Control register.	0x00000000
0x31152540	EMSI0_AT_CTL	EMSI0 Tuning and Auto-tuning control register.	0x1F000005
0x31152544	EMSI0_AT_STAT	EMSI0 Tuning and Auto-tuning status register.	0x80000006

Table A-78: ADSP-2184x EMDMA MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310E002C	EMDMA_CTL	EMDMA External Memory DMA Control Register	0x00000000
0x310E002C	EMDMA0_CTL	EMDMA0 External Memory DMA Control Register	0x00000000
0x310E0080	EMDMA_INDX1	EMDMA External Index Register	0x00000000
0x310E0080	EMDMA0_INDX1	EMDMA0 External Index Register	0x00000000
0x310E0084	EMDMA_MOD1	EMDMA External Modifier Register	0x00000000
0x310E0084	EMDMA0_MOD1	EMDMA0 External Modifier Register	0x00000000
0x310E0088	EMDMA_CNT1	EMDMA External Count Register	0x00000000
0x310E0088	EMDMA0_CNT1	EMDMA0 External Count Register	0x00000000
0x310E008C	EMDMA_INDX0	EMDMA Internal Index Register	0x00000000
0x310E008C	EMDMA0_INDX0	EMDMA0 Internal Index Register	0x00000000

Table A-78: ADSP-2184x EMDMA MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310E0090	EMDMA_MOD0	EMDMA Internal Modifier Register	0x00000000
0x310E0090	EMDMA0_MOD0	EMDMA0 Internal Modifier Register	0x00000000
0x310E0094	EMDMA_CNT0	EMDMA Internal Count Register	0x00000000
0x310E0094	EMDMA0_CNT0	EMDMA0 Internal Count Register	0x00000000
0x310E0098	EMDMA_CHNPTR	EMDMA Chain Pointer Register	0x00000000
0x310E0098	EMDMA0_CHNPTR	EMDMA0 Chain Pointer Register	0x00000000
0x310E009C	EMDMA_BASE	EMDMA External Base Address Register	0x00000000
0x310E009C	EMDMA0_BASE	EMDMA0 External Base Address Register	0x00000000
0x310E00A0	EMDMA_TPTR	EMDMA Tap List Pointer Register	0x00000000
0x310E00A0	EMDMA0_TPTR	EMDMA0 Tap List Pointer Register	0x00000000
0x310E00A4	EMDMA_BUFLEN	EMDMA Circular Buffer Length Register	0x00000000
0x310E00A4	EMDMA0_BUFLEN	EMDMA0 Circular Buffer Length Register	0x00000000
0x310E00AC	EMDMA_TCNT	EMDMA Delay Line Tap Count Register	0x00000000
0x310E00AC	EMDMA0_TCNT	EMDMA0 Delay Line Tap Count Register	0x00000000

Table A-79: ADSP-2184x EMDMA1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310E0030	EMDMA1_CTL	EMDMA1 External Memory DMA Control Register	0x00000000
0x310E00C0	EMDMA1_INDX1	EMDMA1 External Index Register	0x00000000
0x310E00C4	EMDMA1_MOD1	EMDMA1 External Modifier Register	0x00000000
0x310E00C8	EMDMA1_CNT1	EMDMA1 External Count Register	0x00000000
0x310E00CC	EMDMA1_INDX0	EMDMA1 Internal Index Register	0x00000000
0x310E00D0	EMDMA1_MOD0	EMDMA1 Internal Modifier Register	0x00000000
0x310E00D4	EMDMA1_CNT0	EMDMA1 Internal Count Register	0x00000000
0x310E00D8	EMDMA1_CHNPTR	EMDMA1 Chain Pointer Register	0x00000000
0x310E00DC	EMDMA1_BASE	EMDMA1 External Base Address Register	0x00000000
0x310E00E0	EMDMA1_TPTR	EMDMA1 Tap List Pointer Register	0x00000000
0x310E00E4	EMDMA1_BUFLEN	EMDMA1 Circular Buffer Length Register	0x00000000
0x310E00EC	EMDMA1_TCNT	EMDMA1 Delay Line Tap Count Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31040000	EMAC0_MAC_CFG	EMAC0 MAC Configuration Register	0x00000000
0x31040004	EMAC0_MAC_EXT_CFG	EMAC0 MAC Extended Configuration Register	0x00000000
0x31040008	EMAC0_MACPKT_FILT	EMAC0 MAC Packet Filter Register	0x00000000
0x3104000C	EMAC0_WDT_JB_TMOUT	EMAC0 Watchdog and Jabber Timeout Register	0x00000000
0x31040010	EMAC0_HASHTBL_REG0	EMAC0 Hash Table Register 0	0x00000000
0x31040014	EMAC0_HASHTBL_REG1	EMAC0 Hash Table Register 1	0x00000000
0x31040018	EMAC0_HASHTBL_REG2	EMAC0 Hash Table Register 2	0x00000000
0x3104001C	EMAC0_HASHTBL_REG3	EMAC0 Hash Table Register 3	0x00000000
0x31040020	EMAC0_HASHTBL_REG4	EMAC0 Hash Table Register 4	0x00000000
0x31040024	EMAC0_HASHTBL_REG5	EMAC0 Hash Table Register 5	0x00000000
0x31040028	EMAC0_HASHTBL_REG6	EMAC0 Hash Table Register 6	0x00000000
0x3104002C	EMAC0_HASHTBL_REG7	EMAC0 Hash Table Register 7	0x00000000
0x31040050	EMAC0_VLANTAG_CTL	EMAC0 VLAN Tag Control Register	0x00000000
0x31040054	EMAC0_VLANTAG_DAT	EMAC0 VLAN Tag Indirect Access Data Register	0x00000000
0x31040058	EMAC0_VLAN_HASHTBL	EMAC0 VLAN Hash Table Register	0x00000000
0x31040060	EMAC0_VLAN_INCL	EMAC0 VLAN Tag Inclusion or Replacement Register	0x00000000
0x31040064	EMAC0_INVLAN_INCL	EMAC0 Inner VLAN Tag Inclusion or Replacement Register	0x00000000
0x31040070	EMAC0_Q0_TXFLOW_CTL	EMAC0 Queue 0 Transmit Flow Control Register	0x00000000
0x31040074	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040078	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x3104007C	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040080	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040084	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040088	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x3104008C	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040090	EMAC0_RXFLOW_CTL	EMAC0 Receive Flow Control Register	0x00000000
0x31040094	EMAC0_RXQ_CTL4	EMAC0 Receive Queue Control 4 Register	0x00000000
0x31040098	EMAC0_TXQ_PRTY_MAP0	EMAC0 Transmit Queue Priority Mapping 0 Register	0x00000000
0x3104009C	EMAC0_TXQ_PRTY_MAP1	EMAC0 Transmit Queue Priority Mapping 1 Register	0x00000000
0x310400A0	EMAC0_RXQ_CTL0	EMAC0 Receive Queue Control 0 Register	0x00000000
0x310400A4	EMAC0_RXQ_CTL1	EMAC0 Receive Queue Control 1 Register	0x00000000
0x310400A8	EMAC0_RXQ_CTL2	EMAC0 Receive Queue Control 2 Register	0x00000000
0x310400AC	EMAC0_RXQ_CTL3	EMAC0 Receive Queue Control 3 Register	0x00000000
0x310400B0	EMAC0_MAC_ISTAT	EMAC0 MAC Interrupt Status Register	0x00000000
0x310400B4	EMAC0_MAC_IEN	EMAC0 MAC Interrupt Enable Register	0x00000000
0x310400B8	EMAC0_RXTX_STAT	EMAC0 Receive Transmit Status Register	0x00000000
0x310400D0	EMAC0_LPI_CTLSTAT	EMAC0 LPI Control and Status Register	0x00000000
0x310400D4	EMAC0_LPI_TMRSCTL	EMAC0 LPI Timers Control Register	0x03E80000
0x310400D8	EMAC0_LPI_ENTR_TMR	EMAC0 Transmit LPI Entry Timer Register	0x00000000
0x310400DC	EMAC0_ONEUS_TIC_CNT	EMAC0 LPI One Microsecond Tic Counter Register	0x00000063
0x310400F8	EMAC0_PHYIF_CTLSTAT	EMAC0 PHY Interface Control and Status Register	0x00000000
0x31040114	EMAC0_MAC_DBG	EMAC0 Debug Register	0x00000000
0x3104011C	EMAC0_HW_FTR_PSNT0	EMAC0 Hardware Feature Presence Register 0	0x0A7D73F7
0x31040120	EMAC0_HW_FTR_PSNT1	EMAC0 Hardware Feature Presence Register 1	0x439739E7
0x31040124	EMAC0_HW_FTR_PSNT2	EMAC0 Hardware Feature Presence Register 2	0x441C71C7
0x31040128	EMAC0_HW_FTR_PSNT3	EMAC0 Hardware Feature Presence Register 3	0x0C370231
0x3104012C	EMAC0_MAC_HW_FEATURE4	EMAC0 MAC Hardware Feature Presence Register 4	0x00000000
0x31040200	EMAC0_MDIO_ADDR	EMAC0 MDIO Address Register	0x00000000
0x31040204	EMAC0_MDIO_DATA	EMAC0 MDIO Data Register	0x00000000
0x31040210	EMAC0_ARP_ADDR	EMAC0 Address Resolution Protocol (ARP) Register	0x00000000
0x31040230	EMAC0_CSR_SW_CTL	EMAC0 CSR Software Control Register	0x00000000
0x31040234	EMAC0_FPE_CTLSTAT	EMAC0 Frame Preemption Control and Status Register	0x00000000
0x31040238	EMAC0_SPLM_OFST_CFG	EMAC0 MAC Split Mode and Offset Configuration Register	0x00020002
0x31040240	EMAC0_PRSNTM_NSEC	EMAC0 Presentation Time Nanosecond Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040244	EMAC0_PRSNTM_UPDT	EMAC0 Presentation Time Update Register	0x00000000
0x31040300	EMAC0_ADDR0_HI	EMAC0 MAC Address 0 High Register	0x8000FFFF
0x31040304	EMAC0_ADDR0_LO	EMAC0 MAC Address 0 Low Register	0xFFFFFFFF
0x31040308	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104030C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040310	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040314	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040318	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104031C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040320	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040324	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040328	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104032C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040330	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040334	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040338	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104033C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040340	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040344	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040348	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104034C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040350	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040354	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040358	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104035C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040360	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040364	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040368	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104036C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040370	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040374	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040378	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104037C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040380	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040384	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040388	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104038C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040390	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040394	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040398	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104039C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403A0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403A4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403A8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403AC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403B0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403B4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403B8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403BC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403C0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403C4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403C8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403CC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403D0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403D4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403D8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403DC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403E0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403E4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403E8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403EC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403F0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310403F4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403F8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403FC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040700	EMAC0_MMC_CTRL	EMAC0 MMC Control Register	0x00000000
0x31040704	EMAC0_MMC_RXINT	EMAC0 MMC Receive Interrupt Status Register	0x00000000
0x31040708	EMAC0_MMC_TXINT	EMAC0 MMC Transmit Interrupt Status Register	0x00000000
0x3104070C	EMAC0_MMC_RXIMSK	EMAC0 MMC Receive Interrupt Mask Register	0x00000000
0x31040710	EMAC0_MMC_TXIMSK	EMAC0 MMC Transmit Interrupt Mask Register	0x00000000
0x31040714	EMAC0_TX_OCNT_GB	EMAC0 Transmit Octet Good Bad Counter Register	0x00000000
0x31040718	EMAC0_TX_PCNT_GB	EMAC0 Transmit Packet Good Bad Counter Register	0x00000000
0x3104071C	EMAC0_TX_BCST_PCNT_G	EMAC0 Transmit Broadcast Packets Good Counter Register	0x00000000
0x31040720	EMAC0_TX_MCST_PCNT_G	EMAC0 Transmit Multicast Packets Good Counter Register	0x00000000
0x31040724	EMAC0_TX64_GB	EMAC0 Transmit 64 Octets Packets Good Bad Counter Register	0x00000000
0x31040728	EMAC0_TX65TO127_GB	EMAC0 Transmit 65-127 Octets Packets Good Bad Counter Register	0x00000000
0x3104072C	EMAC0_TX128TO255_GB	EMAC0 Transmit 128-255 Octets Packets Good Bad Counter Register	0x00000000
0x31040730	EMAC0_TX256TO511_GB	EMAC0 Transmit 256-511 Octets Packets Good Bad Counter Register	0x00000000
0x31040734	EMAC0_TX512TO1023_GB	EMAC0 Transmit 512-1023 Octets Packets Good Bad Counter Register	0x00000000
0x31040738	EMAC0_TX1024TOMAX_GB	EMAC0 Transmit 1024-Max Octets Packets Good Bad Counter Register	0x00000000
0x3104073C	EMAC0_TX_UCST_PCNT_GB	EMAC0 Transmit Unicast Packets Good Bad Counter Register	0x00000000
0x31040740	EMAC0_TX_MCST_PCNT_GB	EMAC0 Transmit Multicast Packets Good Bad Counter Register	0x00000000
0x31040744	EMAC0_TX_BCST_PCNT_GB	EMAC0 Transmit Broadcast Packets Good Bad Counter Register	0x00000000
0x31040748	EMAC0_TX_URFL_PCNT	EMAC0 Transmit Underflow Error Packets Counter Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104074C	EMAC0_TX_SNGLCOL_PCNT_G	EMAC0 Transmit Single Collision Packets Good Counter Register	0x00000000
0x31040750	EMAC0_TX_MCOL_PCNT_G	EMAC0 Transmit Multiple Collision Packets Good Counter Register	0x00000000
0x31040754	EMAC0_TX_DFRD_PCNT	EMAC0 Transmit Deferred Packets Counter Register	0x00000000
0x31040758	EMAC0_TX_LTCOL_PCNT	EMAC0 Transmit Late Collision Packets Counter Register	0x00000000
0x3104075C	EMAC0_TX_EXCOL_PCNT	EMAC0 Transmit Excessive Collision Packets Counter Register	0x00000000
0x31040760	EMAC0_TX_CRERR_PCNT	EMAC0 Transmit Carrier Error Packets Counter Register	0x00000000
0x31040764	EMAC0_TX_OCNT_G	EMAC0 Transmit Octet Good Counter Register	0x00000000
0x31040768	EMAC0_TX_PCNT_G	EMAC0 Transmit Packet Good Counter Register	0x00000000
0x3104076C	EMAC0_TX_EXDFRL_PCNT	EMAC0 Transmit Excessive Deferral Error Counter Register	0x00000000
0x31040770	EMAC0_TX_PAUSE_PCNT	EMAC0 Transmit Pause Packets Counter Register	0x00000000
0x31040774	EMAC0_TX_VLAN_PCNT_G	EMAC0 Transmit VLAN Packets Good Counter Register	0x00000000
0x31040778	EMAC0_TX_OSIZE_PCNT_G	EMAC0 Transmit Osize Packets Good Counter Register	0x00000000
0x31040780	EMAC0_RX_PCNT_GB	EMAC0 Receive Packets Good Bad Counter Register	0x00000000
0x31040784	EMAC0_RX_OCNT_GB	EMAC0 Receive Octet Good Bad Counter Register	0x00000000
0x31040788	EMAC0_RX_OCNT_G	EMAC0 Receive Octet Good Counter Register	0x00000000
0x3104078C	EMAC0_RX_BCERR_PCNT_G	EMAC0 Receive Broadcast Packets Good Counter Register	0x00000000
0x31040790	EMAC0_RX_MCST_PCNT_G	EMAC0 Receive Multicast Packets Good Counter Register	0x00000000
0x31040794	EMAC0_RX_CRCERR_PCNT	EMAC0 Receive CRC Error Packets Counter Register	0x00000000
0x31040798	EMAC0_RX_ALGNERR_PCNT	EMAC0 Receive Alignment Error Packets Counter Register	0x00000000
0x3104079C	EMAC0_RX_RNTERR_PCNT	EMAC0 Receive Runt Error Packets Counter Register	0x00000000
0x310407A0	EMAC0_RX_JBERR_PCNT	EMAC0 Receive Jabber Error Packets Counter Register	0x00000000
0x310407A4	EMAC0_RX_UNDRSZ_PCNT_G	EMAC0 Receive Undersize Packets Good Counter Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310407A8	EMAC0_RX_OVRSZ_PCNT	EMAC0 Receive Oversize Packets Good Counter Register	0x00000000
0x310407AC	EMAC0_RX64_GB	EMAC0 Receive 64 Octets Packets Good Bad Counter Register	0x00000000
0x310407B0	EMAC0_RX65TO127_GB	EMAC0 Receive 65 - 127 Octets Packets Good Bad Counter Register	0x00000000
0x310407B4	EMAC0_RX128TO255_GB	EMAC0 Receive 128 - 255 Octets Packets Good Bad Counter Register	0x00000000
0x310407B8	EMAC0_RX256TO511_GB	EMAC0 Receive 256 - 511 Octets Packets Good Bad Counter Register	0x00000000
0x310407BC	EMAC0_RX512TO1023_GB	EMAC0 Receive 512 - 1023 Octets Packets Good Bad Counter Register	0x00000000
0x310407C0	EMAC0_RX1024TOMAX_GB	EMAC0 Receive 1024 - Maximum Octets Packets Good Bad Counter Register	0x00000000
0x310407C4	EMAC0_RX_UCST_PCNT_G	EMAC0 Receive Unicast Packets Good Counter Register	0x00000000
0x310407C8	EMAC0_RX_LENERR_PCNT	EMAC0 Receive Length Error Packets Counter Register	0x00000000
0x310407CC	EMAC0_RX_ORTYPE_PCNT	EMAC0 Receive Out of Range Type Packets Counter Register	0x00000000
0x310407D0	EMAC0_RX_PAUSE_PCNT	EMAC0 Receive Pause Packets Counter Register	0x00000000
0x310407D4	EMAC0_RX_FIFOOVFL_PCNT	EMAC0 Receive FIFO Overflow Packets Counter Register	0x00000000
0x310407D8	EMAC0_RX_VLAN_PCNT_GB	EMAC0 Receive VLAN Packets Good Bad Counter Register	0x00000000
0x310407DC	EMAC0_RX_WDTERR_PCNT	EMAC0 Receive Watchdog Error Packets Counter Register	0x00000000
0x310407E0	EMAC0_RX_ERR_PCNT	EMAC0 Receive Error Packets Counter Register	0x00000000
0x310407E4	EMAC0_RX_CTL_PCNT_G	EMAC0 Receive Control Packets Good Counter Register	0x00000000
0x310407EC	EMAC0_TX_LPIUSEC_CNT	EMAC0 Transmit LPI Microseconds Counter Register	0x00000000
0x310407F0	EMAC0_TX_LPITRAN_CNT	EMAC0 Transmit LPI Transition Counter Register	0x00000000
0x310407F4	EMAC0_RX_LPIUSEC_CNT	EMAC0 Receive LPI Microseconds Counter Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310407F8	EMAC0_RX_LPI-TRAN_CNT	EMAC0 Receive LPI Transition Counter Register	0x00000000
0x31040800	EMAC0_MMC_IPC_RXIMSK	EMAC0 MMC Receive IPC Interrupt Mask Register	0x00000000
0x31040808	EMAC0_MMC_IPC_RXINT	EMAC0 MMC Receive IPC Interrupt Status Register	0x00000000
0x31040810	EMAC0_RXIPV4_PCNT_G	EMAC0 Receive IPv4 Good Packets Counter Register	0x00000000
0x31040814	EMAC0_RXIPV4_HDRERR_PCNT	EMAC0 Receive IPv4 Header Error Packets Counter Register	0x00000000
0x31040818	EMAC0_RXIPV4_PYLD_PCNT	EMAC0 Receive IPv4 Payload Packets Counter Register	0x00000000
0x3104081C	EMAC0_RXIPV4_FRGMT_PCNT	EMAC0 Receive IPv4 Fragmented Packets Counter Register	0x00000000
0x31040820	EMAC0_RXIPV4_UDPCSD_PCNT	EMAC0 Receive IPv4 UDP Checksum Disable Packets Counter Register	0x00000000
0x31040824	EMAC0_RXIPV6_PCNT_G	EMAC0 Receive IPv6 Good Packets Counter Register	0x00000000
0x31040828	EMAC0_RXIPV6_HDRERR_PCNT	EMAC0 Receive IPv6 Header Error Packets Counter Register	0x00000000
0x3104082C	EMAC0_RXIPV6_PYLD_PCNT	EMAC0 Receive IPv6 Payload Packets Counter Register	0x00000000
0x31040830	EMAC0_RXUDP_PCNT_G	EMAC0 Receive UDP Good Packets Counter Register	0x00000000
0x31040834	EMAC0_RXUDP_ERR_PCNT	EMAC0 Receive UDP Error Packets Counter Register	0x00000000
0x31040838	EMAC0_RXTCP_PCNT_G	EMAC0 Receive TCP Good Packets Counter Register	0x00000000
0x3104083C	EMAC0_RXTCP_ERR_PCNT	EMAC0 Receive TCP Error Packets Counter Register	0x00000000
0x31040840	EMAC0_RXICMP_PCNT_G	EMAC0 Receive ICMP Good Packets Counter Register	0x00000000
0x31040844	EMAC0_RXICMP_ERR_PCNT	EMAC0 Receive ICMP Error Packets Counter Register	0x00000000
0x31040850	EMAC0_RXIPV4_OCNT_G	EMAC0 Receive IPv4 Good Octets Counter Register	0x00000000
0x31040854	EMAC0_RXIPV4_HDRERR_OCNT	EMAC0 Receive IPv4 Header Error Octets Counter Register	0x00000000
0x31040858	EMAC0_RXIPV4_PYLD_OCNT	EMAC0 Receive IPv4 Payload Octets Counter Register	0x00000000
0x3104085C	EMAC0_RXIPV4_FRGMT_OCNT	EMAC0 Receive IPv4 Fragmented Octets Counter Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040860	EMAC0_RXIPV4_UDPCSD_OCNT	EMAC0 Receive IPv4 UDP Checksum Disable Octets Counter Register	0x00000000
0x31040864	EMAC0_RXIPV6_OCNT_G	EMAC0 Receive IPv6 Good Octets Counter Register	0x00000000
0x31040868	EMAC0_RXIPV6_HDRERR_OCNT	EMAC0 Receive IPv6 Header Error Octets Counter Register	0x00000000
0x3104086C	EMAC0_RXIPV6_PYLD_OCNT	EMAC0 Receive IPv6 Payload Octets Counter Register	0x00000000
0x31040870	EMAC0_RXUDP_OCNT_G	EMAC0 Receive UDP Good Octets Counter Register	0x00000000
0x31040874	EMAC0_RXUDP_ERR_OCNT	EMAC0 Receive UDP Error Octets Counter Register	0x00000000
0x31040878	EMAC0_RXTCP_OCNT_G	EMAC0 Receive TCP Good Octets Counter Register	0x00000000
0x3104087C	EMAC0_RXTCP_ERR_OCNT	EMAC0 Receive TCP Error Octets Counter Register	0x00000000
0x31040880	EMAC0_RXICMP_OCNT_G	EMAC0 Receive ICMP Good Octets Counter Register	0x00000000
0x31040884	EMAC0_RXICMP_ERR_OCNT	EMAC0 Receive ICMP Error Octets Counter Register	0x00000000
0x310408A0	EMAC0_MMC_FPE_TXINT	EMAC0 MMC Transmit FPE Interrupt Status Register	0x00000000
0x310408A4	EMAC0_MMC_FPE_TXIMSK	EMAC0 MMC Transmit FPE Interrupt Mask Register	0x00000000
0x310408A8	EMAC0_TXFPE_FRGMNT_CNT	EMAC0 MMC Transmit FPE Packet Fragment Counter Register	0x00000000
0x310408AC	EMAC0_TXHLDRQ_CNT	EMAC0 MMC Transmit FPE Hold Request Counter Register	0x00000000
0x310408C0	EMAC0_MMC_FPE_RXINT	EMAC0 MMC Receive FPE Interrupt Status Register	0x00000000
0x310408C4	EMAC0_MMC_FPE_RXIMSK	EMAC0 MMC Receive FPE Interrupt Mask Register	0x00000000
0x310408C8	EMAC0_RXPKT_ASERR_CNT	EMAC0 MMC Receive FPE Packet Assembly Error Counter Register	0x00000000
0x310408CC	EMAC0_RXPKT_SMDERR_CNT	EMAC0 MMC Receive FPE Packet SMD Error Counter Register	0x00000000
0x310408D0	EMAC0_RXPKT_ASOK_CNT	EMAC0 MMC Receive FPE Packet Assembly Okay Counter Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310408D4	EMAC0_RXFPE_FRGMNT_CNT	EMAC0 MMC Receive FPE Fragment Counter Register	0x00000000
0x31040900	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040904	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040910	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040914	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040918	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x3104091C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040930	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040934	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040940	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040944	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040948	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x3104094C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040960	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040964	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040970	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040974	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040978	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x3104097C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040990	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040994	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310409A0	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x310409A4	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x310409A8	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x310409AC	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x310409C0	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x310409C4	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x310409D0	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x310409D4	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x310409D8	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x310409DC	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x310409F0	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x310409F4	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040A00	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040A04	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040A08	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x31040A0C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040A20	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040A24	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040A30	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040A34	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040A38	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040A3C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040A50	EMAC0_L3L4_CTL7	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040A54	EMAC0_L4_ADDR7	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040A60	EMAC0_L3_ADDR0_REG7	EMAC0 Layer 3 Address 0 Register 7	0x00000000
0x31040A64	EMAC0_L3_ADDR1_REG7	EMAC0 Layer 3 Address 1 Register 7	0x00000000
0x31040A68	EMAC0_L3_ADDR2_REG7	EMAC0 Layer 3 Address 2 Register 7	0x00000000
0x31040A6C	EMAC0_L3_ADDR3_REG7	EMAC0 Layer 3 Address 3 Register 7	0x00000000
0x31040A70	EMAC0_INDR_ACC_CTL	EMAC0 Indirect Access Control Register	0x00000000
0x31040A74	EMAC0_INDR_ACC_DAT	EMAC0 Indirect Access Data Register	0x00000000
0x31040B00	EMAC0_TM_CTL	EMAC0 Time Stamp Control Register	0x00002000
0x31040B04	EMAC0_TM_SUBSEC	EMAC0 Subsecond Increment Register	0x00000000
0x31040B08	EMAC0_TM_SEC	EMAC0 System Time Seconds Register	0x00000000
0x31040B0C	EMAC0_TM_NSEC	EMAC0 System Time Nanoseconds Register	0x00000000
0x31040B10	EMAC0_TM_SECUPDT	EMAC0 System Time Seconds Update Register	0x00000000
0x31040B14	EMAC0_TM_NSECUPDT	EMAC0 System Time Nanoseconds Update Register	0x00000000
0x31040B18	EMAC0_TM_ADDEND	EMAC0 Time Stamp Addend Register	0x00000000
0x31040B1C	EMAC0_TM_HISEC	EMAC0 System Time Higher Word Seconds Register	0x00000000
0x31040B20	EMAC0_TM_STAT	EMAC0 Time Stamp Status Register	0x00000000
0x31040B24	EMAC0_MAC_RX_DOMAIN_TIME_INCR	EMAC0 MAC Receive Time Domain Increment Register	0x00000000
0x31040B28	EMAC0_MAC_TX_DOMAIN_TIME_INCR	EMAC0 MAC Transmit Time Domain Increment Register	0x00000000
0x31040B30	EMAC0_TXTMSTAT_NSEC	EMAC0 Transmit Time Stamp Status Nanoseconds Register	0x00000000
0x31040B34	EMAC0_TXTMSTAT_SEC	EMAC0 Transmit Time Stamp Status Seconds Register	0x00000000
0x31040B40	EMAC0_TM_AUX_CTL	EMAC0 Auxiliary Time Stamp Control Register	0x00000000
0x31040B48	EMAC0_TM_AUXTS_NSEC	EMAC0 Auxiliary Time Stamp Nanoseconds Register	0x00000000
0x31040B4C	EMAC0_TM_AUXTS_SEC	EMAC0 Auxiliary Time Stamp Seconds Register	0x00000000
0x31040B50	EMAC0_TM_INGASYM_CORR	EMAC0 Time Stamp Ingress Asymmetry Correction Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040B54	EMAC0_TM_EGASYM_CORR	EMAC0 Time Stamp Egress Asymmetry Correction Register	0x00000000
0x31040B58	EMAC0_TM_INGCORR_NSEC	EMAC0 Time Stamp Ingress Correction Nanosecond Register	0x00000000
0x31040B5C	EMAC0_TM_EGCORR_NSEC	EMAC0 Time Stamp Egress Correction Nanosecond Register	0x00000000
0x31040B60	EMAC0_TM_INGCORR_SNSEC	EMAC0 Time Stamp Ingress Correction Sub Nanosecond Register	0x00000000
0x31040B64	EMAC0_TM_EGCORR_SNSEC	EMAC0 Time Stamp Egress Correction Sub Nanosecond Register	0x00000000
0x31040B68	EMAC0_TM_ING_LTNCY	EMAC0 Time Stamp Ingress Latency Register	0x00000000
0x31040B6C	EMAC0_TM_EG_LTNCY	EMAC0 Time Stamp Egress Latency Register	0x00000000
0x31040B70	EMAC0_PPS_CTL	EMAC0 PPS Control Register	0x00000000
0x31040B80	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000
0x31040B84	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040B88	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040B8C	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040B90	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000
0x31040B94	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040B98	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040B9C	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040BA0	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000
0x31040BA4	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040BA8	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040BAC	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040BB0	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000
0x31040BB4	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040BB8	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040BBC	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040BC0	EMAC0_PTO_CTL	EMAC0 PTP Time Stamp Offload Control Register	0x00000000
0x31040BC4	EMAC0_SRCPRT_IDNTY0	EMAC0 Source Port Identity 0 Register	0x00000000
0x31040BC8	EMAC0_SRCPRT_IDNTY1	EMAC0 Source Port Identity 1 Register	0x00000000
0x31040BCC	EMAC0_SRCPRT_IDNTY2	EMAC0 Source Port Identity 2 Register	0x00000000
0x31040BD0	EMAC0_LOG_MSG_INTVL	EMAC0 PTP Log Message Interval Register	0x00000000
0x31040C00	EMAC0_MTL_OPMODE	EMAC0 MTL Operation Mode Register	0x00000000
0x31040C20	EMAC0_MTL_ISTAT	EMAC0 MTL Interrupt Status Register	0x00000000
0x31040C30	EMAC0_RXQ_DMA_MAP0	EMAC0 Receive Queue to DMA Channel Mapping Register 0	0x00000000
0x31040C34	EMAC0_RXQ_DMA_MAP1	EMAC0 Receive Queue to DMA Channel Mapping Register 1	0x00000000
0x31040C40	EMAC0_TBS_CTRL	EMAC0 MTL Time Based Scheduling Control Register	0x00000000
0x31040C50	EMAC0_EST_CTL	EMAC0 EST Control Register	0x00000000
0x31040C54	EMAC0_EST_EXT_CTL	EMAC0 EST Extended Control Register	0x00000000
0x31040C58	EMAC0_EST_STAT	EMAC0 EST Status Register	0x00000000
0x31040C60	EMAC0_EST_SCH_ERR	EMAC0 EST Scheduling Error Register	0x00000000
0x31040C64	EMAC0_EST_FRMSZ_ERR	EMAC0 EST Frame Size Error Register	0x00000000
0x31040C68	EMAC0_EST_FRMSZ_CAP	EMAC0 EST Frame Size Capture Register	0x00000000
0x31040C70	EMAC0_EST_IEN	EMAC0 EST Interrupt Enable Register	0x00000000
0x31040C80	EMAC0_EST_GCL_CTL	EMAC0 EST Gate Control List Control Register	0x00000000
0x31040C84	EMAC0_EST_GCL_DAT	EMAC0 EST Gate Control List Data Register	0x00000000
0x31040C90	EMAC0_FPE_CTRLSTS	EMAC0 MTL Frame Preemption Control and Status Register	0x00000000
0x31040C94	EMAC0_FPE_ADV	EMAC0 Frame Preemption Advance Register	0x00000000
0x31040D00	EMAC0_TQ0_OPMODE	EMAC0 Transmit Queue 0 Operation Mode Register	0x00000000
0x31040D04	EMAC0_TQ0_URFL	EMAC0 Transmit Queue 0 Underflow Packet Counter Register	0x00000000
0x31040D08	EMAC0_TQ0_DBG	EMAC0 Transmit Queue 0 Debug Register	0x00000000
0x31040D14	EMAC0_TQ0_ETS_STAT	EMAC0 Transmit Queue 0 ETS Status Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040D18	EMAC0_TQ0_QTMWGT	EMAC0 Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D2C	EMAC0_Q0_INT_CTLSTAT	EMAC0 Queue 0 Interrupt Control and Status Register	0x00000000
0x31040D30	EMAC0_RQ0_OPMODE	EMAC0 Receive Queue 0 Operation Mode Register	0x00000000
0x31040D34	EMAC0_RQ0_MSPKTOF_CNT	EMAC0 Receive Queue 0 Missed Packet and Overflow Counter Register	0x00000000
0x31040D38	EMAC0_RQ0_DBG	EMAC0 Receive Queue 0 Debug Register	0x00000000
0x31040D3C	EMAC0_RQ0_CTL	EMAC0 Receive Queue Control Register	0x00000000
0x31040D40	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040D44	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040D48	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040D50	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040D54	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040D58	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D5C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040D60	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040D64	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040D6C	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040D70	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040D74	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040D78	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040D7C	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040D80	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040D84	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040D88	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040D90	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040D94	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040D98	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D9C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040DA0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040DA4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040DAC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040DB0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040DB4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040DB8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040DBC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040DC0	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040DC4	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040DC8	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040DD0	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040DD4	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040DD8	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040DDC	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040DE0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040DE4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040DEC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040DF0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040DF4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040DF8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040DFC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040E00	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040E04	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040E08	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040E10	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040E14	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040E18	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E1C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E20	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040E24	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040E2C	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040E30	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040E34	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040E38	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040E3C	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040E40	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040E44	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E48	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040E50	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040E54	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040E58	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E5C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E60	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040E64	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040E6C	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040E70	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040E74	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040E78	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040E7C	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040E80	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040E84	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E88	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040E90	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040E94	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040E98	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E9C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040EA0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040EA4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040EAC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040EB0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040EB4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040EB8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040EBC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040EC0	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040EC4	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040EC8	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040ED0	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040ED4	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040ED8	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040EDC	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040EE0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040EE4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040EEC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040EF0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040EF4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040EF8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040EFC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31041000	EMAC0_DMA_MODE	EMAC0 DMA Mode Register	0x00000000
0x31041004	EMAC0_DMA_SYSBMODE	EMAC0 DMA System Bus Mode Register	0x01010000
0x31041008	EMAC0_DMA_ISTAT	EMAC0 DMA Interrupt Status Register	0x00000000
0x3104100C	EMAC0_DMA_DBG_STAT 0	EMAC0 DMA Debug Status 0 Register	0x00000000
0x31041010	EMAC0_DMA_DBG_STAT 1	EMAC0 DMA Debug Status 1 Register	0x00000000
0x31041014	EMAC0_DMA_DBG_STAT 2	EMAC0 DMA Debug Status 2 Register	0x00000000
0x31041050	EMAC0_DMA_TBS_CTL0	EMAC0 DMA TBS Control 0 Register	0x00000000
0x31041054	EMAC0_DMA_TBS_CTL1	EMAC0 DMA TBS Control 1 Register	0x00000000
0x31041058	EMAC0_DMA_TBS_CTL2	EMAC0 DMA TBS Control 2 Register	0x00000000
0x3104105C	EMAC0_DMA_TBS_CTL3	EMAC0 DMA TBS Control 3 Register	0x00000000
0x31041100	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041104	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041108	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041114	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104111C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31041120	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041128	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104112C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041130	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041134	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041138	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104113C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041144	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104114C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31041154	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104115C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041160	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041164	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041180	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041184	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041188	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041194	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104119C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310411A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310411A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310411AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310411B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310411B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310411B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310411BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310411C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310411CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310411D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310411DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310411E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x310411E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041200	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041204	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041208	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041214	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104121C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31041220	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041228	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104122C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041230	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041234	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041238	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104123C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041244	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104124C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041254	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104125C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041260	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041264	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041280	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041284	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041288	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041294	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104129C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310412A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310412A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310412AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310412B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310412B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310412B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310412BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x310412C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310412CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310412D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310412DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310412E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310412E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041300	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041304	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041308	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041314	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104131C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31041320	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041328	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104132C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041330	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041334	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041338	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104133C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041344	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104134C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31041354	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104135C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041360	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041364	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041380	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041384	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041388	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041394	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104139C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310413A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310413A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310413AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310413B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310413B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310413B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310413BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x310413C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310413CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310413D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310413DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310413E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x310413E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041400	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041404	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041408	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041414	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104141C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041420	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041428	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104142C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041430	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041434	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041438	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104143C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041444	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104144C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31041454	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104145C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041460	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041464	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041480	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041484	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041488	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041494	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104149C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310414A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310414A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000

Table A-80: ADSP-2184x EMAC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310414AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310414B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310414B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310414B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310414BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x310414C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310414CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310414D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310414DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310414E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x310414E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000

Table A-81: ADSP-2184x FIR0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C3000	FIR0_CTL1	FIR0 FIR Global Control Register	0x00000000
0x310C3004	FIR0_DMASTAT	FIR0 FIR DMA Status Register	0x00000000
0x310C3008	FIR0_MACSTAT	FIR0 FIR MAC Status Register	0x00000000
0x310C3010	FIR0_DBG_CTL	FIR0 FIR Debug Control Register	0x00000000
0x310C3014	FIR0_DBG_ADDR	FIR0 Debug Address Register	0x00000000
0x310C3018	FIR0_DBG_WRDAT	FIR0 FIR Debug Data Write Register	0x00000000
0x310C301C	FIR0_DBG_RDDAT	FIR0 FIR Debug Data Read Register	0x00000000
0x310C3040	FIR0_CTL2	FIR0 FIR Channel Control Register	0x00000000
0x310C3044	FIR0_INIDX	FIR0 FIR Input Data Index Register	0x00000000
0x310C3048	FIR0_INMOD	FIR0 FIR Input Data Modifier Register	0x00000000

Table A-81: ADSP-2184x FIR0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C304C	FIR0_INCNT	FIR0 FIR Input Data Count Register	0x00000000
0x310C3050	FIR0_INBASE	FIR0 FIR Input Data Base Register	0x00000000
0x310C3054	FIR0_OUTIDX	FIR0 FIR Output Data Index Register	0x00000000
0x310C3058	FIR0_OUTMOD	FIR0 FIR Output Data Modifier Register	0x00000000
0x310C305C	FIR0_OUTCNT	FIR0 FIR Output Data Count Register	0x00000000
0x310C3060	FIR0_OUTBASE	FIR0 FIR Output Data Base Register	0x00000000
0x310C3064	FIR0_COEFIDX	FIR0 FIR Coefficient Index Register	0x00000000
0x310C3068	FIR0_COEFMOD	FIR0 FIR Coefficient Modifier Register	0x00000000
0x310C306C	FIR0_COEFCNT	FIR0 FIR Coefficient Count Register	0x00000000
0x310C3070	FIR0_CHNPTR	FIR0 FIR Chain Pointer Register	0x00000000
0x310C3074	FIR0_SCTL1	FIR0 Software Control Register 1	0x00000000
0x310C3078	FIR0_SCTL2	FIR0 Software Control Register 2	0x00000000
0x310C307C	FIR0_SGCTL	FIR0 Secondary Global Control Register	0x00000000

Table A-82: ADSP-2184x FIR1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BF000	FIR1_CTL1	FIR1 FIR Global Control Register	0x00000000
0x310BF004	FIR1_DMASTAT	FIR1 FIR DMA Status Register	0x00000000
0x310BF008	FIR1_MACSTAT	FIR1 FIR MAC Status Register	0x00000000
0x310BF010	FIR1_DBG_CTL	FIR1 FIR Debug Control Register	0x00000000
0x310BF014	FIR1_DBG_ADDR	FIR1 Debug Address Register	0x00000000
0x310BF018	FIR1_DBG_WRDAT	FIR1 FIR Debug Data Write Register	0x00000000
0x310BF01C	FIR1_DBG_RDDAT	FIR1 FIR Debug Data Read Register	0x00000000
0x310BF040	FIR1_CTL2	FIR1 FIR Channel Control Register	0x00000000
0x310BF044	FIR1_INIDX	FIR1 FIR Input Data Index Register	0x00000000
0x310BF048	FIR1_INMOD	FIR1 FIR Input Data Modifier Register	0x00000000
0x310BF04C	FIR1_INCNT	FIR1 FIR Input Data Count Register	0x00000000
0x310BF050	FIR1_INBASE	FIR1 FIR Input Data Base Register	0x00000000
0x310BF054	FIR1_OUTIDX	FIR1 FIR Output Data Index Register	0x00000000
0x310BF058	FIR1_OUTMOD	FIR1 FIR Output Data Modifier Register	0x00000000

Table A-82: ADSP-2184x FIR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310BF05C	FIR1_OUTCNT	FIR1 FIR Output Data Count Register	0x00000000
0x310BF060	FIR1_OUTBASE	FIR1 FIR Output Data Base Register	0x00000000
0x310BF064	FIR1_COEFIDX	FIR1 FIR Coefficient Index Register	0x00000000
0x310BF068	FIR1_COEFMOD	FIR1 FIR Coefficient Modifier Register	0x00000000
0x310BF06C	FIR1_COEFCNT	FIR1 FIR Coefficient Count Register	0x00000000
0x310BF070	FIR1_CHNPTR	FIR1 FIR Chain Pointer Register	0x00000000
0x310BF074	FIR1_SCTL1	FIR1 Software Control Register 1	0x00000000
0x310BF078	FIR1_SCTL2	FIR1 Software Control Register 2	0x00000000
0x310BF07C	FIR1_SGCTL	FIR1 Secondary Global Control Register	0x00000000

Table A-83: ADSP-2184x FRACNPLL0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31144000	FRACNPLL0_FPLL_CTL0	FRACNPLL0 Fractional PLL Control Register 0	0x00000000
0x31144004	FRACNPLL0_FPLL_CTL1	FRACNPLL0 Fractional PLL Control Register 1	0x00001000
0x31144008	FRACNPLL0_DPLL_CTL0	FRACNPLL0 DPLL Control Register 0	0x00000000
0x3114400C	FRACNPLL0_DPLL_CTL1	FRACNPLL0 DPLL Control Register 1	0x00000000
0x31144010	FRACNPLL0_DPLL_CTL2	FRACNPLL0 DPLL Control Register 2	0x00000000
0x31144014	FRACNPLL0_DPLL_CTL3	FRACNPLL0 DPLL Control Register 3	0x00000000
0x31144018	FRACNPLL0_DPLL_CTL4	FRACNPLL0 DPLL Control Register 4	0x00000000
0x3114401C	FRACNPLL0_DPLL_CTL5	FRACNPLL0 DPLL Control Register 5	0x00000000
0x31144020	FRACNPLL0_DPLL_CTL6	FRACNPLL0 DPLL Control Register 6	0x00000000
0x31144024	FRACNPLL0_DPLL_CTL7	FRACNPLL0 DPLL Control Register 7	0x00000000
0x31144028	FRACNPLL0_DPLL_CTL8	FRACNPLL0 DPLL Control Register 8	0x00000000
0x3114402C	FRACNPLL0_DPLL_CTL9	FRACNPLL0 DPLL Control Register 9	0x00000000
0x31144030	FRACNPLL0_CTL	FRACNPLL0 Integration Control Register	0xC0355424
0x31144034	FRACNPLL0_STAT	FRACNPLL0 Integration Status Register	0x00000000
0x31144038	FRACNPLL0_DPLL_STAT0	FRACNPLL0 DPLL Status Register 0	0x00000000
0x3114403C	FRACNPLL0_DPLL_STAT1	FRACNPLL0 DPLL Status Register 1	0x00000000
0x31144040	FRACNPLL0_DPLL_STAT2	FRACNPLL0 DPLL Status Register 2	0x00000000

Table A-84: ADSP-2184x FRACNPLL1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31157000	FRACNPLL1_FPLL_CTL0	FRACNPLL1 Fractional PLL Control Register 0	0x00000000
0x31157004	FRACNPLL1_FPLL_CTL1	FRACNPLL1 Fractional PLL Control Register 1	0x00001000
0x31157008	FRACNPLL1_DPLL_CTL0	FRACNPLL1 DPLL Control Register 0	0x00000000
0x3115700C	FRACNPLL1_DPLL_CTL1	FRACNPLL1 DPLL Control Register 1	0x00000000
0x31157010	FRACNPLL1_DPLL_CTL2	FRACNPLL1 DPLL Control Register 2	0x00000000
0x31157014	FRACNPLL1_DPLL_CTL3	FRACNPLL1 DPLL Control Register 3	0x00000000
0x31157018	FRACNPLL1_DPLL_CTL4	FRACNPLL1 DPLL Control Register 4	0x00000000
0x3115701C	FRACNPLL1_DPLL_CTL5	FRACNPLL1 DPLL Control Register 5	0x00000000
0x31157020	FRACNPLL1_DPLL_CTL6	FRACNPLL1 DPLL Control Register 6	0x00000000
0x31157024	FRACNPLL1_DPLL_CTL7	FRACNPLL1 DPLL Control Register 7	0x00000000
0x31157028	FRACNPLL1_DPLL_CTL8	FRACNPLL1 DPLL Control Register 8	0x00000000
0x3115702C	FRACNPLL1_DPLL_CTL9	FRACNPLL1 DPLL Control Register 9	0x00000000
0x31157030	FRACNPLL1_CTL	FRACNPLL1 Integration Control Register	0xC0355424
0x31157034	FRACNPLL1_STAT	FRACNPLL1 Integration Status Register	0x00000000
0x31157038	FRACNPLL1_DPLL_STAT0	FRACNPLL1 DPLL Status Register 0	0x00000000
0x3115703C	FRACNPLL1_DPLL_STAT1	FRACNPLL1 DPLL Status Register 1	0x00000000
0x31157040	FRACNPLL1_DPLL_STAT2	FRACNPLL1 DPLL Status Register 2	0x00000000

Table A-85: ADSP-2184x HADC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31016000	HADC0_CTL	HADC0 Control Register	0x000001AA
0x31016004	HADC0_CHAN_MSK	HADC0 Channel Mask Register	0x0000FF00
0x31016008	HADC0_IMSK	HADC0 Interrupt Mask Register	0x00000000
0x3101600C	HADC0_STAT	HADC0 Status Register	0x00000000
0x31016010	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016014	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016018	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101601C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016020	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016024	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000

Table A-85: ADSP-2184x HADC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31016028	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101602C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016030	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016034	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016038	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101603C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016040	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016044	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016048	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101604C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000

Table A-86: ADSP-2184x HSM_MISC MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3115D000	HSM_MISC_CTL	HSM_MISC Control Register	0x00000000
0x3115D004	HSM_MISC_STS	HSM_MISC Status Register	0x00000000
0x3115D008	HSM_MISC_LIFE_CYCLE_STS	HSM_MISC Life Cycle Status Register	0x00000000
0x3115D00C	HSM_MISC_FEATURE_OUT_BITS	HSM_MISC Feature Out Bits, GPIO	0x00000000
0x3115D010	HSM_MISC_TDV_0	HSM_MISC TDV Value 4-63 Bits	0x00000000
0x3115D014	HSM_MISC_TDV_1	HSM_MISC TDV Value 64-127 Bits	0x00000000
0x3115D018	HSM_MISC_TDV_2	HSM_MISC TDV Value 128-191 Bits	0x00000000
0x3115D01C	HSM_MISC_TDV_3	HSM_MISC TDV Value 192-255 Bits	0x00000000
0x3115D020	HSM_MISC_KDAT_STS	HSM_MISC Secure Key Data Status Register	0x00000000
0x3115D024	HSM_MISC_KDAT0_0	HSM_MISC Secure Key Data 0-63 Bits	0x00000000
0x3115D028	HSM_MISC_KDAT0_1	HSM_MISC Secure Key Data 64-127 Bits	0x00000000
0x3115D02C	HSM_MISC_KDAT0_2	HSM_MISC Secure Key Data 128-191 Bits	0x00000000
0x3115D030	HSM_MISC_KDAT0_3	HSM_MISC Secure Key Data 192-255 Bits	0x00000000
0x3115D034	HSM_MISC_KMETDAT0_0	HSM_MISC Secure Key Meta Data 0-63 Bits	0x00000000
0x3115D038	HSM_MISC_KMETDAT0_1	HSM_MISC Secure Key Meta Data 64-127 Bits	0x00000000
0x3115D03C	HSM_MISC_ALM_CTL	HSM_MISC Alarm Control Register	0x00000000

Table A-86: ADSP-2184x HSM_MISC MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3115D040	HSM_MISC_ALM_STS	HSM_MISC Alarm Status Register	0x00000000
0x3115D044	HSM_MISC_PWR_MGMT_STS	HSM_MISC Power Management Status Register	0x00000000
0x3115D048	HSM_MISC_JTG_ID	HSM_MISC JTAG ID Register	0x00000000
0x3115D04C	HSM_MISC_TRNG_CHR_CTL	HSM_MISC TRNG Characterization Control Register	0x00000000
0x3115D050	HSM_MISC_TRNG_VLD_CTL	HSM_MISC TRNG Validation Control Register	0x00000000
0x3115D054	HSM_MISC_TRNG_VLD_DATIN	HSM_MISC TNRG Validation Data Input Register	0x00000000
0x3115D058	HSM_MISC_TRNG_VLD_DATOUT	HSM_MISC TRNG Validation Data Out Register	0x00000000
0x3115D05C	HSM_MISC_INTR_IN	HSM_MISC Interrupt Input Register	0x00000000
0x3115D060	HSM_MISC_ECC_CTL	HSM_MISC ECC Control Register	0x00000000
0x3115D080	HSM_MISC_OTP_PRGM_CTL0	HSM_MISC OTP Program Control0 Register	0x00000001
0x3115D084	HSM_MISC_OTP_PRGM_CTL1	HSM_MISC OTP Program Control1 Register	0x0000000A
0x3115D0A0	HSM_MISC_OTP_CTL	HSM_MISC OTP Control Register	0x00000020
0x3115D0A4	HSM_MISC_OTP_PMC_CTL	HSM_MISC OTP PMC Control Register	0x0000000A
0x3115D0A8	HSM_MISC_OTP_PMC_MODE0	HSM_MISC OTP PMC Mode0 Register	0x00005F2A
0x3115D0AC	HSM_MISC_OTP_PMC_MODE1	HSM_MISC OTP PMC Mode1 Register	0x0000DF16
0x3115D0B0	HSM_MISC_OTP_PMC_MODE2	HSM_MISC OTP PMC Mode2 Register	0x00005F0F
0x3115D0B4	HSM_MISC_OTP_PMC_MODE3	HSM_MISC OTP PMC Mode3 Register	0x00005F0A
0x3115D0B8	HSM_MISC_OTP_PMC_PGM_TMG0	HSM_MISC OTP PMC Program Timer0 Register	0x00000023
0x3115D0BC	HSM_MISC_OTP_PMC_PGM_TMG1	HSM_MISC OTP PMC Program Timer1 Register	0x00000071
0x3115D0C0	HSM_MISC_OTP_PMC_PGM_TMG2	HSM_MISC OTP PMC Program Timer2 Register	0x000000D5

Table A-86: ADSP-2184x HSM_MISC MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3115D0C4	HSM_MISC_OTP_PMC_DAP_ADDR	HSM_MISC OTP DAP Address Register	0x00000002
0x3115D0C8	HSM_MISC_OTP_PMC_PGM_CQ	HSM_MISC OTP PMC Program CQ Register	0x00004004
0x3115D0CC	HSM_MISC_OTP_DAP_RFMR	HSM_MISC OTP DAP RFMR Register	0x0000002A
0x3115D0D0	HSM_MISC_OTP_DAP_VRMR	HSM_MISC OTP DAP VRMR Register	0x0000005D
0x3115D0D4	HSM_MISC_OTP_DAP_OVLR	HSM_MISC OTP DAP OVLR Register	0x00000083
0x3115D0D8	HSM_MISC_OTP_DAP_IPCR	HSM_MISC OTP DAP IPCR Register	0x00000000
0x3115D0DC	HSM_MISC_OTP_DAP_OSCR	HSM_MISC OTP DAP OSCR Register	0x00000000
0x3115D0E0	HSM_MISC_OTP_DAP_ORCR	HSM_MISC OTP DAP ORCR Register	0x000000C0
0x3115D0E4	HSM_MISC_OTP_DAP_ODCR	HSM_MISC OTP DAP ODCR Register	0x00000000
0x3115D0E8	HSM_MISC_OTP_DAP_IPCR2	HSM_MISC OTP DAP IPCR2 Register	0x000000EE
0x3115D0EC	HSM_MISC_OTP_DAP_OCER	HSM_MISC OTP DAP OCER Register	0x00000008
0x3115D0F0	HSM_MISC_OTP_DAP_DPCR	HSM_MISC OTP DAP DPCR Register	0x00000001
0x3115D0F4	HSM_MISC_OTP_DAP_DPCR2	HSM_MISC OTP DAP DPCR2 Register	0x00000000
0x3115D0F8	HSM_MISC_OTP_PMC_INST_START	HSM_MISC OTP PMC Instruction Start Register	0x00000001
0x3115D0FC	HSM_MISC_OTP_PMC_INST_STOP	HSM_MISC OTP PMC Instruction Stop Register	0x00000002
0x3115D100	HSM_MISC_OTP_PMC_CTL_DEFAULT	HSM_MISC OTP PMC Control Default Register	0x00000009
0x3115D104	HSM_MISC_OTP_DAP_DPCR_ECC_DIS	HSM_MISC OTP DAP DPCR ECC Disable Register	0x00000001

Table A-87: ADSP-2184x HSM_OTP MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31160004	HSM_OTP_STAT	HSM_OTP OTP Status Register	0x00000000

Table A-88: ADSP-2184x IIR MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0000	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0000	IIR_CTL1	IIR Global Control Register	0x00000000
0x310C0000	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0004	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0004	IIR_DMASTAT	IIR DMA Status Register	0x00000000
0x310C0004	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0008	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0008	IIR_MACSTAT	IIR MAC Status Register	0x00000000
0x310C0008	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C000C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C000C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C000C	IIR_DBG_CTL	IIR IIR Debug Control Register	0x00000000
0x310C0010	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0010	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0010	IIR_DBG_ADDR	IIR IIR Debug Address Register	0x00000000
0x310C0014	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0014	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0014	IIR_DBG_WRDAT_LO	IIR IIR Debug Write Data Low Register	0x00000000
0x310C0018	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0018	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0018	IIR_DBG_WRDAT_HI	IIR IIR Debug Write Data High Register	0x00000000
0x310C001C	IIR_DBG_RDDAT_LO	IIR IIR Debug Read Data Low Register	0x00000000
0x310C001C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C001C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0020	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0020	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000

Table A-88: ADSP-2184x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0020	IIR_DBG_RDDAT_HI	IIR IIR Debug Read Data High Register	0x00000000
0x310C0024	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0024	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0028	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0028	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C002C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C002C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0030	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0030	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0034	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0034	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0038	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0038	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C003C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C003C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0040	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0040	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0040	IIR_CTL2	IIR Channel Control Register	0x00000000
0x310C0044	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0044	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0044	IIR_INIDX	IIR Input Data Index Register	0x00000000
0x310C0048	IIR_INMOD	IIR Input Data Index Modifier Register	0x00000000
0x310C0048	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0048	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C004C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C004C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C004C	IIR_INLEN	IIR Input Data Buffer Length Register	0x00000000
0x310C0050	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0050	IIR_INBASE	IIR Input Buffer Base Register	0x00000000
0x310C0050	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0054	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000

Table A-88: ADSP-2184x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0054	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0054	IIR_OUTIDX	IIR Output Data Buffer Index Register	0x00000000
0x310C0058	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0058	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0058	IIR_OUTMOD	IIR IIR Output Data Index Modifier Register	0x00000000
0x310C005C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C005C	IIR_OUTLEN	IIR IIR Output Data Buffer Length Register	0x00000000
0x310C005C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0060	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0060	IIR_OUTBASE	IIR Output Buffer Base Register	0x00000000
0x310C0060	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0064	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0064	IIR_COEFIDX	IIR Coefficient Buffer Index Register	0x00000000
0x310C0064	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0068	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0068	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0068	IIR_COEFMOD	IIR Coefficient Index Modifier Register	0x00000000
0x310C006C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C006C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C006C	IIR_COEFLEN	IIR Coefficient Buffer Length Register	0x00000000
0x310C0070	IIR_CHNPTR	IIR Chain Pointer Register	0x00000000
0x310C0070	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0070	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0074	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0074	IIR_SCTL1	IIR Software Control Register1	0x00000000
0x310C0074	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C0078	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0078	IIR_SCTL2	IIR Software Control Register2	0x00000000
0x310C0078	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000
0x310C007C	IIR_SLEW_CNFG[n]	IIR Channel 0 Slewing Duration and Frequency	0x00000000
0x310C007C	IIR_ALPHA[n]	IIR Channel n Alpha	0x00000000

Table A-88: ADSP-2184x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C007C	IIR_SGCTL	IIR Secondary Global Control Register	0x00000000
0x310C4000	IIR0_CTL1	IIR0 Global Control Register	0x00000000
0x310C4004	IIR0_DMASTAT	IIR0 DMA Status Register	0x00000000
0x310C4008	IIR0_MACSTAT	IIR0 MAC Status Register	0x00000000
0x310C400C	IIR0_DBG_CTL	IIR0 IIR Debug Control Register	0x00000000
0x310C4010	IIR0_DBG_ADDR	IIR0 IIR Debug Address Register	0x00000000
0x310C4014	IIR0_DBG_WRDAT_LO	IIR0 IIR Debug Write Data Low Register	0x00000000
0x310C4018	IIR0_DBG_WRDAT_HI	IIR0 IIR Debug Write Data High Register	0x00000000
0x310C401C	IIR0_DBG_RDDAT_LO	IIR0 IIR Debug Read Data Low Register	0x00000000
0x310C4020	IIR0_DBG_RDDAT_HI	IIR0 IIR Debug Read Data High Register	0x00000000
0x310C4040	IIR0_CTL2	IIR0 Channel Control Register	0x00000000
0x310C4044	IIR0_INIDX	IIR0 Input Data Index Register	0x00000000
0x310C4048	IIR0_INMOD	IIR0 Input Data Index Modifier Register	0x00000000
0x310C404C	IIR0_INLEN	IIR0 Input Data Buffer Length Register	0x00000000
0x310C4050	IIR0_INBASE	IIR0 Input Buffer Base Register	0x00000000
0x310C4054	IIR0_OUTIDX	IIR0 Output Data Buffer Index Register	0x00000000
0x310C4058	IIR0_OUTMOD	IIR0 IIR Output Data Index Modifier Register	0x00000000
0x310C405C	IIR0_OUTLEN	IIR0 IIR Output Data Buffer Length Register	0x00000000
0x310C4060	IIR0_OUTBASE	IIR0 Output Buffer Base Register	0x00000000
0x310C4064	IIR0_COEFIDX	IIR0 Coefficient Buffer Index Register	0x00000000
0x310C4068	IIR0_COEFMOD	IIR0 Coefficient Index Modifier Register	0x00000000
0x310C406C	IIR0_COEFLEN	IIR0 Coefficient Buffer Length Register	0x00000000
0x310C4070	IIR0_CHNPTR	IIR0 Chain Pointer Register	0x00000000
0x310C4074	IIR0_SCTL1	IIR0 Software Control Register1	0x00000000
0x310C4078	IIR0_SCTL2	IIR0 Software Control Register2	0x00000000
0x310C407C	IIR0_SGCTL	IIR0 Secondary Global Control Register	0x00000000
0x310C4100	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4104	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4108	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C410C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4110	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000

Table A-88: ADSP-2184x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C4114	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4118	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C411C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4120	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4124	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4128	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C412C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4130	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4134	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4138	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C413C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4140	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4144	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4148	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C414C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4150	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4154	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4158	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C415C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4160	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4164	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4168	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C416C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4170	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4174	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4178	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C417C	IIR0_ALPHA[n]	IIR0 Channel n Alpha	0x00000000
0x310C4180	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C4184	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C4188	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C418C	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000

Table A-88: ADSP-2184x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C4190	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C4194	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C4198	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C419C	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41A0	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41A4	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41A8	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41AC	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41B0	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41B4	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41B8	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41BC	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41C0	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41C4	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41C8	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41CC	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41D0	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41D4	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41D8	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41DC	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41E0	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41E4	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41E8	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41EC	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41F0	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41F4	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41F8	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C41FC	IIR0_SLEW_CNFG[n]	IIR0 Channel 0 Slewing Duration and Frequency	0x00000000

Table A-89: ADSP-2184x IIR1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0000	IIR1_CTL1	IIR1 Global Control Register	0x00000000
0x310C0004	IIR1_DMASTAT	IIR1 DMA Status Register	0x00000000
0x310C0008	IIR1_MACSTAT	IIR1 MAC Status Register	0x00000000
0x310C000C	IIR1_DBG_CTL	IIR1 IIR Debug Control Register	0x00000000
0x310C0010	IIR1_DBG_ADDR	IIR1 IIR Debug Address Register	0x00000000
0x310C0014	IIR1_DBG_WRDAT_LO	IIR1 IIR Debug Write Data Low Register	0x00000000
0x310C0018	IIR1_DBG_WRDAT_HI	IIR1 IIR Debug Write Data High Register	0x00000000
0x310C001C	IIR1_DBG_RDDAT_LO	IIR1 IIR Debug Read Data Low Register	0x00000000
0x310C0020	IIR1_DBG_RDDAT_HI	IIR1 IIR Debug Read Data High Register	0x00000000
0x310C0040	IIR1_CTL2	IIR1 Channel Control Register	0x00000000
0x310C0044	IIR1_INIDX	IIR1 Input Data Index Register	0x00000000
0x310C0048	IIR1_INMOD	IIR1 Input Data Index Modifier Register	0x00000000
0x310C004C	IIR1_INLEN	IIR1 Input Data Buffer Length Register	0x00000000
0x310C0050	IIR1_INBASE	IIR1 Input Buffer Base Register	0x00000000
0x310C0054	IIR1_OUTIDX	IIR1 Output Data Buffer Index Register	0x00000000
0x310C0058	IIR1_OUTMOD	IIR1 IIR Output Data Index Modifier Register	0x00000000
0x310C005C	IIR1_OUTLEN	IIR1 IIR Output Data Buffer Length Register	0x00000000
0x310C0060	IIR1_OUTBASE	IIR1 Output Buffer Base Register	0x00000000
0x310C0064	IIR1_COEFIDX	IIR1 Coefficient Buffer Index Register	0x00000000
0x310C0068	IIR1_COEFMOD	IIR1 Coefficient Index Modifier Register	0x00000000
0x310C006C	IIR1_COEFLEN	IIR1 Coefficient Buffer Length Register	0x00000000
0x310C0070	IIR1_CHNPTR	IIR1 Chain Pointer Register	0x00000000
0x310C0074	IIR1_SCTL1	IIR1 Software Control Register1	0x00000000
0x310C0078	IIR1_SCTL2	IIR1 Software Control Register2	0x00000000
0x310C007C	IIR1_SGCTL	IIR1 Secondary Global Control Register	0x00000000
0x310C0100	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0104	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0108	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C010C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0110	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0114	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000

Table A-89: ADSP-2184x IIR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0118	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C011C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0120	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0124	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0128	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C012C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0130	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0134	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0138	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C013C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0140	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0144	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0148	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C014C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0150	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0154	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0158	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C015C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0160	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0164	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0168	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C016C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0170	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0174	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0178	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C017C	IIR1_ALPHA[n]	IIR1 Channel n Alpha	0x00000000
0x310C0180	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0184	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0188	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C018C	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0190	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000

Table A-89: ADSP-2184x IIR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0194	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C0198	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C019C	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01A0	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01A4	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01A8	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01AC	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01B0	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01B4	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01B8	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01BC	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01C0	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01C4	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01C8	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01CC	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01D0	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01D4	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01D8	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01DC	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01E0	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01E4	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01E8	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01EC	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01F0	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01F4	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01F8	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000
0x310C01FC	IIR1_SLEW_CNFG[n]	IIR1 Channel 0 Slewing Duration and Frequency	0x00000000

Table A-90: ADSP-2184x IIR2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C1000	IIR2_CTL1	IIR2 Global Control Register	0x00000000
0x310C1004	IIR2_DMASTAT	IIR2 DMA Status Register	0x00000000
0x310C1008	IIR2_MACSTAT	IIR2 MAC Status Register	0x00000000
0x310C100C	IIR2_DBG_CTL	IIR2 IIR Debug Control Register	0x00000000
0x310C1010	IIR2_DBG_ADDR	IIR2 IIR Debug Address Register	0x00000000
0x310C1014	IIR2_DBG_WRDAT_LO	IIR2 IIR Debug Write Data Low Register	0x00000000
0x310C1018	IIR2_DBG_WRDAT_HI	IIR2 IIR Debug Write Data High Register	0x00000000
0x310C101C	IIR2_DBG_RDDAT_LO	IIR2 IIR Debug Read Data Low Register	0x00000000
0x310C1020	IIR2_DBG_RDDAT_HI	IIR2 IIR Debug Read Data High Register	0x00000000
0x310C1040	IIR2_CTL2	IIR2 Channel Control Register	0x00000000
0x310C1044	IIR2_INIDX	IIR2 Input Data Index Register	0x00000000
0x310C1048	IIR2_INMOD	IIR2 Input Data Index Modifier Register	0x00000000
0x310C104C	IIR2_INLEN	IIR2 Input Data Buffer Length Register	0x00000000
0x310C1050	IIR2_INBASE	IIR2 Input Buffer Base Register	0x00000000
0x310C1054	IIR2_OUTIDX	IIR2 Output Data Buffer Index Register	0x00000000
0x310C1058	IIR2_OUTMOD	IIR2 IIR Output Data Index Modifier Register	0x00000000
0x310C105C	IIR2_OUTLEN	IIR2 IIR Output Data Buffer Length Register	0x00000000
0x310C1060	IIR2_OUTBASE	IIR2 Output Buffer Base Register	0x00000000
0x310C1064	IIR2_COEFIDX	IIR2 Coefficient Buffer Index Register	0x00000000
0x310C1068	IIR2_COEFMOD	IIR2 Coefficient Index Modifier Register	0x00000000
0x310C106C	IIR2_COEFLEN	IIR2 Coefficient Buffer Length Register	0x00000000
0x310C1070	IIR2_CHNPTR	IIR2 Chain Pointer Register	0x00000000
0x310C1074	IIR2_SCTL1	IIR2 Software Control Register1	0x00000000
0x310C1078	IIR2_SCTL2	IIR2 Software Control Register2	0x00000000
0x310C107C	IIR2_SGCTL	IIR2 Secondary Global Control Register	0x00000000

Table A-91: ADSP-2184x IIR3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C2000	IIR3_CTL1	IIR3 Global Control Register	0x00000000
0x310C2004	IIR3_DMASTAT	IIR3 DMA Status Register	0x00000000

Table A-91: ADSP-2184x IIR3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C2008	IIR3_MACSTAT	IIR3 MAC Status Register	0x00000000
0x310C200C	IIR3_DBG_CTL	IIR3 IIR Debug Control Register	0x00000000
0x310C2010	IIR3_DBG_ADDR	IIR3 IIR Debug Address Register	0x00000000
0x310C2014	IIR3_DBG_WRDAT_LO	IIR3 IIR Debug Write Data Low Register	0x00000000
0x310C2018	IIR3_DBG_WRDAT_HI	IIR3 IIR Debug Write Data High Register	0x00000000
0x310C201C	IIR3_DBG_RDDAT_LO	IIR3 IIR Debug Read Data Low Register	0x00000000
0x310C2020	IIR3_DBG_RDDAT_HI	IIR3 IIR Debug Read Data High Register	0x00000000
0x310C2040	IIR3_CTL2	IIR3 Channel Control Register	0x00000000
0x310C2044	IIR3_INIDX	IIR3 Input Data Index Register	0x00000000
0x310C2048	IIR3_INMOD	IIR3 Input Data Index Modifier Register	0x00000000
0x310C204C	IIR3_INLEN	IIR3 Input Data Buffer Length Register	0x00000000
0x310C2050	IIR3_INBASE	IIR3 Input Buffer Base Register	0x00000000
0x310C2054	IIR3_OUTIDX	IIR3 Output Data Buffer Index Register	0x00000000
0x310C2058	IIR3_OUTMOD	IIR3 IIR Output Data Index Modifier Register	0x00000000
0x310C205C	IIR3_OUTLEN	IIR3 IIR Output Data Buffer Length Register	0x00000000
0x310C2060	IIR3_OUTBASE	IIR3 Output Buffer Base Register	0x00000000
0x310C2064	IIR3_COEFIDX	IIR3 Coefficient Buffer Index Register	0x00000000
0x310C2068	IIR3_COEFMOD	IIR3 Coefficient Index Modifier Register	0x00000000
0x310C206C	IIR3_COEFLEN	IIR3 Coefficient Buffer Length Register	0x00000000
0x310C2070	IIR3_CHNPTR	IIR3 Chain Pointer Register	0x00000000
0x310C2074	IIR3_SCTL1	IIR3 Software Control Register1	0x00000000
0x310C2078	IIR3_SCTL2	IIR3 Software Control Register2	0x00000000
0x310C207C	IIR3_SGCTL	IIR3 Secondary Global Control Register	0x00000000

Table A-92: ADSP-2184x L2CTL0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31080000	L2CTL0_CTL	L2CTL0 Control Register	0x00000000
0x31080010	L2CTL0_STAT	L2CTL0 Status Register	0x00000000
0x31080014	L2CTL0_RPCR0	L2CTL0 Read Priority Count Register	0x0F0F0F0F
0x31080018	L2CTL0_WPCR0	L2CTL0 Write Priority Count Register	0x0F0F0F0F

Table A-92: ADSP-2184x L2CTL0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31080024	L2CTL0_INIT	L2CTL0 Initialization Register	0x00000000
0x31080038	L2CTL0_ISTAT	L2CTL0 Initialization Status Register	0x00000000
0x3108003C	L2CTL0_PCTL	L2CTL0 Power Control Register	0x00000000
0x31080040	L2CTL0_ERRADDR0	L2CTL0 ECC Error Address 0 Register	0x20000000
0x31080044	L2CTL0_ERRADDR1	L2CTL0 ECC Error Address 1 Register	0x20040000
0x31080048	L2CTL0_ERRADDR2	L2CTL0 ECC Error Address 2 Register	0x20080000
0x3108004C	L2CTL0_ERRADDR3	L2CTL0 ECC Error Address 3 Register	0x200C0000
0x31080050	L2CTL0_ERRADDR4	L2CTL0 ECC Error Address 4 Register	0x20100000
0x31080054	L2CTL0_ERRADDR5	L2CTL0 ECC Error Address 5 Register	0x20140000
0x31080058	L2CTL0_ERRADDR6	L2CTL0 ECC Error Address 6 Register	0x20180000
0x3108005C	L2CTL0_ERRADDR7	L2CTL0 ECC Error Address 7 Register	0x201C0000
0x31080060	L2CTL0_ERRADDR8	L2CTL0 ECC Error Address 8 Register	0x20200000
0x31080080	L2CTL0_ET0	L2CTL0 Error Type 0 Register	0x00000000
0x31080084	L2CTL0_EADDR0	L2CTL0 Error Type 0 Address Register	0x00000000
0x31080088	L2CTL0_ET1	L2CTL0 Error Type 1 Register	0x00000000
0x3108008C	L2CTL0_EADDR1	L2CTL0 Error Type 1 Address Register	0x00000000
0x310800EC	L2CTL0_SCTL	L2CTL0 Scrub Control Register	0x00000000
0x310800F0	L2CTL0_SADR	L2CTL0 Scrub Start Address Register	0x00000000
0x310800F4	L2CTL0_SCNT	L2CTL0 Scrub Count Register	0x00000000
0x310800FC	L2CTL0_REVID	L2CTL0 Revision ID Register	0x00000003
0x31080100	L2CTL0_STAT_1	L2CTL0 L2 Port Error Status Register	0x00000000
0x31080104	L2CTL0_ET2	L2CTL0 Error Type 2 Register	0x00000000
0x31080108	L2CTL0_EADDR2	L2CTL0 Error Type 2 Address Register	0x00000000
0x3108010C	L2CTL0_ET3	L2CTL0 Error Type 3 Register	0x00000000
0x31080110	L2CTL0_EADDR3	L2CTL0 Error Type 3 Address Register	0x00000000
0x31080114	L2CTL0_ET4	L2CTL0 Error Type 4 Register	0x00000000
0x31080118	L2CTL0_EADDR4	L2CTL0 Error Type 4 Address Register	0x00000000
0x31080130	L2CTL0_RPCR1	L2CTL0 Read Priority Count Register	0x0000000F
0x31080134	L2CTL0_WPCR1	L2CTL0 Write Priority Count Register	0x0000000F

Table A-93: ADSP-2184x L2CTL1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31081000	L2CTL1_CTL	L2CTL1 Control Register	0x00000000
0x31081010	L2CTL1_STAT	L2CTL1 Status Register	0x00000000
0x31081014	L2CTL1_RPCR0	L2CTL1 Read Priority Count Register	0x0F0F0F0F
0x31081018	L2CTL1_WPCR0	L2CTL1 Write Priority Count Register	0x0F0F0F0F
0x31081024	L2CTL1_INIT	L2CTL1 Initialization Register	0x00000000
0x31081038	L2CTL1_ISTAT	L2CTL1 Initialization Status Register	0x00000000
0x3108103C	L2CTL1_PCTL	L2CTL1 Power Control Register	0x00000000
0x31081040	L2CTL1_ERRADDR0	L2CTL1 ECC Error Address 0 Register	0x20000000
0x31081044	L2CTL1_ERRADDR1	L2CTL1 ECC Error Address 1 Register	0x20040000
0x31081048	L2CTL1_ERRADDR2	L2CTL1 ECC Error Address 2 Register	0x20080000
0x3108104C	L2CTL1_ERRADDR3	L2CTL1 ECC Error Address 3 Register	0x200C0000
0x31081050	L2CTL1_ERRADDR4	L2CTL1 ECC Error Address 4 Register	0x20100000
0x31081054	L2CTL1_ERRADDR5	L2CTL1 ECC Error Address 5 Register	0x20140000
0x31081058	L2CTL1_ERRADDR6	L2CTL1 ECC Error Address 6 Register	0x20180000
0x3108105C	L2CTL1_ERRADDR7	L2CTL1 ECC Error Address 7 Register	0x201C0000
0x31081060	L2CTL1_ERRADDR8	L2CTL1 ECC Error Address 8 Register	0x20200000
0x31081080	L2CTL1_ET0	L2CTL1 Error Type 0 Register	0x00000000
0x31081084	L2CTL1_EADDR0	L2CTL1 Error Type 0 Address Register	0x00000000
0x31081088	L2CTL1_ET1	L2CTL1 Error Type 1 Register	0x00000000
0x3108108C	L2CTL1_EADDR1	L2CTL1 Error Type 1 Address Register	0x00000000
0x310810EC	L2CTL1_SCTL	L2CTL1 Scrub Control Register	0x00000000
0x310810F0	L2CTL1_SADR	L2CTL1 Scrub Start Address Register	0x00000000
0x310810F4	L2CTL1_SCNT	L2CTL1 Scrub Count Register	0x00000000
0x310810FC	L2CTL1_REVID	L2CTL1 Revision ID Register	0x00000003
0x31081100	L2CTL1_STAT_1	L2CTL1 L2 Port Error Status Register	0x00000000
0x31081104	L2CTL1_ET2	L2CTL1 Error Type 2 Register	0x00000000
0x31081108	L2CTL1_EADDR2	L2CTL1 Error Type 2 Address Register	0x00000000
0x3108110C	L2CTL1_ET3	L2CTL1 Error Type 3 Register	0x00000000
0x31081110	L2CTL1_EADDR3	L2CTL1 Error Type 3 Address Register	0x00000000
0x31081114	L2CTL1_ET4	L2CTL1 Error Type 4 Register	0x00000000
0x31081118	L2CTL1_EADDR4	L2CTL1 Error Type 4 Address Register	0x00000000

Table A-93: ADSP-2184x L2CTL1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31081130	L2CTL1_RPCR1	L2CTL1 Read Priority Count Register	0x0000000F
0x31081134	L2CTL1_WPCR1	L2CTL1 Write Priority Count Register	0x0000000F

Table A-94: ADSP-2184x LP0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFE000	LP0_CTL	LP0 Control Register	0x00000000
0x30FFE004	LP0_STAT	LP0 Status Register	0x00000000
0x30FFE008	LP0_DIV	LP0 Clock Divider Value Register	0x00000000
0x30FFE010	LP0_TX	LP0 Transmit Buffer Register	0x00000000
0x30FFE014	LP0_RX	LP0 Receive Buffer Register	0x00000000
0x30FFE018	LP0_TXIN_SHDW	LP0 Shadow Input Transmit Buffer Register	0x00000000
0x30FFE01C	LP0_TXOUT_SHDW	LP0 Shadow Output Transmit Buffer Register	0x00000000

Table A-95: ADSP-2184x LP1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFE100	LP1_CTL	LP1 Control Register	0x00000000
0x30FFE104	LP1_STAT	LP1 Status Register	0x00000000
0x30FFE108	LP1_DIV	LP1 Clock Divider Value Register	0x00000000
0x30FFE110	LP1_TX	LP1 Transmit Buffer Register	0x00000000
0x30FFE114	LP1_RX	LP1 Receive Buffer Register	0x00000000
0x30FFE118	LP1_TXIN_SHDW	LP1 Shadow Input Transmit Buffer Register	0x00000000
0x30FFE11C	LP1_TXOUT_SHDW	LP1 Shadow Output Transmit Buffer Register	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1000C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1001C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1002C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1003C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1004C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1005C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1006C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1007C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1008C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1009C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x100C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x100E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x100FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1010C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1011C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1012C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1013C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1014C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1015C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1016C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1017C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1018C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1019C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x101C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x101CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x101FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1020C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1021C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1022C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1023C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1024C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1025C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1026C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1027C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1028C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1029C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x102B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x102C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x102FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1030C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1031C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1032C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1033C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1034C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1035C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1036C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1037C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1038C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1039C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x103A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x103C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x103FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1040C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1041C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1042C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1043C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1044C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1045C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1046C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1047C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1048C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1049C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x104C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x104F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x104FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1050C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1051C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1052C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1053C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1054C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1055C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1056C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1057C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1058C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1059C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x105C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x105E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x105FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1060C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1061C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1062C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1063C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1064C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1065C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1066C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1067C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1068C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1069C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x106C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x106CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x106FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1070C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1071C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1072C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1073C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1074C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1075C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1076C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1077C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1078C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1079C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x107B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x107C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x107FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1080C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1081C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1082C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1083C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1084C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1085C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1086C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1087C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1088C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1089C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x108C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x108FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1090C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1091C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1092C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1093C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1094C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1095C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1096C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1097C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1098C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1099C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x109C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x109F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x109FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x10FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x10FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x10FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1100C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1101C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1102C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1103C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1104C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1105C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1106C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1107C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1108C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1109C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x110C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x110C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x110FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1110C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1111C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1112C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1113C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1114C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1115C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1116C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1117C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1118C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1119C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x111B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x111C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x111FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1120C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1121C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1122C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1123C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1124C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1125C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1126C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1127C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1128C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1129C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x112C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x112FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1130C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1131C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1132C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1133C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1134C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1135C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1136C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1137C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1138C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1139C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x113C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x113F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x113FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1140C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1141C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1142C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1143C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1144C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1145C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1146C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1147C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1148C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1149C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x114C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x114E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x114FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1150C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1151C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1152C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1153C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1154C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1155C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1156C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1157C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1158C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1159C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x115C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x115C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x115FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1160C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1161C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1162C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1163C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1164C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1165C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1166C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1167C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1168C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1169C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x116B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x116C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x116FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1170C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1171C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1172C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1173C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1174C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1175C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1176C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1177C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1178C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1179C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x117C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x117FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1180C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1181C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1182C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1183C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1184C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1185C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1186C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1187C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1188C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1189C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x118C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x118F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x118FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1190C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1191C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1192C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1193C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1194C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1195C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1196C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1197C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1198C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1199C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x119C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x119E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x119FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x11FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x11FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x11FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1200C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1201C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1202C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1203C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1204C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1205C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1206C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1207C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1208C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1209C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x120B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x120C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x120FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1210C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1211C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1212C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1213C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1214C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1215C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1216C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1217C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1218C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1219C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x121C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x121FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1220C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1221C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1222C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1223C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1224C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1225C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1226C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1227C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1228C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1229C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x122C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x122F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x122FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1230C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1231C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1232C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1233C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1234C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1235C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1236C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1237C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1238C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1239C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x123C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x123E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x123FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1240C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1241C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1242C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1243C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1244C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1245C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1246C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1247C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1248C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1249C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x124C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x124C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x124FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1250C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1251C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1252C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1253C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1254C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1255C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1256C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1257C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1258C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1259C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x125B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x125C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x125FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1260C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1261C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1262C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1263C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1264C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1265C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1266C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1267C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1268C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1269C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x126C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x126FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1270C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1271C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1272C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1273C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1274C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1275C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1276C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1277C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1278C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1279C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x127C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x127F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x127FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1280C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1281C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1282C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1283C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1284C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1285C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1286C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1287C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1288C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1289C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x128C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x128DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x128FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1290C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1291C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1292C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1293C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1294C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1295C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1296C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1297C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1298C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1299C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x129C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x129C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x129FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x12FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x12FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x12FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1300C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1301C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1302C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1303C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1304C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1305C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1306C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1307C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1308C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1309C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x130C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x130FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1310C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1311C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1312C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1313C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1314C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1315C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1316C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1317C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1318C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1319C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x131C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x131F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x131FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1320C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1321C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1322C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1323C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1324C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1325C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1326C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1327C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1328C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1329C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x132C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x132DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x132FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1330C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1331C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1332C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1333C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1334C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1335C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1336C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1337C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1338C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1339C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x133C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x133C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x133FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1340C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1341C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1342C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1343C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1344C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1345C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1346C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1347C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1348C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1349C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x134AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x134C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x134FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1350C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1351C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1352C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1353C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1354C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1355C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1356C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1357C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1358C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1359C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x135C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x135FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1360C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1361C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1362C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1363C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1364C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1365C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1366C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1367C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1368C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1369C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x136C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x136F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x136FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1370C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1371C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1372C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1373C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1374C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1375C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1376C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1377C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1378C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1379C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x137C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x137DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x137FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1380C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1381C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1382C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1383C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1384C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1385C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1386C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1387C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1388C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1389C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x138C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x138FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1390C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1391C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1392C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1393C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1394C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1395C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1396C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1397C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1398C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1399C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x139AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x139C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x139FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x13F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x13FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x13FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1400C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1401C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1402C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1403C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1404C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1405C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1406C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1407C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1408C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1409C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x140C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x140EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x140FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1410C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1411C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1412C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1413C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1414C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1415C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1416C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1417C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1418C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1419C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x141C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x141D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x141FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1420C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1421C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1422C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1423C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1424C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1425C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1426C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1427C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1428C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1429C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x142C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x142C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x142FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1430C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1431C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1432C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1433C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1434C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1435C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1436C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1437C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1438C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1439C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x143A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x143C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x143FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1440C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1441C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1442C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1443C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1444C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1445C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1446C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1447C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1448C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1449C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x144C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x144FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1450C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1451C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1452C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1453C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1454C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1455C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1456C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1457C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1458C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1459C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x145C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x145EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x145FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1460C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1461C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1462C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1463C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1464C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1465C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1466C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1467C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1468C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1469C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x146C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x146D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x146FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1470C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1471C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1472C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1473C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1474C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1475C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1476C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1477C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1478C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1479C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x147C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x147C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x147FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1480C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1481C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1482C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1483C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1484C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1485C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1486C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1487C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1488C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1489C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x148A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x148C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x148FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1490C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1491C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1492C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1493C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1494C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1495C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1496C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1497C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1498C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1499C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x149C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x149FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x14FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x14FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x14FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1500C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1501C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1502C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1503C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1504C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1505C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1506C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1507C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1508C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1509C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x150C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x150D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x150FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1510C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1511C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1512C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1513C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1514C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1515C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1516C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1517C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1518C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1519C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x151BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x151C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x151FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1520C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1521C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1522C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1523C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1524C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1525C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1526C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1527C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1528C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1529C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x152A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x152C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x152FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1530C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1531C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1532C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1533C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1534C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1535C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1536C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1537C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1538C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1539C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x153C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x153FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1540C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1541C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1542C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1543C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1544C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1545C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1546C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1547C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1548C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1549C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x154C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x154E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x154FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1550C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1551C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1552C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1553C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1554C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1555C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1556C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1557C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1558C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1559C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x155C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x155D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x155FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1560C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1561C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1562C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1563C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1564C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1565C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1566C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1567C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1568C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1569C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x156BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x156C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x156FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1570C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1571C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1572C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1573C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1574C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1575C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1576C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1577C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1578C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1579C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x157A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x157C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x157FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1580C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1581C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1582C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1583C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1584C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1585C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1586C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1587C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1588C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1589C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x158C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x158FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1590C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1591C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1592C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1593C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1594C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1595C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1596C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1597C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1598C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1599C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x159C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x159E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x159FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x15FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x15FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x15FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1600C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1601C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1602C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1603C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1604C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1605C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1606C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1607C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1608C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1609C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x160B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x160C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x160FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1610C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1611C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1612C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1613C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1614C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1615C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1616C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1617C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1618C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1619C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x161A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x161C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x161FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1620C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1621C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1622C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1623C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1624C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1625C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1626C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1627C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1628C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1629C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x162C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x162FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1630C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1631C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1632C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1633C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1634C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1635C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1636C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1637C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1638C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1639C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x163C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x163E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x163FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1640C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1641C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1642C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1643C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1644C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1645C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1646C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1647C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1648C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1649C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x164C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x164CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x164FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1650C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1651C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1652C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1653C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1654C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1655C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1656C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1657C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1658C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1659C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x165B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x165C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x165FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1660C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1661C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1662C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1663C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1664C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1665C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1666C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1667C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1668C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1669C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x166A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x166C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x166FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1670C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1671C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1672C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1673C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1674C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1675C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1676C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1677C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1678C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1679C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x167C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x167FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1680C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1681C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1682C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1683C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1684C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1685C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1686C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1687C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1688C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1689C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x168C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x168E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x168FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1690C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1691C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1692C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1693C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1694C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1695C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1696C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1697C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1698C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1699C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x169C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x169CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x169FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x16FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x16FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x16FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17000	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17004	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17008	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1700C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17010	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17014	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17018	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1701C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17020	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17024	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17028	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1702C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17030	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17034	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17038	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1703C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17040	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17044	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17048	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1704C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17050	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17054	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17058	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1705C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17060	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17064	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17068	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1706C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17070	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17074	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17078	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1707C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17080	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17084	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17088	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1708C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17090	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17094	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17098	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1709C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x170C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x170FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17100	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17104	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17108	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1710C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17110	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17114	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17118	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1711C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17120	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17124	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17128	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1712C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17130	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17134	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17138	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1713C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17140	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17144	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17148	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1714C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17150	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17154	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17158	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1715C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17160	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17164	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17168	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1716C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17170	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17174	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17178	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1717C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17180	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17184	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17188	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1718C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17190	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17194	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17198	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1719C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x171C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x171F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x171FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17200	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17204	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17208	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1720C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17210	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17214	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17218	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1721C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17220	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17224	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17228	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1722C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17230	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17234	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17238	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1723C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17240	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17244	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17248	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1724C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17250	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17254	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17258	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1725C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17260	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17264	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17268	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1726C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17270	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17274	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17278	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1727C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17280	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17284	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17288	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1728C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17290	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17294	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17298	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1729C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x172C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x172E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x172FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17300	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17304	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17308	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1730C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17310	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17314	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17318	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1731C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17320	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17324	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17328	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1732C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17330	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17334	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17338	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1733C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17340	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17344	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17348	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1734C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17350	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17354	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17358	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1735C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17360	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17364	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17368	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1736C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17370	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17374	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17378	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1737C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17380	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17384	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17388	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1738C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17390	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17394	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17398	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1739C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x173C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x173CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x173FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17400	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17404	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17408	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1740C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17410	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17414	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17418	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1741C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17420	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17424	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17428	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1742C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17430	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17434	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17438	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1743C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17440	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17444	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17448	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1744C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17450	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17454	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17458	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1745C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17460	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17464	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17468	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1746C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17470	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17474	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17478	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1747C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17480	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17484	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17488	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1748C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17490	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17494	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17498	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1749C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x174B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x174C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x174FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17500	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17504	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17508	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1750C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17510	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17514	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17518	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1751C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17520	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17524	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17528	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1752C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17530	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17534	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17538	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1753C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17540	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17544	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17548	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1754C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17550	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17554	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17558	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1755C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17560	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17564	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17568	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1756C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17570	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17574	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17578	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1757C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17580	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17584	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17588	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1758C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17590	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17594	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17598	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1759C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x175C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x175FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17600	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17604	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17608	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1760C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17610	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17614	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17618	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1761C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17620	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17624	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17628	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1762C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17630	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17634	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17638	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1763C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17640	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17644	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17648	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1764C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17650	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17654	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17658	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1765C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17660	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17664	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17668	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1766C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17670	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17674	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17678	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1767C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17680	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17684	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17688	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1768C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17690	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17694	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17698	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1769C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x176C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x176F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x176FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17700	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17704	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17708	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1770C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17710	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17714	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17718	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1771C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17720	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17724	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17728	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1772C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17730	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17734	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17738	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1773C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17740	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17744	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17748	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1774C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17750	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17754	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17758	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1775C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17760	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17764	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17768	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1776C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17770	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17774	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17778	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1777C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17780	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17784	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17788	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1778C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17790	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17794	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17798	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1779C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x177C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x177E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x177FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17800	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17804	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17808	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1780C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17810	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17814	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17818	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1781C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17820	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17824	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17828	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1782C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17830	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17834	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17838	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1783C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17840	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17844	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17848	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1784C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17850	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17854	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17858	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1785C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17860	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17864	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17868	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1786C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17870	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17874	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17878	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1787C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17880	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17884	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17888	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1788C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17890	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17894	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17898	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1789C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x178C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x178C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x178FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17900	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17904	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17908	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1790C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17910	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17914	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17918	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1791C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17920	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17924	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17928	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1792C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17930	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17934	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17938	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1793C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17940	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17944	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17948	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1794C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17950	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17954	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17958	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1795C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17960	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17964	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17968	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1796C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17970	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17974	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17978	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1797C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17980	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17984	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17988	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1798C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17990	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17994	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17998	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x1799C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179A0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179A4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179A8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179AC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179B0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x179B4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179B8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179BC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x179C0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179C4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179C8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179CC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179D0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179D4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179D8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179DC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179E0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179E4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179E8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179EC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179F0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179F4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179F8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x179FC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17A00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17A28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17A40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17A80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17A98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17A9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ABC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17AC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ACC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ADC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17AFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17B00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17B10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17B40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17B80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17B84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17B9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17BC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17BF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17BFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17C00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17C40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17C6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17C80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17C9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17CC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17CE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17CFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17D00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17D40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17D54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17D80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17D9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17DC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17DC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17DFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17E00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17E40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17E80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17E9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17EB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17EC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ECC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ED0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ED4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17ED8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17EFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17F00	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F04	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F08	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F0C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F10	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F14	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F18	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F1C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F20	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F24	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17F28	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F2C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F30	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F34	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F38	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F3C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17F40	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F44	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F48	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F4C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F50	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F54	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F58	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F5C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F60	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F64	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F68	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F6C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F70	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F74	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F78	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F7C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17F80	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F84	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F88	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F8C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F90	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F94	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17F98	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x17F9C	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FA0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FA4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FA8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FAC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FB0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FB4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FB8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FBC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x17FC0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FC4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FC8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FCC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FD0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FD4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FD8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FDC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FE0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FE4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FE8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FEC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FF0	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FF4	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FF8	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x17FFC	LPDDRMEM_DCCM_[n]	LPDDRMEM DCCM mem space	0x00000000
0x18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x180C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x181C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x182C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x183C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x184C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x185C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x186C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x187C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x188C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x189C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x18F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x18FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x190C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x191C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x192C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x193C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x194C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x195C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x196C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x197C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x198C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x199C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x19E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x19FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x1FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x1FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x200C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x201C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x202C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x203C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x204C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x205C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x206C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x207C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x208C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x209C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x20B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x20FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x210C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x211C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x212C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x213C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x214C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x215C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x216C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x217C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x218C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x219C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x21FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x220C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x221C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x222C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x223C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x224C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x225C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x226C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x227C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x228C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x229C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x22F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x22FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x230C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x231C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x232C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x233C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x234C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x235C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x236C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x237C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x238C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x239C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x23E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x23FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x240C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x241C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x242C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x243C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x244C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x245C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x246C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x247C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x248C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x249C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x24C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x24FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x250C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x251C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x252C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x253C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x254C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x255C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x256C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x257C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x258C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x259C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x25B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x25FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x260C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x261C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x262C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x263C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x264C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x265C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x266C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x267C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x268C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x269C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x26FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x270C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x271C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x272C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x273C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x274C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x275C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x276C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x277C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x278C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x279C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x27F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x27FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x280C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x281C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x282C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x283C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x284C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x285C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x286C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x287C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x288C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x289C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x28DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x28FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x290C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x291C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x292C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x293C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x294C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x295C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x296C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x297C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x298C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x299C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x29C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x29FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x2FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x2FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x300C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x301C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x302C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x303C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x304C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x305C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x306C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x307C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x308C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x309C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x30FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x310C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x311C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x312C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x313C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x314C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x315C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x316C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x317C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x318C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x319C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x31FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x320C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x321C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x322C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x323C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x324C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x325C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x326C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x327C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x328C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x329C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x32DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x32FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x330C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x331C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x332C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x333C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x334C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x335C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x336C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x337C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x338C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x339C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x33C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x33FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x340C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x341C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x342C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x343C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x344C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x345C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x346C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x347C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x348C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x349C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x34AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x34FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x350C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x351C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x352C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x353C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x354C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x355C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x356C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x357C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x358C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x359C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x35FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x360C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x361C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x362C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x363C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x364C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x365C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x366C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x367C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x368C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x369C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x36F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x36FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x370C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x371C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x372C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x373C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x374C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x375C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x376C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x377C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x378C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x379C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x37DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x37FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x380C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x381C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x382C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x383C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x384C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x385C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x386C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x387C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x388C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x389C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x38C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x38FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x390C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x391C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x392C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x393C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x394C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x395C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x396C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x397C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x398C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x399C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x39AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x39FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x3FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x400C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x401C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x402C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x403C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x404C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x405C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x406C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x407C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x408C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x409C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x40F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x40FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x410C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x411C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x412C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x413C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x414C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x415C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x416C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x417C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x418C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x419C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x41D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x41FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x420C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x421C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x422C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x423C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x424C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x425C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x426C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x427C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x428C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x429C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x42C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x42FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x430C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x431C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x432C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x433C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x434C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x435C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x436C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x437C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x438C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x439C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x43AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x43FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x440C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x441C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x442C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x443C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x444C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x445C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x446C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x447C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x448C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x449C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x44FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x450C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x451C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x452C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x453C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x454C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x455C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x456C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x457C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x458C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x459C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x45F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x45FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x460C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x461C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x462C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x463C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x464C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x465C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x466C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x467C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x468C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x469C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x46D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x46FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x470C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x471C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x472C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x473C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x474C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x475C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x476C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x477C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x478C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x479C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x47C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x47FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x480C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x481C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x482C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x483C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x484C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x485C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x486C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x487C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x488C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x489C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x48A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x48FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x490C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x491C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x492C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x493C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x494C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x495C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x496C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x497C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x498C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x499C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x49FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x4FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x4FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x500C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x501C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x502C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x503C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x504C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x505C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x506C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x507C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x508C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x509C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x50D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x50FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x510C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x511C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x512C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x513C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x514C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x515C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x516C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x517C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x518C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x519C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x51FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x520C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x521C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x522C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x523C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x524C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x525C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x526C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x527C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x528C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x529C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x52A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x52FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x530C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x531C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x532C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x533C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x534C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x535C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x536C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x537C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x538C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x539C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x53FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x540C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x541C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x542C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x543C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x544C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x545C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x546C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x547C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x548C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x549C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x54EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x54FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x550C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x551C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x552C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x553C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x554C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x555C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x556C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x557C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x558C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x559C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x55D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x55FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x560C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x561C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x562C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x563C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x564C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x565C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x566C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x567C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x568C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x569C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x56FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x570C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x571C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x572C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x573C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x574C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x575C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x576C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x577C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x578C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x579C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x57A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x57FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x580C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x581C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x582C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x583C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x584C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x585C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x586C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x587C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x588C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x589C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x58FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x590C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x591C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x592C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x593C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x594C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x595C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x596C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x597C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x598C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x599C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x59EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x59FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x5FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x5FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x600C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x601C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x602C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x603C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x604C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x605C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x606C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x607C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x608C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x609C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x60BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x60FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x610C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x611C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x612C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x613C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x614C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x615C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x616C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x617C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x618C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x619C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x61A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x61FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x620C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x621C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x622C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x623C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x624C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x625C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x626C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x627C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x628C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x629C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x62FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x630C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x631C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x632C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x633C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x634C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x635C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x636C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x637C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x638C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x639C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x63EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x63FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x640C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x641C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x642C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x643C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x644C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x645C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x646C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x647C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x648C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x649C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x64D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x64FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x650C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x651C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x652C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x653C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x654C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x655C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x656C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x657C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x658C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x659C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x65BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x65FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x660C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x661C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x662C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x663C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x664C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x665C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x666C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x667C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x668C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x669C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x66A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x66FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x670C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x671C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x672C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x673C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x674C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x675C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x676C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x677C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x678C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x679C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x67FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x680C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x681C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x682C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x683C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x684C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x685C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x686C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x687C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x688C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x689C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x68E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x68FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x690C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x691C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x692C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x693C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x694C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x695C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x696C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x697C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x698C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x699C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x69D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x69FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x6FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x6FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x700C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x701C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x702C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x703C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x704C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x705C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x706C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x707C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x708C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x709C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x70A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x70FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x710C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x711C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x712C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x713C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x714C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x715C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x716C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x717C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x718C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x719C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x71FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x720C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x721C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x722C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x723C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x724C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x725C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x726C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x727C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x728C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x729C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x72E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x72FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x730C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x731C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x732C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x733C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x734C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x735C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x736C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x737C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x738C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x739C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x73D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x73FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x740C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x741C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x742C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x743C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x744C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x745C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x746C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x747C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x748C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x749C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x74B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x74FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x750C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x751C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x752C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x753C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x754C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x755C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x756C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x757C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x758C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x759C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x75A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x75FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x760C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x761C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x762C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x763C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x764C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x765C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x766C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x767C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x768C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x769C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x76FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x770C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x771C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x772C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x773C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x774C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x775C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x776C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x777C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x778C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x779C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x77E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x77FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x780C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x781C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x782C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x783C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x784C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x785C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x786C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x787C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x788C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x789C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x78CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x78FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x790C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x791C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x792C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x793C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x794C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x795C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x796C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x797C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x798C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x799C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x79B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x79FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x7FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x7FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x800C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x801C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x802C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x803C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x804C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x805C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x806C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x807C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x808C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x809C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x80F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x80FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x810C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x811C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x812C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x813C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x814C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x815C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x816C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x817C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x818C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x819C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x81E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x81FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x820C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x821C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x822C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x823C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x824C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x825C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x826C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x827C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x828C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x829C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x82CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x82FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x830C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x831C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x832C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x833C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x834C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x835C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x836C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x837C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x838C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x839C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x83B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x83FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x840C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x841C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x842C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x843C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x844C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x845C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x846C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x847C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x848C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x849C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x84FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x850C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x851C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x852C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x853C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x854C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x855C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x856C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x857C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x858C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x859C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x85F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x85FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x860C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x861C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x862C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x863C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x864C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x865C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x866C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x867C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x868C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x869C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x86E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x86FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x870C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x871C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x872C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x873C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x874C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x875C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x876C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x877C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x878C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x879C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x87CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x87FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x880C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x881C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x882C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x883C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x884C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x885C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x886C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x887C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x888C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x889C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x88B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x88FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x890C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x891C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x892C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x893C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x894C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x895C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x896C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x897C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x898C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x899C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x89FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x8FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x8FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x900C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x901C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x902C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x903C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x904C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x905C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x906C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x907C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x908C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x909C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x90E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x90FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x910C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x911C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x912C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x913C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x914C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x915C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x916C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x917C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x918C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x919C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x91C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x91FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x920C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x921C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x922C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x923C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x924C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x925C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x926C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x927C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x928C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x929C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x92B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x92FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x930C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x931C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x932C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x933C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x934C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x935C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x936C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x937C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x938C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x939C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x93FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x940C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x941C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x942C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x943C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x944C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x945C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x946C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x947C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x948C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x949C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x94F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x94FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x950C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x951C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x952C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x953C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x954C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x955C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x956C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x957C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x958C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x959C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x95E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x95FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x960C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x961C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x962C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x963C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x964C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x965C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x966C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x967C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x968C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x969C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x96C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x96FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x970C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x971C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x972C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x973C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x974C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x975C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x976C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x977C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x978C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x979C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x97B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x97FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x980C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x981C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x982C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x983C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x984C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x985C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x986C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x987C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x988C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x989C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x98FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x990C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x991C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x992C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x993C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x994C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x995C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x996C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x997C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x998C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x999C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x99F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x99FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9A6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9A9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9AE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9AFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9B54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9B9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9BC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9BFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9C3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9C9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9CB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9CFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9D24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9D98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9D9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9DFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9E0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9E80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9E9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9ED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9EF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9EFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9F6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9F9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x9FE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0x9FFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA00C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA01C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA02C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA03C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA04C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA05C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA06C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA07C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA08C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA09C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA0C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA0FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xA9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xA9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xABB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xABFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xACFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAD08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAD9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xADF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xADFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xAFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xAFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB00C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB01C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB02C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB03C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB04C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB05C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB06C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB07C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB08C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB09C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB0AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB0FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xB9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xB9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBBFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBCF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBCFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBD64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBD9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBDD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBDFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xBFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xBFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC00C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC01C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC02C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC03C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC04C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC05C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC06C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC07C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC08C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC09C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC0FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xC9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xC9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCBF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCBFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCCD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCCFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCD48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCD9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCDFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xCF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xCFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD00C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD01C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD02C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD03C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD04C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD05C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD06C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD07C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD08C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD09C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD0EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD0FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xD98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xD9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDBD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDBFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDCBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDCFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDD30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDD9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDDA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDDFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xDFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xDFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE00C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE01C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE02C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE03C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE04C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE05C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE06C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE07C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE08C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE09C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE0D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE0FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xE9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xE9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEBBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEBFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xECA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xECFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xED14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xED88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xED9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEDFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xEFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xEFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF000	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF004	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF008	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF00C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF010	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF014	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF018	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF01C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF020	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF024	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF028	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF02C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF030	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF034	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF038	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF03C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF040	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF044	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF048	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF04C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF050	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF054	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF058	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF05C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF060	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF064	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF068	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF06C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF070	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF074	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF078	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF07C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF080	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF084	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF088	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF08C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF090	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF094	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF098	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF09C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF0B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF0FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF100	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF104	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF108	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF10C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF110	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF114	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF118	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF11C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF120	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF124	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF128	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF12C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF130	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF134	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF138	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF13C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF140	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF144	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF148	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF14C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF150	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF154	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF158	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF15C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF160	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF164	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF168	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF16C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF170	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF174	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF178	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF17C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF180	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF184	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF188	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF18C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF190	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF194	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF198	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF19C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF1A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF1FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF200	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF204	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF208	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF20C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF210	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF214	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF218	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF21C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF220	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF224	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF228	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF22C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF230	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF234	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF238	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF23C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF240	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF244	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF248	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF24C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF250	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF254	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF258	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF25C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF260	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF264	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF268	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF26C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF270	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF274	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF278	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF27C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF280	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF284	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF288	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF28C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF290	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF294	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF298	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF29C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF2FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF300	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF304	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF308	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF30C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF310	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF314	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF318	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF31C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF320	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF324	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF328	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF32C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF330	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF334	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF338	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF33C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF340	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF344	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF348	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF34C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF350	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF354	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF358	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF35C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF360	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF364	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF368	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF36C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF370	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF374	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF378	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF37C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF380	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF384	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF388	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF38C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF390	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF394	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF398	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF39C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF3E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF3FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF400	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF404	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF408	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF40C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF410	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF414	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF418	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF41C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF420	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF424	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF428	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF42C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF430	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF434	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF438	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF43C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF440	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF444	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF448	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF44C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF450	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF454	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF458	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF45C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF460	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF464	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF468	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF46C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF470	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF474	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF478	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF47C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF480	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF484	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF488	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF48C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF490	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF494	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF498	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF49C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF4CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF4FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF500	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF504	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF508	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF50C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF510	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF514	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF518	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF51C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF520	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF524	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF528	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF52C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF530	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF534	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF538	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF53C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF540	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF544	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF548	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF54C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF550	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF554	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF558	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF55C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF560	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF564	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF568	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF56C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF570	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF574	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF578	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF57C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF580	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF584	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF588	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF58C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF590	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF594	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF598	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF59C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF5B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF5FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF600	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF604	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF608	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF60C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF610	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF614	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF618	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF61C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF620	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF624	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF628	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF62C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF630	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF634	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF638	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF63C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF640	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF644	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF648	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF64C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF650	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF654	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF658	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF65C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF660	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF664	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF668	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF66C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF670	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF674	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF678	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF67C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF680	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF684	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF688	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF68C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF690	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF694	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF698	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF69C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF6A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF6FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF700	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF704	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF708	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF70C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF710	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF714	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF718	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF71C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF720	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF724	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF728	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF72C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF730	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF734	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF738	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF73C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF740	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF744	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF748	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF74C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF750	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF754	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF758	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF75C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF760	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF764	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF768	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF76C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF770	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF774	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF778	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF77C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF780	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF784	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF788	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF78C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF790	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF794	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF798	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF79C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF7FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF800	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF804	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF808	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF80C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF810	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF814	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF818	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF81C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF820	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF824	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF828	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF82C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF830	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF834	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF838	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF83C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF840	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF844	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF848	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF84C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF850	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF854	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF858	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF85C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF860	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF864	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF868	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF86C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF870	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF874	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF878	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF87C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF880	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF884	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF888	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF88C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF890	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF894	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF898	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF89C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF8E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF8FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF900	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF904	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF908	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF90C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF910	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF914	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF918	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF91C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF920	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF924	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF928	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF92C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF930	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF934	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF938	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF93C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF940	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF944	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF948	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF94C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF950	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF954	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF958	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF95C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF960	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF964	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF968	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF96C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF970	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF974	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF978	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF97C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF980	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF984	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF988	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF98C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF990	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF994	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF998	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF99C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9A0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9A4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9A8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9AC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9B0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9B4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9B8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9BC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9C0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9C4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9C8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xF9CC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9D0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9D4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9D8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9DC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9E0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9E4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9E8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9EC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9F0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9F4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9F8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xF9FC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFA40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFA9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFAB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFABC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFACC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFADC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFAFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFB2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFB9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFBA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFBFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFC10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFC84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFC9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFCF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFCFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFD70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFD9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFDE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFDFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFE58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFE9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFECC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFED0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFED4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFED8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFEFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF00	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF04	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF08	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF0C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF10	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF14	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF18	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF1C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF20	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF24	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF28	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF2C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF30	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF34	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF38	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF3C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFF40	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF44	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF48	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF4C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF50	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF54	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF58	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF5C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF60	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF64	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF68	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF6C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF70	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF74	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF78	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF7C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF80	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF84	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF88	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF8C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF90	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF94	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF98	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFF9C	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFA0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFA4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFA8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFAC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFB0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFB4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-96: ADSP-2184x LPDDRMEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0xFFB8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFBC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFC0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFC4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFC8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFCC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFD0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFD4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFD8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFDC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFE0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFE4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFE8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFEC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFF0	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFF4	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFF8	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000
0xFFFC	LPDDRMEM_ICCM_[n]	LPDDRMEM ICCM mem space	0x00000000

Table A-97: ADSP-2184x LPDDR MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3115C000	LPDDR_STAT_CTRL0	LPDDR Misc Status Control	0x00000000
0x3115C004	LPDDR_MRR_DATA	LPDDR Mode Register Read Data	0x00000000

Table A-98: ADSP-2184x MEC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A2000	MEC0_PEIRQ_GCTL[n]	MEC0 Parity Error Interrupt Request Global Control Register	0x00000000
0x310A2010	MEC0_PEIRQ_GSTAT[n]	MEC0 Parity Error Interrupt Request Global Status Register	0x00000000

Table A-98: ADSP-2184x MEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A2040	MEC0_PERR_CTL0	MEC0 Parity Error Control Register	0x00000000
0x310A2044	MEC0_PERR_CTL1	MEC0 Parity Error Control Register	0x00000000
0x310A2080	MEC0_PERR_STAT0	MEC0 Parity Error Status Register	0x00000000
0x310A2084	MEC0_PERR_STAT1	MEC0 Parity Error Status Register	0x00000000
0x310A20C0	MEC0_PERR_IMASK0	MEC0 Parity Error Interrupt Mask Register	0x00000000
0x310A20C4	MEC0_PERR_IMASK1	MEC0 Parity Error Interrupt Mask Register	0x00000000
0x310A2100	MEC0_EEIRQ_GCTL[n]	MEC0 ECC Error Interrupt Request Global Control Register	0x00000000
0x310A2110	MEC0_EEIRQ_GSTAT[n]	MEC0 ECC Error Interrupt Request Global Status Register	0x00000000
0x310A2140	MEC0_ECCERR_CTL[n]	MEC0 ECC Error Control Register	0x00000000
0x310A2180	MEC0_ECCERR_STAT[n]	MEC0 ECC Error Status Register	0x00000000
0x310A21C0	MEC0_ECCERR_IMASK[n]	MEC0 ECC Error Interrupt Mask Register	0x00000000
0x310A2300	MEC0_CMEIRQ_GCTL[n]	MEC0 Core Mem ECC Error Interrupt Request Global Control Register	0x00000000
0x310A2310	MEC0_CMEIRQ_GSTAT[n]	MEC0 Core Memory ECC Error Interrupt Request Global Status Register	0x00000000
0x310A2340	MEC0_CMERR_CTL[n]	MEC0 Core Memory ECC Error Control Register	0x00000000
0x310A2380	MEC0_CMERR_STAT[n]	MEC0 Core Memory ECC Error Status Register	0x00000000
0x310A23C0	MEC0_CMERR_IMASK[n]	MEC0 Core Memory ECC Error Interrupt Mask Register	0x00000000
0x310A2500	MEC0_A55C0EIRQ_GCTL[n]	MEC0 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A2510	MEC0_A55C0EIRQ_GSTAT[n]	MEC0 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A2540	MEC0_A55C0ERR_CTL[n]	MEC0 A55 Error Control Register	0x00000000
0x310A2580	MEC0_A55C0ERR_STAT[n]	MEC0 A55 Error Status Register	0x00000000
0x310A25C0	MEC0_A55C0ERR_IMASK[n]	MEC0 A55 Error Interrupt Mask Register	0x00000000
0x310A2700	MEC0_A55C2EIRQ_GCTL[n]	MEC0 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A2710	MEC0_A55C2EIRQ_GSTAT[n]	MEC0 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A2740	MEC0_A55C2ERR_CTL[n]	MEC0 A55 Error Control Register	0x00000000
0x310A2780	MEC0_A55C2ERR_STAT[n]	MEC0 A55 Error Status Register	0x00000000

Table A-98: ADSP-2184x MEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A27C0	MEC0_A55C2ERR_IMASK[n]	MEC0 A55 Error Interrupt Mask Register	0x00000000
0x310A2F00	MEC0_CLR	MEC0 Clear Register	0x00000000
0x310A2FD0	MEC0_PID4	MEC0 Peripheral ID4 Register	0x00000000
0x310A2FE0	MEC0_PID0	MEC0 Peripheral ID0 Register	0x00000003
0x310A2FE4	MEC0_PID1	MEC0 Peripheral ID1 Register	0x00000050
0x310A2FE8	MEC0_PID2	MEC0 Peripheral ID2 Register	0x0000000E
0x310A2FEC	MEC0_PID3	MEC0 Peripheral ID3 Register	0x00000000
0x310A2FF0	MEC0_CID0	MEC0 Component ID0 Register	0x0000000D
0x310A2FF4	MEC0_CID1	MEC0 Component ID1 Register	0x000000F0
0x310A2FF8	MEC0_CID2	MEC0 Component ID2 Register	0x00000005
0x310A2FFC	MEC0_CID3	MEC0 Component ID3 Register	0x000000B1

Table A-99: ADSP-2184x MEC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A3000	MEC1_PEIRQ_GCTL[n]	MEC1 Parity Error Interrupt Request Global Control Register	0x00000000
0x310A3010	MEC1_PEIRQ_GSTAT[n]	MEC1 Parity Error Interrupt Request Global Status Register	0x00000000
0x310A3040	MEC1_PERR_CTL0	MEC1 Parity Error Control Register	0x00000000
0x310A3044	MEC1_PERR_CTL1	MEC1 Parity Error Control Register	0x00000000
0x310A3080	MEC1_PERR_STAT0	MEC1 Parity Error Status Register	0x00000000
0x310A3084	MEC1_PERR_STAT1	MEC1 Parity Error Status Register	0x00000000
0x310A30C0	MEC1_PERR_IMASK0	MEC1 Parity Error Interrupt Mask Register	0x00000000
0x310A30C4	MEC1_PERR_IMASK1	MEC1 Parity Error Interrupt Mask Register	0x00000000
0x310A3100	MEC1_EEIRQ_GCTL[n]	MEC1 ECC Error Interrupt Request Global Control Register	0x00000000
0x310A3110	MEC1_EEIRQ_GSTAT[n]	MEC1 ECC Error Interrupt Request Global Status Register	0x00000000
0x310A3140	MEC1_ECCERR_CTL[n]	MEC1 ECC Error Control Register	0x00000000
0x310A3180	MEC1_ECCERR_STAT[n]	MEC1 ECC Error Status Register	0x00000000
0x310A31C0	MEC1_ECCERR_IMASK[n]	MEC1 ECC Error Interrupt Mask Register	0x00000000

Table A-99: ADSP-2184x MEC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A3300	MEC1_CMEIRQ_GCTL[n]	MEC1 Core Mem ECC Error Interrupt Request Global Control Register	0x00000000
0x310A3310	MEC1_CMEIRQ_GSTAT[n]	MEC1 Core Memory ECC Error Interrupt Request Global Status Register	0x00000000
0x310A3340	MEC1_CMERR_CTL[n]	MEC1 Core Memory ECC Error Control Register	0x00000000
0x310A3380	MEC1_CMERR_STAT[n]	MEC1 Core Memory ECC Error Status Register	0x00000000
0x310A33C0	MEC1_CMERR_IMASK[n]	MEC1 Core Memory ECC Error Interrupt Mask Register	0x00000000
0x310A3500	MEC1_A55C0EIRQ_GCTL[n]	MEC1 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A3510	MEC1_A55C0EIRQ_GSTAT[n]	MEC1 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A3540	MEC1_A55C0ERR_CTL[n]	MEC1 A55 Error Control Register	0x00000000
0x310A3580	MEC1_A55C0ERR_STAT[n]	MEC1 A55 Error Status Register	0x00000000
0x310A35C0	MEC1_A55C0ERR_IMASK[n]	MEC1 A55 Error Interrupt Mask Register	0x00000000
0x310A3700	MEC1_A55C2EIRQ_GCTL[n]	MEC1 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A3710	MEC1_A55C2EIRQ_GSTAT[n]	MEC1 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A3740	MEC1_A55C2ERR_CTL[n]	MEC1 A55 Error Control Register	0x00000000
0x310A3780	MEC1_A55C2ERR_STAT[n]	MEC1 A55 Error Status Register	0x00000000
0x310A37C0	MEC1_A55C2ERR_IMASK[n]	MEC1 A55 Error Interrupt Mask Register	0x00000000
0x310A3F00	MEC1_CLR	MEC1 Clear Register	0x00000000
0x310A3FD0	MEC1_PID4	MEC1 Peripheral ID4 Register	0x00000000
0x310A3FE0	MEC1_PID0	MEC1 Peripheral ID0 Register	0x00000003
0x310A3FE4	MEC1_PID1	MEC1 Peripheral ID1 Register	0x00000050
0x310A3FE8	MEC1_PID2	MEC1 Peripheral ID2 Register	0x0000000E
0x310A3FEC	MEC1_PID3	MEC1 Peripheral ID3 Register	0x00000000
0x310A3FF0	MEC1_CID0	MEC1 Component ID0 Register	0x0000000D
0x310A3FF4	MEC1_CID1	MEC1 Component ID1 Register	0x000000F0
0x310A3FF8	MEC1_CID2	MEC1 Component ID2 Register	0x00000005
0x310A3FFC	MEC1_CID3	MEC1 Component ID3 Register	0x000000B1

Table A-100: ADSP-2184x MEC2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A4000	MEC2_PEIRQ_GCTL[n]	MEC2 Parity Error Interrupt Request Global Control Register	0x00000000
0x310A4010	MEC2_PEIRQ_GSTAT[n]	MEC2 Parity Error Interrupt Request Global Status Register	0x00000000
0x310A4040	MEC2_PERR_CTL0	MEC2 Parity Error Control Register	0x00000000
0x310A4044	MEC2_PERR_CTL1	MEC2 Parity Error Control Register	0x00000000
0x310A4080	MEC2_PERR_STAT0	MEC2 Parity Error Status Register	0x00000000
0x310A4084	MEC2_PERR_STAT1	MEC2 Parity Error Status Register	0x00000000
0x310A40C0	MEC2_PERR_IMASK0	MEC2 Parity Error Interrupt Mask Register	0x00000000
0x310A40C4	MEC2_PERR_IMASK1	MEC2 Parity Error Interrupt Mask Register	0x00000000
0x310A4100	MEC2_EEIRQ_GCTL[n]	MEC2 ECC Error Interrupt Request Global Control Register	0x00000000
0x310A4110	MEC2_EEIRQ_GSTAT[n]	MEC2 ECC Error Interrupt Request Global Status Register	0x00000000
0x310A4140	MEC2_ECCERR_CTL[n]	MEC2 ECC Error Control Register	0x00000000
0x310A4180	MEC2_ECCERR_STAT[n]	MEC2 ECC Error Status Register	0x00000000
0x310A41C0	MEC2_ECCERR_IMASK[n]	MEC2 ECC Error Interrupt Mask Register	0x00000000
0x310A4300	MEC2_CMEIRQ_GCTL[n]	MEC2 Core Mem ECC Error Interrupt Request Global Control Register	0x00000000
0x310A4310	MEC2_CMEIRQ_GSTAT[n]	MEC2 Core Memory ECC Error Interrupt Request Global Status Register	0x00000000
0x310A4340	MEC2_CMERR_CTL[n]	MEC2 Core Memory ECC Error Control Register	0x00000000
0x310A4380	MEC2_CMERR_STAT[n]	MEC2 Core Memory ECC Error Status Register	0x00000000
0x310A43C0	MEC2_CMERR_IMASK[n]	MEC2 Core Memory ECC Error Interrupt Mask Register	0x00000000
0x310A4500	MEC2_A55C0EIRQ_GCTL[n]	MEC2 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A4510	MEC2_A55C0EIRQ_GSTAT[n]	MEC2 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A4540	MEC2_A55C0ERR_CTL[n]	MEC2 A55 Error Control Register	0x00000000
0x310A4580	MEC2_A55C0ERR_STAT[n]	MEC2 A55 Error Status Register	0x00000000
0x310A45C0	MEC2_A55C0ERR_IMASK[n]	MEC2 A55 Error Interrupt Mask Register	0x00000000
0x310A4700	MEC2_A55C2EIRQ_GCTL[n]	MEC2 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000

Table A-100: ADSP-2184x MEC2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A4710	MEC2_A55C2EIRQ_GSTAT[n]	MEC2 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A4740	MEC2_A55C2ERR_CTL[n]	MEC2 A55 Error Control Register	0x00000000
0x310A4780	MEC2_A55C2ERR_STAT[n]	MEC2 A55 Error Status Register	0x00000000
0x310A47C0	MEC2_A55C2ERR_IMASK[n]	MEC2 A55 Error Interrupt Mask Register	0x00000000
0x310A4F00	MEC2_CLR	MEC2 Clear Register	0x00000000
0x310A4FD0	MEC2_PID4	MEC2 Peripheral ID4 Register	0x00000000
0x310A4FE0	MEC2_PID0	MEC2 Peripheral ID0 Register	0x00000003
0x310A4FE4	MEC2_PID1	MEC2 Peripheral ID1 Register	0x00000050
0x310A4FE8	MEC2_PID2	MEC2 Peripheral ID2 Register	0x0000000E
0x310A4FEC	MEC2_PID3	MEC2 Peripheral ID3 Register	0x00000000
0x310A4FF0	MEC2_CID0	MEC2 Component ID0 Register	0x0000000D
0x310A4FF4	MEC2_CID1	MEC2 Component ID1 Register	0x000000F0
0x310A4FF8	MEC2_CID2	MEC2 Component ID2 Register	0x00000005
0x310A4FFC	MEC2_CID3	MEC2 Component ID3 Register	0x000000B1

Table A-101: ADSP-2184x MISCREG MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A9000	MISCREG_SH0_PFB_RANGE_SELECT	MISCREG SH0 Prefetch Range Selection Register	0xFFFFFFFF
0x310A9008	MISCREG_ECO_REG1	MISCREG LPDDR Miscellaneous Register	0x00000000
0x310A9010	MISCREG_CAN_SYSCTL	MISCREG CANFD Low Power Selection Mode	0x00000000
0x310A9014	MISCREG_SHARC_BRIDGE_REMAP	MISCREG SPORT Direct Interrupt Enable for SH0 and SH1	0x00000000
0x310A9028	MISCREG_ECO_REG9	MISCREG EMAC Error Register	0x00000000
0x310A903C	MISCREG_CLK_MUX_SEL	MISCREG Miscellaneous Clock Multiplexer Selection Register	0x00000000
0x310A9040	MISCREG_CLK_MUX_SEL1	MISCREG Miscellaneous Clock Multiplexer Selection 1 Register	0x00000000

Table A-101: ADSP-2184x MISCREG MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A9044	MISCREG_SH1_PFB_RANGE_SELECT	MISCREG SH1 Prefetch Range Selection Register	0xFFFFFFFF
0x310A9100	MISCREG_XSPI_CTL0	MISCREG XSPI Boot/Init Control Register 0	0x00000002
0x310A9104	MISCREG_XSPI_CTL1	MISCREG XSPI Boot/Init Control Register 1	0x00000000
0x310A9108	MISCREG_XSPI_CTL2	MISCREG XSPI Boot/Init Control Register 2	0x80000101
0x310A910C	MISCREG_XSPI_CTL3	MISCREG XSPI Boot/Init Control Register 3	0x00300404
0x310A9110	MISCREG_XSPI_CTL4	MISCREG XSPI Boot/Init Control Register 4	0x00180030
0x310A9114	MISCREG_XSPI_CTL5	MISCREG XSPI Boot/Init Control Register 5	0x00800004
0x310A9118	MISCREG_XSPI_CTL6	MISCREG XSPI Boot/Init Control Register 6	0x0000357F
0x310A911C	MISCREG_XSPI_CTL7	MISCREG XSPI Boot/Init Control Register 7	0x00000000
0x310A9120	MISCREG_XSPI_CTL8	MISCREG XSPI Boot/Init Control Register 8	0x00000080
0x310A9130	MISCREG_XSPI_RSTCTL	MISCREG XSPI Boot/Init Reset Control Register	0x00000000
0x310A9140	MISCREG_XSPI_STAT0	MISCREG XSPI Boot/Init Status Register	0x00000000
0x310A9208	MISCREG_EMAC_MISC_REG	MISCREG EMAC Miscellaneous Register	0x00000000
0x310A920C	MISCREG_MINI_CGU0	MISCREG MINI CGU0 MISC Register	0x00000000
0x310A9210	MISCREG_MINI_CGU1	MISCREG MINI CGU0 MISC Register	0x00000000
0x310A9214	MISCREG_EMMC_MISC_REG	MISCREG EMMC MISC Register	0x00000000
0x310A9218	MISCREG_EMAC_MISC_REG2	MISCREG EMAC Miscellaneous Register 2	0x00000000
0x310A9300	MISCREG_XSPI1_CTL0	MISCREG XSPI1 Boot/Init Control Register 0	0x00000002
0x310A9304	MISCREG_XSPI1_CTL1	MISCREG XSPI1 Boot/Init Control Register 1	0x00000000
0x310A9308	MISCREG_XSPI1_CTL2	MISCREG XSPI1 Boot/Init Control Register 2	0x80000101
0x310A930C	MISCREG_XSPI1_CTL3	MISCREG XSPI1 Boot/Init Control Register 3	0x00300404
0x310A9310	MISCREG_XSPI1_CTL4	MISCREG XSPI1 Boot/Init Control Register 4	0x00180030
0x310A9314	MISCREG_XSPI1_CTL5	MISCREG XSPI1 Boot/Init Control Register 5	0x00800004
0x310A9318	MISCREG_XSPI1_CTL6	MISCREG XSPI1 Boot/Init Control Register 6	0x0000357F
0x310A931C	MISCREG_XSPI1_CTL7	MISCREG XSPI1 Boot/Init Control Register 7	0x00000000
0x310A9320	MISCREG_XSPI1_CTL8	MISCREG XSPI1 Boot/Init Control Register 8	0x00000080
0x310A9330	MISCREG_XSPI1_RSTCTL	MISCREG XSPI1 Boot/Init Reset Control Register	0x00000000

Table A-101: ADSP-2184x MISCREG MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A9340	MISCREG_XSPI1_STAT0	MISCREG XSPI1 Boot/Init Status Register	0x00000000

Table A-102: ADSP-2184x MLB0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109D000	MLB0_CTL0	MLB0 MediaLB Control 0 Register	0x00000000
0x3109D00C	MLB0_MS0	MLB0 Channel Status 0 Register	0x00000000
0x3109D014	MLB0_MS1	MLB0 Channel Status 1 Register	0x00000000
0x3109D020	MLB0_MSS	MLB0 System Status Register	0x00000000
0x3109D024	MLB0_MSD	MLB0 System Data Register	0x00000000
0x3109D02C	MLB0_MIEN	MLB0 Interrupt Enable Register	0x00000000
0x3109D034	MLB0_GCTL	MLB0 MLB Global Control Register	0x00000000
0x3109D03C	MLB0_CTL1	MLB0 Control 1 Register	0x00000000
0x3109D080	MLB0_HCTL	MLB0 HBI Control Register	0x00000000
0x3109D088	MLB0_HCMR0	MLB0 HBI Channel Mask 0 Register	0x00000000
0x3109D08C	MLB0_HCMR1	MLB0 HBI Channel Mask 1 Register	0x00000000
0x3109D090	MLB0_HCER0	MLB0 HBI Channel Error 0 Register	0x00000000
0x3109D094	MLB0_HCER1	MLB0 HBI Channel Error 1 Register	0x00000000
0x3109D098	MLB0_HCBR0	MLB0 HBI Channel Busy 0 Register	0x00000000
0x3109D09C	MLB0_HCBR1	MLB0 HBI Channel Busy 1 Register	0x00000000
0x3109D0C0	MLB0_MDAT0	MLB0 Memory Interface Control Data 0 Register	0x00000000
0x3109D0C4	MLB0_MDAT1	MLB0 Memory Interface Control Data 1 Register	0x00000000
0x3109D0C8	MLB0_MDAT2	MLB0 Memory Interface Control Data 2 Register	0x00000000
0x3109D0CC	MLB0_MDAT3	MLB0 Memory Interface Control Data 3 Register	0x00000000
0x3109D0D0	MLB0_MDWE0	MLB0 Memory Interface Control Data Write Enable 0 Register	0x00000000
0x3109D0D4	MLB0_MDWE1	MLB0 Memory Interface Control Data Write Enable 1 Register	0x00000000
0x3109D0D8	MLB0_MDWE2	MLB0 Memory Interface Control Data Write Enable 2 Register	0x00000000
0x3109D0DC	MLB0_MDWE3	MLB0 Memory Interface Control Data Write Enable 3 Register	0x00000000
0x3109D0E0	MLB0_MCTL	MLB0 Memory Interface Control Register	0x00000000

Table A-102: ADSP-2184x MLB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109D0E4	MLB0_MADR	MLB0 Memory Interface Address Register	0x00000000
0x3109D3C0	MLB0_ACTL	MLB0 Bus Control Register	0x00000000
0x3109D3D0	MLB0_ACSR0	MLB0 Peripheral Channel Status 0 Register	0x00000000
0x3109D3D4	MLB0_ACSR1	MLB0 Peripheral Channel Status 1 Register	0x00000000
0x3109D3D8	MLB0_ACMR0	MLB0 Peripheral Channel Mask 0 Register	0x00000000
0x3109D3DC	MLB0_ACMR1	MLB0 Peripheral Channel Mask 1 Register	0x00000000

Table A-103: ADSP-2184x SCB1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30200020	SCB1_MST_IB_SYNC_MODE	SCB1	0x00000004

Table A-104: ADSP-2184x A55SCB MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30501FD0	A55SCB_PERIPH_ID_4	A55SCB	0x00000004
0x30501FD4	A55SCB_PERIPH_ID_5	A55SCB	0x00000000
0x30501FD8	A55SCB_PERIPH_ID_6	A55SCB	0x00000000
0x30501FDC	A55SCB_PERIPH_ID_7	A55SCB	0x00000000
0x30501FE0	A55SCB_PERIPH_ID_0	A55SCB	0x00000000
0x30501FE4	A55SCB_PERIPH_ID_1	A55SCB	0x000000B4
0x30501FE8	A55SCB_PERIPH_ID_2	A55SCB	0x0000006B
0x30501FEC	A55SCB_PERIPH_ID_3	A55SCB	0x00000000
0x30501FF0	A55SCB_COMP_ID_0	A55SCB	0x0000000D
0x30501FF4	A55SCB_COMP_ID_1	A55SCB	0x000000F0
0x30501FF8	A55SCB_COMP_ID_2	A55SCB	0x00000005
0x30501FFC	A55SCB_COMP_ID_3	A55SCB	0x000000B1
0x30502008	A55SCB_S_AHB_S_FN_MOD_ISS_BM	A55SCB	0x00000000
0x30502044	A55SCB_S_AHB_S_AHB_CNTL	A55SCB	0x00000000

Table A-104: ADSP-2184x A55SCB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30505008	A55SCB_M_AXI_S_IB_FN_MOD_ISS_BM	A55SCB	0x00000000
0x30505020	A55SCB_M_AXI_S_IB_SYNC_MODE	A55SCB	0x00000004
0x30505024	A55SCB_M_AXI_S_IB_FN_MOD2	A55SCB	0x00000000
0x30505108	A55SCB_M_AXI_S_IB_FN_MOD	A55SCB	0x00000000
0x30542024	A55SCB_M_AXI_M_IB_FN_MOD2	A55SCB	0x00000000
0x30542108	A55SCB_M_AXI_M_IB_FN_MOD	A55SCB	0x00000000
0x30543020	A55SCB_S_AXI_S_IB_SYNC_MODE	A55SCB	0x00000004
0x30543024	A55SCB_S_AXI_S_IB_FN_MOD2	A55SCB	0x00000000
0x30543108	A55SCB_S_AXI_S_IB_FN_MOD	A55SCB	0x00000000

Table A-105: ADSP-2184x SCB3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30101FD0	SCB3_PERIPH_ID_4	SCB3	0x00000004
0x30101FD4	SCB3_PERIPH_ID_5	SCB3	0x00000000
0x30101FD8	SCB3_PERIPH_ID_6	SCB3	0x00000000
0x30101FDC	SCB3_PERIPH_ID_7	SCB3	0x00000000
0x30101FE0	SCB3_PERIPH_ID_0	SCB3	0x00000000
0x30101FE4	SCB3_PERIPH_ID_1	SCB3	0x000000B4
0x30101FE8	SCB3_PERIPH_ID_2	SCB3	0x0000006B
0x30101FEC	SCB3_PERIPH_ID_3	SCB3	0x00000000
0x30101FF0	SCB3_COMP_ID_0	SCB3	0x0000000D
0x30101FF4	SCB3_COMP_ID_1	SCB3	0x000000F0
0x30101FF8	SCB3_COMP_ID_2	SCB3	0x00000005
0x30101FFC	SCB3_COMP_ID_3	SCB3	0x000000B1

Table A-105: ADSP-2184x SCB3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30102008	SCB3_APB_CLKO8_IB_FN_MOD_ISS_BM	SCB3	0x00000000
0x30102020	SCB3_APB_CLKO8_IB_SYNC_MODE	SCB3	0x00000004
0x30103008	SCB3_APB_SCLK0_1_IB_FN_MOD_ISS_BM	SCB3	0x00000000
0x30104008	SCB3_APB_CLKO6_IB_FN_MOD_ISS_BM	SCB3	0x00000000
0x30105008	SCB3_APB_CLKO3_IB_FN_MOD_ISS_BM	SCB3	0x00000000
0x30105020	SCB3_APB_CLKO3_IB_SYNC_MODE	SCB3	0x00000004
0x30108008	SCB3_DAI1_FN_MOD_ISS_BM	SCB3	0x00000000
0x30108044	SCB3_DAI1_AHB_CNTL	SCB3	0x00000000
0x30109008	SCB3_SPIF_FABRIC_FN_MOD_ISS_BM	SCB3	0x00000000
0x3010A008	SCB3_DAI0_FN_MOD_ISS_BM	SCB3	0x00000000
0x3010A044	SCB3_DAI0_AHB_CNTL	SCB3	0x00000000
0x3010B008	SCB3_APB_CLKPWM_IB_FN_MOD_ISS_BM	SCB3	0x00000000
0x3010B020	SCB3_APB_CLKPWM_IB_SYNC_MODE	SCB3	0x00000004
0x3010C008	SCB3_CRYPTO_0_FN_MOD_ISS_BM	SCB3	0x00000000
0x3010C044	SCB3_CRYPTO_0_AHB_CNTL	SCB3	0x00000000
0x3010E008	SCB3_CRYPTO_1_FN_MOD_ISS_BM	SCB3	0x00000000
0x3010E044	SCB3_CRYPTO_1_AHB_CNTL	SCB3	0x00000000
0x3010F008	SCB3_APB_SCLK0_0_IB_FN_MOD_ISS_BM	SCB3	0x00000000
0x30110008	SCB3_APB_CLKO4_IB_FN_MOD_ISS_BM	SCB3	0x00000000

Table A-105: ADSP-2184x SCB3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30110020	SCB3_APB_CLKO4_IB_SYNC_MODE	SCB3	0x00000004
0x3012C008	SCB3_DLMDMA0_FN_MOD_ISS_BM	SCB3	0x00000000
0x3012C044	SCB3_DLMDMA0_AHB_CNTL	SCB3	0x00000000
0x30142108	SCB3_SMMR_FN_MOD	SCB3	0x00000000

Table A-106: ADSP-2184x SCB4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30300000	SCB4_REMAP	SCB4	0x00000000
0x30301FD0	SCB4_PERIPH_ID_4	SCB4	0x00000004
0x30301FD4	SCB4_PERIPH_ID_5	SCB4	0x00000000
0x30301FD8	SCB4_PERIPH_ID_6	SCB4	0x00000000
0x30301FDC	SCB4_PERIPH_ID_7	SCB4	0x00000000
0x30301FE0	SCB4_PERIPH_ID_0	SCB4	0x00000000
0x30301FE4	SCB4_PERIPH_ID_1	SCB4	0x000000B4
0x30301FE8	SCB4_PERIPH_ID_2	SCB4	0x0000006B
0x30301FEC	SCB4_PERIPH_ID_3	SCB4	0x00000000
0x30301FF0	SCB4_COMP_ID_0	SCB4	0x0000000D
0x30301FF4	SCB4_COMP_ID_1	SCB4	0x000000F0
0x30301FF8	SCB4_COMP_ID_2	SCB4	0x00000005
0x30301FFC	SCB4_COMP_ID_3	SCB4	0x000000B1
0x30302008	SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD_ISS_BM	SCB4	0x00000000
0x30302020	SCB4_SHARC_DPORT_TO_SF_IB_SYNC_MODE	SCB4	0x00000004
0x30302108	SCB4_SHARC_DPORT_TO_SF_IB_FN_MOD	SCB4	0x00000000
0x30304008	SCB4_FABRIC_MMR_IB_FN_MOD_ISS_BM	SCB4	0x00000000

Table A-106: ADSP-2184x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30304020	SCB4_FABRIC_MMR_IB_SYNC_MODE	SCB4	0x00000004
0x30304024	SCB4_FABRIC_MMR_IB_FN_MOD2	SCB4	0x00000000
0x30304108	SCB4_FABRIC_MMR_IB_FN_MOD	SCB4	0x00000000
0x30306008	SCB4_FABRIC_FIR_CH0_IB_FN_MOD_ISS_BM	SCB4	0x00000000
0x30306020	SCB4_FABRIC_FIR_CH0_IB_SYNC_MODE	SCB4	0x00000004
0x30306108	SCB4_FABRIC_FIR_CH0_IB_FN_MOD	SCB4	0x00000000
0x30307008	SCB4_FABRIC_FIR_CH1_IB_FN_MOD_ISS_BM	SCB4	0x00000000
0x30307020	SCB4_FABRIC_FIR_CH1_IB_SYNC_MODE	SCB4	0x00000004
0x30307108	SCB4_FABRIC_FIR_CH1_IB_FN_MOD	SCB4	0x00000000
0x3030D008	SCB4_FABRIC_IIR_CH0_IB_FN_MOD_ISS_BM	SCB4	0x00000000
0x3030D020	SCB4_FABRIC_IIR_CH0_IB_SYNC_MODE	SCB4	0x00000004
0x3030D108	SCB4_FABRIC_IIR_CH0_IB_FN_MOD	SCB4	0x00000000
0x3030E008	SCB4_FABRIC_IIR_CH1_IB_FN_MOD_ISS_BM	SCB4	0x00000000

Table A-106: ADSP-2184x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3030E020	SCB4_FABRIC_IIR_CH1_IB_SYNC_MODE	SCB4	0x00000004
0x3030E108	SCB4_FABRIC_IIR_CH1_IB_FN_MOD	SCB4	0x00000000
0x3030F020	SCB4_SHARC_IPORT_IB_SYNC_MODE	SCB4	0x00000004
0x3030F024	SCB4_SHARC_IPORT_IB_FN_MOD2	SCB4	0x00000000
0x30310008	SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD_ISS_BM	SCB4	0x00000000
0x30310020	SCB4_SHARC_DPORT_TO_SF_L2_IB_SYNC_MODE	SCB4	0x00000004
0x30310108	SCB4_SHARC_DPORT_TO_SF_L2_IB_FN_MOD	SCB4	0x00000000
0x30311020	SCB4_IDMA_IB_SYNC_MODE	SCB4	0x00000004
0x30311024	SCB4_IDMA_IB_FN_MOD2	SCB4	0x00000000
0x30342024	SCB4_IIR2_CH0_IB_FN_MOD2	SCB4	0x00000000
0x30342100	SCB4_IIR2_CH0_IB_READ_QOS	SCB4	0x0000000C
0x30342104	SCB4_IIR2_CH0_IB_WRITE_QOS	SCB4	0x0000000C
0x30342108	SCB4_IIR2_CH0_IB_FN_MOD	SCB4	0x00000000
0x30343100	SCB4_SHARC_DPORT_READ_QOS	SCB4	0x0000000C
0x30343104	SCB4_SHARC_DPORT_WRITE_QOS	SCB4	0x0000000C
0x30343108	SCB4_SHARC_DPORT_FN_MOD	SCB4	0x00000000
0x30344020	SCB4_FABRIC_S2PORT_IB_SYNC_MODE	SCB4	0x00000004

Table A-106: ADSP-2184x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30344100	SCB4_FABRIC_S2PORT_IB_READ_QOS	SCB4	0x0000000C
0x30344104	SCB4_FABRIC_S2PORT_IB_WRITE_QOS	SCB4	0x0000000C
0x30344108	SCB4_FABRIC_S2PORT_IB_FN_MOD	SCB4	0x00000000
0x30345100	SCB4_FIR_CH0_READ_QOS	SCB4	0x0000000C
0x30345104	SCB4_FIR_CH0_WRITE_QOS	SCB4	0x0000000C
0x30345108	SCB4_FIR_CH0_FN_MOD	SCB4	0x00000000
0x30346100	SCB4_FIR_CH1_READ_QOS	SCB4	0x0000000C
0x30346104	SCB4_FIR_CH1_WRITE_QOS	SCB4	0x0000000C
0x30346108	SCB4_FIR_CH1_FN_MOD	SCB4	0x00000000
0x30347024	SCB4_IIR0_CH0_IB_FN_MOD2	SCB4	0x00000000
0x30347100	SCB4_IIR0_CH0_IB_READ_QOS	SCB4	0x0000000C
0x30347104	SCB4_IIR0_CH0_IB_WRITE_QOS	SCB4	0x0000000C
0x30347108	SCB4_IIR0_CH0_IB_FN_MOD	SCB4	0x00000000
0x30348024	SCB4_IIR0_CH1_IB_FN_MOD2	SCB4	0x00000000
0x30348100	SCB4_IIR0_CH1_IB_READ_QOS	SCB4	0x0000000C
0x30348104	SCB4_IIR0_CH1_IB_WRITE_QOS	SCB4	0x0000000C
0x30348108	SCB4_IIR0_CH1_IB_FN_MOD	SCB4	0x00000000
0x30349024	SCB4_IIR2_CH1_IB_FN_MOD2	SCB4	0x00000000

Table A-106: ADSP-2184x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30349100	SCB4_IIR2_CH1_IB_READ_QOS	SCB4	0x0000000C
0x30349104	SCB4_IIR2_CH1_IB_WRITE_QOS	SCB4	0x0000000C
0x30349108	SCB4_IIR2_CH1_IB_FN_MOD	SCB4	0x00000000
0x3034A020	SCB4_FABRIC_S1PORT_IB_SYNC_MODE	SCB4	0x00000004
0x3034A100	SCB4_FABRIC_S1PORT_IB_READ_QOS	SCB4	0x0000000C
0x3034A104	SCB4_FABRIC_S1PORT_IB_WRITE_QOS	SCB4	0x0000000C
0x3034A108	SCB4_FABRIC_S1PORT_IB_FN_MOD	SCB4	0x00000000
0x3034B024	SCB4_IIR3_CH0_IB_FN_MOD2	SCB4	0x00000000
0x3034B100	SCB4_IIR3_CH0_IB_READ_QOS	SCB4	0x0000000C
0x3034B104	SCB4_IIR3_CH0_IB_WRITE_QOS	SCB4	0x0000000C
0x3034B108	SCB4_IIR3_CH0_IB_FN_MOD	SCB4	0x00000000
0x3034C024	SCB4_IIR3_CH1_IB_FN_MOD2	SCB4	0x00000000
0x3034C100	SCB4_IIR3_CH1_IB_READ_QOS	SCB4	0x0000000C
0x3034C104	SCB4_IIR3_CH1_IB_WRITE_QOS	SCB4	0x0000000C
0x3034C108	SCB4_IIR3_CH1_IB_FN_MOD	SCB4	0x00000000
0x3034E024	SCB4_IIR1_CH0_IB_FN_MOD2	SCB4	0x00000000
0x3034E100	SCB4_IIR1_CH0_IB_READ_QOS	SCB4	0x0000000C

Table A-106: ADSP-2184x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3034E104	SCB4_IIR1_CH0_IB_WRITE_QOS	SCB4	0x0000000C
0x3034E108	SCB4_IIR1_CH0_IB_FN_MOD	SCB4	0x00000000
0x3034F024	SCB4_IIR1_CH1_IB_FN_MOD2	SCB4	0x00000000
0x3034F100	SCB4_IIR1_CH1_IB_READ_QOS	SCB4	0x0000000C
0x3034F104	SCB4_IIR1_CH1_IB_WRITE_QOS	SCB4	0x0000000C
0x3034F108	SCB4_IIR1_CH1_IB_FN_MOD	SCB4	0x00000000
0x30350100	SCB4_FIR1_CH1_READ_QOS	SCB4	0x0000000C
0x30350104	SCB4_FIR1_CH1_WRITE_QOS	SCB4	0x0000000C
0x30350108	SCB4_FIR1_CH1_FN_MOD	SCB4	0x00000000
0x30351100	SCB4_FIR1_CH0_READ_QOS	SCB4	0x0000000C
0x30351104	SCB4_FIR1_CH0_WRITE_QOS	SCB4	0x0000000C
0x30351108	SCB4_FIR1_CH0_FN_MOD	SCB4	0x00000000

Table A-107: ADSP-2184x INST1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30400000	INST1_REMAP	INST1	0x00000000
0x30401FD0	INST1_PERIPH_ID_4	INST1	0x00000004
0x30401FD4	INST1_PERIPH_ID_5	INST1	0x00000000
0x30401FD8	INST1_PERIPH_ID_6	INST1	0x00000000
0x30401FDC	INST1_PERIPH_ID_7	INST1	0x00000000
0x30401FE0	INST1_PERIPH_ID_0	INST1	0x00000000
0x30401FE4	INST1_PERIPH_ID_1	INST1	0x000000B4

Table A-107: ADSP-2184x INST1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30401FE8	INST1_PERIPH_ID_2	INST1	0x0000006B
0x30401FEC	INST1_PERIPH_ID_3	INST1	0x00000000
0x30401FF0	INST1_COMP_ID_0	INST1	0x0000000D
0x30401FF4	INST1_COMP_ID_1	INST1	0x000000F0
0x30401FF8	INST1_COMP_ID_2	INST1	0x00000005
0x30401FFC	INST1_COMP_ID_3	INST1	0x000000B1
0x30403008	INST1_SPI2_S_IB_FN_MOD_ISS_BM	INST1	0x00000000
0x30403024	INST1_SPI2_S_IB_FN_MOD2	INST1	0x00000000
0x30403108	INST1_SPI2_S_IB_FN_MOD	INST1	0x00000000
0x30404008	INST1_OSPI_S_IB_FN_MOD_ISS_BM	INST1	0x00000000
0x30404024	INST1_OSPI_S_IB_FN_MOD2	INST1	0x00000000
0x3040402C	INST1_OSPI_S_IB_FN_MOD_LB	INST1	0x00000000
0x30404108	INST1_OSPI_S_IB_FN_MOD	INST1	0x00000000
0x30442108	INST1_SPIF_M_FN_MOD	INST1	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30001FD0	SCB0_PERIPH_ID_4	SCB0	0x00000004
0x30001FD4	SCB0_PERIPH_ID_5	SCB0	0x00000000
0x30001FD8	SCB0_PERIPH_ID_6	SCB0	0x00000000
0x30001FDC	SCB0_PERIPH_ID_7	SCB0	0x00000000
0x30001FE0	SCB0_PERIPH_ID_0	SCB0	0x00000000
0x30001FE4	SCB0_PERIPH_ID_1	SCB0	0x000000B4
0x30001FE8	SCB0_PERIPH_ID_2	SCB0	0x0000006B
0x30001FEC	SCB0_PERIPH_ID_3	SCB0	0x00000000
0x30001FF0	SCB0_COMP_ID_0	SCB0	0x0000000D

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30001FF4	SCB0_COMP_ID_1	SCB0	0x000000F0
0x30001FF8	SCB0_COMP_ID_2	SCB0	0x00000005
0x30001FFC	SCB0_COMP_ID_3	SCB0	0x000000B1
0x30002008	SCB0_SPIF_FN_MOD_ISS_BM	SCB0	0x00000000
0x30003008	SCB0_DL2_0_FN_MOD_ISS_BM	SCB0	0x00000000
0x30004008	SCB0_OTP_IB_FN_MOD_ISS_BM	SCB0	0x00000000
0x30004108	SCB0_OTP_IB_FN_MOD	SCB0	0x00000000
0x30005008	SCB0_SMMR_FN_MOD_ISS_BM	SCB0	0x00000000
0x30006008	SCB0_CL2_1_FN_MOD_ISS_BM	SCB0	0x00000000
0x30008008	SCB0_SH0_S1PORT_FN_MOD_ISS_BM	SCB0	0x00000000
0x30009008	SCB0_SH0_S2PORT_FN_MOD_ISS_BM	SCB0	0x00000000
0x3000A008	SCB0_M85_S_AXI_IB_FN_MOD_ISS_BM	SCB0	0x00000000
0x3000A108	SCB0_M85_S_AXI_IB_FN_MOD	SCB0	0x00000000
0x3000B008	SCB0_M85_S2_AXI_IB_FN_MOD_ISS_BM	SCB0	0x00000000
0x3000B108	SCB0_M85_S2_AXI_IB_FN_MOD	SCB0	0x00000000
0x3000D008	SCB0_SMC_IB_FN_MOD_ISS_BM	SCB0	0x00000000
0x3000D108	SCB0_SMC_IB_FN_MOD	SCB0	0x00000000
0x3000E008	SCB0_CL2_2_FN_MOD_ISS_BM	SCB0	0x00000000
0x3000F008	SCB0_DL2_1_FN_MOD_ISS_BM	SCB0	0x00000000
0x30010008	SCB0_DMC0_FN_MOD_ISS_BM	SCB0	0x00000000
0x30042100	SCB0_SP0A_READ_QOS	SCB0	0x0000000C

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30042104	SCB0_SP0A_WRITE_QOS	SCB0	0x0000000C
0x30042108	SCB0_SP0A_FN_MOD	SCB0	0x00000000
0x30043100	SCB0_SP0B_READ_QOS	SCB0	0x0000000C
0x30043104	SCB0_SP0B_WRITE_QOS	SCB0	0x0000000C
0x30043108	SCB0_SP0B_FN_MOD	SCB0	0x00000000
0x30044100	SCB0_SP1A_READ_QOS	SCB0	0x0000000C
0x30044104	SCB0_SP1A_WRITE_QOS	SCB0	0x0000000C
0x30044108	SCB0_SP1A_FN_MOD	SCB0	0x00000000
0x30045100	SCB0_SP1B_READ_QOS	SCB0	0x0000000C
0x30045104	SCB0_SP1B_WRITE_QOS	SCB0	0x0000000C
0x30045108	SCB0_SP1B_FN_MOD	SCB0	0x00000000
0x30046100	SCB0_SP2A_READ_QOS	SCB0	0x0000000C
0x30046104	SCB0_SP2A_WRITE_QOS	SCB0	0x0000000C
0x30046108	SCB0_SP2A_FN_MOD	SCB0	0x00000000
0x30047100	SCB0_SP2B_READ_QOS	SCB0	0x0000000C
0x30047104	SCB0_SP2B_WRITE_QOS	SCB0	0x0000000C
0x30047108	SCB0_SP2B_FN_MOD	SCB0	0x00000000
0x30048100	SCB0_SP3A_READ_QOS	SCB0	0x0000000C
0x30048104	SCB0_SP3A_WRITE_QOS	SCB0	0x0000000C
0x30048108	SCB0_SP3A_FN_MOD	SCB0	0x00000000
0x30049100	SCB0_SP3B_READ_QOS	SCB0	0x0000000C
0x30049104	SCB0_SP3B_WRITE_QOS	SCB0	0x0000000C
0x30049108	SCB0_SP3B_FN_MOD	SCB0	0x00000000
0x3004A024	SCB0_CRC0_CH0_IB_FN_MOD2	SCB0	0x00000000
0x3004A100	SCB0_CRC0_CH0_IB_READ_QOS	SCB0	0x00000001
0x3004A104	SCB0_CRC0_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x3004A108	SCB0_CRC0_CH0_IB_FN_MOD	SCB0	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3004B024	SCB0_CRC0_CH1_IB_FN_MOD2	SCB0	0x00000000
0x3004B100	SCB0_CRC0_CH1_IB_READ_QOS	SCB0	0x00000001
0x3004B104	SCB0_CRC0_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x3004B108	SCB0_CRC0_CH1_IB_FN_MOD	SCB0	0x00000000
0x3004C024	SCB0_CRC1_CH0_IB_FN_MOD2	SCB0	0x00000000
0x3004C100	SCB0_CRC1_CH0_IB_READ_QOS	SCB0	0x00000001
0x3004C104	SCB0_CRC1_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x3004C108	SCB0_CRC1_CH0_IB_FN_MOD	SCB0	0x00000000
0x3004D024	SCB0_CRC1_CH1_IB_FN_MOD2	SCB0	0x00000000
0x3004D100	SCB0_CRC1_CH1_IB_READ_QOS	SCB0	0x00000001
0x3004D104	SCB0_CRC1_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x3004D108	SCB0_CRC1_CH1_IB_FN_MOD	SCB0	0x00000000
0x3004E100	SCB0_UART2_RX_READ_QOS	SCB0	0x0000000C
0x3004E104	SCB0_UART2_RX_WRITE_QOS	SCB0	0x0000000C
0x3004E108	SCB0_UART2_RX_FN_MOD	SCB0	0x00000000
0x3004F100	SCB0_SH0_DPORT_READ_QOS	SCB0	0x00000007
0x3004F104	SCB0_SH0_DPORT_WRITE_QOS	SCB0	0x00000007
0x3004F108	SCB0_SH0_DPORT_FN_MOD	SCB0	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30050100	SCB0_SH0_IPORT_READ_QOS	SCB0	0x00000007
0x30050104	SCB0_SH0_IPORT_WRITE_QOS	SCB0	0x00000007
0x30050108	SCB0_SH0_IPORT_FN_MOD	SCB0	0x00000000
0x30051100	SCB0_SP4A_READ_QOS	SCB0	0x0000000C
0x30051104	SCB0_SP4A_WRITE_QOS	SCB0	0x0000000C
0x30051108	SCB0_SP4A_FN_MOD	SCB0	0x00000000
0x30052100	SCB0_SP4B_READ_QOS	SCB0	0x0000000C
0x30052104	SCB0_SP4B_WRITE_QOS	SCB0	0x0000000C
0x30052108	SCB0_SP4B_FN_MOD	SCB0	0x00000000
0x30053024	SCB0_XSPI_M_IB_FN_MOD2	SCB0	0x00000000
0x30053108	SCB0_XSPI_M_IB_FN_MOD	SCB0	0x00000000
0x30054024	SCB0_GIGE_IB_FN_MOD2	SCB0	0x00000000
0x30054100	SCB0_GIGE_IB_READ_QOS	SCB0	0x00000007
0x30054104	SCB0_GIGE_IB_WRITE_QOS	SCB0	0x00000007
0x30054108	SCB0_GIGE_IB_FN_MOD	SCB0	0x00000000
0x30055024	SCB0_HSMDMA_CH0_IB_FN_MOD2	SCB0	0x00000000
0x30055100	SCB0_HSMDMA_CH0_IB_READ_QOS	SCB0	0x00000001
0x30055104	SCB0_HSMDMA_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x30055108	SCB0_HSMDMA_CH0_IB_FN_MOD	SCB0	0x00000000
0x30056100	SCB0_SP5A_READ_QOS	SCB0	0x0000000C
0x30056104	SCB0_SP5A_WRITE_QOS	SCB0	0x0000000C
0x30056108	SCB0_SP5A_FN_MOD	SCB0	0x00000000
0x30057100	SCB0_MLB_READ_QOS	SCB0	0x0000000C

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30057104	SCB0_MLB_WRITE_QOS	SCB0	0x0000000C
0x30057108	SCB0_MLB_FN_MOD	SCB0	0x00000000
0x30058100	SCB0_UART0_TX_READ_QOS	SCB0	0x0000000C
0x30058104	SCB0_UART0_TX_WRITE_QOS	SCB0	0x0000000C
0x30058108	SCB0_UART0_TX_FN_MOD	SCB0	0x00000000
0x30059100	SCB0_UART0_RX_READ_QOS	SCB0	0x0000000C
0x30059104	SCB0_UART0_RX_WRITE_QOS	SCB0	0x0000000C
0x30059108	SCB0_UART0_RX_FN_MOD	SCB0	0x00000000
0x3005A100	SCB0_SP5B_READ_QOS	SCB0	0x0000000C
0x3005A104	SCB0_SP5B_WRITE_QOS	SCB0	0x0000000C
0x3005A108	SCB0_SP5B_FN_MOD	SCB0	0x00000000
0x3005B024	SCB0_HSMDMA_CH1_IB_FN_MOD2	SCB0	0x00000000
0x3005B100	SCB0_HSMDMA_CH1_IB_READ_QOS	SCB0	0x00000001
0x3005B104	SCB0_HSMDMA_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x3005B108	SCB0_HSMDMA_CH1_IB_FN_MOD	SCB0	0x00000000
0x3005C100	SCB0_UART2_TX_READ_QOS	SCB0	0x0000000C
0x3005C104	SCB0_UART2_TX_WRITE_QOS	SCB0	0x0000000C
0x3005C108	SCB0_UART2_TX_FN_MOD	SCB0	0x00000000
0x3005D100	SCB0_SPI0TX_READ_QOS	SCB0	0x0000000C
0x3005D104	SCB0_SPI0TX_WRITE_QOS	SCB0	0x0000000C
0x3005D108	SCB0_SPI0TX_FN_MOD	SCB0	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3005E100	SCB0_SPI0RX_READ_QOS	SCB0	0x0000000C
0x3005E104	SCB0_SPI0RX_WRITE_QOS	SCB0	0x0000000C
0x3005E108	SCB0_SPI0RX_FN_MOD	SCB0	0x00000000
0x3005F100	SCB0_SPI1TX_READ_QOS	SCB0	0x0000000C
0x3005F104	SCB0_SPI1TX_WRITE_QOS	SCB0	0x0000000C
0x3005F108	SCB0_SPI1TX_FN_MOD	SCB0	0x00000000
0x30060100	SCB0_SPI1RX_READ_QOS	SCB0	0x0000000C
0x30060104	SCB0_SPI1RX_WRITE_QOS	SCB0	0x0000000C
0x30060108	SCB0_SPI1RX_FN_MOD	SCB0	0x00000000
0x30061100	SCB0_SPI2TX_READ_QOS	SCB0	0x0000000C
0x30061104	SCB0_SPI2TX_WRITE_QOS	SCB0	0x0000000C
0x30061108	SCB0_SPI2TX_FN_MOD	SCB0	0x00000000
0x30062100	SCB0_SPI2RX_READ_QOS	SCB0	0x0000000C
0x30062104	SCB0_SPI2RX_WRITE_QOS	SCB0	0x0000000C
0x30062108	SCB0_SPI2RX_FN_MOD	SCB0	0x00000000
0x30063100	SCB0_SP6A_READ_QOS	SCB0	0x0000000C
0x30063104	SCB0_SP6A_WRITE_QOS	SCB0	0x0000000C
0x30063108	SCB0_SP6A_FN_MOD	SCB0	0x00000000
0x30064100	SCB0_SP6B_READ_QOS	SCB0	0x0000000C
0x30064104	SCB0_SP6B_WRITE_QOS	SCB0	0x0000000C
0x30064108	SCB0_SP6B_FN_MOD	SCB0	0x00000000
0x30065100	SCB0_LP0_READ_QOS	SCB0	0x00000007
0x30065104	SCB0_LP0_WRITE_QOS	SCB0	0x00000007
0x30065108	SCB0_LP0_FN_MOD	SCB0	0x00000000
0x30066100	SCB0_SP7A_READ_QOS	SCB0	0x0000000C
0x30066104	SCB0_SP7A_WRITE_QOS	SCB0	0x0000000C
0x30066108	SCB0_SP7A_FN_MOD	SCB0	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30067100	SCB0_SP7B_READ_QOS	SCB0	0x0000000C
0x30067104	SCB0_SP7B_WRITE_QOS	SCB0	0x0000000C
0x30067108	SCB0_SP7B_FN_MOD	SCB0	0x00000000
0x30068024	SCB0_SH0_MMR_IB_FN_MOD2	SCB0	0x00000000
0x30068100	SCB0_SH0_MMR_IB_READ_QOS	SCB0	0x00000007
0x30068104	SCB0_SH0_MMR_IB_WRITE_QOS	SCB0	0x00000007
0x30068108	SCB0_SH0_MMR_IB_FN_MOD	SCB0	0x00000000
0x30069100	SCB0_IDMA_READ_QOS	SCB0	0x00000007
0x30069104	SCB0_IDMA_WRITE_QOS	SCB0	0x00000007
0x30069108	SCB0_IDMA_FN_MOD	SCB0	0x00000000
0x3006A100	SCB0_UART1_TX_READ_QOS	SCB0	0x0000000C
0x3006A104	SCB0_UART1_TX_WRITE_QOS	SCB0	0x0000000C
0x3006A108	SCB0_UART1_TX_FN_MOD	SCB0	0x00000000
0x3006B100	SCB0_UART1_RX_READ_QOS	SCB0	0x0000000C
0x3006B104	SCB0_UART1_RX_WRITE_QOS	SCB0	0x0000000C
0x3006B108	SCB0_UART1_RX_FN_MOD	SCB0	0x00000000
0x3006C100	SCB0_SH0_DPORT_L2_READ_QOS	SCB0	0x00000007
0x3006C104	SCB0_SH0_DPORT_L2_WRITE_QOS	SCB0	0x00000007
0x3006C108	SCB0_SH0_DPORT_L2_FN_MOD	SCB0	0x00000000
0x3006D100	SCB0_LP1_READ_QOS	SCB0	0x00000007
0x3006D104	SCB0_LP1_WRITE_QOS	SCB0	0x00000007
0x3006D108	SCB0_LP1_FN_MOD	SCB0	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3006E024	SCB0_CRC2_CH1_IB_FN_MOD2	SCB0	0x00000000
0x3006E100	SCB0_CRC2_CH1_IB_READ_QOS	SCB0	0x00000001
0x3006E104	SCB0_CRC2_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x3006E108	SCB0_CRC2_CH1_IB_FN_MOD	SCB0	0x00000000
0x3006F024	SCB0_CRC2_CH0_IB_FN_MOD2	SCB0	0x00000000
0x3006F100	SCB0_CRC2_CH0_IB_READ_QOS	SCB0	0x00000001
0x3006F104	SCB0_CRC2_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x3006F108	SCB0_CRC2_CH0_IB_FN_MOD	SCB0	0x00000000
0x30070024	SCB0_CRC3_CH1_IB_FN_MOD2	SCB0	0x00000000
0x30070100	SCB0_CRC3_CH1_IB_READ_QOS	SCB0	0x00000001
0x30070104	SCB0_CRC3_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x30070108	SCB0_CRC3_CH1_IB_FN_MOD	SCB0	0x00000000
0x30071024	SCB0_CRYPTO_IB_FN_MOD2	SCB0	0x00000000
0x30071100	SCB0_CRYPTO_IB_READ_QOS	SCB0	0x00000001
0x30071104	SCB0_CRYPTO_IB_WRITE_QOS	SCB0	0x00000001
0x30071108	SCB0_CRYPTO_IB_FN_MOD	SCB0	0x00000000
0x30072024	SCB0_CRC3_CH0_IB_FN_MOD2	SCB0	0x00000000
0x30072100	SCB0_CRC3_CH0_IB_READ_QOS	SCB0	0x00000001

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30072104	SCB0_CRC3_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x30072108	SCB0_CRC3_CH0_IB_FN_MOD	SCB0	0x00000000
0x30073024	SCB0_MSMDMA1_CH1_IB_FN_MOD2	SCB0	0x00000000
0x30073100	SCB0_MSMDMA1_CH1_IB_READ_QOS	SCB0	0x00000001
0x30073104	SCB0_MSMDMA1_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x30073108	SCB0_MSMDMA1_CH1_IB_FN_MOD	SCB0	0x00000000
0x30075100	SCB0_SH0_FIR_CH1_READ_QOS	SCB0	0x00000007
0x30075104	SCB0_SH0_FIR_CH1_WRITE_QOS	SCB0	0x00000007
0x30075108	SCB0_SH0_FIR_CH1_FN_MOD	SCB0	0x00000000
0x30076024	SCB0_MSMDMA1_CH0_IB_FN_MOD2	SCB0	0x00000000
0x30076100	SCB0_MSMDMA1_CH0_IB_READ_QOS	SCB0	0x00000001
0x30076104	SCB0_MSMDMA1_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x30076108	SCB0_MSMDMA1_CH0_IB_FN_MOD	SCB0	0x00000000
0x30077024	SCB0_HSMDMA1_CH1_IB_FN_MOD2	SCB0	0x00000000
0x30077100	SCB0_HSMDMA1_CH1_IB_READ_QOS	SCB0	0x00000001
0x30077104	SCB0_HSMDMA1_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x30077108	SCB0_HSMDMA1_CH1_IB_FN_MOD	SCB0	0x00000000
0x30078024	SCB0_DLDMA0_CH0_IB_FN_MOD2	SCB0	0x00000000

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30078100	SCB0_DLDMA0_CH0_IB_READ_QOS	SCB0	0x00000001
0x30078104	SCB0_DLDMA0_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x30078108	SCB0_DLDMA0_CH0_IB_FN_MOD	SCB0	0x00000000
0x30079024	SCB0_DLDMA0_CH1_IB_FN_MOD2	SCB0	0x00000000
0x30079100	SCB0_DLDMA0_CH1_IB_READ_QOS	SCB0	0x00000001
0x30079104	SCB0_DLDMA0_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x30079108	SCB0_DLDMA0_CH1_IB_FN_MOD	SCB0	0x00000000
0x3007A024	SCB0_DLDMA1_CH0_IB_FN_MOD2	SCB0	0x00000000
0x3007A100	SCB0_DLDMA1_CH0_IB_READ_QOS	SCB0	0x00000001
0x3007A104	SCB0_DLDMA1_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x3007A108	SCB0_DLDMA1_CH0_IB_FN_MOD	SCB0	0x00000000
0x3007B024	SCB0_DLDMA1_CH1_IB_FN_MOD2	SCB0	0x00000000
0x3007B100	SCB0_DLDMA1_CH1_IB_READ_QOS	SCB0	0x00000001
0x3007B104	SCB0_DLDMA1_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x3007B108	SCB0_DLDMA1_CH1_IB_FN_MOD	SCB0	0x00000000
0x3007C024	SCB0_MSMDMA_CH0_IB_FN_MOD2	SCB0	0x00000000
0x3007C100	SCB0_MSMDMA_CH0_IB_READ_QOS	SCB0	0x00000001
0x3007C104	SCB0_MSMDMA_CH0_IB_WRITE_QOS	SCB0	0x00000001

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3007C108	SCB0_MSMDMA_CH0_IB_FN_MOD	SCB0	0x00000000
0x3007D024	SCB0_MSMDMA_CH1_IB_FN_MOD2	SCB0	0x00000000
0x3007D100	SCB0_MSMDMA_CH1_IB_READ_QOS	SCB0	0x00000001
0x3007D104	SCB0_MSMDMA_CH1_IB_WRITE_QOS	SCB0	0x00000001
0x3007D108	SCB0_MSMDMA_CH1_IB_FN_MOD	SCB0	0x00000000
0x3007E024	SCB0_HSMDMA1_CH0_IB_FN_MOD2	SCB0	0x00000000
0x3007E100	SCB0_HSMDMA1_CH0_IB_READ_QOS	SCB0	0x00000001
0x3007E104	SCB0_HSMDMA1_CH0_IB_WRITE_QOS	SCB0	0x00000001
0x3007E108	SCB0_HSMDMA1_CH0_IB_FN_MOD	SCB0	0x00000000
0x3007F024	SCB0_GIGE1_IB_FN_MOD2	SCB0	0x00000000
0x3007F100	SCB0_GIGE1_IB_READ_QOS	SCB0	0x00000007
0x3007F104	SCB0_GIGE1_IB_WRITE_QOS	SCB0	0x00000007
0x3007F108	SCB0_GIGE1_IB_FN_MOD	SCB0	0x00000000
0x30080100	SCB0_M85_M2_AXI_READ_QOS	SCB0	0x00000007
0x30080104	SCB0_M85_M2_AXI_WRITE_QOS	SCB0	0x00000007
0x30080108	SCB0_M85_M2_AXI_FN_MOD	SCB0	0x00000000
0x30081024	SCB0_XSPI_M1_IB_FN_MOD2	SCB0	0x00000000
0x30081108	SCB0_XSPI_M1_IB_FN_MOD	SCB0	0x00000000
0x30082100	SCB0_SPI5TX_READ_QOS	SCB0	0x0000000C

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30082104	SCB0_SPI5TX_WRITE_QOS	SCB0	0x0000000C
0x30082108	SCB0_SPI5TX_FN_MOD	SCB0	0x00000000
0x30083024	SCB0_EMMC_IB_FN_MOD2	SCB0	0x00000000
0x30083100	SCB0_EMMC_IB_READ_QOS	SCB0	0x00000007
0x30083104	SCB0_EMMC_IB_WRITE_QOS	SCB0	0x00000007
0x30083108	SCB0_EMMC_IB_FN_MOD	SCB0	0x00000000
0x30084100	SCB0_SPI5RX_READ_QOS	SCB0	0x0000000C
0x30084104	SCB0_SPI5RX_WRITE_QOS	SCB0	0x0000000C
0x30084108	SCB0_SPI5RX_FN_MOD	SCB0	0x00000000
0x30085024	SCB0_HSM_IB_FN_MOD2	SCB0	0x00000000
0x30085100	SCB0_HSM_IB_READ_QOS	SCB0	0x00000007
0x30085104	SCB0_HSM_IB_WRITE_QOS	SCB0	0x00000007
0x30085108	SCB0_HSM_IB_FN_MOD	SCB0	0x00000000
0x30086100	SCB0_M85_M_AXI_READ_QOS	SCB0	0x00000007
0x30086104	SCB0_M85_M_AXI_WRITE_QOS	SCB0	0x00000007
0x30086108	SCB0_M85_M_AXI_FN_MOD	SCB0	0x00000000
0x30087100	SCB0_SH0_FIR_CH0_READ_QOS	SCB0	0x00000007
0x30087104	SCB0_SH0_FIR_CH0_WRITE_QOS	SCB0	0x00000007
0x30087108	SCB0_SH0_FIR_CH0_FN_MOD	SCB0	0x00000000
0x30088024	SCB0_DBG_IB_FN_MOD2	SCB0	0x00000000
0x30088100	SCB0_DBG_IB_READ_QOS	SCB0	0x00000001

Table A-108: ADSP-2184x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30088104	SCB0_DBG_IB_WRITE_QOS	SCB0	0x00000001
0x30088108	SCB0_DBG_IB_FN_MOD	SCB0	0x00000000
0x30089024	SCB0_ETR_IB_FN_MOD2	SCB0	0x00000000
0x30089100	SCB0_ETR_IB_READ_QOS	SCB0	0x00000001
0x30089104	SCB0_ETR_IB_WRITE_QOS	SCB0	0x00000001
0x30089108	SCB0_ETR_IB_FN_MOD	SCB0	0x00000000
0x3008D100	SCB0_SH0_IIR_CH0_READ_QOS	SCB0	0x00000007
0x3008D104	SCB0_SH0_IIR_CH0_WRITE_QOS	SCB0	0x00000007
0x3008D108	SCB0_SH0_IIR_CH0_FN_MOD	SCB0	0x00000000
0x3008E100	SCB0_SH0_IIR_CH1_READ_QOS	SCB0	0x00000007
0x3008E104	SCB0_SH0_IIR_CH1_WRITE_QOS	SCB0	0x00000007
0x3008E108	SCB0_SH0_IIR_CH1_FN_MOD	SCB0	0x00000000
0x300C2008	SCB0_SCB6_IB_FN_MOD_ISS_BM	SCB0	0x00000000
0x300C2020	SCB0_SCB6_IB_SYNC_MODE	SCB0	0x00000004
0x300C2024	SCB0_SCB6_IB_FN_MOD2	SCB0	0x00000000
0x300C2108	SCB0_SCB6_IB_FN_MOD	SCB0	0x00000000

Table A-109: ADSP-2184x OTPC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31011004	OTPC0_STAT	OTPC0 OTP Status Register	0x00000000
0x3101102C	OTPC0_SECU_STATE	OTPC0 OTP Security State Register	0x00000001

Table A-110: ADSP-2184x PADS0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004604	PADS0_PCFG0	PADS0 Peripheral PAD Configuration 0 Register	0x00018009
0x31004608	PADS0_PCFG1	PADS0 Peripheral Configuration 1 Register	0x00000000
0x3100460C	PADS0_PORTA0_DS	PADS0 PORTA Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004610	PADS0_PORTA1_DS	PADS0 PORTA Pins 8 to 15 Drive Strength Control Register	0x99999999
0x31004614	PADS0_PORTB0_DS	PADS0 PORTB Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004618	PADS0_PORTB1_DS	PADS0 PORTB Pins 8 to 15 Drive Strength Control Register	0x99999999
0x3100461C	PADS0_PORTC0_DS	PADS0 PORTC Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004620	PADS0_PORTC1_DS	PADS0 PORTC Pins 8 to 15 Drive Strength Control Register	0x99999999
0x31004624	PADS0_PORTD0_DS	PADS0 PORTD Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004628	PADS0_PORTD1_DS	PADS0 PORTD Pins 8 to 15 Drive Strength Control Register	0x99999999
0x3100462C	PADS0_PORTE0_DS	PADS0 PORTE Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004630	PADS0_PORTE1_DS	PADS0 PORTE Pins 8 to 15 Drive Strength Control Register	0x99999999
0x31004634	PADS0_PORTF0_DS	PADS0 PORTF Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004638	PADS0_PORTF1_DS	PADS0 PORTF Pins 8 to 15 Drive Strength Control Register	0x99999999
0x3100463C	PADS0_PORTG0_DS	PADS0 PORTG Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004640	PADS0_PORTG1_DS	PADS0 PORTG Pins 8 to 15 Drive Strength Control Register	0x99999999
0x31004644	PADS0_PORTH0_DS	PADS0 PORTH Pins 0 to 7 Drive Strength Control Register	0x99999999
0x31004648	PADS0_PORTH1_DS	PADS0 PORTH Pins 8 and 9 Drive Strength Control Register	0x99999999
0x31004650	PADS0_NONPORTS_DS	PADS0 Non-GPIO Drive Strength Register	0xEEEE999E
0x31004678	PADS0_DAI0_0_DS	PADS0 DAI0 Pins 1 to 8 Drive Strength Control Register	0x99999999

Table A-110: ADSP-2184x PADS0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100467C	PADS0_DAI0_1_DS	PADS0 DAI0 Pins 9 to 16 Drive Strength Control Register	0x99999999
0x31004680	PADS0_DAI1_0_DS	PADS0 DAI1 Pins 1 to 8 Drive Strength Control Register	0x99999999
0x31004684	PADS0_DAI1_1_DS	PADS0 DAI1 Pins 9 to 16 Drive Strength Control Register	0x99999999
0x31004690	PADS0_DAI0_IE	PADS0 DAI0 Port Input Enable Control Register	0x00000000
0x31004694	PADS0_DAI1_IE	PADS0 DAI1 Port Input Enable Control Register	0x00000000
0x310046F4	PADS0_OTPC_BOOT_STAT	PADS0 OTPC Boot Status Register	0x00000000
0x310046F8	PADS0_NONPORTS_CTL	PADS0 Non-GPIO Control Register	0x00003180
0x31004708	PADS0_STAT	PADS0 Status Register	0x00000001
0x31004724	PADS0_DAI0_2_DS	PADS0 DAI0 Pins 17 to 20 Drive Strength Control Register	0x00009999
0x31004728	PADS0_DAI1_2_DS	PADS0 DAI1 Pins 17 to 20 Drive Strength Control Register	0x00009999
0x3100472C	PADS0_PORT_AB_PE	PADS0 PORTA and PORTB Pull Enable Register	0x00000000
0x31004730	PADS0_PORT_CD_PE	PADS0 PORTC and PORTD Pull Enable Register	0x00000000
0x31004734	PADS0_PORT_EF_PE	PADS0 PORTE and PORTF Pull Enable Register	0x00000000
0x31004738	PADS0_PORT_GH_PE	PADS0 PORTG and PORTH Pull Enable Register	0x00000000
0x31004740	PADS0_DAI0_PE	PADS0 DAI0 Pull Enable Register	0x00000000
0x31004744	PADS0_DAI1_PE	PADS0 DAI1 Pull Enable Register	0x00000000
0x31004748	PADS0_PORT_AB_PS	PADS0 PORTA and PORTB Pull Selector Register	0x00000000
0x3100474C	PADS0_PORT_CD_PS	PADS0 PORTC and PORTD Pull Selector Register	0x00000000
0x31004750	PADS0_PORT_EF_PS	PADS0 PORTE and PORTF Pull Selector Register	0x00000000
0x31004754	PADS0_PORT_GH_PS	PADS0 PORTG and PORTH Pull Selector Register	0x00000000
0x3100475C	PADS0_DAI0_PS	PADS0 DAI0 Pull Selector Register	0x00000000
0x31004760	PADS0_DAI1_PS	PADS0 DAI1 Pull Selector Register	0x00000000
0x31004764	PADS0_PORT_AB_SL	PADS0 PORTA and PORTB Slew Rate Control Register	0xFFFFFFFF
0x31004768	PADS0_PORT_CD_SL	PADS0 PORTC and PORTD Slew Rate Control Register	0xFFFFFFFF
0x3100476C	PADS0_PORT_EF_SL	PADS0 PORTE and PORTF Slew Rate Control Register	0xFFFFFFFF
0x31004770	PADS0_PORT_GH_SL	PADS0 PORTG and PORTH Slew Rate Control Register	0xFFFFFFFF
0x31004778	PADS0_DAI0_SL	PADS0 DAI0 Slew Rate Control Register	0x000FFFFF

Table A-110: ADSP-2184x PADS0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100477C	PADS0_DAI1_SL	PADS0 DAI1 Slew Rate Control Register	0x000FFFFF
0x31004780	PADS0_PORT_AB_SPU	PADS0 PORTA and PORTB Strong Pull Up Control Register	0x00000000
0x31004784	PADS0_PORT_CD_SPU	PADS0 PORTC and PORTD Strong Pull Up Control Register	0x00000000
0x31004788	PADS0_PORT_EF_SPU	PADS0 PORTE and PORTF Strong Pull Up Control Register	0x00000000
0x3100478C	PADS0_PORT_GH_SPU	PADS0 PORTG and PORTH Strong Pull Up Control Register	0x00000000
0x31004794	PADS0_DAI0_SPU	PADS0 DAI0 Strong Pull Up Control Register	0x00000000
0x31004798	PADS0_DAI1_SPU	PADS0 DAI1 Strong Pull up Control Register	0x00000000
0x3100479C	PADS0_NONPORTS_DS_1	PADS0 EMAC Non-GPIO Drive Strength Register	0x0000000E

Table A-111: ADSP-2184x PCG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9300	PCG0_CTLA0	PCG0 Precision Clock A Control 0 Register	0x00000000
0x310C9304	PCG0_CTLA1	PCG0 Precision Clock A Control 1 Register	0x00000000
0x310C9308	PCG0_CTLB0	PCG0 Precision Clock B Control 0 Register	0x00000000
0x310C930C	PCG0_CTLB1	PCG0 Precision Clock B Control 1 Register	0x00000000
0x310C9310	PCG0_PW1	PCG0 Precision Clock Pulse Width Control 1 Register	0x00000000
0x310C9314	PCG0_SYNC1	PCG0 Precision Clock Frame Sync Synchronization 1 Register	0x00000000
0x310C9318	PCG0_CTLE0	PCG0 Precision Clock E Control 0 Register	0x00000000
0x310C931C	PCG0_CTLE1	PCG0 Precision Clock E Control 1 Register	0x00000000
0x310C9320	PCG0_CTLF0	PCG0 Precision Clock F Control 0 Register	0x00000000
0x310C9324	PCG0_CTLF1	PCG0 Precision Clock F Control 1 Register	0x00000000
0x310C9328	PCG0_PW3	PCG0 Precision Clock Pulse Width Control 3 Register	0x00000000
0x310C932C	PCG0_SYNC3	PCG0 Precision Clock Frame Sync Synchronization 3 Register	0x00000000
0x310CA300	PCG0_CTLC0	PCG0 Precision Clock C Control 0 Register	0x00000000
0x310CA304	PCG0_CTLC1	PCG0 Precision Clock C Control 1 Register	0x00000000
0x310CA308	PCG0_CTLD0	PCG0 Precision Clock D Control 0 Register	0x00000000

Table A-111: ADSP-2184x PCG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA30C	PCG0_CTLD1	PCG0 Precision Clock D Control 1 Register	0x00000000
0x310CA310	PCG0_PW2	PCG0 Precision Clock Pulse Width Control 2 Register	0x00000000
0x310CA314	PCG0_SYNC2	PCG0 Precision Clock Frame Sync Synchronization 2 Register	0x00000000
0x310CA318	PCG0_CTLG0	PCG0 Precision Clock G Control 0 Register	0x00000000
0x310CA31C	PCG0_CTLG1	PCG0 Precision Clock G Control 1 Register	0x00000000
0x310CA320	PCG0_CTLH0	PCG0 Precision Clock H Control 0 Register	0x00000000
0x310CA324	PCG0_CTLH1	PCG0 Precision Clock H Control 1 Register	0x00000000
0x310CA328	PCG0_PW4	PCG0 Precision Clock Pulse Width Control 4 Register	0x00000000
0x310CA32C	PCG0_SYNC4	PCG0 Precision Clock Frame Sync Synchronization 4 Register	0x00000000

Table A-112: ADSP-2184x PDM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C90A4	PDM0_CTL0	PDM0 PDM Control Register	0x00000013
0x310C90A8	PDM0_HPF_CTL	PDM0 High Pass Filter Control Register	0x000000D0
0x310C90AC	PDM0_SP_CTL0	PDM0 Serial Port Control0 Register	0x00000041
0x310C90B0	PDM0_SP_CTL1	PDM0 Serial Port Control1 Register	0x31211101
0x310C90B8	PDM0_RESET	PDM0 Software Reset Register	0x00000000

Table A-113: ADSP-2184x PDM1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA0A4	PDM1_CTL0	PDM1 PDM Control Register	0x00000013
0x310CA0A8	PDM1_HPF_CTL	PDM1 High Pass Filter Control Register	0x000000D0
0x310CA0AC	PDM1_SP_CTL0	PDM1 Serial Port Control0 Register	0x00000041
0x310CA0B0	PDM1_SP_CTL1	PDM1 Serial Port Control1 Register	0x31211101
0x310CA0B8	PDM1_RESET	PDM1 Software Reset Register	0x00000000

Table A-114: ADSP-2184x PFB MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
	PFB_CTL0	PFB Control Register 0	0x00000000

Table A-114: ADSP-2184x PFB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31076000			
0x31076000	PFB0_CTL0	PFB0 Control Register 0	0x00000000
0x31076004	PFB_CTL1	PFB Control Register 1	0x00001107
0x31076004	PFB0_CTL1	PFB0 Control Register 1	0x00001107
0x31076008	PFB_BCR_START	PFB Prefetch buffer third control register	0x00000000
0x3107600C	PFB_BCR_END	PFB Prefetch buffer fourth control register	0x00000000
0x31076014	PFB_ZERO_START	PFB Return Zero Start Address Register	0x00000000
0x31076014	PFB0_ZERO_START	PFB0 Return Zero Start Address Register	0x00000000
0x31076018	PFB_ZERO_END	PFB Return Zero End Address Register	0x00000000
0x31076018	PFB0_ZERO_END	PFB0 Return Zero End Address Register	0x00000000
0x3107601C	PFB_RANGE_SELECT	PFB Range Select Register	0x0000FFFF
0x3107601C	PFB0_RANGE_SELECT	PFB0 Range Select Register	0x0000FFFF
0x31076020	PFB_STS0	PFB Prefetch Buffer First Status Register	0x00000002
0x31076020	PFB0_STS0	PFB0 Prefetch Buffer First Status Register	0x00000002
0x31076050	PFB_RQSTR_ID_EN	PFB Requester ID Enable Register	0x0000000F
0x31076050	PFB0_RQSTR_ID_EN	PFB0 Requester ID Enable Register	0x0000000F
0x31076054	PFB_RQSTR0_ID	PFB Requester ID 0 Register	0x0000000F
0x31076054	PFB0_RQSTR0_ID	PFB0 Requester ID 0 Register	0x0000000F
0x31076058	PFB_RQSTR0_ID_MSK	PFB Requester ID 0 Mask Register	0x00000000
0x31076058	PFB0_RQSTR0_ID_MSK	PFB0 Requester ID 0 Mask Register	0x00000000
0x3107605C	PFB_RQSTR1_ID	PFB Requester ID 1 Register	0x0000000F
0x3107605C	PFB0_RQSTR1_ID	PFB0 Requester ID 1 Register	0x0000000F
0x31076060	PFB_RQSTR1_ID_MSK	PFB Requester ID 1 Mask Register	0x00000000
0x31076060	PFB0_RQSTR1_ID_MSK	PFB0 Requester ID 1 Mask Register	0x00000000
0x31076064	PFB_RQSTR2_ID	PFB Requester ID 2 Register	0x0000000F
0x31076064	PFB0_RQSTR2_ID	PFB0 Requester ID 2 Register	0x0000000F
0x31076068	PFB_RQSTR2_ID_MSK	PFB Requester ID 2 Mask Register	0x00000000
0x31076068	PFB0_RQSTR2_ID_MSK	PFB0 Requester ID 2 Mask Register	0x00000000
0x3107606C	PFB_RQSTR3_ID	PFB Requester ID 3 Register	0x0000000F
0x3107606C	PFB0_RQSTR3_ID	PFB0 Requester ID 3 Register	0x0000000F
0x31076070	PFB_RQSTR3_ID_MSK	PFB Requester ID 3 Mask Register	0x00000000

Table A-114: ADSP-2184x PFB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31076070	PFB0_RQSTR3_ID_MSK	PFB0 Requester ID 3 Mask Register	0x00000000

Table A-115: ADSP-2184x PFB1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31145000	PFB1_CTL0	PFB1 Control Register 0	0x00000000
0x31145004	PFB1_CTL1	PFB1 Control Register 1	0x00001107
0x31145008	PFB1_BCR_START	PFB1 Prefetch buffer third control register	0x00000000
0x3114500C	PFB1_BCR_END	PFB1 Prefetch buffer fourth control register	0x00000000
0x31145014	PFB1_ZERO_START	PFB1 Return Zero Start Address Register	0x00000000
0x31145018	PFB1_ZERO_END	PFB1 Return Zero End Address Register	0x00000000
0x3114501C	PFB1_RANGE_SELECT	PFB1 Range Select Register	0x0000FFFF
0x31145020	PFB1_STS0	PFB1 Prefetch Buffer First Status Register	0x00000002
0x31145050	PFB1_RQSTR_ID_EN	PFB1 Requester ID Enable Register	0x0000000F
0x31145054	PFB1_RQSTR0_ID	PFB1 Requester ID 0 Register	0x0000000F
0x31145058	PFB1_RQSTR0_ID_MSK	PFB1 Requester ID 0 Mask Register	0x00000000
0x3114505C	PFB1_RQSTR1_ID	PFB1 Requester ID 1 Register	0x0000000F
0x31145060	PFB1_RQSTR1_ID_MSK	PFB1 Requester ID 1 Mask Register	0x00000000
0x31145064	PFB1_RQSTR2_ID	PFB1 Requester ID 2 Register	0x0000000F
0x31145068	PFB1_RQSTR2_ID_MSK	PFB1 Requester ID 2 Mask Register	0x00000000
0x3114506C	PFB1_RQSTR3_ID	PFB1 Requester ID 3 Register	0x0000000F
0x31145070	PFB1_RQSTR3_ID_MSK	PFB1 Requester ID 3 Mask Register	0x00000000

Table A-116: ADSP-2184x PFB3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3114B000	PFB3_CTL0	PFB3 Control Register 0	0x00000000
0x3114B004	PFB3_CTL1	PFB3 Control Register 1	0x00001107
0x3114B014	PFB3_ZERO_START	PFB3 Return Zero Start Address Register	0x00000000
0x3114B018	PFB3_ZERO_END	PFB3 Return Zero End Address Register	0x00000000
0x3114B01C	PFB3_RANGE_SELECT	PFB3 Range Select Register	0x0000FFFF
0x3114B020	PFB3_STS0	PFB3 Prefetch Buffer First Status Register	0x00000002

Table A-116: ADSP-2184x PFB3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114B050	PFB3_RQSTR_ID_EN	PFB3 Requester ID Enable Register	0x0000000F
0x3114B054	PFB3_RQSTR0_ID	PFB3 Requester ID 0 Register	0x0000000F
0x3114B058	PFB3_RQSTR0_ID_MSK	PFB3 Requester ID 0 Mask Register	0x00000000
0x3114B05C	PFB3_RQSTR1_ID	PFB3 Requester ID 1 Register	0x0000000F
0x3114B060	PFB3_RQSTR1_ID_MSK	PFB3 Requester ID 1 Mask Register	0x00000000
0x3114B064	PFB3_RQSTR2_ID	PFB3 Requester ID 2 Register	0x0000000F
0x3114B068	PFB3_RQSTR2_ID_MSK	PFB3 Requester ID 2 Mask Register	0x00000000
0x3114B06C	PFB3_RQSTR3_ID	PFB3 Requester ID 3 Register	0x0000000F
0x3114B070	PFB3_RQSTR3_ID_MSK	PFB3 Requester ID 3 Mask Register	0x00000000

Table A-117: ADSP-2184x PINT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005000	PINT0_MSK_SET	PINT0 PINT Mask Set Register	0x00000000
0x31005004	PINT0_MSK_CLR	PINT0 PINT Mask Clear Register	0x00000000
0x31005008	PINT0_REQ	PINT0 PINT Request Register	0x00000000
0x3100500C	PINT0_ASSIGN	PINT0 PINT Assign Register	0x00000101
0x31005010	PINT0_EDGE_SET	PINT0 PINT Edge Set Register	0x00000000
0x31005014	PINT0_EDGE_CLR	PINT0 PINT Edge Clear Register	0x00000000
0x31005018	PINT0_INV_SET	PINT0 PINT Invert Set Register	0x00000000
0x3100501C	PINT0_INV_CLR	PINT0 PINT Invert Clear Register	0x00000000
0x31005020	PINT0_PINSTATE	PINT0 PINT Pin State Register	0x00000000
0x31005024	PINT0_LATCH	PINT0 PINT Latch Register	0x00000000

Table A-118: ADSP-2184x PINT1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005100	PINT1_MSK_SET	PINT1 PINT Mask Set Register	0x00000000
0x31005104	PINT1_MSK_CLR	PINT1 PINT Mask Clear Register	0x00000000
0x31005108	PINT1_REQ	PINT1 PINT Request Register	0x00000000
0x3100510C	PINT1_ASSIGN	PINT1 PINT Assign Register	0x00000101
0x31005110	PINT1_EDGE_SET	PINT1 PINT Edge Set Register	0x00000000

Table A-118: ADSP-2184x PINT1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31005114	PINT1_EDGE_CLR	PINT1 PINT Edge Clear Register	0x00000000
0x31005118	PINT1_INV_SET	PINT1 PINT Invert Set Register	0x00000000
0x3100511C	PINT1_INV_CLR	PINT1 PINT Invert Clear Register	0x00000000
0x31005120	PINT1_PINSTATE	PINT1 PINT Pin State Register	0x00000000
0x31005124	PINT1_LATCH	PINT1 PINT Latch Register	0x00000000

Table A-119: ADSP-2184x PINT2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005200	PINT2_MSK_SET	PINT2 PINT Mask Set Register	0x00000000
0x31005204	PINT2_MSK_CLR	PINT2 PINT Mask Clear Register	0x00000000
0x31005208	PINT2_REQ	PINT2 PINT Request Register	0x00000000
0x3100520C	PINT2_ASSIGN	PINT2 PINT Assign Register	0x00000101
0x31005210	PINT2_EDGE_SET	PINT2 PINT Edge Set Register	0x00000000
0x31005214	PINT2_EDGE_CLR	PINT2 PINT Edge Clear Register	0x00000000
0x31005218	PINT2_INV_SET	PINT2 PINT Invert Set Register	0x00000000
0x3100521C	PINT2_INV_CLR	PINT2 PINT Invert Clear Register	0x00000000
0x31005220	PINT2_PINSTATE	PINT2 PINT Pin State Register	0x00000000
0x31005224	PINT2_LATCH	PINT2 PINT Latch Register	0x00000000

Table A-120: ADSP-2184x PINT3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005300	PINT3_MSK_SET	PINT3 PINT Mask Set Register	0x00000000
0x31005304	PINT3_MSK_CLR	PINT3 PINT Mask Clear Register	0x00000000
0x31005308	PINT3_REQ	PINT3 PINT Request Register	0x00000000
0x3100530C	PINT3_ASSIGN	PINT3 PINT Assign Register	0x00000101
0x31005310	PINT3_EDGE_SET	PINT3 PINT Edge Set Register	0x00000000
0x31005314	PINT3_EDGE_CLR	PINT3 PINT Edge Clear Register	0x00000000
0x31005318	PINT3_INV_SET	PINT3 PINT Invert Set Register	0x00000000
0x3100531C	PINT3_INV_CLR	PINT3 PINT Invert Clear Register	0x00000000
0x31005320	PINT3_PINSTATE	PINT3 PINT Pin State Register	0x00000000

Table A-120: ADSP-2184x PINT3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31005324	PINT3_LATCH	PINT3 PINT Latch Register	0x00000000

Table A-121: ADSP-2184x PINT4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005400	PINT4_MSK_SET	PINT4 PINT Mask Set Register	0x00000000
0x31005404	PINT4_MSK_CLR	PINT4 PINT Mask Clear Register	0x00000000
0x31005408	PINT4_REQ	PINT4 PINT Request Register	0x00000000
0x3100540C	PINT4_ASSIGN	PINT4 PINT Assign Register	0x00000101
0x31005410	PINT4_EDGE_SET	PINT4 PINT Edge Set Register	0x00000000
0x31005414	PINT4_EDGE_CLR	PINT4 PINT Edge Clear Register	0x00000000
0x31005418	PINT4_INV_SET	PINT4 PINT Invert Set Register	0x00000000
0x3100541C	PINT4_INV_CLR	PINT4 PINT Invert Clear Register	0x00000000
0x31005420	PINT4_PINSTATE	PINT4 PINT Pin State Register	0x00000000
0x31005424	PINT4_LATCH	PINT4 PINT Latch Register	0x00000000

Table A-122: ADSP-2184x PINT5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005500	PINT5_MSK_SET	PINT5 PINT Mask Set Register	0x00000000
0x31005504	PINT5_MSK_CLR	PINT5 PINT Mask Clear Register	0x00000000
0x31005508	PINT5_REQ	PINT5 PINT Request Register	0x00000000
0x3100550C	PINT5_ASSIGN	PINT5 PINT Assign Register	0x00000101
0x31005510	PINT5_EDGE_SET	PINT5 PINT Edge Set Register	0x00000000
0x31005514	PINT5_EDGE_CLR	PINT5 PINT Edge Clear Register	0x00000000
0x31005518	PINT5_INV_SET	PINT5 PINT Invert Set Register	0x00000000
0x3100551C	PINT5_INV_CLR	PINT5 PINT Invert Clear Register	0x00000000
0x31005520	PINT5_PINSTATE	PINT5 PINT Pin State Register	0x00000000
0x31005524	PINT5_LATCH	PINT5 PINT Latch Register	0x00000000

Table A-123: ADSP-2184x PINT6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005600	PINT6_MSK_SET	PINT6 PINT Mask Set Register	0x00000000
0x31005604	PINT6_MSK_CLR	PINT6 PINT Mask Clear Register	0x00000000
0x31005608	PINT6_REQ	PINT6 PINT Request Register	0x00000000
0x3100560C	PINT6_ASSIGN	PINT6 PINT Assign Register	0x00000101
0x31005610	PINT6_EDGE_SET	PINT6 PINT Edge Set Register	0x00000000
0x31005614	PINT6_EDGE_CLR	PINT6 PINT Edge Clear Register	0x00000000
0x31005618	PINT6_INV_SET	PINT6 PINT Invert Set Register	0x00000000
0x3100561C	PINT6_INV_CLR	PINT6 PINT Invert Clear Register	0x00000000
0x31005620	PINT6_PINSTATE	PINT6 PINT Pin State Register	0x00000000
0x31005624	PINT6_LATCH	PINT6 PINT Latch Register	0x00000000

Table A-124: ADSP-2184x PINT7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005700	PINT7_MSK_SET	PINT7 PINT Mask Set Register	0x00000000
0x31005704	PINT7_MSK_CLR	PINT7 PINT Mask Clear Register	0x00000000
0x31005708	PINT7_REQ	PINT7 PINT Request Register	0x00000000
0x3100570C	PINT7_ASSIGN	PINT7 PINT Assign Register	0x00000101
0x31005710	PINT7_EDGE_SET	PINT7 PINT Edge Set Register	0x00000000
0x31005714	PINT7_EDGE_CLR	PINT7 PINT Edge Clear Register	0x00000000
0x31005718	PINT7_INV_SET	PINT7 PINT Invert Set Register	0x00000000
0x3100571C	PINT7_INV_CLR	PINT7 PINT Invert Clear Register	0x00000000
0x31005720	PINT7_PINSTATE	PINT7 PINT Pin State Register	0x00000000
0x31005724	PINT7_LATCH	PINT7 PINT Latch Register	0x00000000

Table A-125: ADSP-2184x PKA0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310D4000	PKA0_APTR	PKA0 PKA Vector_A Address	0x00000000
0x310D4004	PKA0_BPTR	PKA0 PKA Vector_B Address	0x00000000
0x310D4008	PKA0_CPTR	PKA0 PKA Vector_C Address	0x00000000
0x310D400C	PKA0_DPTR	PKA0 PKA Vector_D Address	0x00000000

Table A-125: ADSP-2184x PKA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310D4010	PKA0_ALEN	PKA0 PKA Vector_A Length	0x00000000
0x310D4014	PKA0_BLEN	PKA0 PKA Vector_B Length	0x00000000
0x310D4018	PKA0_SHIFT	PKA0 PKA Bit Shift Value	0x00000000
0x310D401C	PKA0_FUNC	PKA0 PKA Function	0x00000000
0x310D4020	PKA0_COMPARE	PKA0 PKA Compare Result	0x00000001
0x310D4024	PKA0_RESULTMSW	PKA0 PKA Most-Significant-Word of Result Vector	0x00008000
0x310D4028	PKA0_DIVMSW	PKA0 PKA Most-Significant-Word of Divide Remainder	0x00008000
0x310D6000	PKA0_RAM	PKA0 Start of PKA RAM space	0x00000000

Table A-126: ADSP-2184x PKIC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310D8000	PKIC0_POL_CTL	PKIC0 Polarity Control Register	0x00000000
0x310D8004	PKIC0_TYPE_CTL	PKIC0 Type Control Register	0x00000000
0x310D8008	PKIC0_EN_CTL	PKIC0 Enable Control Register	0x00000000
0x310D800C	PKIC0_RAW_STAT	PKIC0 Raw Status Register	0x00000000
0x310D800C	PKIC0_EN_SET	PKIC0 Enable Set Register	0x00000000
0x310D8010	PKIC0_ACK	PKIC0 Acknowledge Register	0x00000000
0x310D8010	PKIC0_EN_STAT	PKIC0 Enabled Status Register	0x00000000
0x310D8014	PKIC0_EN_CLR	PKIC0 Enable Clear Register	0x00000000

Table A-127: ADSP-2184x PKTE0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD000	PKTE0_CTL_STAT	PKTE0 Packet Engine Control Register	0x00000002
0x310CD004	PKTE0_SRC_ADDR	PKTE0 Packet Engine Source Address	0x00000000
0x310CD008	PKTE0_DEST_ADDR	PKTE0 Packet Engine Destination Address	0x00000000
0x310CD00C	PKTE0_SA_ADDR	PKTE0 Packet Engine SA Address	0x00000000
0x310CD010	PKTE0_STATE_ADDR	PKTE0 Packet Engine State Record Address	0x00000000
0x310CD014	PKTE0_ARC4STATE_ADDR	PKTE0 Packet Engine ARC4 State Record Address	0x00000000
0x310CD018	PKTE0_USERID	PKTE0 Packet Engine User ID	0x00000000

Table A-127: ADSP-2184x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD01C	PKTE0_LEN	PKTE0 Packet Engine Length Register	0x00800000
0x310CD080	PKTE0_CDRBASE_ADDR	PKTE0 Packet Engine Command Descriptor Ring Base Address	0x00000000
0x310CD084	PKTE0_RDRBASE_ADDR	PKTE0 Packet Engine Result Descriptor Ring Base Address	0x00000000
0x310CD088	PKTE0_RING_CFG	PKTE0 Packet Engine Ring Configuration	0x00000000
0x310CD08C	PKTE0_RING_THRESH	PKTE0 Packet Engine Ring Threshold Registers	0x00000000
0x310CD090	PKTE0_CDSC_INCR	PKTE0 Packet Engine Command Descriptor Count Increment Register	0x00000000
0x310CD090	PKTE0_CDSC_CNT	PKTE0 Packet Engine Command Descriptor Count Register	0x00000000
0x310CD094	PKTE0_RDSC_CNT	PKTE0 Packet Engine Result Descriptor Count Registers	0x00000000
0x310CD094	PKTE0_RDSC_DECR	PKTE0 Packet Engine Result Descriptor Count Decrement Registers	0x00000000
0x310CD098	PKTE0_RING_PTR	PKTE0 Packet Engine Ring Pointer Status	0x00000000
0x310CD09C	PKTE0_RING_STAT	PKTE0 Packet Engine Ring Status	0x00000000
0x310CD100	PKTE0_CFG	PKTE0 Packet Engine Configuration Register	0x00000000
0x310CD104	PKTE0_STAT	PKTE0 Packet Engine Status Register	0x00040402
0x310CD108	PKTE0_PE_ALT_KEY_STATUS	PKTE0 PE Alternative Key stat register	0x00000000
0x310CD10C	PKTE0_BUF_THRESH	PKTE0 Packet Engine Buffer Threshold Register	0x00800080
0x310CD110	PKTE0_INBUF_INCR	PKTE0 Packet Engine Input Buffer Count Increment Register	0x00000000
0x310CD110	PKTE0_INBUF_CNT	PKTE0 Packet Engine Input Buffer Count Register	0x00000000
0x310CD114	PKTE0_OUTBUF_CNT	PKTE0 Packet Engine Output Buffer Count Register	0x00000000
0x310CD114	PKTE0_OUTBUF_DECR	PKTE0 Packet Engine Output Buffer Count Decrement Register	0x00000000
0x310CD118	PKTE0_BUF_PTR	PKTE0 Packet Engine Buffer Pointer Register	0x00000000
0x310CD120	PKTE0_DMA_CFG	PKTE0 Packet Engine DMA Configuration Register	0x00180006
0x310CD124	PKTE0_PE_CACHE_CTRL_0	PKTE0 Packet Engine Cache Control 0	0x00000000
0x310CD128	PKTE0_PE_CACHE_CTRL_1	PKTE0 Packet Engine Cache Control 1	0x00000000
0x310CD1D0	PKTE0_ENDIAN_CFG	PKTE0 Packet Engine Endian Configuration Register	0x00E400E4

Table A-127: ADSP-2184x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD1E0	PKTE0_HLT_STAT	PKTE0 Packet Engine Halt Status Register	0x00000000
0x310CD1E0	PKTE0_HLT_CTL	PKTE0 Packet Engine Halt Control Register	0x00000000
0x310CD1E4	PKTE0_CONT	PKTE0 PKTE Continue Register	0x00000000
0x310CD1E8	PKTE0_CLK_CTL	PKTE0 PE Clock Control Register	0x0000001F
0x310CD200	PKTE0_IUMSK_STAT	PKTE0 Interrupt Unmasked Status Register	0x00000401
0x310CD204	PKTE0_INT_CLR	PKTE0 Interrupt Clear Register	0x00000000
0x310CD204	PKTE0_IMSK_STAT	PKTE0 Interrupt Masked Status Register	0x00000000
0x310CD208	PKTE0_INT_EN	PKTE0 Interrupt Enable Register	0x00000000
0x310CD20C	PKTE0_INT_CFG	PKTE0 Interrupt Configuration Register	0x00000000
0x310CD210	PKTE0_IMSK_EN	PKTE0 Interrupt Mask Enable Register	0x00000000
0x310CD214	PKTE0_IMSK_DIS	PKTE0 Interrupt Mask Disable Register	0x00000000
0x310CD400	PKTE0_SA_CMD0	PKTE0 SA Command 0	0x00000000
0x310CD404	PKTE0_SA_CMD1	PKTE0 SA Command 1	0x00000000
0x310CD408	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD40C	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD410	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD414	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD418	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD41C	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD420	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD424	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD428	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD42C	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD430	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD434	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD438	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD43C	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD440	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD444	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD448	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD44C	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000

Table A-127: ADSP-2184x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD450	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD454	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD458	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD45C	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD460	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD464	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD468	PKTE0_SA_SPI	PKTE0 SA SPI Register	0x00000000
0x310CD46C	PKTE0_SA_SEQNUM[n]	PKTE0 SA Sequence Number Register	0x00000000
0x310CD470	PKTE0_SA_SEQNUM[n]	PKTE0 SA Sequence Number Register	0x00000000
0x310CD474	PKTE0_SA_SEQNUM_MSK[n]	PKTE0 SA Sequence Number Mask Registers	0x00000000
0x310CD478	PKTE0_SA_SEQNUM_MSK[n]	PKTE0 SA Sequence Number Mask Registers	0x00000000
0x310CD47C	PKTE0_SA_NONCE	PKTE0 SA Initialization Vector Register	0x00000000
0x310CD47C	PKTE0_SA_ARC4IJPTR	PKTE0 ARC4 i and j Pointer Register	0x00000000
0x310CD47C	PKTE0_SA_RDY	PKTE0 SA Ready Indicator	0x00000000
0x310CD500	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD504	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD508	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD50C	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD510	PKTE0_STATE_BYTE_CNT[n]	PKTE0 State Hash Byte Count Registers	0x00000000
0x310CD514	PKTE0_STATE_BYTE_CNT[n]	PKTE0 State Hash Byte Count Registers	0x00000000
0x310CD518	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD51C	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD520	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD524	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD528	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD52C	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD530	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD534	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000

Table A-127: ADSP-2184x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD700	PKTE0_ARC4STATE_BUF	PKTE0 Starting Entry of 256-byte ARC4 State Buffer	0x00000000
0x310CD800	PKTE0_DATAIO_BUF	PKTE0 Starting Entry of 256-byte Data Input/Output Buffer	0x00000000

Table A-128: ADSP-2184x PORT MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004000	PORT_FER	PORT Port x Function Enable Register	0x00000000
0x31004000	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004004	PORT_FER_SET	PORT Port x Function Enable Set Register	0x00000000
0x31004004	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004008	PORT_FER_CLR	PORT Port x Function Enable Clear Register	0x00000000
0x31004008	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100400C	PORT_DATA	PORT Port x GPIO Data Register	0x00000000
0x3100400C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004010	PORT_DATA_SET	PORT Port x GPIO Data Set Register	0x00000000
0x31004010	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004014	PORT_DATA_CLR	PORT Port x GPIO Data Clear Register	0x00000000
0x31004014	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004018	PORT_DIR	PORT Port x GPIO Direction Register	0x00000000
0x31004018	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100401C	PORT_DIR_SET	PORT Port x GPIO Direction Set Register	0x00000000
0x3100401C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004020	PORT_DIR_CLR	PORT Port x GPIO Direction Clear Register	0x00000000
0x31004020	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004024	PORT_INEN	PORT Port x GPIO Input Enable Register	0x00000000
0x31004024	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004028	PORT_INEN_SET	PORT Port x GPIO Input Enable Set Register	0x00000000
0x31004028	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100402C	PORT_INEN_CLR	PORT Port x GPIO Input Enable Clear Register	0x00000000
0x3100402C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004030	PORT_MUX	PORT Port x Multiplexer Control Register	0x00000000

Table A-128: ADSP-2184x PORT MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004030	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004034	PORT_DATA_TGL	PORT Port x GPIO Output Toggle Register	0x00000000
0x31004034	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004038	PORT_POL	PORT Port x GPIO Polarity Invert Register	0x00000000
0x31004038	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100403C	PORT_POL_SET	PORT Port x GPIO Polarity Invert Set Register	0x00000000
0x3100403C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004040	PORT_POL_CLR	PORT Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004040	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004044	PORT_LOCK	PORT Port x GPIO Lock Register	0x00000000
0x31004044	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004048	PORT_TRIG_TGL	PORT Port x GPIO Trigger Toggle Register	0x00000000
0x31004048	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-129: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004080	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004084	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004088	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100408C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004090	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004094	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004098	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100409C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310040A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310040A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310040A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310040AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310040B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310040B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000

Table A-129: ADSP-2184x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310040B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310040BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x310040C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310040C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310040C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-130: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004100	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004104	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004108	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100410C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004110	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004114	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004118	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100411C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004120	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004124	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004128	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100412C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004130	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004134	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004138	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100413C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004140	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004144	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004148	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-131: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004180	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004184	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004188	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100418C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004190	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004194	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004198	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100419C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310041A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310041A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310041A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310041AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310041B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310041B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310041B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310041BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x310041C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310041C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310041C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-132: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004200	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004204	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004208	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100420C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004210	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004214	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004218	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100421C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000

Table A-132: ADSP-2184x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004220	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004224	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004228	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100422C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004230	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004234	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004238	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100423C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004240	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004244	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004248	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-133: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004280	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004284	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004288	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100428C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004290	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004294	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004298	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100429C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310042A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310042A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310042A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310042AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310042B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310042B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310042B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310042BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000

Table A-133: ADSP-2184x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310042C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310042C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310042C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-134: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004300	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004304	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004308	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100430C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004310	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004314	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004318	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100431C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004320	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004324	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004328	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100432C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004330	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004334	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004338	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100433C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004340	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004344	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004348	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-135: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004380	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004384	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000

Table A-135: ADSP-2184x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004388	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100438C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004390	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004394	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004398	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100439C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310043A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310043A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310043A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310043AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310043B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310043B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310043B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310043BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x310043C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310043C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310043C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-136: ADSP-2184x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004400	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004404	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004408	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100440C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004410	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004414	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004418	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100441C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004420	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004424	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000

Table A-136: ADSP-2184x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004428	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100442C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004430	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004434	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004438	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100443C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004440	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004444	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004448	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-137: ADSP-2184x PWM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31146000	PWM0_CTL	PWM0 Control Register	0x00020000
0x31146004	PWM0_CHANCFG	PWM0 Channel Configuration Register	0x00000000
0x31146008	PWM0_TRIPCFG	PWM0 Trip Configuration Register	0x00000000
0x3114600C	PWM0_STAT	PWM0 Status Register	0x00000000
0x31146010	PWM0_IMSK	PWM0 Interrupt Mask Register	0x00000000
0x31146014	PWM0_ILAT	PWM0 Interrupt Latch Register	0x00000000
0x31146018	PWM0_CHOPCFG	PWM0 Chop Configuration Register	0x00000000
0x31146020	PWM0_SYNC_WID	PWM0 Sync Pulse Width Register	0x000003FF
0x31146024	PWM0_TM0	PWM0 Timer 0 Period Register	0x00000000
0x31146028	PWM0_TM1	PWM0 Timer 1 Period Register	0x00000000
0x3114602C	PWM0_TM2	PWM0 Timer 2 Period Register	0x00000000
0x31146030	PWM0_TM3	PWM0 Timer 3 Period Register	0x00000000
0x31146034	PWM0_TM4	PWM0 Timer 4 Period Register	0x00000000
0x31146038	PWM0_DLYA	PWM0 Channel A Delay Register	0x00000000
0x3114603C	PWM0_DLYB	PWM0 Channel B Delay Register	0x00000000
0x31146040	PWM0_DLYC	PWM0 Channel C Delay Register	0x00000000
0x31146044	PWM0_DLYD	PWM0 Channel D Delay Register	0x00000000
0x31146048	PWM0_ACTL	PWM0 Channel A Control Register	0x00000000

Table A-137: ADSP-2184x PWM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114604C	PWM0_AH0	PWM0 Channel A-High Duty-0 Register	0x00000000
0x31146050	PWM0_AH1	PWM0 Channel A-High Duty-1 Register	0x00000000
0x3114605C	PWM0_AL0	PWM0 Channel A-Low Duty-0 Register	0x00000000
0x31146060	PWM0_AL1	PWM0 Channel A-Low Duty-1 Register	0x00000000
0x3114606C	PWM0_BCTL	PWM0 Channel B Control Register	0x00000000
0x31146070	PWM0_BH0	PWM0 Channel B-High Duty-0 Register	0x00000000
0x31146074	PWM0_BH1	PWM0 Channel B-High Duty-1 Register	0x00000000
0x31146080	PWM0_BL0	PWM0 Channel B-Low Duty-0 Register	0x00000000
0x31146084	PWM0_BL1	PWM0 Channel B-Low Duty-1 Register	0x00000000
0x31146090	PWM0_CCTL	PWM0 Channel C Control Register	0x00000000
0x31146094	PWM0_CH0	PWM0 Channel C-High Pulse Duty Register 0	0x00000000
0x31146098	PWM0_CH1	PWM0 Channel C-High Pulse Duty Register 1	0x00000000
0x311460A4	PWM0_CL0	PWM0 Channel C-Low Pulse Duty Register 0	0x00000000
0x311460A8	PWM0_CL1	PWM0 Channel C-Low Duty-1 Register	0x00000000
0x311460B4	PWM0_DCTL	PWM0 Channel D Control Register	0x00000000
0x311460B8	PWM0_DH0	PWM0 Channel D-High Duty-0 Register	0x00000000
0x311460BC	PWM0_DH1	PWM0 Channel D-High Pulse Duty Register 1	0x00000000
0x311460C8	PWM0_DL0	PWM0 Channel D-Low Pulse Duty Register 0	0x00000000
0x311460CC	PWM0_DL1	PWM0 Channel D-Low Pulse Duty Register 1	0x00000000
0x311460D8	PWM0_AH_DUTY0	PWM0 Channel A-High Full Duty0 Register	0x00000000
0x311460DC	PWM0_AH_DUTY1	PWM0 Channel A-High Full Duty1 Register	0x00000000
0x311460E0	PWM0_AL_DUTY0	PWM0 Channel A-Low Full Duty0 Register	0x00000000
0x311460E4	PWM0_AL_DUTY1	PWM0 Channel A-Low Full Duty1 Register	0x00000000
0x311460E8	PWM0_BH_DUTY0	PWM0 Channel B-High Full Duty0 Register	0x00000000
0x311460EC	PWM0_BH_DUTY1	PWM0 Channel B-High Full Duty1 Register	0x00000000
0x311460F0	PWM0_BL_DUTY0	PWM0 Channel B-Low Full Duty0 Register	0x00000000
0x311460F4	PWM0_BL_DUTY1	PWM0 Channel B-Low Full Duty1 Register	0x00000000
0x311460F8	PWM0_CH_DUTY0	PWM0 Channel C-High Full Duty0 Register	0x00000000
0x311460FC	PWM0_CH_DUTY1	PWM0 Channel C-High Full Duty1 Register	0x00000000
0x31146100	PWM0_CL_DUTY0	PWM0 Channel C-Low Full Duty0 Register	0x00000000
0x31146104	PWM0_CL_DUTY1	PWM0 Channel C-Low Full Duty1 Register	0x00000000

Table A-137: ADSP-2184x PWM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31146108	PWM0_DH_DUTY0	PWM0 Channel D-High Full Duty0 Register	0x00000000
0x3114610C	PWM0_DH_DUTY1	PWM0 Channel D-High Full Duty1 Register	0x00000000
0x31146110	PWM0_DL_DUTY0	PWM0 Channel D-Low Full Duty0 Register	0x00000000
0x31146114	PWM0_DL_DUTY1	PWM0 Channel D-Low Full Duty1 Register	0x00000000
0x31146118	PWM0_CHA_DT	PWM0 Channel A Dead-time Register	0x00000000
0x3114611C	PWM0_CHB_DT	PWM0 Channel B Dead-time Register	0x00000000
0x31146120	PWM0_CHC_DT	PWM0 Channel C Dead-time Register	0x00000000
0x31146124	PWM0_CHD_DT	PWM0 Channel D Dead-time Register	0x00000000
0x31146130	PWM0_SWTRIP	PWM0 Software Trip Register	0x00000000
0x31146134	PWM0_TRIP_POL	PWM0 Trip Polarity Register	0x00000000

Table A-138: ADSP-2184x RCU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108C000	RCU0_CTL	RCU0 Control Register	0x00000700
0x3108C004	RCU0_STAT	RCU0 Status Register	0x00000021
0x3108C008	RCU0_CRCTL	RCU0 Core Reset Outputs Control Register	0x00000006
0x3108C00C	RCU0_CRSTAT	RCU0 Core Reset Outputs Status Register	0x00000006
0x3108C018	RCU0_SRRQSTAT	RCU0 System Reset Request Status Register	0x00000000
0x3108C01C	RCU0_SIDIS	RCU0 System Interface Disable Register	0x00000000
0x3108C020	RCU0_SISTAT	RCU0 System Interface Status Register	0x00000000
0x3108C024	RCU0_SVECT_LCK	RCU0 SVECT Lock Register	0x00000000
0x3108C028	RCU0_BCODE	RCU0 Boot Code Register	0x00000000
0x3108C02C	RCU0_SVECT0	RCU0 Software Vector Register 0	0x000007D1
0x3108C030	RCU0_SVECT1	RCU0 Software Vector Register 1	0x202000E4
0x3108C034	RCU0_SVECT2	RCU0 Software Vector Register 2	0x000007D1
0x3108C06C	RCU0_MSG	RCU0 Message Register	0x00000000
0x3108C070	RCU0_MSG_SET	RCU0 Message Set Bits Register	0x00000000
0x3108C074	RCU0_MSG_CLR	RCU0 Message Clear Bits Register	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31089000	SEC0_GCTL	SEC0 Global Control Register	0x00000000
0x31089004	SEC0_GSTAT	SEC0 Global Status Register	0x00000000
0x31089008	SEC0_RAISE	SEC0 Global Raise Register	0x00000000
0x3108900C	SEC0_END	SEC0 Global End Register	0x00000000
0x31089010	SEC0_FCTL	SEC0 Fault Control Register	0x00000000
0x31089014	SEC0_FSTAT	SEC0 Fault Status Register	0x00000000
0x31089018	SEC0_FSID	SEC0 Fault Source ID Register	0x00000000
0x3108901C	SEC0_FEND	SEC0 Fault End Register	0x00000000
0x31089020	SEC0_FDLY	SEC0 Fault Delay Register	0x00000000
0x31089024	SEC0_FDLY_CUR	SEC0 Fault Delay Current Register	0x00000000
0x31089028	SEC0_FSRDLY	SEC0 Fault System Reset Delay Register	0x00000000
0x3108902C	SEC0_FSRDLY_CUR	SEC0 Fault System Reset Delay Current Register	0x00000000
0x31089030	SEC0_FCOPP	SEC0 Fault COP Period Register	0x00000000
0x31089034	SEC0_FCOPP_CUR	SEC0 Fault COP Period Current Register	0x00000000
0x31089440	SEC0_CCTL[n]	SEC0 SCI Control Register n	0x00000000
0x31089444	SEC0_CSTAT[n]	SEC0 SCI Status Register n	0x00000000
0x31089448	SEC0_CPND[n]	SEC0 Core Pending Register n	0x00000000
0x3108944C	SEC0_CACT[n]	SEC0 SCI Active Register n	0x00000000
0x31089450	SEC0_CPMSK[n]	SEC0 SCI Priority Mask Register n	0x000000FF
0x31089454	SEC0_CGMSK[n]	SEC0 SCI Group Mask Register n	0x00000000
0x31089458	SEC0_CPLVL[n]	SEC0 SCI Priority Level Register n	0x00000007
0x3108945C	SEC0_CSID[n]	SEC0 SCI Source ID Register n	0x00000000
0x31089480	SEC0_CCTL[n]	SEC0 SCI Control Register n	0x00000000
0x31089484	SEC0_CSTAT[n]	SEC0 SCI Status Register n	0x00000000
0x31089488	SEC0_CPND[n]	SEC0 Core Pending Register n	0x00000000
0x3108948C	SEC0_CACT[n]	SEC0 SCI Active Register n	0x00000000
0x31089490	SEC0_CPMSK[n]	SEC0 SCI Priority Mask Register n	0x000000FF
0x31089494	SEC0_CGMSK[n]	SEC0 SCI Group Mask Register n	0x00000000
0x31089498	SEC0_CPLVL[n]	SEC0 SCI Priority Level Register n	0x00000007
0x3108949C	SEC0_CSID[n]	SEC0 SCI Source ID Register n	0x00000000
0x31089800	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089804	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089808	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108980C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089810	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089814	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089818	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108981C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089820	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089824	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089828	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108982C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089830	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089834	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089838	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108983C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089840	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089844	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089848	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108984C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089850	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089854	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089858	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108985C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089860	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089864	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089868	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108986C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089870	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089874	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089878	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108987C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089880	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089884	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089888	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108988C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089890	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089894	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089898	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108989C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898A0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898A4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898A8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898AC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898B0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898B4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898B8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898BC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898C0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898C4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898C8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898CC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898D0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898D4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898D8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898DC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898E0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898E4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898E8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898EC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898F0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898F4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898F8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310898FC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089900	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089904	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089908	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108990C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089910	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089914	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089918	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108991C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089920	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089924	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089928	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108992C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089930	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089934	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089938	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108993C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089940	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089944	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089948	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108994C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089950	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089954	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089958	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108995C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089960	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089964	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089968	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108996C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089970	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089974	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089978	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108997C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089980	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089984	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089988	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108998C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089990	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089994	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089998	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108999C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899A0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899A4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899A8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899AC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899B0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899B4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899B8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899BC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899C0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899C4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899C8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899CC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899D0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899D4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899D8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899DC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899E0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899E4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899E8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899EC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899F0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310899F4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899F8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899FC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089A70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ABC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ACC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ADC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089AEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089B68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089BE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089C60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089CDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089D58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089DD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089E50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089ECC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089ED0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ED4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089ED8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089F48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-139: ADSP-2184x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089FC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-140: ADSP-2184x HSM MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31153000	HSM_HWC_CORE_ID	HSM	0x00000000
0x31153004	HSM_HWC_CORE_INFO	HSM	0x00000000
0x31153008	HSM_HWC_CORE_INFO_1	HSM	0x00000000
0x31153010	HSM_HWC_CORE_KEY_CONTRIBUTION	HSM	0x00000000
0x31153020	HSM_HWC_CORE_ACCESS_CONTROL	HSM	0x00000000
0x31153030	HSM_HWC_CORE_PERMISSIONS	HSM	0x00000000
0x31153100	HSM_HWC_CORE_LOCK_MASTER	HSM	0x00000000
0x31153110	HSM_HWC_CORE_INTERRUPT	HSM	0x00000000

Table A-140: ADSP-2184x HSM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31153114	HSM_HWC_CORE_INTERRUPT_MASK	HSM	0x00000000
0x31153120	HSM_SIC_STATUS	HSM	0x00000000
0x31153124	HSM_BOOT_BYPASS_STATUS	HSM	0x00000000
0x31153128	HSM_BOOT_BYPASS_VECTOR	HSM	0x00000000
0x3115312C	HSM_POWER_DOWN_ALLOWED	HSM	0x00000000
0x31153134	HSM_SIC_HWC_ABORT	HSM	0x00000000
0x31153138	HSM_SIC_HWC_ABORT_DONE	HSM	0x00000000
0x3115313C	HSM_SIC_HWC_CCC	HSM	0x00000000
0x31153140	HSM_SIC_HWC_CCC_DONE	HSM	0x00000000
0x31153144	HSM_ESW_RESET	HSM	0x00000000
0x31153148	HSM_SAFETY_BIST_CONTROL	HSM	0x00000000
0x31153200	HSM_PERSO_PUBLIC_KEY_HASH_DW0	HSM	0x00000000
0x31153204	HSM_PERSO_PUBLIC_KEY_HASH_DW1	HSM	0x00000000
0x31153208	HSM_PERSO_PUBLIC_KEY_HASH_DW2	HSM	0x00000000
0x3115320C	HSM_PERSO_PUBLIC_KEY_HASH_DW3	HSM	0x00000000
0x31153210	HSM_PERSO_PUBLIC_KEY_HASH_DW4	HSM	0x00000000
0x31153214	HSM_PERSO_PUBLIC_KEY_HASH_DW5	HSM	0x00000000
0x31153218	HSM_PERSO_PUBLIC_KEY_HASH_DW6	HSM	0x00000000
0x3115321C	HSM_PERSO_PUBLIC_KEY_HASH_DW7	HSM	0x00000000
0x31153220	HSM_OTP_ESW_START_REGION	HSM	0x00000000

Table A-140: ADSP-2184x HSM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31153224	HSM_FIPS_MODE	HSM	0x00000000
0x31153228	HSM_DEFAULT_TMC_TRNG_CONTROL	HSM	0x00000000
0x3115322C	HSM_DEFAULT_TMC_TRNG_CONFIG	HSM	0x00000000
0x31153400	HSM_BOOT_STATUS	HSM	0x00000000
0x31153404	HSM_SW_BOOT_STATUS	HSM	0x00000000
0x31153408	HSM_CYCLE_COUNTER_DW0	HSM	0x00000000
0x3115340C	HSM_CYCLE_COUNTER_DW1	HSM	0x00000000
0x31153410	HSM_TRNG_INFO	HSM	0x00000000
0x31153420	HSM_SOC_VERSION	HSM	0x01234567
0x31153424	HSM_BUILDER_VERSION	HSM	0x60000A42
0x31153428	HSM_KERNEL_VERSION	HSM	0x00000000
0x3115342C	HSM_SDK_VERSION	HSM	0x00000000
0x31153430	HSM_HW_VERSION	HSM	0x622166C5
0x31153434	HSM_HW_VERSION1	HSM	0x9176B330
0x31153438	HSM_HW_VERSION2	HSM	0xD33980B3
0x3115343C	HSM_HW_VERSION3	HSM	0xC2C1E0C1
0x31153500	HSM_SOC_INTERRUPT_OUT	HSM	0x00000000
0x31153504	HSM_SOC_INTERRUPT_OUT_MASK	HSM	0x00000000
0x31153508	HSM_INTERNAL_HW_ERROR_INFO	HSM	0x00000000
0x3115350C	HSM_INTERNAL_SW_ERROR_INFO	HSM	0x00000000
0x31153510	HSM_OTP_HW_ERROR_INFO	HSM	0x00000000
0x31153520	HSM_SAFETY_DIAGNOSIS_0	HSM	0x00000000

Table A-140: ADSP-2184x HSM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31153524	HSM_SAFETY_DIAGNOSIS_1	HSM	0x00000000
0x31153528	HSM_SAFETY_DIAGNOSIS_2	HSM	0x00000000
0x31153530	HSM_SW_SCRATCH_0	HSM	0x00000000
0x31153540	HSM_DEVICE_LIFECYCLE	HSM	0x00000000
0x31153544	HSM_CUSTOM_0	HSM	0x00000000
0x31153548	HSM_CUSTOM_1	HSM	0x00000000
0x3115354C	HSM_CUSTOM_2	HSM	0x00000000
0x31153550	HSM_CUSTOM_3	HSM	0x00000000
0x31153554	HSM_CUSTOM_4	HSM	0x00000000
0x31153558	HSM_CUSTOM_5	HSM	0x00000000
0x3115355C	HSM_CUSTOM_6	HSM	0x00000000
0x31153560	HSM_CUSTOM_7	HSM	0x00000000
0x31153564	HSM_CUSTOM_8	HSM	0x00000000
0x31153568	HSM_CUSTOM_9	HSM	0x00000000
0x3115356C	HSM_CUSTOM_10	HSM	0x00000000
0x31153570	HSM_CUSTOM_11	HSM	0x00000000
0x31153574	HSM_CUSTOM_12	HSM	0x00000000
0x31153578	HSM_CUSTOM_13	HSM	0x00000000
0x3115357C	HSM_CUSTOM_14	HSM	0x00000000
0x31153580	HSM_CUSTOM_15	HSM	0x00000000
0x31153584	HSM_CUSTOM_16	HSM	0x00000000
0x31153588	HSM_CUSTOM_17	HSM	0x00000000
0x3115358C	HSM_CUSTOM_18	HSM	0x00000000
0x31153590	HSM_CUSTOM_19	HSM	0x00000000
0x31153594	HSM_CUSTOM_20	HSM	0x00000000
0x31153598	HSM_CUSTOM_21	HSM	0x00000000
0x3115359C	HSM_CUSTOM_22	HSM	0x00000000
0x311535A0	HSM_CUSTOM_23	HSM	0x00000000
0x311535B4	HSM_DEVICE_KEYS_PERSO_STATUS	HSM	0x40000000

Table A-140: ADSP-2184x HSM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311535B8	HSM_DEVICE_ROOT_TABLE_STATUS	HSM	0x40000055
0x311535E0	HSM_SBOOT_HASH	HSM	0x00000000
0x31153600	HSM_ESW_2_HLOS_D0	HSM	0x00000000
0x31153604	HSM_ESW_2_HLOS_D1	HSM	0x00000000
0x31153608	HSM_ESW_2_HLOS_D2	HSM	0x00000000
0x3115360C	HSM_ESW_2_HLOS_D3	HSM	0x00000000
0x31153610	HSM_ESW_2_HLOS_D4	HSM	0x00000000
0x31153614	HSM_ESW_2_HLOS_D5	HSM	0x00000000
0x31153618	HSM_ESW_2_HLOS_D6	HSM	0x00000000
0x3115361C	HSM_ESW_2_HLOS_D7	HSM	0x00000000
0x31153620	HSM_ESW_2_HLOS_LENGTH	HSM	0x00000000
0x31153630	HSM_HLOS_2_ESW_D0	HSM	0x00000000
0x31153634	HSM_HLOS_2_ESW_D1	HSM	0x00000000
0x31153638	HSM_HLOS_2_ESW_D2	HSM	0x00000000
0x3115363C	HSM_HLOS_2_ESW_D3	HSM	0x00000000
0x31153640	HSM_HLOS_2_ESW_D4	HSM	0x00000000
0x31153644	HSM_HLOS_2_ESW_D5	HSM	0x00000000
0x31153648	HSM_HLOS_2_ESW_D6	HSM	0x00000000
0x3115364C	HSM_HLOS_2_ESW_D7	HSM	0x00000000
0x31153650	HSM_HLOS_2_ESW_LENGTH	HSM	0x00000000

Table A-141: ADSP-2184x SMC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3100600C	SMC0_B0CTL	SMC0 Bank 0 Control Register	0x01000000
0x31006010	SMC0_B0TIM	SMC0 Bank 0 Timing Register	0x01010101
0x31006014	SMC0_B0ETIM	SMC0 Bank 0 Extended Timing Register	0x00020200
0x3100601C	SMC0_B1CTL	SMC0 Bank 1 Control Register	0x01000000
0x31006020	SMC0_B1TIM	SMC0 Bank 1 Timing Register	0x01010101

Table A-141: ADSP-2184x SMC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31006024	SMC0_B1ETIM	SMC0 Bank 1 Extended Timing Register	0x00020200
0x3100602C	SMC0_B2CTL	SMC0 Bank 2 Control Register	0x01000000
0x31006030	SMC0_B2TIM	SMC0 Bank 2 Timing Register	0x01010101
0x31006034	SMC0_B2ETIM	SMC0 Bank 2 Extended Timing Register	0x00020200
0x3100603C	SMC0_B3CTL	SMC0 Bank 3 Control Register	0x01000000
0x31006040	SMC0_B3TIM	SMC0 Bank 3 Timing Register	0x01010101
0x31006044	SMC0_B3ETIM	SMC0 Bank 3 Extended Timing Register	0x00020200

Table A-142: ADSP-2184x SMPU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31007000	SMPU0_CTL	SMPU0 SMPU Control Register	0x00000000
0x31007004	SMPU0_STAT	SMPU0 SMPU Status Register	0x00000000
0x31007008	SMPU0_IADDR	SMPU0 Interrupt Address Register	0x00000000
0x3100700C	SMPU0_IDTLS	SMPU0 Interrupt Details Register	0x00000000
0x31007010	SMPU0_BADDR	SMPU0 Bus Error Address Register	0x00000000
0x31007014	SMPU0_BDTLS	SMPU0 Bus Error Details Register	0x00000000
0x31007020	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x31007024	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007028	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x3100702C	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007030	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x31007034	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007038	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x3100703C	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007040	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x31007044	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007048	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x3100704C	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007050	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x31007054	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000

Table A-142: ADSP-2184x SMPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31007058	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x3100705C	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007060	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x31007064	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007068	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x3100706C	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007070	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x31007074	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007078	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x3100707C	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007080	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x31007084	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007088	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x3100708C	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007090	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x31007094	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007098	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x3100709C	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x310070A0	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x310070A4	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x310070A8	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x310070AC	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x310070B0	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x310070B4	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x310070B8	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x310070BC	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x310070C0	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x310070C4	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x310070C8	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x310070CC	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x310070D0	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000

Table A-142: ADSP-2184x SMPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310070D4	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x310070D8	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x310070DC	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007220	SMPU0_REVID	SMPU0 SMPU Revision ID Register	0x00000010
0x31007800	SMPU0_SECURECTL	SMPU0 SMPU Control Secure Accesses Register	0x00000000
0x31007820	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007824	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007828	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x3100782C	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007830	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007834	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007838	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x3100783C	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31012000	SMPU_CTL	SMPU SMPU Control Register	0x00000000
0x31012004	SMPU_STAT	SMPU SMPU Status Register	0x00000000
0x31012008	SMPU_IADDR	SMPU Interrupt Address Register	0x00000000
0x3101200C	SMPU_IDTLS	SMPU Interrupt Details Register	0x00000000
0x31012010	SMPU_BADDR	SMPU Bus Error Address Register	0x00000000
0x31012014	SMPU_BDTLS	SMPU Bus Error Details Register	0x00000000
0x31012020	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x31012024	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012028	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x3101202C	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012030	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x31012034	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012038	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x3101203C	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012040	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x31012044	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012048	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x3101204C	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000

Table A-142: ADSP-2184x SMPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31012050	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x31012054	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012058	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x3101205C	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012060	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x31012064	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012068	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x3101206C	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012070	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x31012074	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012078	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x3101207C	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012080	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x31012084	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012088	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x3101208C	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012090	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x31012094	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012098	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x3101209C	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x310120A0	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x310120A4	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x310120A8	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x310120AC	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x310120B0	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x310120B4	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x310120B8	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x310120BC	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x310120C0	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x310120C4	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x310120C8	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000

Table A-142: ADSP-2184x SMPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310120CC	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x310120D0	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x310120D4	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x310120D8	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x310120DC	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x310121A0	SMPU_EXACADD[n]	SMPU Exclusive Access IDn Address	0x00000000
0x310121A4	SMPU_EXACSTAT[n]	SMPU Exclusive Access Status	0x00000000
0x310121A8	SMPU_EXACADD[n]	SMPU Exclusive Access IDn Address	0x00000000
0x310121AC	SMPU_EXACSTAT[n]	SMPU Exclusive Access Status	0x00000000
0x310121B0	SMPU_EXACADD[n]	SMPU Exclusive Access IDn Address	0x00000000
0x310121B4	SMPU_EXACSTAT[n]	SMPU Exclusive Access Status	0x00000000
0x31012220	SMPU_REVID	SMPU SMPU Revision ID Register	0x00000010
0x31012800	SMPU_SECURECTL	SMPU SMPU Control Secure Accesses Register	0x00000000
0x31012820	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012824	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012828	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x3101282C	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012830	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012834	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012838	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x3101283C	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000

Table A-143: ADSP-2184x SMPU11 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A1000	SMPU11_CTL	SMPU11 SMPU Control Register	0x00000000
0x310A1004	SMPU11_STAT	SMPU11 SMPU Status Register	0x00000000
0x310A1008	SMPU11_IADDR	SMPU11 Interrupt Address Register	0x00000000
0x310A100C	SMPU11_IDTLS	SMPU11 Interrupt Details Register	0x00000000
0x310A1010	SMPU11_BADDR	SMPU11 Bus Error Address Register	0x00000000
0x310A1014	SMPU11_BDTLS	SMPU11 Bus Error Details Register	0x00000000

Table A-143: ADSP-2184x SMPU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A1020	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A1024	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1028	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A102C	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1030	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A1034	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1038	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A103C	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1040	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A1044	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1048	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A104C	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1050	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A1054	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1058	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A105C	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1060	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A1064	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1068	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A106C	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1070	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A1074	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1078	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A107C	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1080	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A1084	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1088	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A108C	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1090	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A1094	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1098	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000

Table A-143: ADSP-2184x SMPU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A109C	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A10A0	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A10A4	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A10A8	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A10AC	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A10B0	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A10B4	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A10B8	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A10BC	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A10C0	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A10C4	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A10C8	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A10CC	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A10D0	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A10D4	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A10D8	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A10DC	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1220	SMPU11_REVID	SMPU11 SMPU Revision ID Register	0x00000010
0x310A1800	SMPU11_SECURECTL	SMPU11 SMPU Control Secure Accesses Register	0x00000000
0x310A1820	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1824	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1828	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A182C	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1830	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1834	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1838	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A183C	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000

Table A-144: ADSP-2184x SMPU12 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31012000	SMPU12_CTL	SMPU12 SMPU Control Register	0x00000000
0x31012004	SMPU12_STAT	SMPU12 SMPU Status Register	0x00000000
0x31012008	SMPU12_IADDR	SMPU12 Interrupt Address Register	0x00000000
0x3101200C	SMPU12_IDTLS	SMPU12 Interrupt Details Register	0x00000000
0x31012010	SMPU12_BADDR	SMPU12 Bus Error Address Register	0x00000000
0x31012014	SMPU12_BDTLS	SMPU12 Bus Error Details Register	0x00000000
0x31012020	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x31012024	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012028	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x3101202C	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012030	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x31012034	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012038	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x3101203C	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012040	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x31012044	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012048	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x3101204C	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012050	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x31012054	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012058	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x3101205C	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012060	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x31012064	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012068	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x3101206C	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012070	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x31012074	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012078	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x3101207C	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012080	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000

Table A-144: ADSP-2184x SMPU12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31012084	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012088	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x3101208C	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012090	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x31012094	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012098	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x3101209C	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x310120A0	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x310120A4	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x310120A8	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x310120AC	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x310120B0	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x310120B4	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x310120B8	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x310120BC	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x310120C0	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x310120C4	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x310120C8	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x310120CC	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x310120D0	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x310120D4	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x310120D8	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x310120DC	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012220	SMPU12_REVID	SMPU12 SMPU Revision ID Register	0x00000010
0x31012800	SMPU12_SECURECTL	SMPU12 SMPU Control Secure Accesses Register	0x00000000
0x31012820	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012824	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012828	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x3101282C	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012830	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012834	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000

Table A-144: ADSP-2184x SMPU12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31012838	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x3101283C	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000

Table A-145: ADSP-2184x SMPU2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31083000	SMPU2_CTL	SMPU2 SMPU Control Register	0x00000000
0x31083004	SMPU2_STAT	SMPU2 SMPU Status Register	0x00000000
0x31083008	SMPU2_IADDR	SMPU2 Interrupt Address Register	0x00000000
0x3108300C	SMPU2_IDTLS	SMPU2 Interrupt Details Register	0x00000000
0x31083010	SMPU2_BADDR	SMPU2 Bus Error Address Register	0x00000000
0x31083014	SMPU2_BDTLS	SMPU2 Bus Error Details Register	0x00000000
0x31083020	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x31083024	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083028	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x3108302C	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083030	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x31083034	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083038	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x3108303C	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083040	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x31083044	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083048	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x3108304C	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083050	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x31083054	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083058	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x3108305C	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083060	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x31083064	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083068	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000

Table A-145: ADSP-2184x SMPU2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108306C	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083070	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x31083074	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083078	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x3108307C	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083080	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x31083084	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083088	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x3108308C	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083090	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x31083094	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083098	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x3108309C	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x310830A0	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x310830A4	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x310830A8	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x310830AC	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x310830B0	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x310830B4	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x310830B8	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x310830BC	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x310830C0	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x310830C4	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x310830C8	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x310830CC	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x310830D0	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x310830D4	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x310830D8	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x310830DC	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x310831A0	SMPU2_EXACADD[n]	SMPU2 Exclusive Access IDn Address	0x00000000
0x310831A4	SMPU2_EXACSTAT[n]	SMPU2 Exclusive Access Status	0x00000000

Table A-145: ADSP-2184x SMPU2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310831A8	SMPU2_EXACADD[n]	SMPU2 Exclusive Access IDn Address	0x00000000
0x310831AC	SMPU2_EXACSTAT[n]	SMPU2 Exclusive Access Status	0x00000000
0x310831B0	SMPU2_EXACADD[n]	SMPU2 Exclusive Access IDn Address	0x00000000
0x310831B4	SMPU2_EXACSTAT[n]	SMPU2 Exclusive Access Status	0x00000000
0x31083220	SMPU2_REVID	SMPU2 SMPU Revision ID Register	0x00000010
0x31083800	SMPU2_SECURECTL	SMPU2 SMPU Control Secure Accesses Register	0x00000000
0x31083820	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083824	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083828	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x3108382C	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083830	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083834	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083838	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x3108383C	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000

Table A-146: ADSP-2184x SMPU3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31084000	SMPU3_CTL	SMPU3 SMPU Control Register	0x00000000
0x31084004	SMPU3_STAT	SMPU3 SMPU Status Register	0x00000000
0x31084008	SMPU3_IADDR	SMPU3 Interrupt Address Register	0x00000000
0x3108400C	SMPU3_IDTLS	SMPU3 Interrupt Details Register	0x00000000
0x31084010	SMPU3_BADDR	SMPU3 Bus Error Address Register	0x00000000
0x31084014	SMPU3_BDTLS	SMPU3 Bus Error Details Register	0x00000000
0x31084020	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x31084024	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084028	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x3108402C	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084030	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x31084034	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084038	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000

Table A-146: ADSP-2184x SMPU3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108403C	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084040	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x31084044	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084048	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x3108404C	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084050	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x31084054	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084058	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x3108405C	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084060	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x31084064	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084068	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x3108406C	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084070	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x31084074	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084078	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x3108407C	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084080	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x31084084	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084088	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x3108408C	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084090	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x31084094	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084098	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x3108409C	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x310840A0	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x310840A4	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x310840A8	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x310840AC	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x310840B0	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x310840B4	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000

Table A-146: ADSP-2184x SMPU3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310840B8	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x310840BC	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x310840C0	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x310840C4	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x310840C8	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x310840CC	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x310840D0	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x310840D4	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x310840D8	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x310840DC	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084220	SMPU3_REVID	SMPU3 SMPU Revision ID Register	0x00000010
0x31084800	SMPU3_SECURECTL	SMPU3 SMPU Control Secure Accesses Register	0x00000000
0x31084820	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084824	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084828	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x3108482C	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084830	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084834	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084838	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x3108483C	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000

Table A-147: ADSP-2184x SMPU4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31085000	SMPU4_CTL	SMPU4 SMPU Control Register	0x00000000
0x31085004	SMPU4_STAT	SMPU4 SMPU Status Register	0x00000000
0x31085008	SMPU4_IADDR	SMPU4 Interrupt Address Register	0x00000000
0x3108500C	SMPU4_IDTLS	SMPU4 Interrupt Details Register	0x00000000
0x31085010	SMPU4_BADDR	SMPU4 Bus Error Address Register	0x00000000
0x31085014	SMPU4_BDTLS	SMPU4 Bus Error Details Register	0x00000000
0x31085020	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000

Table A-147: ADSP-2184x SMPU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31085024	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085028	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x3108502C	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085030	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x31085034	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085038	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x3108503C	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085040	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x31085044	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085048	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x3108504C	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085050	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x31085054	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085058	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x3108505C	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085060	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x31085064	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085068	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x3108506C	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085070	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x31085074	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085078	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x3108507C	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085080	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x31085084	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085088	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x3108508C	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085090	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x31085094	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085098	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x3108509C	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000

Table A-147: ADSP-2184x SMPU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310850A0	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x310850A4	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x310850A8	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x310850AC	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x310850B0	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x310850B4	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x310850B8	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x310850BC	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x310850C0	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x310850C4	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x310850C8	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x310850CC	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x310850D0	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x310850D4	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x310850D8	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x310850DC	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x310851A0	SMPU4_EXACADD[n]	SMPU4 Exclusive Access IDn Address	0x00000000
0x310851A4	SMPU4_EXACSTAT[n]	SMPU4 Exclusive Access Status	0x00000000
0x310851A8	SMPU4_EXACADD[n]	SMPU4 Exclusive Access IDn Address	0x00000000
0x310851AC	SMPU4_EXACSTAT[n]	SMPU4 Exclusive Access Status	0x00000000
0x310851B0	SMPU4_EXACADD[n]	SMPU4 Exclusive Access IDn Address	0x00000000
0x310851B4	SMPU4_EXACSTAT[n]	SMPU4 Exclusive Access Status	0x00000000
0x31085220	SMPU4_REVID	SMPU4 SMPU Revision ID Register	0x00000010
0x31085800	SMPU4_SECURECTL	SMPU4 SMPU Control Secure Accesses Register	0x00000000
0x31085820	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085824	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085828	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x3108582C	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085830	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085834	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085838	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000

Table A-147: ADSP-2184x SMPU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108583C	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000

Table A-148: ADSP-2184x SMPU5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31086000	SMPU5_CTL	SMPU5 SMPU Control Register	0x00000000
0x31086004	SMPU5_STAT	SMPU5 SMPU Status Register	0x00000000
0x31086008	SMPU5_IADDR	SMPU5 Interrupt Address Register	0x00000000
0x3108600C	SMPU5_IDTLS	SMPU5 Interrupt Details Register	0x00000000
0x31086010	SMPU5_BADDR	SMPU5 Bus Error Address Register	0x00000000
0x31086014	SMPU5_BDTLS	SMPU5 Bus Error Details Register	0x00000000
0x31086020	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x31086024	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086028	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x3108602C	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086030	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x31086034	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086038	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x3108603C	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086040	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x31086044	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086048	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x3108604C	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086050	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x31086054	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086058	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x3108605C	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086060	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x31086064	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086068	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x3108606C	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000

Table A-148: ADSP-2184x SMPU5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31086070	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x31086074	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086078	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x3108607C	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086080	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x31086084	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086088	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x3108608C	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086090	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x31086094	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086098	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x3108609C	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x310860A0	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x310860A4	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x310860A8	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x310860AC	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x310860B0	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x310860B4	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x310860B8	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x310860BC	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x310860C0	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x310860C4	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x310860C8	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x310860CC	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x310860D0	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x310860D4	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x310860D8	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x310860DC	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086220	SMPU5_REVID	SMPU5 SMPU Revision ID Register	0x00000010
0x31086800	SMPU5_SECURECTL	SMPU5 SMPU Control Secure Accesses Register	0x00000000
0x31086820	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000

Table A-148: ADSP-2184x SMPU5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31086824	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086828	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x3108682C	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086830	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086834	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086838	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x3108683C	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000

Table A-149: ADSP-2184x SMPU6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31087000	SMPU6_CTL	SMPU6 SMPU Control Register	0x00000000
0x31087004	SMPU6_STAT	SMPU6 SMPU Status Register	0x00000000
0x31087008	SMPU6_IADDR	SMPU6 Interrupt Address Register	0x00000000
0x3108700C	SMPU6_IDTLS	SMPU6 Interrupt Details Register	0x00000000
0x31087010	SMPU6_BADDR	SMPU6 Bus Error Address Register	0x00000000
0x31087014	SMPU6_BDTLS	SMPU6 Bus Error Details Register	0x00000000
0x31087020	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x31087024	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087028	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x3108702C	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087030	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x31087034	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087038	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x3108703C	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087040	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x31087044	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087048	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x3108704C	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087050	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x31087054	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000

Table A-149: ADSP-2184x SMPU6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31087058	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x3108705C	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087060	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x31087064	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087068	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x3108706C	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087070	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x31087074	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087078	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x3108707C	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087080	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x31087084	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087088	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x3108708C	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087090	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x31087094	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087098	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x3108709C	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x310870A0	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x310870A4	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x310870A8	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x310870AC	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x310870B0	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x310870B4	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x310870B8	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x310870BC	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x310870C0	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x310870C4	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x310870C8	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x310870CC	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x310870D0	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000

Table A-149: ADSP-2184x SMPU6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310870D4	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x310870D8	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x310870DC	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087220	SMPU6_REVID	SMPU6 SMPU Revision ID Register	0x00000010
0x31087800	SMPU6_SECURECTL	SMPU6 SMPU Control Secure Accesses Register	0x00000000
0x31087820	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087824	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087828	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x3108782C	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087830	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087834	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087838	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x3108783C	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000

Table A-150: ADSP-2184x SMPU9 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A0000	SMPU9_CTL	SMPU9 SMPU Control Register	0x00000000
0x310A0004	SMPU9_STAT	SMPU9 SMPU Status Register	0x00000000
0x310A0008	SMPU9_IADDR	SMPU9 Interrupt Address Register	0x00000000
0x310A000C	SMPU9_IDTLS	SMPU9 Interrupt Details Register	0x00000000
0x310A0010	SMPU9_BADDR	SMPU9 Bus Error Address Register	0x00000000
0x310A0014	SMPU9_BDTLS	SMPU9 Bus Error Details Register	0x00000000
0x310A0020	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A0024	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0028	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A002C	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0030	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A0034	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0038	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A003C	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000

Table A-150: ADSP-2184x SMPU9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A0040	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A0044	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0048	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A004C	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0050	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A0054	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0058	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A005C	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0060	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A0064	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0068	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A006C	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0070	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A0074	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0078	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A007C	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0080	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A0084	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0088	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A008C	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0090	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A0094	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0098	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A009C	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A00A0	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A00A4	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A00A8	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A00AC	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A00B0	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A00B4	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A00B8	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000

Table A-150: ADSP-2184x SMPU9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A00BC	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A00C0	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A00C4	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A00C8	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A00CC	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A00D0	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A00D4	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A00D8	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A00DC	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A01A0	SMPU9_EXACADD[n]	SMPU9 Exclusive Access IDn Address	0x00000000
0x310A01A4	SMPU9_EXACSTAT[n]	SMPU9 Exclusive Access Status	0x00000000
0x310A01A8	SMPU9_EXACADD[n]	SMPU9 Exclusive Access IDn Address	0x00000000
0x310A01AC	SMPU9_EXACSTAT[n]	SMPU9 Exclusive Access Status	0x00000000
0x310A01B0	SMPU9_EXACADD[n]	SMPU9 Exclusive Access IDn Address	0x00000000
0x310A01B4	SMPU9_EXACSTAT[n]	SMPU9 Exclusive Access Status	0x00000000
0x310A0220	SMPU9_REVID	SMPU9 SMPU Revision ID Register	0x00000010
0x310A0800	SMPU9_SECURECTL	SMPU9 SMPU Control Secure Accesses Register	0x00000000
0x310A0820	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0824	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0828	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A082C	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0830	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0834	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0838	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A083C	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000

Table A-151: ADSP-2184x SPDIF0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9280	SPDIF0_TX_CTL	SPDIF0 Transmit Control Register	0x00000000
0x310C9280	SPDIF_TX_CTL	SPDIF Transmit Control Register	0x00000000

Table A-151: ADSP-2184x SPDIF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9284	SPDIF0_TX_STAT_A0	SPDIF0 Transmit Status A0 Register	0x00000000
0x310C9284	SPDIF_TX_STAT_A0	SPDIF Transmit Status A0 Register	0x00000000
0x310C9288	SPDIF0_TX_STAT_B0	SPDIF0 Transmit Status B0 Register	0x00000000
0x310C9288	SPDIF_TX_STAT_B0	SPDIF Transmit Status B0 Register	0x00000000
0x310C92A0	SPDIF0_RX_CTL	SPDIF0 Receive Control	0x00000018
0x310C92A0	SPDIF_RX_CTL	SPDIF Receive Control	0x00000018
0x310C92A4	SPDIF0_RX_STAT	SPDIF0 Receive Status Register	0x00000000
0x310C92A4	SPDIF_RX_STAT	SPDIF Receive Status Register	0x00000000
0x310C92A8	SPDIF0_RX_STAT0_A	SPDIF0 Receive Status A0 Register	0x00000000
0x310C92A8	SPDIF_RX_STAT0_A	SPDIF Receive Status A0 Register	0x00000000
0x310C92AC	SPDIF0_RX_STAT0_B	SPDIF0 Receive Status B0 Register	0x00000000
0x310C92AC	SPDIF_RX_STAT0_B	SPDIF Receive Status B0 Register	0x00000000
0x310C92B0	SPDIF0_RX_STAT1_A	SPDIF0 Receive Status A1 Register	0x00000000
0x310C92B0	SPDIF_RX_STAT1_A	SPDIF Receive Status A1 Register	0x00000000
0x310C92B4	SPDIF0_RX_STAT1_B	SPDIF0 Receive Status B1 Register	0x00000000
0x310C92B4	SPDIF_RX_STAT1_B	SPDIF Receive Status B1 Register	0x00000000
0x310C9350	SPDIF0_TX_STAT_A1	SPDIF0 Transmit Status A1 Register	0x00000000
0x310C9350	SPDIF_TX_STAT_A1	SPDIF Transmit Status A1 Register	0x00000000
0x310C9354	SPDIF0_TX_STAT_A2	SPDIF0 Transmit Status A2 Register	0x00000000
0x310C9354	SPDIF_TX_STAT_A2	SPDIF Transmit Status A2 Register	0x00000000
0x310C9358	SPDIF0_TX_STAT_A3	SPDIF0 Transmit Status A3 Register	0x00000000
0x310C9358	SPDIF_TX_STAT_A3	SPDIF Transmit Status A3 Register	0x00000000
0x310C935C	SPDIF0_TX_STAT_A4	SPDIF0 Transmit Status A4 Register	0x00000000
0x310C935C	SPDIF_TX_STAT_A4	SPDIF Transmit Status A4 Register	0x00000000
0x310C9360	SPDIF0_TX_STAT_A5	SPDIF0 Transmit Status A5 Register	0x00000000
0x310C9360	SPDIF_TX_STAT_A5	SPDIF Transmit Status A5 Register	0x00000000
0x310C9368	SPDIF0_TX_STAT_B1	SPDIF0 Transmit Status B1 Register	0x00000000
0x310C9368	SPDIF_TX_STAT_B1	SPDIF Transmit Status B1 Register	0x00000000
0x310C936C	SPDIF0_TX_STAT_B2	SPDIF0 Transmit Status B2 Register	0x00000000
0x310C936C	SPDIF_TX_STAT_B2	SPDIF Transmit Status B2 Register	0x00000000
0x310C9370	SPDIF0_TX_STAT_B3	SPDIF0 Transmit Status B3 Register	0x00000000

Table A-151: ADSP-2184x SPDIF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9370	SPDIF_TX_STAT_B3	SPDIF Transmit Status B3 Register	0x00000000
0x310C9374	SPDIF0_TX_STAT_B4	SPDIF0 Transmit Status B4 Register	0x00000000
0x310C9374	SPDIF_TX_STAT_B4	SPDIF Transmit Status B4 Register	0x00000000
0x310C9378	SPDIF0_TX_STAT_B5	SPDIF0 Transmit Status B5 Register	0x00000000
0x310C9378	SPDIF_TX_STAT_B5	SPDIF Transmit Status B5 Register	0x00000000
0x310C9380	SPDIF0_TX_UBUFF_A0	SPDIF0 Transmit User Buffer A0 Register	0x00000000
0x310C9380	SPDIF_TX_UBUFF_A0	SPDIF Transmit User Buffer A0 Register	0x00000000
0x310C9384	SPDIF0_TX_UBUFF_A1	SPDIF0 Transmit User Buffer A1 Register	0x00000000
0x310C9384	SPDIF_TX_UBUFF_A1	SPDIF Transmit User Buffer A1 Register	0x00000000
0x310C9388	SPDIF0_TX_UBUFF_A2	SPDIF0 Transmit User Buffer A2 Register	0x00000000
0x310C9388	SPDIF_TX_UBUFF_A2	SPDIF Transmit User Buffer A2 Register	0x00000000
0x310C938C	SPDIF0_TX_UBUFF_A3	SPDIF0 Transmit User Buffer A3 Register	0x00000000
0x310C938C	SPDIF_TX_UBUFF_A3	SPDIF Transmit User Buffer A3 Register	0x00000000
0x310C9390	SPDIF0_TX_UBUFF_A4	SPDIF0 Transmit User Buffer A4 Register	0x00000000
0x310C9390	SPDIF_TX_UBUFF_A4	SPDIF Transmit User Buffer A4 Register	0x00000000
0x310C9394	SPDIF0_TX_UBUFF_A5	SPDIF0 Transmit User Buffer A5 Register	0x00000000
0x310C9394	SPDIF_TX_UBUFF_A5	SPDIF Transmit User Buffer A5 Register	0x00000000
0x310C93A0	SPDIF0_TX_UBUFF_B0	SPDIF0 Transmit User Buffer B0 Register	0x00000000
0x310C93A0	SPDIF_TX_UBUFF_B0	SPDIF Transmit User Buffer B0 Register	0x00000000
0x310C93A4	SPDIF0_TX_UBUFF_B1	SPDIF0 Transmit User Buffer B1 Register	0x00000000
0x310C93A4	SPDIF_TX_UBUFF_B1	SPDIF Transmit User Buffer B1 Register	0x00000000
0x310C93A8	SPDIF0_TX_UBUFF_B2	SPDIF0 Transmit User Buffer B2 Register	0x00000000
0x310C93A8	SPDIF_TX_UBUFF_B2	SPDIF Transmit User Buffer B2 Register	0x00000000
0x310C93AC	SPDIF0_TX_UBUFF_B3	SPDIF0 Transmit User Buffer B3 Register	0x00000000
0x310C93AC	SPDIF_TX_UBUFF_B3	SPDIF Transmit User Buffer B3 Register	0x00000000
0x310C93B0	SPDIF0_TX_UBUFF_B4	SPDIF0 Transmit User Buffer B4 Register	0x00000000
0x310C93B0	SPDIF_TX_UBUFF_B4	SPDIF Transmit User Buffer B4 Register	0x00000000
0x310C93B4	SPDIF0_TX_UBUFF_B5	SPDIF0 Transmit User Buffer B5 Register	0x00000000
0x310C93B4	SPDIF_TX_UBUFF_B5	SPDIF Transmit User Buffer B5 Register	0x00000000
0x310C93BC	SPDIF0_TX_USRUPDT	SPDIF0 User Bit Update Register	0x00000000
0x310C93BC	SPDIF_TX_USRUPDT	SPDIF User Bit Update Register	0x00000000

Table A-152: ADSP-2184x SPDIF1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA280	SPDIF1_TX_CTL	SPDIF1 Transmit Control Register	0x00000000
0x310CA284	SPDIF1_TX_STAT_A0	SPDIF1 Transmit Status A0 Register	0x00000000
0x310CA288	SPDIF1_TX_STAT_B0	SPDIF1 Transmit Status B0 Register	0x00000000
0x310CA2A0	SPDIF1_RX_CTL	SPDIF1 Receive Control	0x00000018
0x310CA2A4	SPDIF1_RX_STAT	SPDIF1 Receive Status Register	0x00000000
0x310CA2A8	SPDIF1_RX_STAT0_A	SPDIF1 Receive Status A0 Register	0x00000000
0x310CA2AC	SPDIF1_RX_STAT0_B	SPDIF1 Receive Status B0 Register	0x00000000
0x310CA2B0	SPDIF1_RX_STAT1_A	SPDIF1 Receive Status A1 Register	0x00000000
0x310CA2B4	SPDIF1_RX_STAT1_B	SPDIF1 Receive Status B1 Register	0x00000000
0x310CA350	SPDIF1_TX_STAT_A1	SPDIF1 Transmit Status A1 Register	0x00000000
0x310CA354	SPDIF1_TX_STAT_A2	SPDIF1 Transmit Status A2 Register	0x00000000
0x310CA358	SPDIF1_TX_STAT_A3	SPDIF1 Transmit Status A3 Register	0x00000000
0x310CA35C	SPDIF1_TX_STAT_A4	SPDIF1 Transmit Status A4 Register	0x00000000
0x310CA360	SPDIF1_TX_STAT_A5	SPDIF1 Transmit Status A5 Register	0x00000000
0x310CA368	SPDIF1_TX_STAT_B1	SPDIF1 Transmit Status B1 Register	0x00000000
0x310CA36C	SPDIF1_TX_STAT_B2	SPDIF1 Transmit Status B2 Register	0x00000000
0x310CA370	SPDIF1_TX_STAT_B3	SPDIF1 Transmit Status B3 Register	0x00000000
0x310CA374	SPDIF1_TX_STAT_B4	SPDIF1 Transmit Status B4 Register	0x00000000
0x310CA378	SPDIF1_TX_STAT_B5	SPDIF1 Transmit Status B5 Register	0x00000000
0x310CA380	SPDIF1_TX_UBUFF_A0	SPDIF1 Transmit User Buffer A0 Register	0x00000000
0x310CA384	SPDIF1_TX_UBUFF_A1	SPDIF1 Transmit User Buffer A1 Register	0x00000000
0x310CA388	SPDIF1_TX_UBUFF_A2	SPDIF1 Transmit User Buffer A2 Register	0x00000000
0x310CA38C	SPDIF1_TX_UBUFF_A3	SPDIF1 Transmit User Buffer A3 Register	0x00000000
0x310CA390	SPDIF1_TX_UBUFF_A4	SPDIF1 Transmit User Buffer A4 Register	0x00000000
0x310CA394	SPDIF1_TX_UBUFF_A5	SPDIF1 Transmit User Buffer A5 Register	0x00000000
0x310CA3A0	SPDIF1_TX_UBUFF_B0	SPDIF1 Transmit User Buffer B0 Register	0x00000000
0x310CA3A4	SPDIF1_TX_UBUFF_B1	SPDIF1 Transmit User Buffer B1 Register	0x00000000
0x310CA3A8	SPDIF1_TX_UBUFF_B2	SPDIF1 Transmit User Buffer B2 Register	0x00000000
0x310CA3AC	SPDIF1_TX_UBUFF_B3	SPDIF1 Transmit User Buffer B3 Register	0x00000000
0x310CA3B0	SPDIF1_TX_UBUFF_B4	SPDIF1 Transmit User Buffer B4 Register	0x00000000
0x310CA3B4	SPDIF1_TX_UBUFF_B5	SPDIF1 Transmit User Buffer B5 Register	0x00000000

Table A-152: ADSP-2184x SPDIF1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA3BC	SPDIF1_TX_USRUPDT	SPDIF1 User Bit Update Register	0x00000000

Table A-153: ADSP-2184x SPI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102E004	SPI0_CTL	SPI0 Control Register	0x00000050
0x3102E004	SPI_CTL	SPI Control Register	0x00000050
0x3102E008	SPI0_RXCTL	SPI0 Receive Control Register	0x00000000
0x3102E008	SPI_RXCTL	SPI Receive Control Register	0x00000000
0x3102E00C	SPI0_TXCTL	SPI0 Transmit Control Register	0x00000000
0x3102E00C	SPI_TXCTL	SPI Transmit Control Register	0x00000000
0x3102E010	SPI0_CLK	SPI0 Clock Rate Register	0x00000000
0x3102E010	SPI_CLK	SPI Clock Rate Register	0x00000000
0x3102E014	SPI0_DLY	SPI0 Delay Register	0x00000301
0x3102E014	SPI_DLY	SPI Delay Register	0x00000301
0x3102E018	SPI0_TSEL	SPI0 Target Select Register	0x0000FE00
0x3102E018	SPI_TSEL	SPI Target Select Register	0x0000FE00
0x3102E01C	SPI0_RWC	SPI0 Received Word Count Register	0x00000000
0x3102E01C	SPI_RWC	SPI Received Word Count Register	0x00000000
0x3102E020	SPI0_RWCR	SPI0 Received Word Count Reload Register	0x00000000
0x3102E020	SPI_RWCR	SPI Received Word Count Reload Register	0x00000000
0x3102E024	SPI0_TWC	SPI0 Transmitted Word Count Register	0x00000000
0x3102E024	SPI_TWC	SPI Transmitted Word Count Register	0x00000000
0x3102E028	SPI0_TWCR	SPI0 Transmitted Word Count Reload Register	0x00000000
0x3102E028	SPI_TWCR	SPI Transmitted Word Count Reload Register	0x00000000
0x3102E030	SPI0_IMSK	SPI0 Interrupt Mask Register	0x00000000
0x3102E030	SPI_IMSK	SPI Interrupt Mask Register	0x00000000
0x3102E034	SPI0_IMSK_CLR	SPI0 Interrupt Mask Clear Register	0x00000000
0x3102E034	SPI_IMSK_CLR	SPI Interrupt Mask Clear Register	0x00000000
0x3102E038	SPI0_IMSK_SET	SPI0 Interrupt Mask Set Register	0x00000000
0x3102E038	SPI_IMSK_SET	SPI Interrupt Mask Set Register	0x00000000

Table A-153: ADSP-2184x SPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102E040	SPI0_STAT	SPI0 Status Register	0x00440001
0x3102E040	SPI_STAT	SPI Status Register	0x00440001
0x3102E044	SPI0_ILAT	SPI0 Masked Interrupt Condition Register	0x00000000
0x3102E044	SPI_ILAT	SPI Masked Interrupt Condition Register	0x00000000
0x3102E048	SPI0_ILAT_CLR	SPI0 Masked Interrupt Clear Register	0x00000000
0x3102E048	SPI_ILAT_CLR	SPI Masked Interrupt Clear Register	0x00000000
0x3102E050	SPI0_RFIFO	SPI0 Receive FIFO Data Register	0x00000000
0x3102E050	SPI_RFIFO	SPI Receive FIFO Data Register	0x00000000
0x3102E058	SPI0_TFIFO	SPI0 Transmit FIFO Data Register	0x00000000
0x3102E058	SPI_TFIFO	SPI Transmit FIFO Data Register	0x00000000
0x3102E060	SPI_MMRDH	SPI Memory Mapped Read Header	0x00000000
0x3102E064	SPI_MMTOP	SPI SPI Memory Top Address	0x00000000

Table A-154: ADSP-2184x SPI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102F004	SPI1_CTL	SPI1 Control Register	0x00000050
0x3102F008	SPI1_RXCTL	SPI1 Receive Control Register	0x00000000
0x3102F00C	SPI1_TXCTL	SPI1 Transmit Control Register	0x00000000
0x3102F010	SPI1_CLK	SPI1 Clock Rate Register	0x00000000
0x3102F014	SPI1_DLY	SPI1 Delay Register	0x00000301
0x3102F018	SPI1_TSEL	SPI1 Target Select Register	0x0000FE00
0x3102F01C	SPI1_RWC	SPI1 Received Word Count Register	0x00000000
0x3102F020	SPI1_RWCR	SPI1 Received Word Count Reload Register	0x00000000
0x3102F024	SPI1_TWC	SPI1 Transmitted Word Count Register	0x00000000
0x3102F028	SPI1_TWCR	SPI1 Transmitted Word Count Reload Register	0x00000000
0x3102F030	SPI1_IMSK	SPI1 Interrupt Mask Register	0x00000000
0x3102F034	SPI1_IMSK_CLR	SPI1 Interrupt Mask Clear Register	0x00000000
0x3102F038	SPI1_IMSK_SET	SPI1 Interrupt Mask Set Register	0x00000000
0x3102F040	SPI1_STAT	SPI1 Status Register	0x00440001
0x3102F044	SPI1_ILAT	SPI1 Masked Interrupt Condition Register	0x00000000

Table A-154: ADSP-2184x SPI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102F048	SPI1_ILAT_CLR	SPI1 Masked Interrupt Clear Register	0x00000000
0x3102F050	SPI1_RFIFO	SPI1 Receive FIFO Data Register	0x00000000
0x3102F058	SPI1_TFIFO	SPI1 Transmit FIFO Data Register	0x00000000

Table A-155: ADSP-2184x SPI2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31030004	SPI2_CTL	SPI2 Control Register	0x00000050
0x31030008	SPI2_RXCTL	SPI2 Receive Control Register	0x00000000
0x3103000C	SPI2_TXCTL	SPI2 Transmit Control Register	0x00000000
0x31030010	SPI2_CLK	SPI2 Clock Rate Register	0x00000000
0x31030014	SPI2_DLY	SPI2 Delay Register	0x00000301
0x31030018	SPI2_TSEL	SPI2 Target Select Register	0x0000FE00
0x3103001C	SPI2_RWC	SPI2 Received Word Count Register	0x00000000
0x31030020	SPI2_RWCR	SPI2 Received Word Count Reload Register	0x00000000
0x31030024	SPI2_TWC	SPI2 Transmitted Word Count Register	0x00000000
0x31030028	SPI2_TWCR	SPI2 Transmitted Word Count Reload Register	0x00000000
0x31030030	SPI2_IMSK	SPI2 Interrupt Mask Register	0x00000000
0x31030034	SPI2_IMSK_CLR	SPI2 Interrupt Mask Clear Register	0x00000000
0x31030038	SPI2_IMSK_SET	SPI2 Interrupt Mask Set Register	0x00000000
0x31030040	SPI2_STAT	SPI2 Status Register	0x00440001
0x31030044	SPI2_ILAT	SPI2 Masked Interrupt Condition Register	0x00000000
0x31030048	SPI2_ILAT_CLR	SPI2 Masked Interrupt Clear Register	0x00000000
0x31030050	SPI2_RFIFO	SPI2 Receive FIFO Data Register	0x00000000
0x31030058	SPI2_TFIFO	SPI2 Transmit FIFO Data Register	0x00000000
0x31030060	SPI2_MMRDH	SPI2 Memory Mapped Read Header	0x00000000
0x31030064	SPI2_MMTOP	SPI2 SPI Memory Top Address	0x00000000

Table A-156: ADSP-2184x SPI5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31033004	SPI5_CTL	SPI5 Control Register	0x00000050

Table A-156: ADSP-2184x SPI5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31033008	SPI5_RXCTL	SPI5 Receive Control Register	0x00000000
0x3103300C	SPI5_TXCTL	SPI5 Transmit Control Register	0x00000000
0x31033010	SPI5_CLK	SPI5 Clock Rate Register	0x00000000
0x31033014	SPI5_DLY	SPI5 Delay Register	0x00000301
0x31033018	SPI5_TSEL	SPI5 Target Select Register	0x0000FE00
0x3103301C	SPI5_RWC	SPI5 Received Word Count Register	0x00000000
0x31033020	SPI5_RWCR	SPI5 Received Word Count Reload Register	0x00000000
0x31033024	SPI5_TWC	SPI5 Transmitted Word Count Register	0x00000000
0x31033028	SPI5_TWCR	SPI5 Transmitted Word Count Reload Register	0x00000000
0x31033030	SPI5_IMSK	SPI5 Interrupt Mask Register	0x00000000
0x31033034	SPI5_IMSK_CLR	SPI5 Interrupt Mask Clear Register	0x00000000
0x31033038	SPI5_IMSK_SET	SPI5 Interrupt Mask Set Register	0x00000000
0x31033040	SPI5_STAT	SPI5 Status Register	0x00440001
0x31033044	SPI5_ILAT	SPI5 Masked Interrupt Condition Register	0x00000000
0x31033048	SPI5_ILAT_CLR	SPI5 Masked Interrupt Clear Register	0x00000000
0x31033050	SPI5_RFIFO	SPI5 Receive FIFO Data Register	0x00000000
0x31033058	SPI5_TFIFO	SPI5 Transmit FIFO Data Register	0x00000000
0x31033060	SPI5_MMRDH	SPI5 Memory Mapped Read Header	0x00000000
0x31033064	SPI5_MMTOP	SPI5 SPI Memory Top Address	0x00000000

Table A-157: ADSP-2184x SPORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002000	SPORT0_CTL_A	SPORT0 Half SPORT 'A' Control Register	0x00000000
0x31002004	SPORT0_DIV_A	SPORT0 Half SPORT 'A' Divisor Register	0x00000000
0x31002008	SPORT0_MCTL_A	SPORT0 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100200C	SPORT0_CS0_A	SPORT0 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002010	SPORT0_CS1_A	SPORT0 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002014	SPORT0_CS2_A	SPORT0 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000

Table A-157: ADSP-2184x SPORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002018	SPORT0_CS3_A	SPORT0 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002020	SPORT0_ERR_A	SPORT0 Half SPORT 'A' Error Register	0x00000000
0x31002024	SPORT0_MSTAT_A	SPORT0 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002028	SPORT0_CTL2_A	SPORT0 Half SPORT 'A' Control 2 Register	0x00000000
0x31002040	SPORT0_TXPRI_A	SPORT0 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002044	SPORT0_RXPRI_A	SPORT0 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002048	SPORT0_TXSEC_A	SPORT0 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100204C	SPORT0_RXSEC_A	SPORT0 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002080	SPORT0_CTL_B	SPORT0 Half SPORT 'B' Control Register	0x00000000
0x31002084	SPORT0_DIV_B	SPORT0 Half SPORT 'B' Divisor Register	0x00000000
0x31002088	SPORT0_MCTL_B	SPORT0 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100208C	SPORT0_CS0_B	SPORT0 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002090	SPORT0_CS1_B	SPORT0 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002094	SPORT0_CS2_B	SPORT0 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002098	SPORT0_CS3_B	SPORT0 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310020A0	SPORT0_ERR_B	SPORT0 Half SPORT 'B' Error Register	0x00000000
0x310020A4	SPORT0_MSTAT_B	SPORT0 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310020A8	SPORT0_CTL2_B	SPORT0 Half SPORT 'B' Control 2 Register	0x00000000
0x310020C0	SPORT0_TXPRI_B	SPORT0 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310020C4	SPORT0_RXPRI_B	SPORT0 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310020C8	SPORT0_TXSEC_B	SPORT0 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310020CC	SPORT0_RXSEC_B	SPORT0 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-158: ADSP-2184x SPORT1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002100	SPORT1_CTL_A	SPORT1 Half SPORT 'A' Control Register	0x00000000
0x31002104	SPORT1_DIV_A	SPORT1 Half SPORT 'A' Divisor Register	0x00000000

Table A-158: ADSP-2184x SPORT1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002108	SPORT1_MCTL_A	SPORT1 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100210C	SPORT1_CS0_A	SPORT1 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002110	SPORT1_CS1_A	SPORT1 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002114	SPORT1_CS2_A	SPORT1 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002118	SPORT1_CS3_A	SPORT1 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002120	SPORT1_ERR_A	SPORT1 Half SPORT 'A' Error Register	0x00000000
0x31002124	SPORT1_MSTAT_A	SPORT1 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002128	SPORT1_CTL2_A	SPORT1 Half SPORT 'A' Control 2 Register	0x00000000
0x31002140	SPORT1_TXPRI_A	SPORT1 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002144	SPORT1_RXPRI_A	SPORT1 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002148	SPORT1_TXSEC_A	SPORT1 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100214C	SPORT1_RXSEC_A	SPORT1 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002180	SPORT1_CTL_B	SPORT1 Half SPORT 'B' Control Register	0x00000000
0x31002184	SPORT1_DIV_B	SPORT1 Half SPORT 'B' Divisor Register	0x00000000
0x31002188	SPORT1_MCTL_B	SPORT1 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100218C	SPORT1_CS0_B	SPORT1 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002190	SPORT1_CS1_B	SPORT1 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002194	SPORT1_CS2_B	SPORT1 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002198	SPORT1_CS3_B	SPORT1 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310021A0	SPORT1_ERR_B	SPORT1 Half SPORT 'B' Error Register	0x00000000
0x310021A4	SPORT1_MSTAT_B	SPORT1 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310021A8	SPORT1_CTL2_B	SPORT1 Half SPORT 'B' Control 2 Register	0x00000000
0x310021C0	SPORT1_TXPRI_B	SPORT1 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310021C4	SPORT1_RXPRI_B	SPORT1 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310021C8	SPORT1_TXSEC_B	SPORT1 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000

Table A-158: ADSP-2184x SPORT1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310021CC	SPORT1_RXSEC_B	SPORT1 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-159: ADSP-2184x SPORT2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002200	SPORT2_CTL_A	SPORT2 Half SPORT 'A' Control Register	0x00000000
0x31002204	SPORT2_DIV_A	SPORT2 Half SPORT 'A' Divisor Register	0x00000000
0x31002208	SPORT2_MCTL_A	SPORT2 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100220C	SPORT2_CS0_A	SPORT2 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002210	SPORT2_CS1_A	SPORT2 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002214	SPORT2_CS2_A	SPORT2 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002218	SPORT2_CS3_A	SPORT2 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002220	SPORT2_ERR_A	SPORT2 Half SPORT 'A' Error Register	0x00000000
0x31002224	SPORT2_MSTAT_A	SPORT2 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002228	SPORT2_CTL2_A	SPORT2 Half SPORT 'A' Control 2 Register	0x00000000
0x31002240	SPORT2_TXPRI_A	SPORT2 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002244	SPORT2_RXPRI_A	SPORT2 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002248	SPORT2_TXSEC_A	SPORT2 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100224C	SPORT2_RXSEC_A	SPORT2 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002280	SPORT2_CTL_B	SPORT2 Half SPORT 'B' Control Register	0x00000000
0x31002284	SPORT2_DIV_B	SPORT2 Half SPORT 'B' Divisor Register	0x00000000
0x31002288	SPORT2_MCTL_B	SPORT2 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100228C	SPORT2_CS0_B	SPORT2 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002290	SPORT2_CS1_B	SPORT2 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002294	SPORT2_CS2_B	SPORT2 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002298	SPORT2_CS3_B	SPORT2 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000

Table A-159: ADSP-2184x SPORT2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310022A0	SPORT2_ERR_B	SPORT2 Half SPORT 'B' Error Register	0x00000000
0x310022A4	SPORT2_MSTAT_B	SPORT2 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310022A8	SPORT2_CTL2_B	SPORT2 Half SPORT 'B' Control 2 Register	0x00000000
0x310022C0	SPORT2_TXPRI_B	SPORT2 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310022C4	SPORT2_RXPRI_B	SPORT2 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310022C8	SPORT2_TXSEC_B	SPORT2 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310022CC	SPORT2_RXSEC_B	SPORT2 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-160: ADSP-2184x SPORT3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002300	SPORT3_CTL_A	SPORT3 Half SPORT 'A' Control Register	0x00000000
0x31002304	SPORT3_DIV_A	SPORT3 Half SPORT 'A' Divisor Register	0x00000000
0x31002308	SPORT3_MCTL_A	SPORT3 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100230C	SPORT3_CS0_A	SPORT3 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002310	SPORT3_CS1_A	SPORT3 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002314	SPORT3_CS2_A	SPORT3 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002318	SPORT3_CS3_A	SPORT3 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002320	SPORT3_ERR_A	SPORT3 Half SPORT 'A' Error Register	0x00000000
0x31002324	SPORT3_MSTAT_A	SPORT3 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002328	SPORT3_CTL2_A	SPORT3 Half SPORT 'A' Control 2 Register	0x00000000
0x31002340	SPORT3_TXPRI_A	SPORT3 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002344	SPORT3_RXPRI_A	SPORT3 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002348	SPORT3_TXSEC_A	SPORT3 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100234C	SPORT3_RXSEC_A	SPORT3 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002380	SPORT3_CTL_B	SPORT3 Half SPORT 'B' Control Register	0x00000000
0x31002384	SPORT3_DIV_B	SPORT3 Half SPORT 'B' Divisor Register	0x00000000
0x31002388	SPORT3_MCTL_B	SPORT3 Half SPORT 'B' Multichannel Control Register	0x00000000

Table A-160: ADSP-2184x SPORT3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100238C	SPORT3_CS0_B	SPORT3 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002390	SPORT3_CS1_B	SPORT3 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002394	SPORT3_CS2_B	SPORT3 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002398	SPORT3_CS3_B	SPORT3 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310023A0	SPORT3_ERR_B	SPORT3 Half SPORT 'B' Error Register	0x00000000
0x310023A4	SPORT3_MSTAT_B	SPORT3 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310023A8	SPORT3_CTL2_B	SPORT3 Half SPORT 'B' Control 2 Register	0x00000000
0x310023C0	SPORT3_TXPRI_B	SPORT3 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310023C4	SPORT3_RXPRI_B	SPORT3 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310023C8	SPORT3_TXSEC_B	SPORT3 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310023CC	SPORT3_RXSEC_B	SPORT3 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-161: ADSP-2184x SPORT4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002400	SPORT4_CTL_A	SPORT4 Half SPORT 'A' Control Register	0x00000000
0x31002404	SPORT4_DIV_A	SPORT4 Half SPORT 'A' Divisor Register	0x00000000
0x31002408	SPORT4_MCTL_A	SPORT4 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100240C	SPORT4_CS0_A	SPORT4 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002410	SPORT4_CS1_A	SPORT4 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002414	SPORT4_CS2_A	SPORT4 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002418	SPORT4_CS3_A	SPORT4 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002420	SPORT4_ERR_A	SPORT4 Half SPORT 'A' Error Register	0x00000000
0x31002424	SPORT4_MSTAT_A	SPORT4 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002428	SPORT4_CTL2_A	SPORT4 Half SPORT 'A' Control 2 Register	0x00000000
0x31002440	SPORT4_TXPRI_A	SPORT4 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000

Table A-161: ADSP-2184x SPORT4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002444	SPORT4_RXPRI_A	SPORT4 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002448	SPORT4_TXSEC_A	SPORT4 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100244C	SPORT4_RXSEC_A	SPORT4 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002480	SPORT4_CTL_B	SPORT4 Half SPORT 'B' Control Register	0x00000000
0x31002484	SPORT4_DIV_B	SPORT4 Half SPORT 'B' Divisor Register	0x00000000
0x31002488	SPORT4_MCTL_B	SPORT4 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100248C	SPORT4_CS0_B	SPORT4 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002490	SPORT4_CS1_B	SPORT4 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002494	SPORT4_CS2_B	SPORT4 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002498	SPORT4_CS3_B	SPORT4 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310024A0	SPORT4_ERR_B	SPORT4 Half SPORT 'B' Error Register	0x00000000
0x310024A4	SPORT4_MSTAT_B	SPORT4 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310024A8	SPORT4_CTL2_B	SPORT4 Half SPORT 'B' Control 2 Register	0x00000000
0x310024C0	SPORT4_TXPRI_B	SPORT4 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310024C4	SPORT4_RXPRI_B	SPORT4 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310024C8	SPORT4_TXSEC_B	SPORT4 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310024CC	SPORT4_RXSEC_B	SPORT4 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-162: ADSP-2184x SPORT5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002500	SPORT5_CTL_A	SPORT5 Half SPORT 'A' Control Register	0x00000000
0x31002504	SPORT5_DIV_A	SPORT5 Half SPORT 'A' Divisor Register	0x00000000
0x31002508	SPORT5_MCTL_A	SPORT5 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100250C	SPORT5_CS0_A	SPORT5 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002510	SPORT5_CS1_A	SPORT5 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002514	SPORT5_CS2_A	SPORT5 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000

Table A-162: ADSP-2184x SPORT5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002518	SPORT5_CS3_A	SPORT5 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002520	SPORT5_ERR_A	SPORT5 Half SPORT 'A' Error Register	0x00000000
0x31002524	SPORT5_MSTAT_A	SPORT5 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002528	SPORT5_CTL2_A	SPORT5 Half SPORT 'A' Control 2 Register	0x00000000
0x31002540	SPORT5_TXPRI_A	SPORT5 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002544	SPORT5_RXPRI_A	SPORT5 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002548	SPORT5_TXSEC_A	SPORT5 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100254C	SPORT5_RXSEC_A	SPORT5 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002580	SPORT5_CTL_B	SPORT5 Half SPORT 'B' Control Register	0x00000000
0x31002584	SPORT5_DIV_B	SPORT5 Half SPORT 'B' Divisor Register	0x00000000
0x31002588	SPORT5_MCTL_B	SPORT5 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100258C	SPORT5_CS0_B	SPORT5 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002590	SPORT5_CS1_B	SPORT5 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002594	SPORT5_CS2_B	SPORT5 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002598	SPORT5_CS3_B	SPORT5 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310025A0	SPORT5_ERR_B	SPORT5 Half SPORT 'B' Error Register	0x00000000
0x310025A4	SPORT5_MSTAT_B	SPORT5 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310025A8	SPORT5_CTL2_B	SPORT5 Half SPORT 'B' Control 2 Register	0x00000000
0x310025C0	SPORT5_TXPRI_B	SPORT5 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310025C4	SPORT5_RXPRI_B	SPORT5 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310025C8	SPORT5_TXSEC_B	SPORT5 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310025CC	SPORT5_RXSEC_B	SPORT5 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-163: ADSP-2184x SPORT6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002600	SPORT6_CTL_A	SPORT6 Half SPORT 'A' Control Register	0x00000000
0x31002604	SPORT6_DIV_A	SPORT6 Half SPORT 'A' Divisor Register	0x00000000

Table A-163: ADSP-2184x SPORT6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002608	SPORT6_MCTL_A	SPORT6 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100260C	SPORT6_CS0_A	SPORT6 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002610	SPORT6_CS1_A	SPORT6 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002614	SPORT6_CS2_A	SPORT6 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002618	SPORT6_CS3_A	SPORT6 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002620	SPORT6_ERR_A	SPORT6 Half SPORT 'A' Error Register	0x00000000
0x31002624	SPORT6_MSTAT_A	SPORT6 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002628	SPORT6_CTL2_A	SPORT6 Half SPORT 'A' Control 2 Register	0x00000000
0x31002640	SPORT6_TXPRI_A	SPORT6 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002644	SPORT6_RXPRI_A	SPORT6 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002648	SPORT6_TXSEC_A	SPORT6 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100264C	SPORT6_RXSEC_A	SPORT6 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002680	SPORT6_CTL_B	SPORT6 Half SPORT 'B' Control Register	0x00000000
0x31002684	SPORT6_DIV_B	SPORT6 Half SPORT 'B' Divisor Register	0x00000000
0x31002688	SPORT6_MCTL_B	SPORT6 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100268C	SPORT6_CS0_B	SPORT6 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002690	SPORT6_CS1_B	SPORT6 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002694	SPORT6_CS2_B	SPORT6 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002698	SPORT6_CS3_B	SPORT6 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310026A0	SPORT6_ERR_B	SPORT6 Half SPORT 'B' Error Register	0x00000000
0x310026A4	SPORT6_MSTAT_B	SPORT6 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310026A8	SPORT6_CTL2_B	SPORT6 Half SPORT 'B' Control 2 Register	0x00000000
0x310026C0	SPORT6_TXPRI_B	SPORT6 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310026C4	SPORT6_RXPRI_B	SPORT6 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310026C8	SPORT6_TXSEC_B	SPORT6 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000

Table A-163: ADSP-2184x SPORT6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310026CC	SPORT6_RXSEC_B	SPORT6 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-164: ADSP-2184x SPORT7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002700	SPORT7_CTL_A	SPORT7 Half SPORT 'A' Control Register	0x00000000
0x31002704	SPORT7_DIV_A	SPORT7 Half SPORT 'A' Divisor Register	0x00000000
0x31002708	SPORT7_MCTL_A	SPORT7 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100270C	SPORT7_CS0_A	SPORT7 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002710	SPORT7_CS1_A	SPORT7 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002714	SPORT7_CS2_A	SPORT7 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002718	SPORT7_CS3_A	SPORT7 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002720	SPORT7_ERR_A	SPORT7 Half SPORT 'A' Error Register	0x00000000
0x31002724	SPORT7_MSTAT_A	SPORT7 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002728	SPORT7_CTL2_A	SPORT7 Half SPORT 'A' Control 2 Register	0x00000000
0x31002740	SPORT7_TXPRI_A	SPORT7 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002744	SPORT7_RXPRI_A	SPORT7 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002748	SPORT7_TXSEC_A	SPORT7 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100274C	SPORT7_RXSEC_A	SPORT7 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002780	SPORT7_CTL_B	SPORT7 Half SPORT 'B' Control Register	0x00000000
0x31002784	SPORT7_DIV_B	SPORT7 Half SPORT 'B' Divisor Register	0x00000000
0x31002788	SPORT7_MCTL_B	SPORT7 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100278C	SPORT7_CS0_B	SPORT7 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002790	SPORT7_CS1_B	SPORT7 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002794	SPORT7_CS2_B	SPORT7 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002798	SPORT7_CS3_B	SPORT7 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000

Table A-164: ADSP-2184x SPORT7 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310027A0	SPORT7_ERR_B	SPORT7 Half SPORT 'B' Error Register	0x00000000
0x310027A4	SPORT7_MSTAT_B	SPORT7 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310027A8	SPORT7_CTL2_B	SPORT7 Half SPORT 'B' Control 2 Register	0x00000000
0x310027C0	SPORT7_TXPRI_B	SPORT7 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310027C4	SPORT7_RXPRI_B	SPORT7 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310027C8	SPORT7_TXSEC_B	SPORT7 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310027CC	SPORT7_RXSEC_B	SPORT7 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B000	SPU0_CTL	SPU0 Control Register	0x000000AD
0x3108B004	SPU0_STAT	SPU0 Status Register	0x00000000
0x3108B400	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B404	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B408	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B40C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B410	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B414	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B418	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B41C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B420	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B424	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B428	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B42C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B430	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B434	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B438	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B43C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B440	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B444	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B448	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B44C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B450	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B454	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B458	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B45C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B460	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B464	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B468	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B46C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B470	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B474	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B478	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B47C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B480	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B484	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B488	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B48C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B490	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B494	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B498	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B49C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4A0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4A4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4A8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4AC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4B0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4B4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4B8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4BC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4C0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B4C4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4C8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4CC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4D0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4D4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4D8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4DC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4E0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4E4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4E8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4EC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4F0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4F4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4F8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4FC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B500	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B504	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B508	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B50C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B510	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B514	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B518	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B51C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B520	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B524	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B528	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B52C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B530	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B534	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B538	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B53C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B540	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B544	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B548	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B54C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B550	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B554	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B558	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B55C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B560	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B564	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B568	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B56C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B570	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B574	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B578	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B57C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B580	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B584	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B588	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B58C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B590	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B594	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B598	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B59C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5A0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5A4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5A8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5AC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5B0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5B4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5B8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B5BC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5C0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5C4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5C8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5CC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5D0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5D4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5D8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5DC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5E0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5E4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5E8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5EC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5F0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5F4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5F8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5FC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B600	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B604	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B608	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B60C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B610	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B614	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B618	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B61C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B620	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B624	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B628	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B62C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B630	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B634	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B638	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B63C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B640	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B644	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B648	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B64C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B650	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B654	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B658	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B65C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B660	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B664	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B668	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B66C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B670	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B674	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B678	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B67C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B680	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B684	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B688	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B68C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B690	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B694	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B698	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B69C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6A0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6A4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6A8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6AC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6B0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B6B4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6B8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6BC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6C0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6C4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6C8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6CC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6D0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6D4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6D8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6DC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6E0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6E4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6E8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6EC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6F0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6F4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6F8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6FC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B700	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B704	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B708	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B70C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B710	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B714	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B718	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B71C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B720	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B724	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B728	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B72C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B730	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B734	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B738	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B73C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B740	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B744	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B748	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B74C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B750	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B840	SPU0_SECURECTL	SPU0 Secure Control Register	0x00000000
0x3108B84C	SPU0_SECURECHK	SPU0 Secure Check Register	0xFFFFFFFF
0x3108B980	SPU0_SECUREC[n]	SPU0 Secure Core Registers	0x00000001
0x3108B984	SPU0_SECUREC[n]	SPU0 Secure Core Registers	0x00000001
0x3108B988	SPU0_SECUREC[n]	SPU0 Secure Core Registers	0x00000001
0x3108BA00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BA44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA58	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA5C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA60	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA64	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA68	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA6C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA70	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA74	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA78	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA7C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA80	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA84	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA88	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA8C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA90	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA94	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA98	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA9C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAA0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAA4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAA8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAAC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAB0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAB4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAB8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BABC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BAC0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAC4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAC8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BACC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAD0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAD4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAD8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BADC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAE0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAE4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAE8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAEC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAF0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAF4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAF8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAFC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BB3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB58	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB5C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB60	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB64	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB68	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB6C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB70	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB74	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB78	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB7C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB80	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB84	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB88	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB8C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB90	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB94	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB98	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB9C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBA0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBA4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBA8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBAC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBB0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBB4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BBB8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBBC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBC0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBC4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBC8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBCC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBD0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBD4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBD8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBDC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBE0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBE4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBE8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBEC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBF0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBF4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBF8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBFC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BC34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC58	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC5C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC60	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC64	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC68	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC6C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC70	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC74	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC78	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC7C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC80	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC84	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC88	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC8C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC90	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC94	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC98	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC9C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCA0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCA4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCA8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCAC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BCB0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCB4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCB8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCBC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCC0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCC4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCC8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCCC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCD0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCD4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCD8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCDC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCE0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCE4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCE8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCEC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCF0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCF4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCF8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCFC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-165: ADSP-2184x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BD2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-166: ADSP-2184x SWU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31015000	SWU0_GCTL	SWU0 Global Control Register	0x00000000
0x31015004	SWU0_GSTAT	SWU0 Global Status Register	0x00000000
0x31015010	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015014	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015018	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101501C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015020	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015024	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015028	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101502C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000
0x31015030	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015034	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015038	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101503C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015040	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015044	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015048	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000

Table A-166: ADSP-2184x SWU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101504C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000
0x31015050	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015054	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015058	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101505C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015060	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015064	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015068	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101506C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000
0x31015070	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015074	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015078	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101507C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015080	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015084	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015088	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101508C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000

Table A-167: ADSP-2184x SWU1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31093000	SWU1_GCTL	SWU1 Global Control Register	0x00000000
0x31093004	SWU1_GSTAT	SWU1 Global Status Register	0x00000000
0x31093010	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093014	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093018	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109301C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093020	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093024	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093028	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109302C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000

Table A-167: ADSP-2184x SWU1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31093030	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093034	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093038	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109303C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093040	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093044	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093048	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109304C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000
0x31093050	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093054	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093058	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109305C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093060	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093064	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093068	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109306C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000
0x31093070	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093074	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093078	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109307C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093080	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093084	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093088	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109308C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000

Table A-168: ADSP-2184x SWU11 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31097000	SWU11_GCTL	SWU11 Global Control Register	0x00000000
0x31097004	SWU11_GSTAT	SWU11 Global Status Register	0x00000000
0x31097010	SWU11_CTL[n]	SWU11 Control Register n	0x00000000

Table A-168: ADSP-2184x SWU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31097014	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097018	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109701C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097020	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097024	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097028	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109702C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000
0x31097030	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097034	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097038	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109703C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097040	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097044	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097048	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109704C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000
0x31097050	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097054	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097058	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109705C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097060	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097064	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097068	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109706C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000
0x31097070	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097074	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097078	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109707C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097080	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097084	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097088	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109708C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000

Table A-169: ADSP-2184x SWU12 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A8000	SWU12_GCTL	SWU12 Global Control Register	0x00000000
0x310A8004	SWU12_GSTAT	SWU12 Global Status Register	0x00000000
0x310A8010	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8014	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8018	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A801C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8020	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8024	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8028	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A802C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000
0x310A8030	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8034	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8038	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A803C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8040	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8044	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8048	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A804C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000
0x310A8050	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8054	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8058	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A805C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8060	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8064	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8068	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A806C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000
0x310A8070	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8074	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8078	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A807C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8080	SWU12_CNT[n]	SWU12 Count Register n	0x00000000

Table A-169: ADSP-2184x SWU12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A8084	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8088	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A808C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000

Table A-170: ADSP-2184x SWU13 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109E000	SWU13_GCTL	SWU13 Global Control Register	0x00000000
0x3109E004	SWU13_GSTAT	SWU13 Global Status Register	0x00000000
0x3109E010	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E014	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E018	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E01C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E020	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E024	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E028	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E02C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000
0x3109E030	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E034	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E038	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E03C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E040	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E044	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E048	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E04C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000
0x3109E050	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E054	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E058	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E05C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E060	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E064	SWU13_TARG[n]	SWU13 Target Register n	0x00000000

Table A-170: ADSP-2184x SWU13 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109E068	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E06C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000
0x3109E070	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E074	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E078	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E07C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E080	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E084	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E088	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E08C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000

Table A-171: ADSP-2184x SWU2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31094000	SWU2_GCTL	SWU2 Global Control Register	0x00000000
0x31094004	SWU2_GSTAT	SWU2 Global Status Register	0x00000000
0x31094010	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094014	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094018	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109401C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094020	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094024	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094028	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109402C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000
0x31094030	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094034	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094038	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109403C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094040	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094044	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094048	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000

Table A-171: ADSP-2184x SWU2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109404C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000
0x31094050	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094054	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094058	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109405C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094060	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094064	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094068	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109406C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000
0x31094070	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094074	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094078	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109407C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094080	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094084	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094088	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109408C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000

Table A-172: ADSP-2184x SWU3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31095000	SWU3_GCTL	SWU3 Global Control Register	0x00000000
0x31095004	SWU3_GSTAT	SWU3 Global Status Register	0x00000000
0x31095010	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095014	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095018	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109501C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095020	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095024	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095028	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109502C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000

Table A-172: ADSP-2184x SWU3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31095030	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095034	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095038	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109503C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095040	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095044	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095048	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109504C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000
0x31095050	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095054	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095058	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109505C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095060	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095064	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095068	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109506C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000
0x31095070	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095074	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095078	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109507C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095080	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095084	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095088	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109508C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000

Table A-173: ADSP-2184x SWU4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31096000	SWU4_GCTL	SWU4 Global Control Register	0x00000000
0x31096004	SWU4_GSTAT	SWU4 Global Status Register	0x00000000
0x31096010	SWU4_CTL[n]	SWU4 Control Register n	0x00000000

Table A-173: ADSP-2184x SWU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31096014	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096018	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109601C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096020	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096024	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096028	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109602C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000
0x31096030	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096034	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096038	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109603C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096040	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096044	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096048	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109604C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000
0x31096050	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096054	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096058	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109605C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096060	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096064	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096068	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109606C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000
0x31096070	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096074	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096078	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109607C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096080	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096084	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096088	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109608C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000

Table A-174: ADSP-2184x SWU5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31098000	SWU5_GCTL	SWU5 Global Control Register	0x00000000
0x31098004	SWU5_GSTAT	SWU5 Global Status Register	0x00000000
0x31098010	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098014	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098018	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109801C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098020	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098024	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098028	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109802C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000
0x31098030	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098034	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098038	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109803C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098040	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098044	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098048	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109804C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000
0x31098050	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098054	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098058	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109805C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098060	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098064	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098068	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109806C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000
0x31098070	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098074	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098078	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109807C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098080	SWU5_CNT[n]	SWU5 Count Register n	0x00000000

Table A-174: ADSP-2184x SWU5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31098084	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098088	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109808C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000

Table A-175: ADSP-2184x SWU7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31140000	SWU7_GCTL	SWU7 Global Control Register	0x00000000
0x31140004	SWU7_GSTAT	SWU7 Global Status Register	0x00000000
0x31140010	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140014	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140018	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114001C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140020	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140024	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140028	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114002C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000
0x31140030	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140034	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140038	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114003C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140040	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140044	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140048	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114004C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000
0x31140050	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140054	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140058	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114005C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140060	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140064	SWU7_TARG[n]	SWU7 Target Register n	0x00000000

Table A-175: ADSP-2184x SWU7 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31140068	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114006C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000
0x31140070	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140074	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140078	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114007C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140080	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140084	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140088	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114008C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000

Table A-176: ADSP-2184x SWU8 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31141000	SWU8_GCTL	SWU8 Global Control Register	0x00000000
0x31141004	SWU8_GSTAT	SWU8 Global Status Register	0x00000000
0x31141010	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141014	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141018	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114101C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141020	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141024	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141028	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114102C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000
0x31141030	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141034	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141038	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114103C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141040	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141044	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141048	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000

Table A-176: ADSP-2184x SWU8 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114104C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000
0x31141050	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141054	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141058	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114105C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141060	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141064	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141068	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114106C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000
0x31141070	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141074	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141078	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114107C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141080	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141084	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141088	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114108C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000

Table A-177: ADSP-2184x TAPC MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31130000	TAPC_IDCODE	TAPC IDCODE Register	0x028390CB
0x31130004	TAPC_USERCODE	TAPC USERCODE Register	0x00000000
0x31130008	TAPC_SDBGKEY_CTL	TAPC Secure Debug Key Control Register	0x00000000
0x3113000C	TAPC_SDBGKEY_STAT	TAPC Secure Debug Key Status Register	0x00000000
0x31130010	TAPC_SDBGKEY0	TAPC Secure Debug Key 0 Register	0x00000000
0x31130014	TAPC_SDBGKEY1	TAPC Secure Debug Key 1 Register	0x00000000
0x31130018	TAPC_SDBGKEY2	TAPC Secure Debug Key 2 Register	0x00000000
0x3113001C	TAPC_SDBGKEY3	TAPC Secure Debug Key 3 Register	0x00000000
0x31130020	TAPC_SDBGKEY4	TAPC Secure Debug Key 4 Register	0x00000000
0x31130024	TAPC_SDBGKEY5	TAPC Secure Debug Key 5 Register	0x00000000

Table A-177: ADSP-2184x TAPC MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31130028	TAPC_SDBGKEY6	TAPC Secure Debug Key 6 Register	0x00000000
0x3113002C	TAPC_SDBGKEY7	TAPC Secure Debug Key 7 Register	0x00000000
0x31131000	TAPC_DBGCTL	TAPC Debug Control Register	0x0000001E

Table A-178: ADSP-2184x TIMER0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31018004	TIMER0_RUN	TIMER0 Run Register	0x00000000
0x31018008	TIMER0_RUN_SET	TIMER0 Run Set Register	0x00000000
0x3101800C	TIMER0_RUN_CLR	TIMER0 Run Clear Register	0x00000000
0x31018010	TIMER0_STOP_CFG	TIMER0 Stop Configuration Register	0x00000000
0x31018014	TIMER0_STOP_CFG_SET	TIMER0 Stop Configuration Set Register	0x00000000
0x31018018	TIMER0_STOP_CFG_CLR	TIMER0 Stop Configuration Clear Register	0x00000000
0x3101801C	TIMER0_DATA_IMSK	TIMER0 Data Interrupt Mask Register	0x0000FFFF
0x31018020	TIMER0_STAT_IMSK	TIMER0 Status Interrupt Mask Register	0x0000FFFF
0x31018024	TIMER0_TRG_MSK	TIMER0 Trigger Generator Mask Register	0x0000FFFF
0x31018028	TIMER0_TRG_IE	TIMER0 Trigger Receiver Enable Register	0x00000000
0x3101802C	TIMER0_DATA_ILAT	TIMER0 Data Interrupt Latch Register	0x00000000
0x31018030	TIMER0_STAT_ILAT	TIMER0 Status Interrupt Latch Register	0x00000000
0x31018034	TIMER0_ERR_TYPE	TIMER0 Error Type Status Register	0x00000000
0x31018038	TIMER0_BCAST_PER	TIMER0 Broadcast Period Register	0x00000000
0x3101803C	TIMER0_BCAST_WID	TIMER0 Broadcast Width Register	0x00000000
0x31018040	TIMER0_BCAST_DLY	TIMER0 Broadcast Delay Register	0x00000000
0x31018060	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018064	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018068	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101806C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018070	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018080	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018084	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018088	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000

Table A-178: ADSP-2184x TIMER0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101808C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018090	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310180A0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310180A4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310180A8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310180AC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310180B0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310180C0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310180C4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310180C8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310180CC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310180D0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310180E0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310180E4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310180E8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310180EC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310180F0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018100	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018104	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018108	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101810C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018110	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018120	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018124	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018128	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101812C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018130	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018140	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018144	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018148	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101814C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000

Table A-178: ADSP-2184x TIMER0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31018150	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018160	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018164	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018168	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101816C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018170	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018180	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018184	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018188	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101818C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018190	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310181A0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310181A4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310181A8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310181AC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310181B0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310181C0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310181C4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310181C8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310181CC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310181D0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310181E0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310181E4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310181E8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310181EC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310181F0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018200	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018204	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018208	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101820C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018210	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000

Table A-178: ADSP-2184x TIMER0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31018220	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018224	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018228	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101822C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018230	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018240	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018244	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018248	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101824C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018250	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000

Table A-179: ADSP-2184x TMU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31016800	TMU0_CTL	TMU0 TMU Control Register	0x00050000
0x31016804	TMU0_TEMP	TMU0 Temperature Value Register	0x00000000
0x31016808	TMU0_AVG	TMU0 Averaging Register	0x00000041
0x3101680C	TMU0_FLT_LIM_HI	TMU0 Fault High Limit Register	0x000000FF
0x31016810	TMU0_ALRT_LIM_HI	TMU0 Alert High Limit Register	0x0FF3FCFF
0x31016814	TMU0_FLT_LIM_LO	TMU0 Fault Low Limit Register	0x00000100
0x31016818	TMU0_ALRT_LIM_LO	TMU0 Alert Low Limit Register	0x00000100
0x3101681C	TMU0_STAT	TMU0 Status Register	0x00000000
0x31016824	TMU0_GAIN	TMU0 Gain Value Register	0x00000000
0x31016828	TMU0_IMSK	TMU0 Interrupt Mask Register	0x0000003F
0x3101682C	TMU0_OFFSET	TMU0 Offset Register	0x00000000
0x31016834	TMU0_CNV_BLANK	TMU0 Temperature Conversion Blank Register	0x01401404
0x31016838	TMU0_REFR_CNTR	TMU0 Temperature Refresh Counter	0x00000000

Table A-180: ADSP-2184x TRNG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310D0000	TRNG0_INPUT[n]	TRNG0 TRNG Input Registers	0x00000000

Table A-180: ADSP-2184x TRNG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310D0000	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D0004	TRNG0_INPUT[n]	TRNG0 TRNG Input Registers	0x00000000
0x310D0004	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D0008	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D000C	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D0010	TRNG0_STAT	TRNG0 TRNG Status Register	0x00000000
0x310D0010	TRNG0_INTACK	TRNG0 TRNG Interrupt Acknowledge Register	0x00000000
0x310D0014	TRNG0_CTL	TRNG0 TRNG Control Register	0x00000000
0x310D0018	TRNG0_CFG	TRNG0 TRNG Configuration Register	0x00000000
0x310D001C	TRNG0_ALMCNT	TRNG0 TRNG Alarm Counter Register	0x000000FF
0x310D0020	TRNG0_FROEN	TRNG0 TRNG FRO Enable Register	0x000000FF
0x310D0024	TRNG0_FRODETUNE	TRNG0 TRNG FRO De-tune Register	0x00000000
0x310D0028	TRNG0_ALMMSK	TRNG0 TRNG Alarm Mask Register	0x00000000
0x310D002C	TRNG0_ALMSTP	TRNG0 TRNG Alarm Stop Register	0x00000000
0x310D0030	TRNG0_LFSR_L	TRNG0 TRNG LFSR Access Register	0x00000000
0x310D0034	TRNG0_LFSR_M	TRNG0 TRNG LFSR Access Register	0x00000000
0x310D0038	TRNG0_LFSR_H	TRNG0 TRNG LFSR Access Register	0x00000000
0x310D003C	TRNG0_CNT	TRNG0 Counter Register	0x00000000
0x310D0040	TRNG0_RUNCNT	TRNG0 TRNG Run Count Registers	0x00000000
0x310D0040	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0044	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0044	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0048	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0048	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D004C	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D004C	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0050	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0050	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0054	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0054	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0058	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000

Table A-180: ADSP-2184x TRNG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310D005C	TRNG0_MONOBITCNT	TRNG0 TRNG Monobit Test Result Register	0x00002710
0x310D0060	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D0060	TRNG0_V[n]	TRNG0 TRNG Post-Process "V" Value Registers	0x00000000
0x310D0064	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D0064	TRNG0_V[n]	TRNG0 TRNG Post-Process "V" Value Registers	0x00000000
0x310D0068	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D006C	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D0070	TRNG0_TEST	TRNG0 TRNG Test Register	0x00000000
0x310D0074	TRNG0_BLKCNT	TRNG0 TRNG Block Count Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A000	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A004	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A008	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A00C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A010	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A014	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A018	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A01C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A020	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A024	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A028	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A02C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A030	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A034	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A038	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A03C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A040	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A044	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A048	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A04C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A050	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A054	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A058	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A05C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A060	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A064	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A068	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A06C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A070	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A074	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A078	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A07C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A080	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A084	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A088	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A08C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A090	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A094	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A098	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A09C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0A0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0A4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0A8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0AC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0B0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0B4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0B8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0BC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0C0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A0C4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0C8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0CC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0D0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0D4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0D8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0DC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0E0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0E4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0E8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0EC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0F0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0F4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0F8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0FC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A100	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A104	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A108	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A10C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A110	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A114	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A118	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A11C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A120	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A124	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A128	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A12C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A130	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A134	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A138	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A13C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A140	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A144	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A148	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A14C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A150	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A154	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A158	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A15C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A160	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A164	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A168	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A16C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A170	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A174	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A178	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A17C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A180	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A184	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A188	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A18C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A190	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A194	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A198	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A19C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1A0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1A4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1A8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1AC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1B0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1B4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1B8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A1BC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1C0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1C4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1C8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1CC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1D0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1D4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1D8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1DC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1E0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1E4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1E8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1EC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1F0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1F4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1F8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1FC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A200	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A204	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A208	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A20C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A210	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A214	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A218	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A21C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A220	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A224	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A228	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A22C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A230	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A234	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A238	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A23C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A240	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A244	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A248	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A24C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A250	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A254	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A258	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A25C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A260	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A264	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A268	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A26C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A270	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A274	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A278	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A27C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A280	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A284	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A288	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A28C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A290	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A294	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A298	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A29C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2A0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2A4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2A8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2AC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2B0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A2B4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2B8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2BC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2C0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2C4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2C8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2CC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2D0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2D4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2D8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2DC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2E0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2E4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2E8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2EC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2F0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2F4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2F8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2FC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A300	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A304	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A308	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A30C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A310	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A314	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A318	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A31C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A320	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A324	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A328	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A32C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A330	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A334	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A338	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A33C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A340	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A344	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A348	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A34C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A350	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A354	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A358	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A35C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A360	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A364	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A368	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A36C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A370	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A374	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A378	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A37C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A380	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A384	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A388	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A38C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A390	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A394	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A398	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A39C	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3A0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3A4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3A8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-181: ADSP-2184x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A3AC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3B0	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3B4	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3B8	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A3BC	TRU0_RSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A7E0	TRU0_GEN	TRU0 Generator Trigger Register	0x00000000
0x3108A7E8	TRU0_ERRADDR	TRU0 Error Address Register	0x00000000
0x3108A7EC	TRU0_STAT	TRU0 Status Information Register	0x00000000
0x3108A7F4	TRU0_GCTL	TRU0 Global Control Register	0x00000000

Table A-182: ADSP-2184x TWI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001400	TWI0_CLKDIV	TWI0 SCL Clock Divider Register	0x00000000
0x31001404	TWI0_CTL	TWI0 Control Register	0x00000000
0x31001408	TWI0_TGTCTL	TWI0 Target Mode Control Register	0x00000000
0x3100140C	TWI0_TGTSTAT	TWI0 Target Mode Status Register	0x00000000
0x31001410	TWI0_TGTADDR	TWI0 Target Mode Address Register	0x00000000
0x31001414	TWI0_CTLRCTL	TWI0 Controller Mode Control Registers	0x00000000
0x31001418	TWI0_CTLRSTAT	TWI0 Controller Mode Status Register	0x00000000
0x3100141C	TWI0_CTLRADDR	TWI0 Controller Mode Address Register	0x00000000
0x31001420	TWI0_ISTAT	TWI0 Interrupt Status Register	0x00000000
0x31001424	TWI0_IMSK	TWI0 Interrupt Mask Register	0x00000000
0x31001428	TWI0_FIFOCTL	TWI0 FIFO Control Register	0x00000000
0x3100142C	TWI0_FIFOSTAT	TWI0 FIFO Status Register	0x00000000
0x31001480	TWI0_TXDATA8	TWI0 Tx Data Single-Byte Register	0x00000000
0x31001484	TWI0_TXDATA16	TWI0 Tx Data Double-Byte Register	0x00000000
0x31001488	TWI0_RXDATA8	TWI0 Rx Data Single-Byte Register	0x00000000
0x3100148C	TWI0_RXDATA16	TWI0 Rx Data Double-Byte Register	0x00000000

Table A-183: ADSP-2184x TWI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001500	TWI1_CLKDIV	TWI1 SCL Clock Divider Register	0x00000000
0x31001504	TWI1_CTL	TWI1 Control Register	0x00000000
0x31001508	TWI1_TGTCTL	TWI1 Target Mode Control Register	0x00000000
0x3100150C	TWI1_TGTSTAT	TWI1 Target Mode Status Register	0x00000000
0x31001510	TWI1_TGTADDR	TWI1 Target Mode Address Register	0x00000000
0x31001514	TWI1_CTLRCTL	TWI1 Controller Mode Control Registers	0x00000000
0x31001518	TWI1_CTLRSTAT	TWI1 Controller Mode Status Register	0x00000000
0x3100151C	TWI1_CTLRADDR	TWI1 Controller Mode Address Register	0x00000000
0x31001520	TWI1_ISTAT	TWI1 Interrupt Status Register	0x00000000
0x31001524	TWI1_IMSK	TWI1 Interrupt Mask Register	0x00000000
0x31001528	TWI1_FIFOCTL	TWI1 FIFO Control Register	0x00000000
0x3100152C	TWI1_FIFOSTAT	TWI1 FIFO Status Register	0x00000000
0x31001580	TWI1_TXDATA8	TWI1 Tx Data Single-Byte Register	0x00000000
0x31001584	TWI1_TXDATA16	TWI1 Tx Data Double-Byte Register	0x00000000
0x31001588	TWI1_RXDATA8	TWI1 Rx Data Single-Byte Register	0x00000000
0x3100158C	TWI1_RXDATA16	TWI1 Rx Data Double-Byte Register	0x00000000

Table A-184: ADSP-2184x TWI2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001600	TWI2_CLKDIV	TWI2 SCL Clock Divider Register	0x00000000
0x31001604	TWI2_CTL	TWI2 Control Register	0x00000000
0x31001608	TWI2_TGTCTL	TWI2 Target Mode Control Register	0x00000000
0x3100160C	TWI2_TGTSTAT	TWI2 Target Mode Status Register	0x00000000
0x31001610	TWI2_TGTADDR	TWI2 Target Mode Address Register	0x00000000
0x31001614	TWI2_CTLRCTL	TWI2 Controller Mode Control Registers	0x00000000
0x31001618	TWI2_CTLRSTAT	TWI2 Controller Mode Status Register	0x00000000
0x3100161C	TWI2_CTLRADDR	TWI2 Controller Mode Address Register	0x00000000
0x31001620	TWI2_ISTAT	TWI2 Interrupt Status Register	0x00000000
0x31001624	TWI2_IMSK	TWI2 Interrupt Mask Register	0x00000000
0x31001628	TWI2_FIFOCTL	TWI2 FIFO Control Register	0x00000000

Table A-184: ADSP-2184x TWI2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100162C	TWI2_FIFOSTAT	TWI2 FIFO Status Register	0x00000000
0x31001680	TWI2_TXDATA8	TWI2 Tx Data Single-Byte Register	0x00000000
0x31001684	TWI2_TXDATA16	TWI2 Tx Data Double-Byte Register	0x00000000
0x31001688	TWI2_RXDATA8	TWI2 Rx Data Single-Byte Register	0x00000000
0x3100168C	TWI2_RXDATA16	TWI2 Rx Data Double-Byte Register	0x00000000

Table A-185: ADSP-2184x TWI3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001000	TWI3_CLKDIV	TWI3 SCL Clock Divider Register	0x00000000
0x31001004	TWI3_CTL	TWI3 Control Register	0x00000000
0x31001008	TWI3_TGTCTL	TWI3 Target Mode Control Register	0x00000000
0x3100100C	TWI3_TGTSTAT	TWI3 Target Mode Status Register	0x00000000
0x31001010	TWI3_TGTADDR	TWI3 Target Mode Address Register	0x00000000
0x31001014	TWI3_CTLRCTL	TWI3 Controller Mode Control Registers	0x00000000
0x31001018	TWI3_CTLRSTAT	TWI3 Controller Mode Status Register	0x00000000
0x3100101C	TWI3_CTLRADDR	TWI3 Controller Mode Address Register	0x00000000
0x31001020	TWI3_ISTAT	TWI3 Interrupt Status Register	0x00000000
0x31001024	TWI3_IMSK	TWI3 Interrupt Mask Register	0x00000000
0x31001028	TWI3_FIFOCTL	TWI3 FIFO Control Register	0x00000000
0x3100102C	TWI3_FIFOSTAT	TWI3 FIFO Status Register	0x00000000
0x31001080	TWI3_TXDATA8	TWI3 Tx Data Single-Byte Register	0x00000000
0x31001084	TWI3_TXDATA16	TWI3 Tx Data Double-Byte Register	0x00000000
0x31001088	TWI3_RXDATA8	TWI3 Rx Data Single-Byte Register	0x00000000
0x3100108C	TWI3_RXDATA16	TWI3 Rx Data Double-Byte Register	0x00000000

Table A-186: ADSP-2184x TWI4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001100	TWI4_CLKDIV	TWI4 SCL Clock Divider Register	0x00000000
0x31001104	TWI4_CTL	TWI4 Control Register	0x00000000
0x31001108	TWI4_TGTCTL	TWI4 Target Mode Control Register	0x00000000

Table A-186: ADSP-2184x TWI4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100110C	TWI4_TGTSTAT	TWI4 Target Mode Status Register	0x00000000
0x31001110	TWI4_TGTADDR	TWI4 Target Mode Address Register	0x00000000
0x31001114	TWI4_CTLRCTL	TWI4 Controller Mode Control Registers	0x00000000
0x31001118	TWI4_CTLRSTAT	TWI4 Controller Mode Status Register	0x00000000
0x3100111C	TWI4_CTLRADDR	TWI4 Controller Mode Address Register	0x00000000
0x31001120	TWI4_ISTAT	TWI4 Interrupt Status Register	0x00000000
0x31001124	TWI4_IMSK	TWI4 Interrupt Mask Register	0x00000000
0x31001128	TWI4_FIFOCTL	TWI4 FIFO Control Register	0x00000000
0x3100112C	TWI4_FIFOSTAT	TWI4 FIFO Status Register	0x00000000
0x31001180	TWI4_TXDATA8	TWI4 Tx Data Single-Byte Register	0x00000000
0x31001184	TWI4_TXDATA16	TWI4 Tx Data Double-Byte Register	0x00000000
0x31001188	TWI4_RXDATA8	TWI4 Rx Data Single-Byte Register	0x00000000
0x3100118C	TWI4_RXDATA16	TWI4 Rx Data Double-Byte Register	0x00000000

Table A-187: ADSP-2184x TWI5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001200	TWI5_CLKDIV	TWI5 SCL Clock Divider Register	0x00000000
0x31001204	TWI5_CTL	TWI5 Control Register	0x00000000
0x31001208	TWI5_TGTCTL	TWI5 Target Mode Control Register	0x00000000
0x3100120C	TWI5_TGTSTAT	TWI5 Target Mode Status Register	0x00000000
0x31001210	TWI5_TGTADDR	TWI5 Target Mode Address Register	0x00000000
0x31001214	TWI5_CTLRCTL	TWI5 Controller Mode Control Registers	0x00000000
0x31001218	TWI5_CTLRSTAT	TWI5 Controller Mode Status Register	0x00000000
0x3100121C	TWI5_CTLRADDR	TWI5 Controller Mode Address Register	0x00000000
0x31001220	TWI5_ISTAT	TWI5 Interrupt Status Register	0x00000000
0x31001224	TWI5_IMSK	TWI5 Interrupt Mask Register	0x00000000
0x31001228	TWI5_FIFOCTL	TWI5 FIFO Control Register	0x00000000
0x3100122C	TWI5_FIFOSTAT	TWI5 FIFO Status Register	0x00000000
0x31001280	TWI5_TXDATA8	TWI5 Tx Data Single-Byte Register	0x00000000
0x31001284	TWI5_TXDATA16	TWI5 Tx Data Double-Byte Register	0x00000000

Table A-187: ADSP-2184x TWI5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31001288	TWI5_RXDATA8	TWI5 Rx Data Single-Byte Register	0x00000000
0x3100128C	TWI5_RXDATA16	TWI5 Rx Data Double-Byte Register	0x00000000

Table A-188: ADSP-2184x UART0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003004	UART0_CTL	UART0 Control Register	0x00000000
0x31003008	UART0_STAT	UART0 Status Register	0x000000A0
0x3100300C	UART0_SCR	UART0 Scratch Register	0x00000000
0x31003010	UART0_CLK	UART0 Clock Rate Register	0x0000FFFF
0x31003014	UART0_IMSK	UART0 Interrupt Mask Register	0x00000000
0x31003018	UART0_IMSK_SET	UART0 Interrupt Mask Set Register	0x00000000
0x3100301C	UART0_IMSK_CLR	UART0 Interrupt Mask Clear Register	0x00000000
0x31003020	UART0_RBR	UART0 Receive Buffer Register	0x00000000
0x31003024	UART0_THR	UART0 Transmit Hold Register	0x00000000
0x31003028	UART0_TAIP	UART0 Transmit Address/Insert Pulse Register	0x00000000
0x3100302C	UART0_TSR	UART0 Transmit Shift Register	0x000007FF
0x31003030	UART0_RSR	UART0 Receive Shift Register	0x00000000
0x31003034	UART0_TXCNT	UART0 Transmit Counter Register	0x00000000
0x31003038	UART0_RXCNT	UART0 Receive Counter Register	0x00000000

Table A-189: ADSP-2184x UART1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003404	UART1_CTL	UART1 Control Register	0x00000000
0x31003408	UART1_STAT	UART1 Status Register	0x000000A0
0x3100340C	UART1_SCR	UART1 Scratch Register	0x00000000
0x31003410	UART1_CLK	UART1 Clock Rate Register	0x0000FFFF
0x31003414	UART1_IMSK	UART1 Interrupt Mask Register	0x00000000
0x31003418	UART1_IMSK_SET	UART1 Interrupt Mask Set Register	0x00000000
0x3100341C	UART1_IMSK_CLR	UART1 Interrupt Mask Clear Register	0x00000000
0x31003420	UART1_RBR	UART1 Receive Buffer Register	0x00000000

Table A-189: ADSP-2184x UART1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31003424	UART1_THR	UART1 Transmit Hold Register	0x00000000
0x31003428	UART1_TAIP	UART1 Transmit Address/Insert Pulse Register	0x00000000
0x3100342C	UART1_TSR	UART1 Transmit Shift Register	0x000007FF
0x31003430	UART1_RSR	UART1 Receive Shift Register	0x00000000
0x31003434	UART1_TXCNT	UART1 Transmit Counter Register	0x00000000
0x31003438	UART1_RXCNT	UART1 Receive Counter Register	0x00000000

Table A-190: ADSP-2184x UART2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003804	UART2_CTL	UART2 Control Register	0x00000000
0x31003808	UART2_STAT	UART2 Status Register	0x000000A0
0x3100380C	UART2_SCR	UART2 Scratch Register	0x00000000
0x31003810	UART2_CLK	UART2 Clock Rate Register	0x0000FFFF
0x31003814	UART2_IMSK	UART2 Interrupt Mask Register	0x00000000
0x31003818	UART2_IMSK_SET	UART2 Interrupt Mask Set Register	0x00000000
0x3100381C	UART2_IMSK_CLR	UART2 Interrupt Mask Clear Register	0x00000000
0x31003820	UART2_RBR	UART2 Receive Buffer Register	0x00000000
0x31003824	UART2_THR	UART2 Transmit Hold Register	0x00000000
0x31003828	UART2_TAIP	UART2 Transmit Address/Insert Pulse Register	0x00000000
0x3100382C	UART2_TSR	UART2 Transmit Shift Register	0x000007FF
0x31003830	UART2_RSR	UART2 Receive Shift Register	0x00000000
0x31003834	UART2_TXCNT	UART2 Transmit Counter Register	0x00000000
0x31003838	UART2_RXCNT	UART2 Receive Counter Register	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31300000	USB_OTG_CTL	USB OTG Control and Status Register	0x00010000
0x31300004	USB_OTG_IRQ	USB OTG Interrupt Register	0x00000000
0x31300008	USB_AHB_CFG	USB Bus Configuration Register	0x00000000
0x3130000C	USB_CFG	USB USB Configuration Register	0x00001410

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31300010	USB_RST_CTL	USB Reset Register	0x80000000
0x31300014	USB_ISTAT	USB Interrupt Status Register	0x04000020
0x31300018	USB_IMSK	USB Interrupt Mask Register	0x00000000
0x3130001C	USB_RXDBG_STAT	USB Receive Status Debug Read Register	0x00000000
0x31300020	USB_RXDATA_STAT	USB Receive Status Read/Pop Register	0x00000000
0x31300024	USB_RXFIFOSZ	USB Receive FIFO Size Register	0x00000800
0x31300028	USB_TXFIFOSZ_NP	USB Non-periodic Transmit FIFO Size Register	0x08000800
0x3130002C	USB_TXFIFO_STAT_NP	USB Non-periodic Transmit FIFO/Queue Status Register	0x00080800
0x31300034	USB_PHYIF_CTL	USB PHY Interface Control Register	0x00000000
0x31300040	USB_MODID	USB Module ID Register	0x4F54400A
0x31300044	USB_HWCFG1	USB User Hardware Configuration 1 Register	0x00000000
0x31300048	USB_HWCFG2	USB User Hardware Configuration 2 Register	0x228BEC90
0x3130004C	USB_HWCFG3	USB User Hardware Configuration 3 Register	0x0FA006E8
0x31300050	USB_HWCFG4	USB User Hardware Configuration 4 Register	0xCE002020
0x3130005C	USB_DFIFO_CFG	USB DFIFO Configuration Register	0x0FA01000
0x31300100	USB_TXFIFOSZ_PER_H	USB Host Periodic Transmit FIFO Size Register	0x04001000
0x31300104	USB_TXFIFOSZ1_IEP_D	USB Device IN Endpoint 1 Transmit FIFO Size Register	0x01000900
0x31300108	USB_TXFIFOSZ2_IEP_D	USB Device IN Endpoint 2 Transmit FIFO Size Register	0x01000A00
0x3130010C	USB_TXFIFOSZ3_IEP_D	USB Device IN Endpoint 3 Transmit FIFO Size Register	0x01000B00
0x31300400	USB_CFG_H	USB Host Configuration Register	0x00000200
0x31300404	USB_FIR_H	USB Host Frame Interval Register	0x0000EA60
0x31300408	USB_FNUM_H	USB Host Frame Number/Frame Time Remaining Register	0x00003FFF
0x31300414	USB_ISTAT_HA	USB Host All Channels Interrupt Register	0x00000000
0x31300418	USB_IMSK_HA	USB Host All Channels Interrupt Mask Register	0x00000000
0x3130041C	USB_FL_BADDR_H	USB Host Frame List Base Address Register	0x00000000
0x31300440	USB_PORT_CTL_H	USB Host Port Control and Status Register	0x00000000
0x31300500	USB_HCCHAR0	USB	0x00000000
0x31300504	USB_HCSPLT0	USB	0x00000000
0x31300508	USB_HCINT0	USB Host Channel Interrupt Register	0x00000000
0x3130050C	USB_HCINTMSK0	USB Host Channel $i Interrupt Mask Register	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31300510	USB_HCTSIZ0	USB	0x00000000
0x31300514	USB_HCDMA0	USB	0x00000000
0x3130051C	USB_HCDMAB0	USB	0x00000000
0x31300520	USB_HCCHAR1	USB	0x00000000
0x31300524	USB_HCSPLT1	USB	0x00000000
0x31300528	USB_HCINT1	USB	0x00000000
0x3130052C	USB_HCINTMSK1	USB	0x00000000
0x31300530	USB_HCTSIZ1	USB	0x00000000
0x31300534	USB_HCDMA1	USB	0x00000000
0x3130053C	USB_HCDMAB1	USB	0x00000000
0x31300540	USB_HCCHAR2	USB	0x00000000
0x31300544	USB_HCSPLT2	USB	0x00000000
0x31300548	USB_HCINT2	USB	0x00000000
0x3130054C	USB_HCINTMSK2	USB	0x00000000
0x31300550	USB_HCTSIZ2	USB	0x00000000
0x31300554	USB_HCDMA2	USB	0x00000000
0x3130055C	USB_HCDMAB2	USB	0x00000000
0x31300560	USB_HCCHAR3	USB	0x00000000
0x31300564	USB_HCSPLT3	USB	0x00000000
0x31300568	USB_HCINT3	USB	0x00000000
0x3130056C	USB_HCINTMSK3	USB	0x00000000
0x31300570	USB_HCTSIZ3	USB	0x00000000
0x31300574	USB_HCDMA3	USB	0x00000000
0x3130057C	USB_HCDMAB3	USB	0x00000000
0x31300580	USB_HCCHAR4	USB	0x00000000
0x31300584	USB_HCSPLT4	USB	0x00000000
0x31300588	USB_HCINT4	USB	0x00000000
0x3130058C	USB_HCINTMSK4	USB	0x00000000
0x31300590	USB_HCTSIZ4	USB	0x00000000
0x31300594	USB_HCDMA4	USB	0x00000000
0x3130059C	USB_HCDMAB4	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x313005A0	USB_HCCHAR5	USB	0x00000000
0x313005A4	USB_HCSPLT5	USB	0x00000000
0x313005A8	USB_HCINT5	USB	0x00000000
0x313005AC	USB_HCINTMSK5	USB	0x00000000
0x313005B0	USB_HCTSIZ5	USB	0x00000000
0x313005B4	USB_HCDMA5	USB	0x00000000
0x313005BC	USB_HCDMAB5	USB	0x00000000
0x313005C0	USB_HCCHAR6	USB	0x00000000
0x313005C4	USB_HCSPLT6	USB	0x00000000
0x313005C8	USB_HCINT6	USB	0x00000000
0x313005CC	USB_HCINTMSK6	USB	0x00000000
0x313005D0	USB_HCTSIZ6	USB	0x00000000
0x313005D4	USB_HCDMA6	USB	0x00000000
0x313005DC	USB_HCDMAB6	USB	0x00000000
0x313005E0	USB_HCCHAR7	USB	0x00000000
0x313005E4	USB_HCSPLT7	USB	0x00000000
0x313005E8	USB_HCINT7	USB	0x00000000
0x313005EC	USB_HCINTMSK7	USB	0x00000000
0x313005F0	USB_HCTSIZ7	USB	0x00000000
0x313005F4	USB_HCDMA7	USB	0x00000000
0x313005FC	USB_HCDMAB7	USB	0x00000000
0x31300600	USB_HCCHAR8	USB	0x00000000
0x31300604	USB_HCSPLT8	USB	0x00000000
0x31300608	USB_HCINT8	USB	0x00000000
0x3130060C	USB_HCINTMSK8	USB	0x00000000
0x31300610	USB_HCTSIZ8	USB	0x00000000
0x31300614	USB_HCDMA8	USB	0x00000000
0x3130061C	USB_HCDMAB8	USB	0x00000000
0x31300620	USB_HCCHAR9	USB	0x00000000
0x31300624	USB_HCSPLT9	USB	0x00000000
0x31300628	USB_HCINT9	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3130062C	USB_HCINTMSK9	USB	0x00000000
0x31300630	USB_HCTSIZ9	USB	0x00000000
0x31300634	USB_HCDMA9	USB	0x00000000
0x3130063C	USB_HCDMAB9	USB	0x00000000
0x31300640	USB_HCCHAR10	USB	0x00000000
0x31300644	USB_HCSPLT10	USB	0x00000000
0x31300648	USB_HCINT10	USB	0x00000000
0x3130064C	USB_HCINTMSK10	USB	0x00000000
0x31300650	USB_HCTSIZ10	USB	0x00000000
0x31300654	USB_HCDMA10	USB	0x00000000
0x3130065C	USB_HCDMAB10	USB	0x00000000
0x31300660	USB_HCCHAR11	USB	0x00000000
0x31300664	USB_HCSPLT11	USB	0x00000000
0x31300668	USB_HCINT11	USB	0x00000000
0x3130066C	USB_HCINTMSK11	USB	0x00000000
0x31300670	USB_HCTSIZ11	USB	0x00000000
0x31300674	USB_HCDMA11	USB	0x00000000
0x3130067C	USB_HCDMAB11	USB	0x00000000
0x31300680	USB_HCCHAR12	USB	0x00000000
0x31300684	USB_HCSPLT12	USB	0x00000000
0x31300688	USB_HCINT12	USB	0x00000000
0x3130068C	USB_HCINTMSK12	USB	0x00000000
0x31300690	USB_HCTSIZ12	USB	0x00000000
0x31300694	USB_HCDMA12	USB	0x00000000
0x3130069C	USB_HCDMAB12	USB	0x00000000
0x313006A0	USB_HCCHAR13	USB	0x00000000
0x313006A4	USB_HCSPLT13	USB	0x00000000
0x313006A8	USB_HCINT13	USB	0x00000000
0x313006AC	USB_HCINTMSK13	USB	0x00000000
0x313006B0	USB_HCTSIZ13	USB	0x00000000
0x313006B4	USB_HCDMA13	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x313006BC	USB_HCDMAB13	USB	0x00000000
0x313006C0	USB_HCCHAR14	USB	0x00000000
0x313006C4	USB_HCSPLT14	USB	0x00000000
0x313006C8	USB_HCINT14	USB	0x00000000
0x313006CC	USB_HCINTMSK14	USB	0x00000000
0x313006D0	USB_HCTSIZ14	USB	0x00000000
0x313006D4	USB_HCDMA14	USB	0x00000000
0x313006DC	USB_HCDMAB14	USB	0x00000000
0x313006E0	USB_HCCHAR15	USB	0x00000000
0x313006E4	USB_HCSPLT15	USB	0x00000000
0x313006E8	USB_HCINT15	USB	0x00000000
0x313006EC	USB_HCINTMSK15	USB	0x00000000
0x313006F0	USB_HCTSIZ15	USB	0x00000000
0x313006F4	USB_HCDMA15	USB	0x00000000
0x313006FC	USB_HCDMAB15	USB	0x00000000
0x31300800	USB_CFG_D	USB Device Configuration Register	0x08100000
0x31300804	USB_CTL_D	USB Device Control Register	0x00000002
0x31300808	USB_STAT_D	USB Device Status Register	0x00000002
0x31300810	USB_IMASK_IEP_D	USB Device IN Endpoint Common Interrupt Mask Register	0x00000000
0x31300814	USB_IMASK_OEP_D	USB Device OUT Endpoint Common Interrupt Mask Register	0x00000000
0x31300818	USB_ISTAT_D	USB Device All Endpoint Interrupt Status Register	0x00000000
0x3130081C	USB_IMSK_D	USB Device All Endpoint Interrupt Mask Register	0x00000000
0x31300828	USB_VBUSDIS_D	USB Device VBUS Discharge Time Register	0x000017D7
0x3130082C	USB_VBUSPULSE_D	USB Device VBUS Pulsing Time Register	0x000005B8
0x31300830	USB_THR_CTL_D	USB Device Threshold Control Register	0x0C100020
0x31300834	USB_IMSK_IEP_FEMPT_D	USB Device IN Endpoint FIFO Empty Interrupt Mask Register	0x00000000
0x31300900	USB_DIEPCTL0	USB	0x00008000
0x31300908	USB_DIEPINT0	USB	0x00000080
0x31300910	USB_DIEPTSIZ0	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31300914	USB_DIEPDMA0	USB	0x00000000
0x31300918	USB_DTXFSTS0	USB	0x00000800
0x3130091C	USB_DIEPDMAB0	USB	0x00000000
0x31300920	USB_DIEPCTL1	USB	0x00000000
0x31300928	USB_DIEPINT1	USB	0x00000080
0x31300930	USB_DIEPTSIZ1	USB	0x00000000
0x31300934	USB_DIEPDMA1	USB	0x00000000
0x31300938	USB_DTXFSTS1	USB	0x00000800
0x3130093C	USB_DIEPDMAB1	USB	0x00000000
0x31300940	USB_DIEPCTL2	USB	0x00000000
0x31300948	USB_DIEPINT2	USB	0x00000080
0x31300950	USB_DIEPTSIZ2	USB	0x00000000
0x31300954	USB_DIEPDMA2	USB	0x00000000
0x31300958	USB_DTXFSTS2	USB	0x00000800
0x3130095C	USB_DIEPDMAB2	USB	0x00000000
0x31300960	USB_DIEPCTL3	USB	0x00000000
0x31300968	USB_DIEPINT3	USB	0x00000080
0x31300970	USB_DIEPTSIZ3	USB	0x00000000
0x31300974	USB_DIEPDMA3	USB	0x00000000
0x31300978	USB_DTXFSTS3	USB	0x00000800
0x3130097C	USB_DIEPDMAB3	USB	0x00000000
0x31300980	USB_DIEPCTL4	USB	0x00000000
0x31300988	USB_DIEPINT4	USB	0x00000080
0x31300990	USB_DIEPTSIZ4	USB	0x00000000
0x31300994	USB_DIEPDMA4	USB	0x00000000
0x31300998	USB_DTXFSTS4	USB	0x00000800
0x3130099C	USB_DIEPDMAB4	USB	0x00000000
0x313009A0	USB_DIEPCTL5	USB	0x00000000
0x313009A8	USB_DIEPINT5	USB	0x00000080
0x313009B0	USB_DIEPTSIZ5	USB	0x00000000
0x313009B4	USB_DIEPDMA5	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x313009B8	USB_DTXFSTS5	USB	0x00000800
0x313009BC	USB_DIEPDMAB5	USB	0x00000000
0x313009C0	USB_DIEPCTL6	USB	0x00000000
0x313009C8	USB_DIEPINT6	USB	0x00000080
0x313009D0	USB_DIEPTSIZ6	USB	0x00000000
0x313009D4	USB_DIEPDMA6	USB	0x00000000
0x313009D8	USB_DTXFSTS6	USB	0x00000800
0x313009DC	USB_DIEPDMAB6	USB	0x00000000
0x313009E0	USB_DIEPCTL7	USB	0x00000000
0x313009E8	USB_DIEPINT7	USB	0x00000080
0x313009F0	USB_DIEPTSIZ7	USB	0x00000000
0x313009F4	USB_DIEPDMA7	USB	0x00000000
0x313009F8	USB_DTXFSTS7	USB	0x00000800
0x313009FC	USB_DIEPDMAB7	USB	0x00000000
0x31300A00	USB_DIEPCTL8	USB	0x00000000
0x31300A08	USB_DIEPINT8	USB	0x00000080
0x31300A10	USB_DIEPTSIZ8	USB	0x00000000
0x31300A14	USB_DIEPDMA8	USB	0x00000000
0x31300A18	USB_DTXFSTS8	USB	0x00000800
0x31300A1C	USB_DIEPDMAB8	USB	0x00000000
0x31300A20	USB_DIEPCTL9	USB	0x00000000
0x31300A28	USB_DIEPINT9	USB	0x00000080
0x31300A30	USB_DIEPTSIZ9	USB	0x00000000
0x31300A34	USB_DIEPDMA9	USB	0x00000000
0x31300A38	USB_DTXFSTS9	USB	0x00000800
0x31300A3C	USB_DIEPDMAB9	USB	0x00000000
0x31300A40	USB_DIEPCTL10	USB	0x00000000
0x31300A48	USB_DIEPINT10	USB	0x00000080
0x31300A50	USB_DIEPTSIZ10	USB	0x00000000
0x31300A54	USB_DIEPDMA10	USB	0x00000000
0x31300A58	USB_DTXFSTS10	USB	0x00000800

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31300A5C	USB_DIEPDMAB10	USB	0x00000000
0x31300A60	USB_DIEPCTL11	USB	0x00000000
0x31300A68	USB_DIEPINT11	USB	0x00000080
0x31300A70	USB_DIEPTSIZ11	USB	0x00000000
0x31300A74	USB_DIEPDMA11	USB	0x00000000
0x31300A78	USB_DTXFSTS11	USB	0x00000800
0x31300A7C	USB_DIEPDMAB11	USB	0x00000000
0x31300B00	USB_DOEPCTL0	USB	0x00008000
0x31300B08	USB_DOEPINT0	USB	0x00000000
0x31300B10	USB_DOEPTSIZ0	USB	0x00000000
0x31300B14	USB_DOEPDMA0	USB	0x00000000
0x31300B1C	USB_DOEPDMAB0	USB	0x00000000
0x31300B20	USB_DOEPCTL1	USB	0x00000000
0x31300B28	USB_DOEPINT1	USB	0x00000000
0x31300B30	USB_DOEPTSIZ1	USB	0x00000000
0x31300B34	USB_DOEPDMA1	USB	0x00000000
0x31300B3C	USB_DOEPDMAB1	USB	0x00000000
0x31300B40	USB_DOEPCTL2	USB	0x00000000
0x31300B48	USB_DOEPINT2	USB	0x00000000
0x31300B50	USB_DOEPTSIZ2	USB	0x00000000
0x31300B54	USB_DOEPDMA2	USB	0x00000000
0x31300B5C	USB_DOEPDMAB2	USB	0x00000000
0x31300B60	USB_DOEPCTL3	USB	0x00000000
0x31300B68	USB_DOEPINT3	USB	0x00000000
0x31300B70	USB_DOEPTSIZ3	USB	0x00000000
0x31300B74	USB_DOEPDMA3	USB	0x00000000
0x31300B7C	USB_DOEPDMAB3	USB	0x00000000
0x31300B80	USB_DOEPCTL4	USB	0x00000000
0x31300B88	USB_DOEPINT4	USB	0x00000000
0x31300B90	USB_DOEPTSIZ4	USB	0x00000000
0x31300B94	USB_DOEPDMA4	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31300B9C	USB_DOEPDMAB4	USB	0x00000000
0x31300BA0	USB_DOEPCTL5	USB	0x00000000
0x31300BA8	USB_DOEPINT5	USB	0x00000000
0x31300BB0	USB_DOEPTSIZ5	USB	0x00000000
0x31300BB4	USB_DOEPDMA5	USB	0x00000000
0x31300BBC	USB_DOEPDMAB5	USB	0x00000000
0x31300BC0	USB_DOEPCTL6	USB	0x00000000
0x31300BC8	USB_DOEPINT6	USB	0x00000000
0x31300BD0	USB_DOEPTSIZ6	USB	0x00000000
0x31300BD4	USB_DOEPDMA6	USB	0x00000000
0x31300BDC	USB_DOEPDMAB6	USB	0x00000000
0x31300BE0	USB_DOEPCTL7	USB	0x00000000
0x31300BE8	USB_DOEPINT7	USB	0x00000000
0x31300BF0	USB_DOEPTSIZ7	USB	0x00000000
0x31300BF4	USB_DOEPDMA7	USB	0x00000000
0x31300BFC	USB_DOEPDMAB7	USB	0x00000000
0x31300C00	USB_DOEPCTL8	USB	0x00000000
0x31300C08	USB_DOEPINT8	USB	0x00000000
0x31300C10	USB_DOEPTSIZ8	USB	0x00000000
0x31300C14	USB_DOEPDMA8	USB	0x00000000
0x31300C1C	USB_DOEPDMAB8	USB	0x00000000
0x31300C20	USB_DOEPCTL9	USB	0x00000000
0x31300C28	USB_DOEPINT9	USB	0x00000000
0x31300C30	USB_DOEPTSIZ9	USB	0x00000000
0x31300C34	USB_DOEPDMA9	USB	0x00000000
0x31300C3C	USB_DOEPDMAB9	USB	0x00000000
0x31300C40	USB_DOEPCTL10	USB	0x00000000
0x31300C48	USB_DOEPINT10	USB	0x00000000
0x31300C50	USB_DOEPTSIZ10	USB	0x00000000
0x31300C54	USB_DOEPDMA10	USB	0x00000000
0x31300C5C	USB_DOEPDMAB10	USB	0x00000000

Table A-191: ADSP-2184x USB MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31300C60	USB_DOEPCTL11	USB	0x00000000
0x31300C68	USB_DOEPINT11	USB	0x00000000
0x31300C70	USB_DOEPTSIZ11	USB	0x00000000
0x31300C74	USB_DOEPDMA11	USB	0x00000000
0x31300C7C	USB_DOEPDMAB11	USB	0x00000000
0x31300E00	USB_PWR_CTL	USB Power and Clock Gating Control Register	0x00000000

Table A-192: ADSP-2184x WDOG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31008000	WDOG0_CTL	WDOG0 Control Register	0x00000AD0
0x31008004	WDOG0_CNT	WDOG0 Count Register	0x00000000
0x31008008	WDOG0_STAT	WDOG0 Watchdog Timer Status Register	0x00000000
0x3100800C	WDOG0_WIN	WDOG0 Watchdog Timer Window Register	0xFFFFFFFF

Table A-193: ADSP-2184x WDOG1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31008800	WDOG1_CTL	WDOG1 Control Register	0x00000AD0
0x31008804	WDOG1_CNT	WDOG1 Count Register	0x00000000
0x31008808	WDOG1_STAT	WDOG1 Watchdog Timer Status Register	0x00000000
0x3100880C	WDOG1_WIN	WDOG1 Watchdog Timer Window Register	0xFFFFFFFF

Table A-194: ADSP-2184x WDOG2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31009000	WDOG2_CTL	WDOG2 Control Register	0x00000AD0
0x31009004	WDOG2_CNT	WDOG2 Count Register	0x00000000
0x31009008	WDOG2_STAT	WDOG2 Watchdog Timer Status Register	0x00000000
0x3100900C	WDOG2_WIN	WDOG2 Watchdog Timer Window Register	0xFFFFFFFF

Table A-195: ADSP-2184x WDOG3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3100A000	WDOG3_CTL	WDOG3 Control Register	0x00000AD0
0x3100A004	WDOG3_CNT	WDOG3 Count Register	0x00000000
0x3100A008	WDOG3_STAT	WDOG3 Watchdog Timer Status Register	0x00000000
0x3100A00C	WDOG3_WIN	WDOG3 Watchdog Timer Window Register	0xFFFFFFFF

Table A-196: ADSP-2184x XMSHARCFX0XTSUBSYSTEM MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31102000	XMSHARCFX0XTSUBSYSTEM_TRAXID	XMSHARCFX0XTSUBSYSTEM	0x00436A00
0x31102004	XMSHARCFX0XTSUBSYSTEM_TRAXCTRL	XMSHARCFX0XTSUBSYSTEM	0xF8001082
0x31102008	XMSHARCFX0XTSUBSYSTEM_TRAXSTAT	XMSHARCFX0XTSUBSYSTEM	0x00000D00
0x3110200C	XMSHARCFX0XTSUBSYSTEM_TRAXDATA	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31102010	XMSHARCFX0XTSUBSYSTEM_TRAXADDR	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31102014	XMSHARCFX0XTSUBSYSTEM_TRIGGERPC	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31102018	XMSHARCFX0XTSUBSYSTEM_PCMATCHCTRL	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110201C	XMSHARCFX0XTSUBSYSTEM_DELAYCNT	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31102020	XMSHARCFX0XTSUBSYSTEM_MEMADDRSTART	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31102024	XMSHARCFX0XTSUBSYSTEM_MEMADDREND	XMSHARCFX0XTSUBSYSTEM	0x000007FF
0x31102040	XMSHARCFX0XTSUBSYSTEM_EXTTIMELO	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31102044	XMSHARCFX0XTSUBSYSTEM_EXTTIMEHI	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103000	XMSHARCFX0XTSUBSYSTEM_PMG	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103010	XMSHARCFX0XTSUBSYSTEM_INTPC	XMSHARCFX0XTSUBSYSTEM	0x00000000

Table A-196: ADSP-2184x XMSHARCFX0XTSUBSYSTEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31103080	XMSHARCFX0XTSUBSYSTEM_PM0	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103084	XMSHARCFX0XTSUBSYSTEM_PM1	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103088	XMSHARCFX0XTSUBSYSTEM_PM2	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110308C	XMSHARCFX0XTSUBSYSTEM_PM3	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103090	XMSHARCFX0XTSUBSYSTEM_PM4	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103094	XMSHARCFX0XTSUBSYSTEM_PM5	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103098	XMSHARCFX0XTSUBSYSTEM_PM6	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110309C	XMSHARCFX0XTSUBSYSTEM_PM7	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103100	XMSHARCFX0XTSUBSYSTEM_PMCTRL0	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103104	XMSHARCFX0XTSUBSYSTEM_PMCTRL1	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103108	XMSHARCFX0XTSUBSYSTEM_PMCTRL2	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110310C	XMSHARCFX0XTSUBSYSTEM_PMCTRL3	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103110	XMSHARCFX0XTSUBSYSTEM_PMCTRL4	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103114	XMSHARCFX0XTSUBSYSTEM_PMCTRL5	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103118	XMSHARCFX0XTSUBSYSTEM_PMCTRL6	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110311C	XMSHARCFX0XTSUBSYSTEM_PMCTRL7	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103180	XMSHARCFX0XTSUBSYSTEM_PMSTAT0	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103184	XMSHARCFX0XTSUBSYSTEM_PMSTAT1	XMSHARCFX0XTSUBSYSTEM	0x00000000

Table A-196: ADSP-2184x XMSHARCFX0XTSUBSYSTEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31103188	XMSHARCFX0XTSUBSYSTEM_PMSTAT2	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110318C	XMSHARCFX0XTSUBSYSTEM_PMSTAT3	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103190	XMSHARCFX0XTSUBSYSTEM_PMSTAT4	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103194	XMSHARCFX0XTSUBSYSTEM_PMSTAT5	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31103198	XMSHARCFX0XTSUBSYSTEM_PMSTAT6	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110319C	XMSHARCFX0XTSUBSYSTEM_PMSTAT7	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31104000	XMSHARCFX0XTSUBSYSTEM_OCDID	XMSHARCFX0XTSUBSYSTEM	0x00430602
0x31104008	XMSHARCFX0XTSUBSYSTEM_DCRCLR	XMSHARCFX0XTSUBSYSTEM	0x00600000
0x3110400C	XMSHARCFX0XTSUBSYSTEM_DCRSET	XMSHARCFX0XTSUBSYSTEM	0x00600000
0x31104010	XMSHARCFX0XTSUBSYSTEM_DSR	XMSHARCFX0XTSUBSYSTEM	0x80000000
0x31104014	XMSHARCFX0XTSUBSYSTEM_DDR	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31104018	XMSHARCFX0XTSUBSYSTEM_DDREXEC	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110401C	XMSHARCFX0XTSUBSYSTEM_DIR0EXEC	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31104020	XMSHARCFX0XTSUBSYSTEM_DIR0	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31104024	XMSHARCFX0XTSUBSYSTEM_DIR1	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31104028	XMSHARCFX0XTSUBSYSTEM_DIR2	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110402C	XMSHARCFX0XTSUBSYSTEM_DIR3	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31105020	XMSHARCFX0XTSUBSYSTEM_PWRCTL	XMSHARCFX0XTSUBSYSTEM	0x00000000

Table A-196: ADSP-2184x XMSHARCFX0XTSUBSYSTEM MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31105024	XMSHARCFX0XTSUBSYSTEM_PWRSTAT	XMSHARCFX0XTSUBSYSTEM	0x10011111
0x31105028	XMSHARCFX0XTSUBSYSTEM_ERISTAT	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x3110502C	XMSHARCFX0XTSUBSYSTEM_FAULTINFOLO	XMSHARCFX0XTSUBSYSTEM	0x00000000
0x31105030	XMSHARCFX0XTSUBSYSTEM_FAULTINFOHI	XMSHARCFX0XTSUBSYSTEM	0x00000000

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3101C000	XSPI0_CMD0	XSPI0 Command Register 0	0x00000000
0x3101C000	XSPI_CMD0	XSPI Command Register 0	0x00000000
0x3101C004	XSPI0_CMD1	XSPI0 Command Register 1	0x00000000
0x3101C004	XSPI_CMD1	XSPI Command Register 1	0x00000000
0x3101C008	XSPI0_CMD2	XSPI0 Command Register 2	0x00000000
0x3101C008	XSPI_CMD2	XSPI Command Register 2	0x00000000
0x3101C00C	XSPI0_CMD3	XSPI0 Command Register 3	0x00000000
0x3101C00C	XSPI_CMD3	XSPI Command Register 3	0x00000000
0x3101C010	XSPI0_CMD4	XSPI0 Command Register 4	0x00000000
0x3101C010	XSPI_CMD4	XSPI Command Register 4	0x00000000
0x3101C014	XSPI0_CMD5	XSPI0 Command Register 5	0x00000000
0x3101C014	XSPI_CMD5	XSPI Command Register 5	0x00000000
0x3101C040	XSPI0_CMD_STAT_PTR	XSPI0 Command Status Pointer Register	0x00000000
0x3101C040	XSPI_CMD_STAT_PTR	XSPI Command Status Pointer Register	0x00000000
0x3101C044	XSPI0_CMD_STAT	XSPI0 Command Status Register	0x00000000
0x3101C044	XSPI_CMD_STAT	XSPI Command Status Register	0x00000000
0x3101C100	XSPI0_GSTAT	XSPI0 General Controller Status Register	0x00000000
0x3101C100	XSPI_GSTAT	XSPI General Controller Status Register	0x00000000
0x3101C104	XSPI0_TRD_STAT	XSPI0 Auto Command Engine Thread Status Register	0x00000000
0x3101C104	XSPI_TRD_STAT	XSPI Auto Command Engine Thread Status Register	0x00000000
0x3101C110	XSPI0_ISTAT	XSPI0 Interrupt Status Register	0x00000000

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101C110	XSPI_ISTAT	XSPI Interrupt Status Register	0x00000000
0x3101C114	XSPI0_INT_EN	XSPI0 Interrupt Enable Register	0x00000000
0x3101C114	XSPI_INT_EN	XSPI Interrupt Enable Register	0x00000000
0x3101C120	XSPI0_TRD_COMP_ISTAT	XSPI0 Auto Command Engine Interrupt Status Thread Register	0x00000000
0x3101C120	XSPI_TRD_COMP_ISTAT	XSPI Auto Command Engine Interrupt Status Thread Register	0x00000000
0x3101C130	XSPI0_TRD_ERR_ISTAT	XSPI0 Thread Error Interrupt Status Register	0x00000000
0x3101C130	XSPI_TRD_ERR_ISTAT	XSPI Thread Error Interrupt Status Register	0x00000000
0x3101C134	XSPI0_TRD_ERR_INT_EN	XSPI0 Thread Error Interrupt Enable Register	0x00000000
0x3101C134	XSPI_TRD_ERR_INT_EN	XSPI Thread Error Interrupt Enable Register	0x00000000
0x3101C150	XSPI0_DMA_ERRADDR_LO	XSPI0 DMA Error Address Low Register	0x00000000
0x3101C150	XSPI_DMA_ERRADDR_LO	XSPI DMA Error Address Low Register	0x00000000
0x3101C154	XSPI0_DMA_ERRADDR_HI	XSPI0 DMA Error Address High Register	0x00000000
0x3101C154	XSPI_DMA_ERRADDR_HI	XSPI DMA Error Address High Register	0x00000000
0x3101C158	XSPI0_BOOT_STAT	XSPI0 Boot Status Register	0x00000000
0x3101C158	XSPI_BOOT_STAT	XSPI Boot Status Register	0x00000000
0x3101C208	XSPI0_LONGPOL_GCTL	XSPI0 Long Polling Count Register	0x000003E8
0x3101C208	XSPI_LONGPOL_GCTL	XSPI Long Polling Count Register	0x000003E8
0x3101C20C	XSPI0_SHORTPOL_GCTL	XSPI0 Short Polling Count Register	0x000001F4
0x3101C20C	XSPI_SHORTPOL_GCTL	XSPI Short Polling Count Register	0x000001F4
0x3101C230	XSPI0_WORKMODE_CTL	XSPI0 Device Control Register	0x00000000
0x3101C230	XSPI_WORKMODE_CTL	XSPI Device Control Register	0x00000000
0x3101C23C	XSPI0_DMA_CTL	XSPI0 DMA Interface Control Register	0x000D0000
0x3101C23C	XSPI_DMA_CTL	XSPI DMA Interface Control Register	0x000D0000
0x3101C240	XSPI0_SDMA_SIZ	XSPI0 Host DMA Block Size Register	0x00000000
0x3101C240	XSPI_SDMA_SIZ	XSPI Host DMA Block Size Register	0x00000000
0x3101C244	XSPI0_SDMA_TRD_STAT	XSPI0 Host DMA Thread Status Register	0x00000000
0x3101C244	XSPI_SDMA_TRD_STAT	XSPI Host DMA Thread Status Register	0x00000000
0x3101C24C	XSPI0_SDMA_ADDR0	XSPI0 Host DMA Buffer Address Register 0	0x00000000

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101C24C	XSPI_SDMA_ADDR0	XSPI Host DMA Buffer Address Register 0	0x00000000
0x3101C250	XSPI0_SDMA_ADDR1	XSPI0 Host DMA Buffer Address Register 1	0x00000000
0x3101C250	XSPI_SDMA_ADDR1	XSPI Host DMA Buffer Address Register 1	0x00000000
0x3101C260	XSPI0_DISCOVERY_GCTL	XSPI0 Device Discovery Control Register	0x00000000
0x3101C260	XSPI_DISCOVERY_GCTL	XSPI Device Discovery Control Register	0x00000000
0x3101C388	XSPI0_XIP_GCTL	XSPI0 XIP Configuration Register	0x00FF0000
0x3101C388	XSPI_XIP_GCTL	XSPI XIP Configuration Register	0x00FF0000
0x3101C390	XSPI0_SEQ_GCTL0	XSPI0 Sequence Configuration Register 0	0x0000208F
0x3101C390	XSPI_SEQ_GCTL0	XSPI Sequence Configuration Register 0	0x0000208F
0x3101C394	XSPI0_SEQ_GCTL1	XSPI0 Sequence Configuration Register 1	0x00000000
0x3101C394	XSPI_SEQ_GCTL1	XSPI Sequence Configuration Register 1	0x00000000
0x3101C398	XSPI0_DAC_CFG	XSPI0 DAC Configuration Register	0x00000000
0x3101C398	XSPI_DAC_CFG	XSPI DAC Configuration Register	0x00000000
0x3101C39C	XSPI0_DAC_REMAPADDR0	XSPI0 Address Remapping Register 0	0x00000000
0x3101C39C	XSPI_DAC_REMAPADDR0	XSPI Address Remapping Register 0	0x00000000
0x3101C3A0	XSPI0_DAC_REMAPADDR1	XSPI0 Address Remapping Register 1	0x00000000
0x3101C3A0	XSPI_DAC_REMAPADDR1	XSPI Address Remapping Register 1	0x00000000
0x3101C400	XSPI0_RST_SEQ_CFG0	XSPI0 Reset Sequence Configuration Register 0	0x00019966
0x3101C400	XSPI_RST_SEQ_CFG0	XSPI Reset Sequence Configuration Register 0	0x00019966
0x3101C404	XSPI0_RST_SEQ_CFG1	XSPI0 Reset Sequence Configuration Register 1	0xD0669900
0x3101C404	XSPI_RST_SEQ_CFG1	XSPI Reset Sequence Configuration Register 1	0xD0669900
0x3101C410	XSPI0_ERS_SEQ_CFG0	XSPI0 Erase Sequence Configuration Register 0	0x00DF3020
0x3101C410	XSPI_ERS_SEQ_CFG0	XSPI Erase Sequence Configuration Register 0	0x00DF3020
0x3101C414	XSPI0_ERS_SEQ_CFG1	XSPI0 Erase Sequence Configuration Register 1	0x0000000C
0x3101C414	XSPI_ERS_SEQ_CFG1	XSPI Erase Sequence Configuration Register 1	0x0000000C
0x3101C418	XSPI0_ERS_SEQ_CFG2	XSPI0 Erase Sequence Configuration Register 2	0x009F0060
0x3101C418	XSPI_ERS_SEQ_CFG2	XSPI Erase Sequence Configuration Register 2	0x009F0060
0x3101C420	XSPI0_PROG_SEQ_CFG0	XSPI0 Program Sequence Configuration Register 0	0x00003002
0x3101C420	XSPI_PROG_SEQ_CFG0	XSPI Program Sequence Configuration Register 0	0x00003002

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101C424	XSPI0_PROG_SEQ_CFG1	XSPI0 Program Sequence Configuration Register 1	0x0000FD00
0x3101C424	XSPI_PROG_SEQ_CFG1	XSPI Program Sequence Configuration Register 1	0x0000FD00
0x3101C428	XSPI0_PROG_SEQ_CFG2	XSPI0 Program Sequence Configuration Register 2	0x00000002
0x3101C428	XSPI_PROG_SEQ_CFG2	XSPI Program Sequence Configuration Register 2	0x00000002
0x3101C430	XSPI0_READ_SEQ_CFG0	XSPI0 Read Sequence Configuration Register 0	0x00003003
0x3101C430	XSPI_READ_SEQ_CFG0	XSPI Read Sequence Configuration Register 0	0x00003003
0x3101C434	XSPI0_READ_SEQ_CFG1	XSPI0 Read Sequence Configuration Register 1	0x0000FC00
0x3101C434	XSPI_READ_SEQ_CFG1	XSPI Read Sequence Configuration Register 1	0x0000FC00
0x3101C438	XSPI0_READ_SEQ_CFG2	XSPI0 Read Sequence Configuration Register 2	0x00000F0A
0x3101C438	XSPI_READ_SEQ_CFG2	XSPI Read Sequence Configuration Register 2	0x00000F0A
0x3101C440	XSPI0_WE_SEQ_CFG0	XSPI0 WEL Sequence Configuration Register	0x01F90006
0x3101C440	XSPI_WE_SEQ_CFG0	XSPI WEL Sequence Configuration Register	0x01F90006
0x3101C450	XSPI0_STAT_SEQ_CFG0	XSPI0 Status Checking Sequence Configuration Register 0	0x00000000
0x3101C450	XSPI_STAT_SEQ_CFG0	XSPI Status Checking Sequence Configuration Register 0	0x00000000
0x3101C454	XSPI0_STAT_SEQ_CFG1	XSPI0 Status Checking Sequence Configuration Register 1	0x00000000
0x3101C454	XSPI_STAT_SEQ_CFG1	XSPI Status Checking Sequence Configuration Register 1	0x00000000
0x3101C458	XSPI0_STAT_SEQ_CFG2	XSPI0 Status Checking Sequence Configuration Register 2	0x05000505
0x3101C458	XSPI_STAT_SEQ_CFG2	XSPI Status Checking Sequence Configuration Register 2	0x05000505
0x3101C45C	XSPI0_STAT_SEQ_CFG3	XSPI0 Status Checking Sequence Configuration Register 3	0xFA00FAFA
0x3101C45C	XSPI_STAT_SEQ_CFG3	XSPI Status Checking Sequence Configuration Register 3	0xFA00FAFA
0x3101C460	XSPI0_STAT_SEQ_CFG4	XSPI0 Status Checking Sequence Configuration Register 4	0x00000F00
0x3101C460	XSPI_STAT_SEQ_CFG4	XSPI Status Checking Sequence Configuration Register 4	0x00000F00
0x3101C464	XSPI0_STAT_SEQ_CFG5	XSPI0 Status Checking Sequence Configuration Register 5	0x00000040
0x3101C464	XSPI_STAT_SEQ_CFG5	XSPI Status Checking Sequence Configuration Register 5	0x00000040
0x3101C46C	XSPI0_STAT_SEQ_CFG7	XSPI0 Status Checking Sequence Configuration Register 7	0x00000000
0x3101C46C	XSPI_STAT_SEQ_CFG7	XSPI Status Checking Sequence Configuration Register 7	0x00000000
0x3101C470	XSPI0_STAT_SEQ_CFG8	XSPI0 Status Checking Sequence Configuration Register 8	0x00000000
0x3101C470	XSPI_STAT_SEQ_CFG8	XSPI Status Checking Sequence Configuration Register 8	0x00000000
0x3101C474	XSPI0_STAT_SEQ_CFG9	XSPI0 Status Checking Sequence Configuration Register 9	0x00000000
0x3101C474	XSPI_STAT_SEQ_CFG9	XSPI Status Checking Sequence Configuration Register 9	0x00000000

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101C478	XSPI0_STAT_SEQ_CFG10	XSPI0 Status Checking Sequence Configuration Register 10	0x00000000
0x3101C478	XSPI_STAT_SEQ_CFG10	XSPI Status Checking Sequence Configuration Register 10	0x00000000
0x3101CF00	XSPI0_REVID	XSPI0 Revision ID Register	0x65220206
0x3101CF00	XSPI_REVID	XSPI Revision ID Register	0x65220206
0x3101CF04	XSPI0_FEATURES	XSPI0 Controller Features Register	0x03710003
0x3101CF04	XSPI_FEATURES	XSPI Controller Features Register	0x03710003
0x3101D000	XSPI0_MINICTL_WP_CTL	XSPI0 Write Protect Register	0x00000001
0x3101D000	XSPI_MINICTL_WP_CTL	XSPI Write Protect Register	0x00000001
0x3101D004	XSPI0_MINICTL_RST_PIN_CTL	XSPI0 Hardware Reset Control Register	0x00000001
0x3101D004	XSPI_MINICTL_RST_PIN_CTL	XSPI Hardware Reset Control Register	0x00000001
0x3101D008	XSPI0_MINICTL_CLKMODE	XSPI0 Clock Mode Control Register	0x00000000
0x3101D008	XSPI_MINICTL_CLKMODE	XSPI Clock Mode Control Register	0x00000000
0x3101D00C	XSPI0_MINICTL_JEDEC_RST_TR	XSPI0 JEDEC Reset Delay Register	0x00008080
0x3101D00C	XSPI_MINICTL_JEDEC_RST_TR	XSPI JEDEC Reset Delay Register	0x00008080
0x3101D010	XSPI0_MINICTL_DEV_DLY	XSPI0 Device Delay Register	0x01000100
0x3101D010	XSPI_MINICTL_DEV_DLY	XSPI Device Delay Register	0x01000100
0x3101D014	XSPI0_MINICTL_RST_RECOV	XSPI0 Reset Recovery Delay Register	0x0000000A
0x3101D014	XSPI_MINICTL_RST_RECOV	XSPI Reset Recovery Delay Register	0x0000000A
0x3101D018	XSPI0_MINICTL_DEV_ACTIVE_MAX	XSPI0 Device Active Maximum Clock Cycle Register	0x00000080
0x3101D018	XSPI_MINICTL_DEV_ACTIVE_MAX	XSPI Device Active Maximum Clock Cycle Register	0x00000080
0x3101D020	XSPI0_MINICTL_HF_OFFSET	XSPI0 HyperFlash Offset Register	0x00000D03

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101D020	XSPI_MINICTL_HF_OFFSET	XSPI HyperFlash Offset Register	0x00000D03
0x3101D030	XSPI0_PHY_DLL_UPDT_CNT	XSPI0 PHY DLL Resynchronization Register	0x00000000
0x3101D030	XSPI_PHY_DLL_UPDT_CNT	XSPI PHY DLL Resynchronization Register	0x00000000
0x3101D034	XSPI0_PHY_DLL_CTL	XSPI0 PHY DLL Control Register	0x01030707
0x3101D034	XSPI_PHY_DLL_CTL	XSPI PHY DLL Control Register	0x01030707
0x3101E000	XSPI0_PHY_DQ_TR	XSPI0 PHY DQ Timing Register	0x00000002
0x3101E000	XSPI_PHY_DQ_TR	XSPI PHY DQ Timing Register	0x00000002
0x3101E004	XSPI0_PHY_DQS_TR	XSPI0 PHY DQS Timing Register	0x00100000
0x3101E004	XSPI_PHY_DQS_TR	XSPI PHY DQS Timing Register	0x00100000
0x3101E008	XSPI0_PHY_GATE_LPBK_CTL	XSPI0 PHY Gate Loopback Control Register	0x00D80000
0x3101E008	XSPI_PHY_GATE_LPBK_CTL	XSPI PHY Gate Loopback Control Register	0x00D80000
0x3101E00C	XSPI0_PHY_DLL_REQ_CTL	XSPI0 Bus Requester DLL Control Register	0x00800000
0x3101E00C	XSPI_PHY_DLL_REQ_CTL	XSPI Bus Requester DLL Control Register	0x00800000
0x3101E010	XSPI0_PHY_DLL_COMP_CTL	XSPI0 Bus Completer DLL Control Register	0x00000000
0x3101E010	XSPI_PHY_DLL_COMP_CTL	XSPI Bus Completer DLL Control Register	0x00000000
0x3101E014	XSPI0_PHY_IE_TR	XSPI0 PHY DQS Input Enable Timing Register	0x00100000
0x3101E014	XSPI_PHY_IE_TR	XSPI PHY DQS Input Enable Timing Register	0x00100000
0x3101E018	XSPI0_PHY_OB0	XSPI0 PHY Observable Points Register	0x00000000
0x3101E018	XSPI_PHY_OB0	XSPI PHY Observable Points Register	0x00000000
0x3101E01C	XSPI0_PHY_DLLOB0	XSPI0 PHY DLL Observable Points 0 Register	0x00000000
0x3101E01C	XSPI_PHY_DLLOB0	XSPI PHY DLL Observable Points 0 Register	0x00000000
0x3101E020	XSPI0_PHY_DLLOB1	XSPI0 PHY DLL Observable Points 1 Register	0x00000000
0x3101E020	XSPI_PHY_DLLOB1	XSPI PHY DLL Observable Points 1 Register	0x00000000
0x3101E028	XSPI0_PHY_STATIC_TGL	XSPI0 PHY Static Aging Register	0x00000000
0x3101E028	XSPI_PHY_STATIC_TGL	XSPI PHY Static Aging Register	0x00000000

Table A-197: ADSP-2184x XSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101E034	XSPI0_PHY_WR_DESKEW_PAD_CTL0	XSPI0 PHY Deskew Write Register	0x00000000
0x3101E034	XSPI_PHY_WR_DESKEW_PAD_CTL0	XSPI PHY Deskew Write Register	0x00000000
0x3101E070	XSPI0_PHY_REVID	XSPI0 PHY Revision ID Register	0x61820107
0x3101E070	XSPI_PHY_REVID	XSPI PHY Revision ID Register	0x61820107
0x3101E074	XSPI0_PHY_FEATURES	XSPI0 PHY Features Register	0x000019E8
0x3101E074	XSPI_PHY_FEATURES	XSPI PHY Features Register	0x000019E8
0x3101E080	XSPI0_PHY_GCTL	XSPI0 PHY Global Control Register	0x00000180
0x3101E080	XSPI_PHY_GCTL	XSPI PHY Global Control Register	0x00000180
0x3101E084	XSPI0_PHY_GTSEL	XSPI0 PHY Global Termination Control Register	0x00000000
0x3101E084	XSPI_PHY_GTSEL	XSPI PHY Global Termination Control Register	0x00000000
0x3101E088	XSPI0_PHY_GPIO_CTL0	XSPI0 PHY GPIO Control Register 0	0x00000000
0x3101E088	XSPI_PHY_GPIO_CTL0	XSPI PHY GPIO Control Register 0	0x00000000
0x3101E08C	XSPI0_PHY_GPIO_CTL1	XSPI0 PHY GPIO Control Register 1	0x00000000
0x3101E08C	XSPI_PHY_GPIO_CTL1	XSPI PHY GPIO Control Register 1	0x00000000
0x3101E090	XSPI0_PHY_GPIO_STAT0	XSPI0 PHY GPIO Status Register 0	0x00000000
0x3101E090	XSPI_PHY_GPIO_STAT0	XSPI PHY GPIO Status Register 0	0x00000000
0x3101E094	XSPI0_PHY_GPIO_STAT1	XSPI0 PHY GPIO Status Register 1	0x00000000
0x3101E094	XSPI_PHY_GPIO_STAT1	XSPI PHY GPIO Status Register 1	0x00000000

Table A-198: ADSP-2184x XSPI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3114C000	XSPI1_CMD0	XSPI1 Command Register 0	0x00000000
0x3114C004	XSPI1_CMD1	XSPI1 Command Register 1	0x00000000
0x3114C008	XSPI1_CMD2	XSPI1 Command Register 2	0x00000000
0x3114C00C	XSPI1_CMD3	XSPI1 Command Register 3	0x00000000
0x3114C010	XSPI1_CMD4	XSPI1 Command Register 4	0x00000000
0x3114C014	XSPI1_CMD5	XSPI1 Command Register 5	0x00000000
0x3114C040	XSPI1_CMD_STAT_PTR	XSPI1 Command Status Pointer Register	0x00000000
0x3114C044	XSPI1_CMD_STAT	XSPI1 Command Status Register	0x00000000

Table A-198: ADSP-2184x XSPI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114C100	XSPI1_GSTAT	XSPI1 General Controller Status Register	0x00000000
0x3114C104	XSPI1_TRD_STAT	XSPI1 Auto Command Engine Thread Status Register	0x00000000
0x3114C110	XSPI1_ISTAT	XSPI1 Interrupt Status Register	0x00000000
0x3114C114	XSPI1_INT_EN	XSPI1 Interrupt Enable Register	0x00000000
0x3114C120	XSPI1_TRD_COMP_ISTAT	XSPI1 Auto Command Engine Interrupt Status Thread Register	0x00000000
0x3114C130	XSPI1_TRD_ERR_ISTAT	XSPI1 Thread Error Interrupt Status Register	0x00000000
0x3114C134	XSPI1_TRD_ERR_INT_EN	XSPI1 Thread Error Interrupt Enable Register	0x00000000
0x3114C150	XSPI1_DMA_ERRADDR_LO	XSPI1 DMA Error Address Low Register	0x00000000
0x3114C154	XSPI1_DMA_ERRADDR_HI	XSPI1 DMA Error Address High Register	0x00000000
0x3114C158	XSPI1_BOOT_STAT	XSPI1 Boot Status Register	0x00000000
0x3114C208	XSPI1_LONGPOL_GCTL	XSPI1 Long Polling Count Register	0x000003E8
0x3114C20C	XSPI1_SHORTPOL_GCTL	XSPI1 Short Polling Count Register	0x000001F4
0x3114C230	XSPI1_WORKMODE_CTL	XSPI1 Device Control Register	0x00000000
0x3114C23C	XSPI1_DMA_CTL	XSPI1 DMA Interface Control Register	0x000D0000
0x3114C240	XSPI1_SDMA_SIZ	XSPI1 Host DMA Block Size Register	0x00000000
0x3114C244	XSPI1_SDMA_TRD_STAT	XSPI1 Host DMA Thread Status Register	0x00000000
0x3114C24C	XSPI1_SDMA_ADDR0	XSPI1 Host DMA Buffer Address Register 0	0x00000000
0x3114C250	XSPI1_SDMA_ADDR1	XSPI1 Host DMA Buffer Address Register 1	0x00000000
0x3114C260	XSPI1_DISCOVERY_GCTL	XSPI1 Device Discovery Control Register	0x00000000
0x3114C388	XSPI1_XIP_GCTL	XSPI1 XIP Configuration Register	0x00FF0000
0x3114C390	XSPI1_SEQ_GCTL0	XSPI1 Sequence Configuration Register 0	0x0000208F
0x3114C394	XSPI1_SEQ_GCTL1	XSPI1 Sequence Configuration Register 1	0x00000000
0x3114C398	XSPI1_DAC_CFG	XSPI1 DAC Configuration Register	0x00000000
0x3114C39C	XSPI1_DAC_REMAPADDR0	XSPI1 Address Remapping Register 0	0x00000000
0x3114C3A0	XSPI1_DAC_REMAPADDR1	XSPI1 Address Remapping Register 1	0x00000000
0x3114C400	XSPI1_RST_SEQ_CFG0	XSPI1 Reset Sequence Configuration Register 0	0x00019966
0x3114C404	XSPI1_RST_SEQ_CFG1	XSPI1 Reset Sequence Configuration Register 1	0xD0669900

Table A-198: ADSP-2184x XSPI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114C410	XSPI1_ERS_SEQ_CFG0	XSPI1 Erase Sequence Configuration Register 0	0x00DF3020
0x3114C414	XSPI1_ERS_SEQ_CFG1	XSPI1 Erase Sequence Configuration Register 1	0x0000000C
0x3114C418	XSPI1_ERS_SEQ_CFG2	XSPI1 Erase Sequence Configuration Register 2	0x009F0060
0x3114C420	XSPI1_PROG_SEQ_CFG0	XSPI1 Program Sequence Configuration Register 0	0x00003002
0x3114C424	XSPI1_PROG_SEQ_CFG1	XSPI1 Program Sequence Configuration Register 1	0x0000FD00
0x3114C428	XSPI1_PROG_SEQ_CFG2	XSPI1 Program Sequence Configuration Register 2	0x00000002
0x3114C430	XSPI1_READ_SEQ_CFG0	XSPI1 Read Sequence Configuration Register 0	0x00003003
0x3114C434	XSPI1_READ_SEQ_CFG1	XSPI1 Read Sequence Configuration Register 1	0x0000FC00
0x3114C438	XSPI1_READ_SEQ_CFG2	XSPI1 Read Sequence Configuration Register 2	0x00000F0A
0x3114C440	XSPI1_WE_SEQ_CFG0	XSPI1 WEL Sequence Configuration Register	0x01F90006
0x3114C450	XSPI1_STAT_SEQ_CFG0	XSPI1 Status Checking Sequence Configuration Register 0	0x00000000
0x3114C454	XSPI1_STAT_SEQ_CFG1	XSPI1 Status Checking Sequence Configuration Register 1	0x00000000
0x3114C458	XSPI1_STAT_SEQ_CFG2	XSPI1 Status Checking Sequence Configuration Register 2	0x05000505
0x3114C45C	XSPI1_STAT_SEQ_CFG3	XSPI1 Status Checking Sequence Configuration Register 3	0xFA00FAFA
0x3114C460	XSPI1_STAT_SEQ_CFG4	XSPI1 Status Checking Sequence Configuration Register 4	0x00000F00
0x3114C464	XSPI1_STAT_SEQ_CFG5	XSPI1 Status Checking Sequence Configuration Register 5	0x00000040
0x3114C46C	XSPI1_STAT_SEQ_CFG7	XSPI1 Status Checking Sequence Configuration Register 7	0x00000000
0x3114C470	XSPI1_STAT_SEQ_CFG8	XSPI1 Status Checking Sequence Configuration Register 8	0x00000000
0x3114C474	XSPI1_STAT_SEQ_CFG9	XSPI1 Status Checking Sequence Configuration Register 9	0x00000000
0x3114C478	XSPI1_STAT_SEQ_CFG10	XSPI1 Status Checking Sequence Configuration Register 10	0x00000000
0x3114CF00	XSPI1_REVID	XSPI1 Revision ID Register	0x65220206
0x3114CF04	XSPI1_FEATURES	XSPI1 Controller Features Register	0x03710003
0x3114D000	XSPI1_MINICTL_WP_CTL	XSPI1 Write Protect Register	0x00000001
0x3114D004	XSPI1_MINICTL_RST_PIN_CTL	XSPI1 Hardware Reset Control Register	0x00000001
0x3114D008	XSPI1_MINICTL_CLKMODE	XSPI1 Clock Mode Control Register	0x00000000
0x3114D00C	XSPI1_MINICTL_JEDEC_RST_TR	XSPI1 JEDEC Reset Delay Register	0x00008080
0x3114D010	XSPI1_MINICTL_DEV_DLY	XSPI1 Device Delay Register	0x01000100

Table A-198: ADSP-2184x XSPI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3114D014	XSPI1_MINICTL_RST_RECOV	XSPI1 Reset Recovery Delay Register	0x0000000A
0x3114D018	XSPI1_MINICTL_DEV_ACTIVE_MAX	XSPI1 Device Active Maximum Clock Cycle Register	0x00000080
0x3114D020	XSPI1_MINICTL_HF_OFFSET	XSPI1 HyperFlash Offset Register	0x00000D03
0x3114D030	XSPI1_PHY_DLL_UPDT_CNT	XSPI1 PHY DLL Resynchronization Register	0x00000000
0x3114D034	XSPI1_PHY_DLL_CTL	XSPI1 PHY DLL Control Register	0x01030707
0x3114E000	XSPI1_PHY_DQ_TR	XSPI1 PHY DQ Timing Register	0x00000002
0x3114E004	XSPI1_PHY_DQS_TR	XSPI1 PHY DQS Timing Register	0x00100000
0x3114E008	XSPI1_PHY_GATE_LPBK_CTL	XSPI1 PHY Gate Loopback Control Register	0x00D80000
0x3114E00C	XSPI1_PHY_DLL_REQ_CTL	XSPI1 Bus Requester DLL Control Register	0x00800000
0x3114E010	XSPI1_PHY_DLL_COMP_CTL	XSPI1 Bus Completer DLL Control Register	0x00000000
0x3114E014	XSPI1_PHY_IE_TR	XSPI1 PHY DQS Input Enable Timing Register	0x00100000
0x3114E018	XSPI1_PHY_OB0	XSPI1 PHY Observable Points Register	0x00000000
0x3114E01C	XSPI1_PHY_DLLOB0	XSPI1 PHY DLL Observable Points 0 Register	0x00000000
0x3114E020	XSPI1_PHY_DLLOB1	XSPI1 PHY DLL Observable Points 1 Register	0x00000000
0x3114E028	XSPI1_PHY_STATIC_TGL	XSPI1 PHY Static Aging Register	0x00000000
0x3114E034	XSPI1_PHY_WR_DESKEW_PAD_CTL0	XSPI1 PHY Deskew Write Register	0x00000000
0x3114E070	XSPI1_PHY_REVID	XSPI1 PHY Revision ID Register	0x61820107
0x3114E074	XSPI1_PHY_FEATURES	XSPI1 PHY Features Register	0x000019E8
0x3114E080	XSPI1_PHY_GCTL	XSPI1 PHY Global Control Register	0x00000180
0x3114E084	XSPI1_PHY_GTSEL	XSPI1 PHY Global Termination Control Register	0x00000000
0x3114E088	XSPI1_PHY_GPIO_CTL0	XSPI1 PHY GPIO Control Register 0	0x00000000
0x3114E08C	XSPI1_PHY_GPIO_CTL1	XSPI1 PHY GPIO Control Register 1	0x00000000
0x3114E090	XSPI1_PHY_GPIO_STAT0	XSPI1 PHY GPIO Status Register 0	0x00000000
0x3114E094	XSPI1_PHY_GPIO_STAT1	XSPI1 PHY GPIO Status Register 1	0x00000000